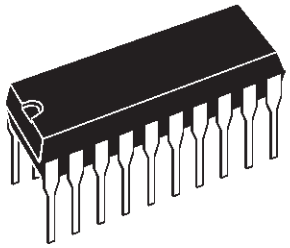
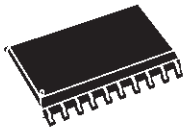
ST62T08C/T09C
ST62T10C/T20C/E20C

8-BIT OTP/EPROM MCUs WITH A/D CONVERTER, OSCILLATOR SAFEGUARD, SAFE RESET AND 20 PINS

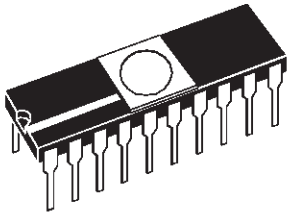
- 3.0 to 6.0V Supply Operating Range
- 8 MHz Maximum Clock Frequency
- -40 to +125°C Operating Temperature Range
- Run, Wait and Stop Modes
- 5 Interrupt Vectors
- Look-up Table capability in Program Memory
- Data Storage in Program Memory: User selectable size
- Data RAM: 64bytes
- User Programmable Options
- 12 I/O pins, fully programmable as:
  - Input with pull-up resistor
  - Input without pull-up resistor
  - Input with interrupt generation
  - Open-drain or push-pull output
  - Analog Input (except ST62T08C)
- 4 I/O lines can sink up to 20mA to drive LEDs or TRIACs directly
- 8-bit Timer/Counter with 7-bit programmable prescaler
- Digital Watchdog
- Oscillator Safe Guard
- Low Voltage Detector for Safe Reset
- 8-bit A/D Converter with up to 8 analog inputs
- On-chip Clock oscillator can be driven by Quartz Crystal Ceramic resonator or RC network
- Power-on Reset
- One external Non-Maskable Interrupt
- ST626x-EMU2 Emulation and Development System (connects to an MS-DOS PC via a parallel port)



PDIP20



PSO20



CDIP20W

(See end of Datasheet for Ordering Information)

DEVICE SUMMARY

DEVICE	OTP (Bytes)	EPROM (Bytes)	I/O Pins	Analog inputs
ST62T08C	1036	-	12	-
ST62T09C	1036	-	12	4
ST62T10C	1836	-	12	8
ST62T20C	3884	-	12	8
ST62E20C	-	3884	12	8

Rev. 2.7

Table of Contents

Document Page

ST62T08C/T09C/ST62T10C/T20C/E20C 1

1 GENERAL DESCRIPTION 4
1.1 INTRODUCTION 4
1.2 PIN DESCRIPTIONS 5
1.3 MEMORY MAP 6
1.3.1 Introduction 6
1.3.2 Program Space 7
1.3.3 Data Space 8
1.3.4 Stack Space 8
1.3.5 Data Window Register (DWR) 9
1.4 PROGRAMMING MODES 10
1.4.1 Option Bytes 10
1.4.2 Program Memory 11
1.4.3 EPROM Erasing 11
2 CENTRAL PROCESSING UNIT 12
2.1 INTRODUCTION 12
2.2 CPU REGISTERS 12
3 CLOCKS, RESET, INTERRUPTS AND POWER SAVING MODES 14
3.1 CLOCK SYSTEM 14
3.1.1 Main Oscillator 14
3.1.2 Low Frequency Auxiliary Oscillator (LFAO) 15
3.1.3 Oscillator Safe Guard 15
3.2 RESETS 18
3.2.1 RESET Input 18
3.2.2 Power-on Reset 18
3.2.3 Watchdog Reset 19
3.2.4 LVD Reset 19
3.2.5 Application Notes 19
3.2.6 MCU Initialization Sequence 20
3.3 DIGITAL WATCHDOG 22
3.3.1 Digital Watchdog Register (DWDR) 24
3.3.2 Application Notes 24
3.4 INTERRUPTS 26
3.4.1 Interrupt request 26
3.4.2 Interrupt Procedure 27
3.4.3 Interrupt Option Register (IOR) 28
3.4.4 Interrupt sources 28
3.5 POWER SAVING MODES 30
3.5.1 WAIT Mode 30
3.5.2 STOP Mode 30
3.5.3 Exit from WAIT and STOP Modes 31
4 ON-CHIP PERIPHERALS 32
4.1 I/O PORTS 32
4.1.1 Operating Modes 33
4.1.2 Safe I/O State Switching Sequence 34
4.1.3 I/O Port Option Registers 35

Table of Contents

Document Page

4.1.4 I/O Port Data Direction Registers . . . 35
4.1.5 I/O Port Data Registers . . . 35
4.2 TIMER . . . 37
4.2.1 Timer Operating Modes . . . 38
4.2.2 Timer Interrupt . . . 38
4.2.3 Application Notes . . . 39
4.2.4 Timer Registers . . . 39
4.3 A/D CONVERTER (ADC) . . . 40
4.3.1 Application Notes . . . 40
5 SOFTWARE . . . 42
5.1 ST6 ARCHITECTURE . . . 42
5.2 ADDRESSING MODES . . . 42
5.3 INSTRUCTION SET . . . 43
6 ELECTRICAL CHARACTERISTICS . . . 48
6.1 ABSOLUTE MAXIMUM RATINGS . . . 48
6.2 RECOMMENDED OPERATING CONDITIONS . . . 49
6.3 DC ELECTRICAL CHARACTERISTICS . . . 50
6.4 AC ELECTRICAL CHARACTERISTICS . . . 51
6.5 A/D CONVERTER CHARACTERISTICS . . . 52
6.6 TIMER CHARACTERISTICS . . . 52
7 GENERAL INFORMATION . . . 58
7.1 PACKAGE MECHANICAL DATA . . . 58
7.2 .ORDERING INFORMATION . . . 59

ST62P08C/P09C/ST62P10C/P20C . . . 61

1 GENERAL DESCRIPTION . . . 62
1.1 INTRODUCTION . . . 62
1.2 ORDERING INFORMATION . . . 62
1.2.1 Transfer of Customer Code . . . 62
1.2.2 Listing Generation and Verification . . . 62

ST6208C/09C/ST6210C/20C . . . 65

1 GENERAL DESCRIPTION . . . 66
1.1 INTRODUCTION . . . 66
1.2 ROM READOUT PROTECTION . . . 66
1.3 ORDERING INFORMATION . . . 68
1.3.1 Transfer of Customer Code . . . 68
1.3.2 Listing Generation and Verification . . . 68

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.

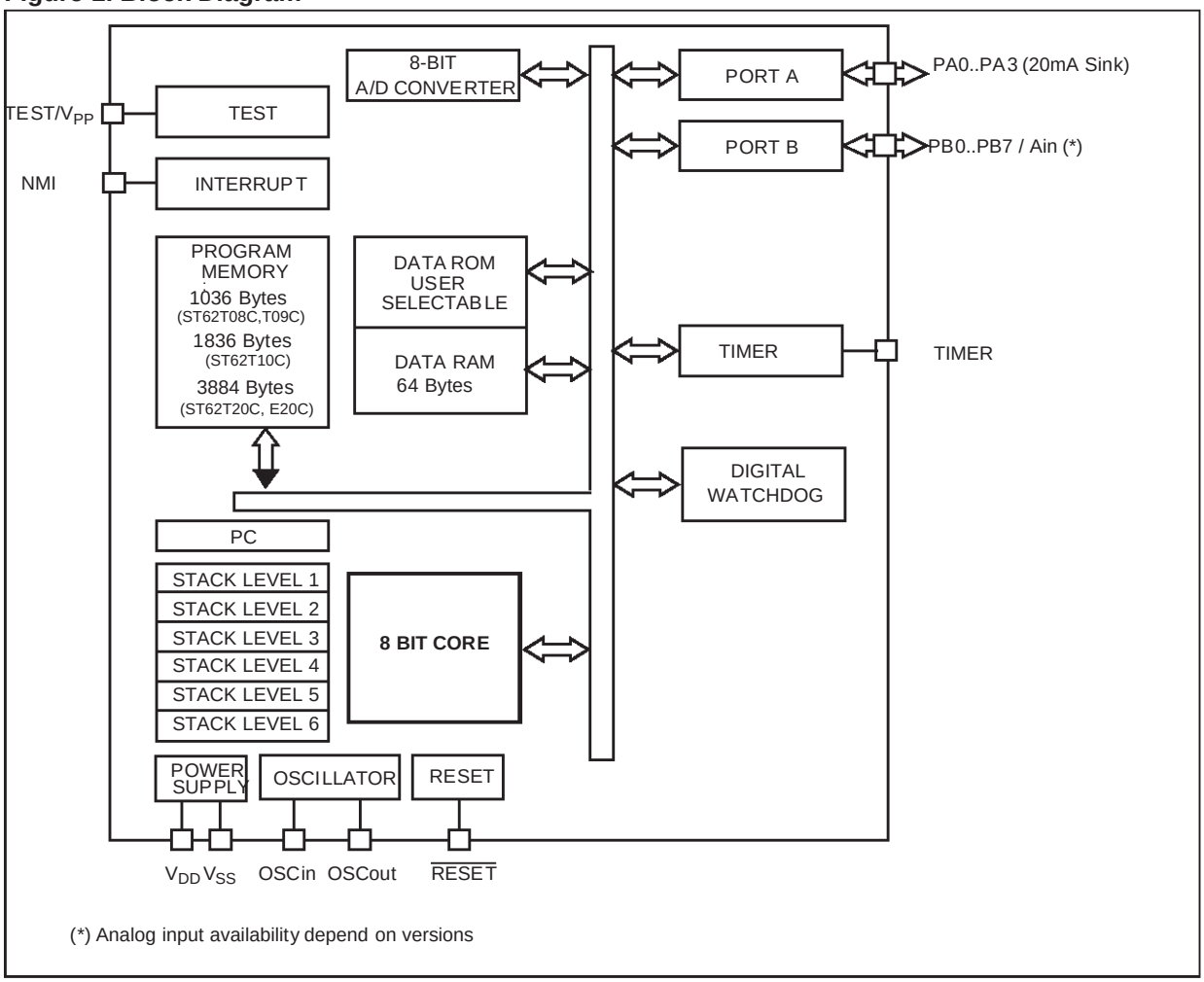
The ST62E20C is the erasable EPROM version of the ST62T08C,T09C,T10C and T20C device, which may be used to emulate the ST62T08C,T09C,T10C and T20C device, as well as the respective ST6208C,09C,10C,20C ROM devices.

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

1.2 PIN DESCRIPTIONS

V_{DD} and V_{SS}. Power is supplied to the MCU via these two pins. V_{DD} is the power connection and V_{SS} is the ground connection.

OSCin and OSCout. These pins are internally connected to the on-chip oscillator circuit. A quartz crystal, a ceramic resonator or an external clock signal can be connected between these two pins. The OSCin pin is the input pin, the OSCout pin is the output pin.

$\overline{\text{RESET}}$. The active-low $\overline{\text{RESET}}$ pin is used to restart the microcontroller. Internal pull-up is provided at this pin.

TEST/V_{PP}. The TEST must be held at V_{SS} for normal operation. If TEST pin is connected to a +12.5V level during the reset phase, the EPROM programming Mode is entered.

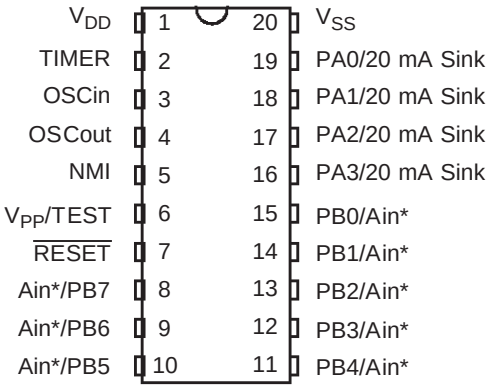
NMI. The NMI pin provides the capability for asynchronous interruption, by applying an external non maskable interrupt to the MCU. The NMI input is falling edge sensitive. The user can select as option the availability of an on-chip pull-up at this pin.

TIMER. This is the timer I/O pin. In input mode it is connected to the prescaler and acts as external timer clock input or as control gate input for the internal timer clock. In output mode the timer pin outputs the data bit when a time-out occurs. The user can select as option the availability of an on-chip pull-up at this pin.

PA0-PA3. These 4 lines are organized as one I/O port (A). Each line may be configured under software control as inputs with or without internal pull-up resistors, interrupt generating inputs with pull-up resistors, open-drain or push-pull outputs. PA0-PA3 can also sink 20mA for direct LED driving.

PB0-PB7. These 8 lines are organized as one I/O port (B). Each line may be configured under software control as inputs with or without internal pull-up resistors, interrupt generating inputs with pull-up resistors, open-drain or push-pull outputs. PB0-PB3 can be used as analog input to the A/D converter on the ST62T10C, T20C and E20C, while PB4-PB7 can be used as analog inputs for the A/D converter on the ST62T09C, T10C, T20C and E20C.

Figure 2. ST62T08C,T09C, T10C, T20C and E20C Pin Configuration



*Analog input availability depend on device

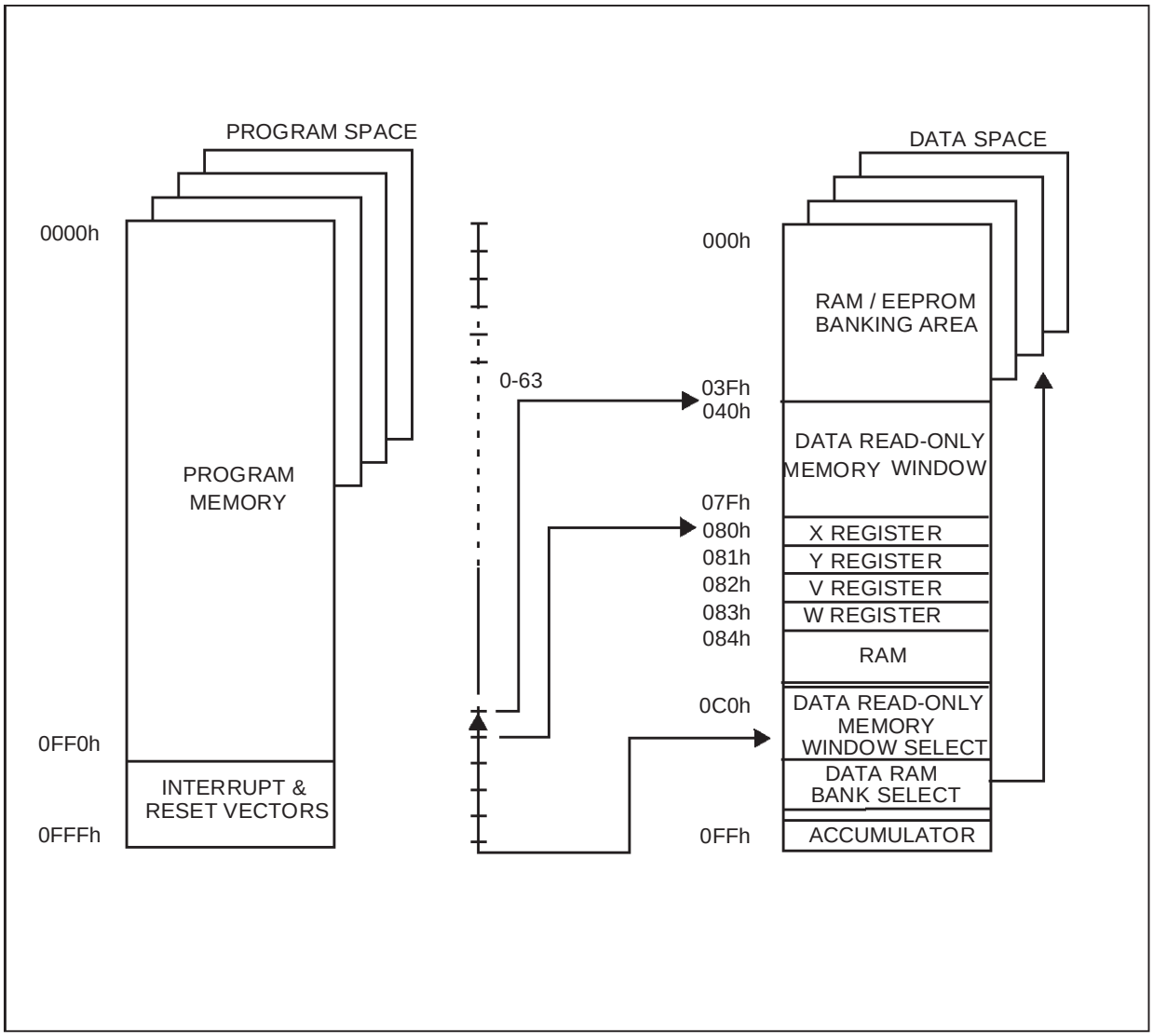
1.3 MEMORY MAP

1.3.1 Introduction

The MCU operates in three separate memory spaces: Program space, Data space, and Stack space. Operation in these three memory spaces is described in the following paragraphs.

Briefly, Program space contains user program code in OTP and user vectors; Data space contains user data in RAM and in OTP, and Stack space accommodates six levels of stack for subroutine and interrupt service routine nesting.

Figure 3. Memory Addressing Diagram



MEMORY MAP (Cont'd)

1.3.2 Program Space

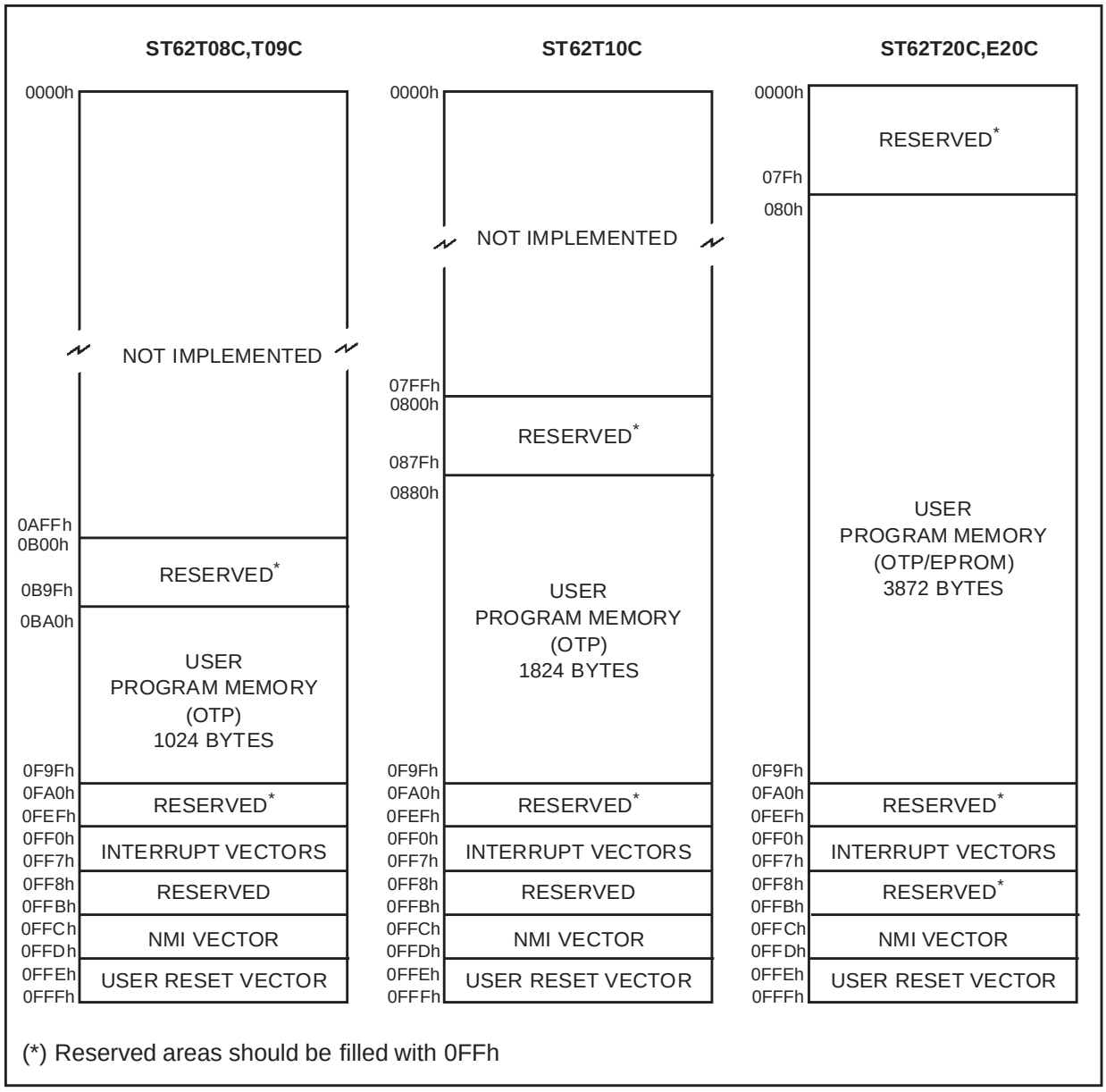
Program Space comprises the instructions to be executed, the data required for immediate addressing mode instructions, the reserved factory test area and the user vectors. Program Space is addressed via the 12-bit Program Counter register (PC register)Program Memory Protection.

The Program Memory in OTP or EPROM devices can be protected against external readout of memory by selecting the READOUT PROTECTION option in the option byte.

In the EPROM parts, READOUT PROTECTION option can be disactivated only by U.V. erasure that also results into the whole EPROM context erasure.

Note: Once the Readout Protection is activated, it is no longer possible, even for STMicroelectronics, to gain access to the OTP contents. Returned parts with a protection set can therefore not be accepted.

Figure 4. Program Memory Map



(*) Reserved areas should be filled with 0FFh

MEMORY MAP (Cont'd)

1.3.3 Data Space

Data Space accommodates all the data necessary for processing the user program. This space comprises the RAM resource, the processor core and peripheral registers, as well as read-only data such as constants and look-up tables in OTP/EPROM.

1.3.3.1 Data ROM

All read-only data is physically stored in program memory, which also accommodates the Program Space. The program memory consequently contains the program code to be executed, as well as the constants and look-up tables required by the application.

The Data Space locations in which the different constants and look-up tables are addressed by the processor core may be thought of as a 64-byte window through which it is possible to access the read-only data stored in OTP/EPROM.

1.3.3.2 Data RAM

In ST6208C/09C/10C/20C devices, the data space includes 60 bytes of RAM, the accumulator (A), the indirect registers (X), (Y), the short direct registers (V), (W), the I/O port registers, the peripheral data and control registers, the interrupt option register and the Data ROM Window register (DRW register).

1.3.4 Stack Space

Stack space consists of six 12-bit registers which are used to stack subroutine and interrupt return addresses, as well as the current program counter contents.

Table 1. ST6208C/09C/10C/20C Data Memory Space

RESERVED	000h 03Fh
DATA ROM WINDOW AREA 64 BYTES	040h 07Fh
X REGISTER	080h
Y REGISTER	081h
V REGISTER	082h
W REGISTER	083h
DATA RAM 60 BYTES	084h 0BFh
PORT A DATA REGISTER	0C0h
PORT B DATA REGISTER	0C1h
RESERVED	0C2h
RESERVED	0C3h
PORT A DIRECTION REGISTER	0C4h
PORT B DIRECTION REGISTER	0C5h
RESERVED	0C6h
RESERVED	0C7h
INTERRUPT OPTION REGISTER	0C8h*
DATA ROM WINDOW REGISTER	0C9h*
RESERVED	0CAh 0CBh
PORT A OPTION REGISTER	0CCh
PORT B OPTION REGISTER	0CDh
RESERVED	0CEh
RESERVED	0CFh
A/D DATA REGISTER(except ST62T08C)	0D0h
A/D CONTROL REGISTER (except ST62T08C)	0D1h
TIMER PRESCALER REGISTER	0D2h
TIMER COUNTER REGISTER	0D3h
TIMER STATUS CONTROL REGISTER	0D4h
RESERVED	0D5h 0D6h 0D7h
WATCHDOG REGISTER	0D8h
RESERVED	0D9h 0FEh
ACCUMULATOR	0FFh

* WRITE ONLY REGISTER

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

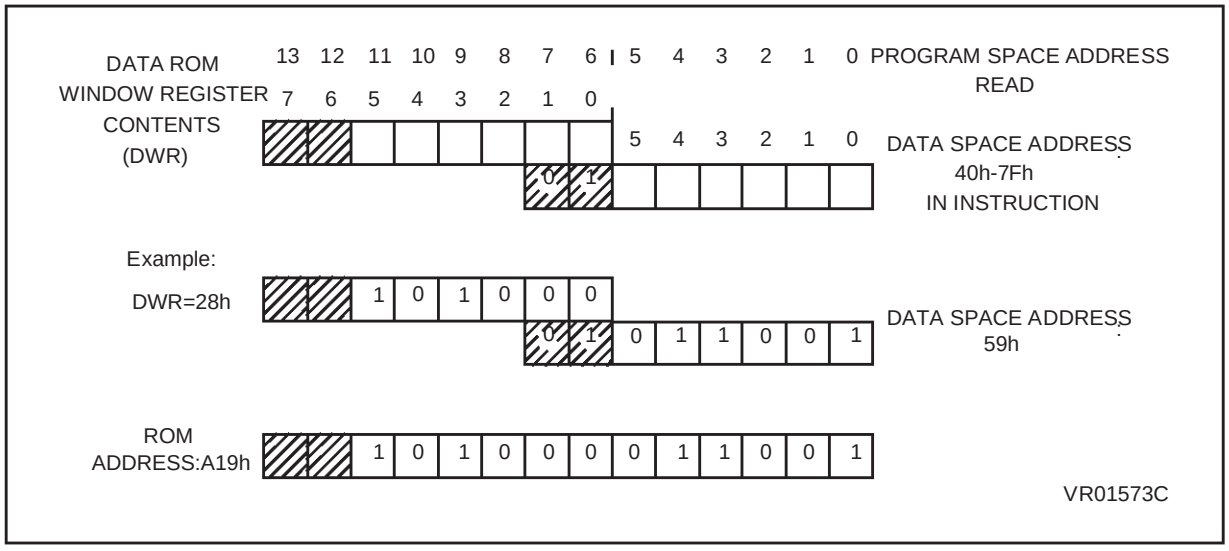
Bits 6, 7 = Not used.

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



1.4 PROGRAMMING MODES

1.4.1 Option Bytes

The two Option Bytes allow configuration capability to the MCUs. Option byte's content is automatically read, and the selected options enabled, when the chip reset is activated.

It can only be accessed during the programming mode. This access is made either automatically (copy from a master device) or by selecting the OPTION BYTE PROGRAMMING mode of the programmer.

The option bytes are located in a non-user map. No address has to be specified.

EPROM Code Option Byte (LSB)

7							0
PRO-TECT	OSCIL	-	-	NMI PULL	TIM PULL	WDACT	OS-GEN

EPROM Code Option Byte (MSB)

15							8
-	-	-	-	-	-	EXTC-NTL	LVD

D15-D10. Reserved. Must be cleared

EXTCNTL. *External STOP MODE control.*. When EXTCNTL is high, STOP mode is available with watchdog active by setting NMI pin to one.. When EXTCNTL is low, STOP mode is not available with the watchdog active.

LVD. *LVD RESET enable.*When this bit is set, safe RESET is performed by MCU when the supply voltage is too low. When this bit is cleared, only power-on reset or external RESET are active.

PROTECT. *Readout Protection.* This bit allows the protection of the software contents against piracy. When the bit PROTECT is set high, readout of the OTP contents is prevented by hardware.. When this bit is low, the user program can be read.

OSCIL. *Oscillator selection*. When this bit is low, the oscillator must be controlled by a quartz crystal, a ceramic resonator or an external frequency. When it is high, the oscillator must be controlled by an RC network, with only the resistor having to be externally provided.

D5. Reserved. Must be cleared to zero.

D4. Reserved. Must be set to one.

NMI PULL. *NMI Pull-Up.* This bit must be set high to configure the NMI pin with a pull-up resistor. When it is low, no pull-up is provided**.**

TIM PULL.*TIM Pull-Up*. This bit must be set high to configure the TIMER pin with a pull-up resistor. When it is low, no pull-up is provided**.**

WDACT. This bit controls the watchdog activation. When it is high, hardware activation is selected. The software activation is selected when WDACT is low.

OSGEN. *Oscillator Safe Guard*. This bit must be set high to enable the Oscillator Safe Guard. When this bit is low, the OSG is disabled.

The Option byte is written during programming either by using the PC menu (PC driven Mode) or automatically (stand-alone mode)

PROGRAMMING MODES (Cont'd)

1.4.2 Program Memory

EPROM/OTP programming mode is set by a +12.5V voltage applied to the TEST/V_{PP} pin. The programming flow of the ST62T08C,T09C,T10C,T20C/E20C is described in the User Manual of the EPROM Programming Board.

Table 2. ST62T08C, T09C Program Memory Map

Device Address	Description
0000h-0B9Fh	Reserved
0BA0h-0F9Fh	User ROM
0FA0h-0FEFh	Reserved
0FF0h-0FF7h	Interrupt Vectors
0FF8h-0FFBh	Reserved
0FFCh-0FFDh	NMI Interrupt Vector
0FFEh-0FFFh	Reset Vector

Table 3. ST62T10C Program Memory Map

Device Address	Description
0000h-087Fh	Reserved
0880h-0F9Fh	User ROM
0FA0h-0FEFh	Reserved
0FF0h-0FF7h	Interrupt Vectors
0FF8h-0FFBh	Reserved
0FFCh-0FFDh	NMI Interrupt Vector
0FFEh-0FFFh	Reset Vector

Table 4. ST62T20C,E20C Program Memory Map

Device Address	Description
0000h-007Fh	Reserved
0080h-0F9Fh	User ROM
0FA0h-0FEFh	Reserved
0FF0h-0FF7h	Interrupt Vectors
0FF8h-0FFBh	Reserved
0FFCh-0FFDh	NMI Interrupt Vector
0FFEh-0FFFh	Reset Vector

Note: OTP/EPROM devices can be programmed with the development tools available from STMicroelectronics (ST62E2X-EPB or ST622X-KIT).

1.4.3 EPROM Erasing

The EPROM of the windowed package of the MCUs may be erased by exposure to Ultra Violet light. The erasure characteristic of the MCUs is such that erasure begins when the memory is exposed to light with a wave lengths shorter than approximately 4000Å. It should be noted that sunlights and some types of fluorescent lamps have wavelengths in the range 3000-4000Å.

It is thus recommended that the window of the MCUs packages be covered by an opaque label to prevent unintentional erasure problems when testing the application in such an environment.

The recommended erasure procedure of the MCUs EPROM is the exposure to short wave ultraviolet light which have a wave-length 2537A. The integrated dose (i.e. U.V. intensity x exposure time) for erasure should be a minimum of 30W-sec/cm^2. The erasure time with this dosage is ap-

2 CENTRAL PROCESSING UNIT

2.1 INTRODUCTION

The CPU Core of ST6 devices is independent of the I/O or Memory configuration. As such, it may be thought of as an independent central processor communicating with on-chip I/O, Memory and Peripherals via internal address, data, and control buses. In-core communication is arranged as shown in Figure 6; the controller being externally linked to both the Reset and Oscillator circuits, while the core is linked to the dedicated on-chip peripherals via the serial data bus and indirectly, for interrupt purposes, through the control registers.

2.2 CPU REGISTERS

The ST6 Family CPU core features six registers and three pairs of flags available to the programmer. These are described in the following paragraphs.

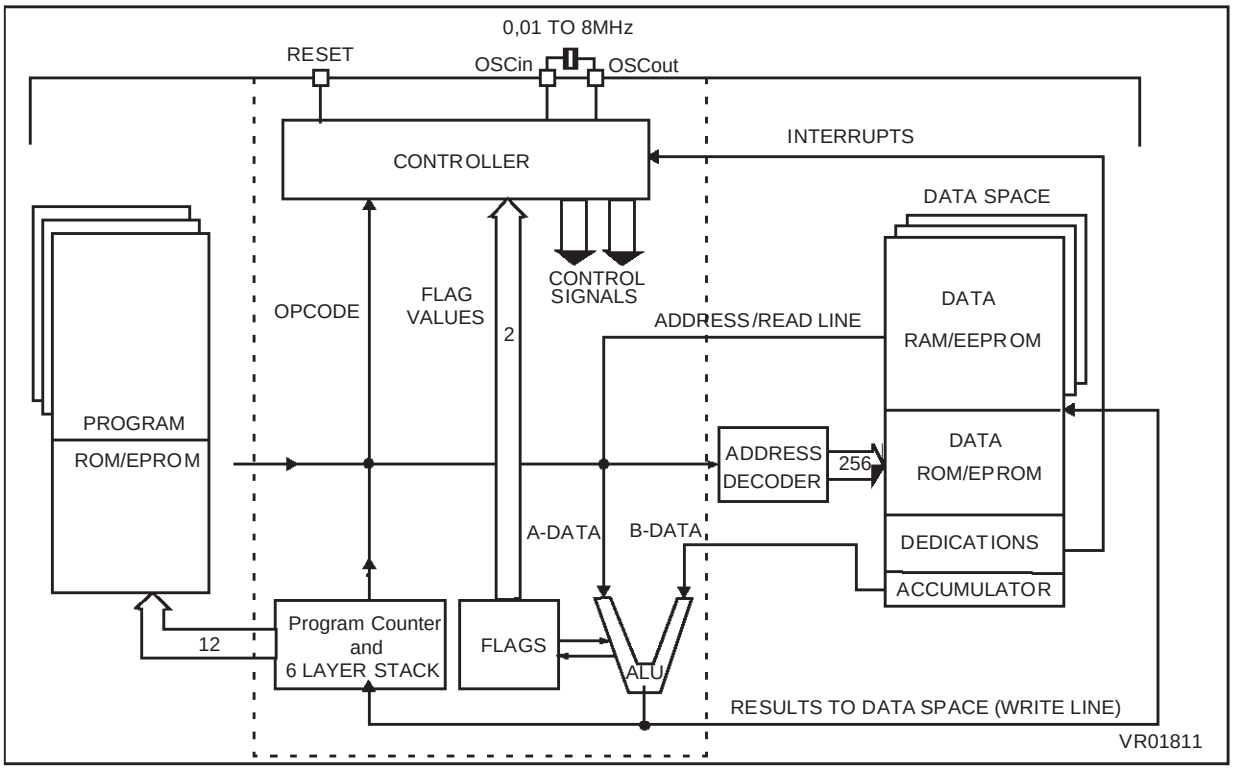
Accumulator (A). The accumulator is an 8-bit general purpose register used in all arithmetic calculations, logical operations, and data manipulations. The accumulator can be addressed in Data space as a RAM location at address FFh. Thus the ST6 can manipulate the accumulator just like any other register in Data space.

Indirect Registers (X, Y). These two indirect registers are used as pointers to memory locations in Data space. They are used in the register-indirect addressing mode. These registers can be addressed in the data space as RAM locations at addresses 80h (X) and 81h (Y). They can also be accessed with the direct, short direct, or bit direct addressing modes. Accordingly, the ST6 instruction set can use the indirect registers as any other register of the data space.

Short Direct Registers (V, W). These two registers are used to save a byte in short direct addressing mode. They can be addressed in Data space as RAM locations at addresses 82h (V) and 83h (W). They can also be accessed using the direct and bit direct addressing modes. Thus, the ST6 instruction set can use the short direct registers as any other register of the data space.

Program Counter (PC). The program counter is a 12-bit register which contains the address of the next ROM location to be processed by the core. This ROM location may be an opcode, an operand, or the address of an operand. The 12-bit length allows the direct addressing of 4096 bytes in Program space.

Figure 6. ST6 Core Block Diagram



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.

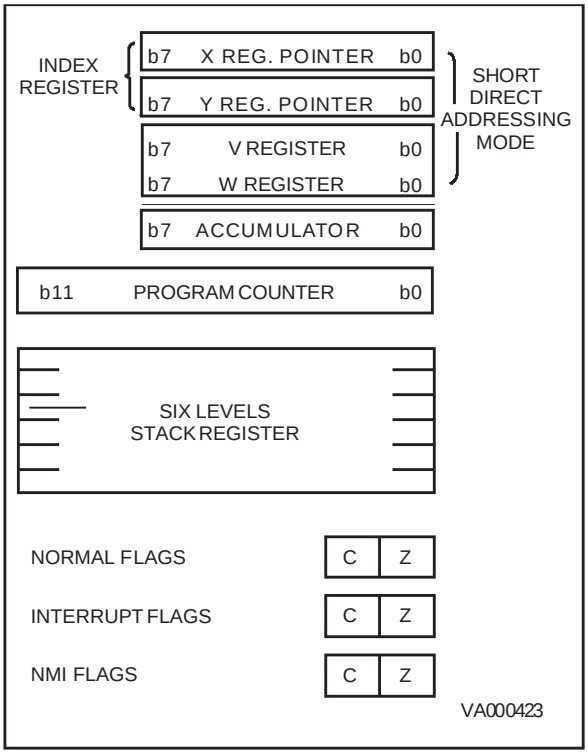
The Carry flag is set when a carry or a borrow occurs during arithmetic operations; otherwise it is cleared. The Carry flag is also set to the value of the bit tested in a bit test instruction; it also participates in the rotate left instruction.

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 CLOCKS, RESET, INTERRUPTS AND POWER SAVING MODES

3.1 CLOCK SYSTEM

The MCU features a Main Oscillator which can be driven by an external clock, or used in conjunction with an AT-cut parallel resonant crystal or a suitable ceramic resonator, or with an external resistor (R_{NET}). In addition, a Low Frequency Auxiliary Oscillator (LFAO) can be switched in for security reasons, to reduce power consumption, or to offer the benefits of a back-up clock system.

The Oscillator Safeguard (OSG) option filters spikes from the oscillator lines, provides access to the LFAO to provide a backup oscillator in the event of main oscillator failure and also automatically limits the internal clock frequency (f_{INT}) as a function of V_{DD}, in order to guarantee correct operation. These functions are illustrated in Figure 9, Figure 10, Figure 11 and Figure 12.

Figure 8 illustrates various possible oscillator configurations using an external crystal or ceramic resonator, an external clock input, an external resistor (R_{NET}), or the lowest cost solution using only the LFAO. C_{L1} an C_{L2} should have a capacitance in the range 12 to 22 pF for an oscillator frequency in the 4-8 MHz range.

The internal MCU clock frequency (f_{INT}) is divided by 12 to drive the Timer, the A/D converter and the Watchdog timer, and by 13 to drive the CPU core, as may be seen in Figure 11.

With an 8MHz oscillator frequency, the fastest machine cycle is therefore 1.625µs.

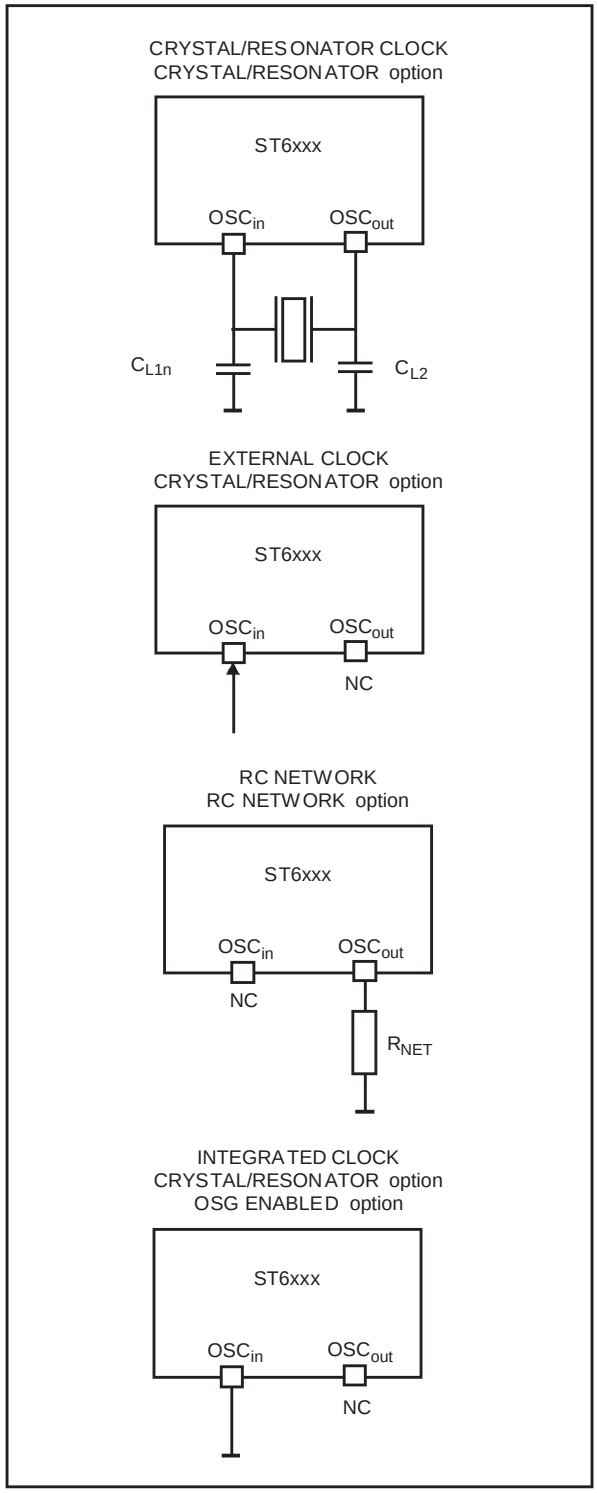
A machine cycle is the smallest unit of time needed to execute any operation (for instance, to increment the Program Counter). An instruction may require two, four, or five machine cycles for execution.

3.1.1 Main Oscillator

The oscillator configuration may be specified by selecting the appropriate option. When the CRYSTAL/RESONATOR option is selected, it must be used with a quartz crystal, a ceramic resonator or an external signal provided on the OSCin pin. When the RC NETWORK option is selected, the system clock is generated by an external resistor.

The main oscillator can be turned off (when the OSG ENABLED option is selected) by setting the OSCOFF bit of the ADC Control Register. The Low Frequency Auxiliary Oscillator is automatically started.

Figure 8. Oscillator Configurations



CLOCK SYSTEM (Cont'd)

Turning on the main oscillator is achieved by resetting the OSCOFF bit of the A/D Converter Control Register or by resetting the MCU. Restarting the main oscillator implies a delay comprising the oscillator start up delay period plus the duration of the software instruction at f_{LFAO} clock frequency.

3.1.2 Low Frequency Auxiliary Oscillator (LFAO)

The Low Frequency Auxiliary Oscillator has three main purposes. Firstly, it can be used to reduce power consumption in non timing critical routines. Secondly, it offers a fully integrated system clock, without any external components. Lastly, it acts as a safety oscillator in case of main oscillator failure.

This oscillator is available when the OSG ENABLED option is selected. In this case, it automatically starts one of its periods after the first missing edge from the main oscillator, whatever the reason (main oscillator defective, no clock circuitry provided, main oscillator switched off...).

User code, normal interrupts, WAIT and STOP instructions, are processed as normal, at the reduced f_{LFAO} frequency. The A/D converter accuracy is decreased, since the internal frequency is below 1MHz.

At power on, the Low Frequency Auxiliary Oscillator starts faster than the Main Oscillator. It therefore feeds the on-chip counter generating the POR delay until the Main Oscillator runs.

The Low Frequency Auxiliary Oscillator is automatically switched off as soon as the main oscillator starts.

ADCR

Address: 0D1h — Read/Write

7							0
ADCR 7	ADCR 6	ADCR 5	ADCR 4	ADCR 3	OSC OFF	ADCR 1	ADCR 0

Bit 7-3, 1-0= **ADCR7-ADCR3, ADCR1-ADCR0**: *ADC Control Register***.** These bits are not used.

Bit 2 = **OSCOFF**. When low, this bit enables main oscillator to run. The main oscillator is switched off when OSCOFF is high.

3.1.3 Oscillator Safe Guard

The Oscillator Safe Guard (OSG) affords drastically increased operational integrity in ST62xx devices. The OSG circuit provides three basic functions: it filters spikes from the oscillator lines which would result in over frequency to the ST62 CPU; it gives access to the Low Frequency Auxiliary Oscillator (LFAO), used to ensure minimum processing in case of main oscillator failure, to offer reduced power consumption or to provide a fixed frequency low cost oscillator; finally, it automatically limits the internal clock frequency as a function of supply voltage, in order to ensure correct operation even if the power supply should drop.

The OSG is enabled or disabled by choosing the relevant OSG option. It may be viewed as a filter whose cross-over frequency is device dependent.

Spikes on the oscillator lines result in an effectively increased internal clock frequency. In the absence of an OSG circuit, this may lead to an over frequency for a given power supply voltage. The OSG filters out such spikes (as illustrated in Figure 9). In all cases, when the OSG is active, the maximum internal clock frequency, f_{INT}, is limited to f_{OSG}, which is supply voltage dependent. This relationship is illustrated in Figure 12.

When the OSG is enabled, the Low Frequency Auxiliary Oscillator may be accessed. This oscillator starts operating after the first missing edge of the main oscillator (see Figure 10).

Over-frequency, at a given power supply level, is seen by the OSG as spikes; it therefore filters out some cycles in order that the internal clock frequency of the device is kept within the range the particular device can stand (depending on V_{DD}), and below f_{OSG}: the maximum authorised frequency with OSG enabled.

Note. The OSG should be used wherever possible as it provides maximum safety. Care must be taken, however, as it can increase power consumption and reduce the maximum operating frequency to f_{OSG}.

Warning: Care has to be taken when using the OSG, as the internal frequency is defined between a minimum and a maximum value and is not accurate.

For precise timing measurements, it is not recommended to use the OSG and it should not be enabled in applications that use the SPI or the UART.

It should also be noted that power consumption in Stop mode is higher when the OSG is enabled (around 50µA at nominal conditions and room temperature).

CLOCK SYSTEM (Cont'd)

Figure 9. OSG Filtering Principle

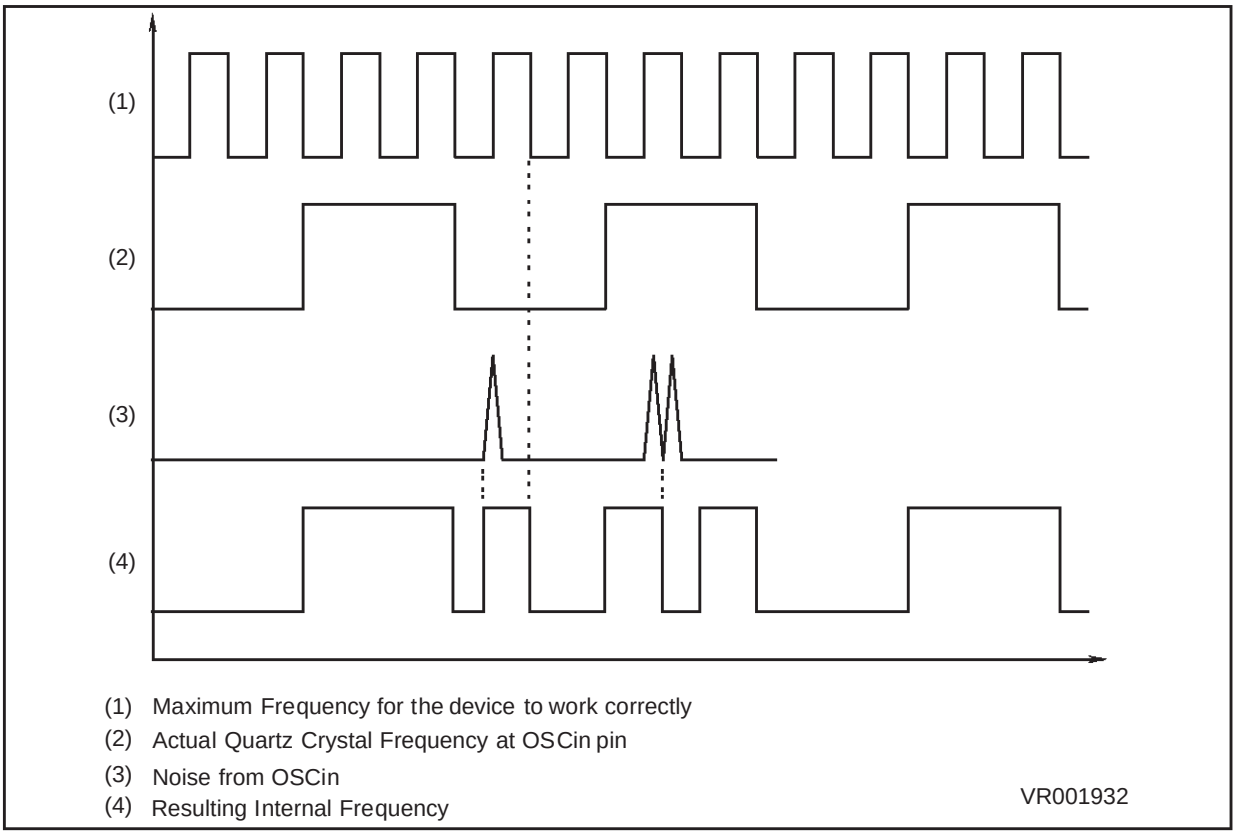
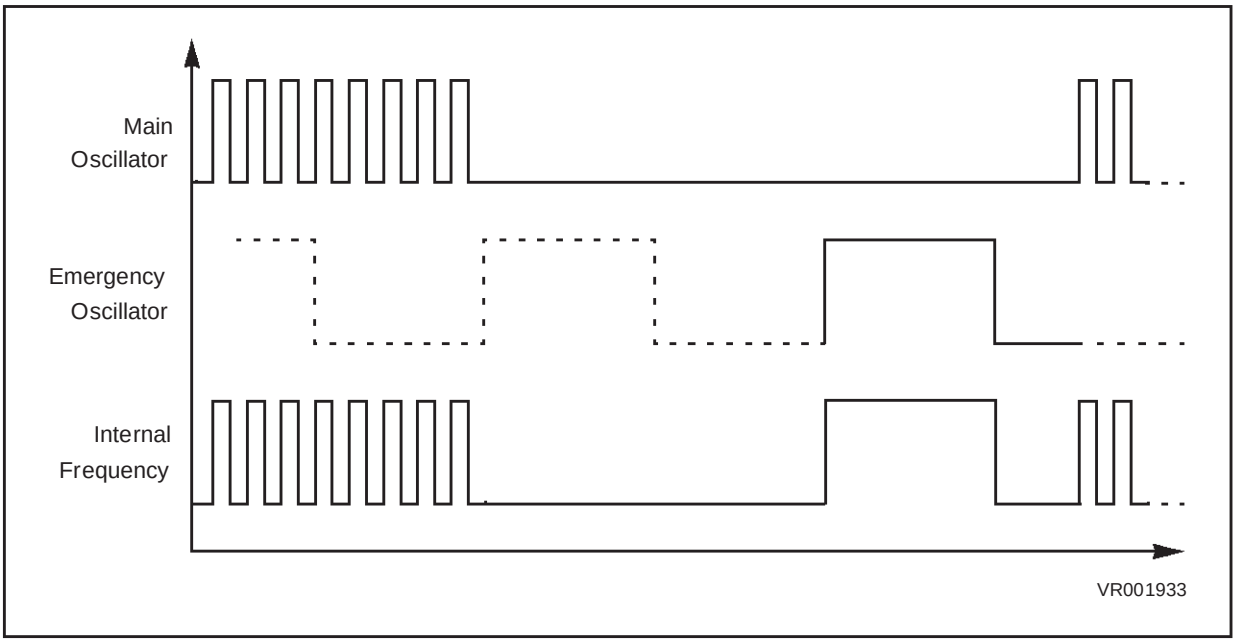


Figure 10. OSG Emergency Oscillator Principle



CLOCK SYSTEM (Cont'd)

Figure 11. Clock Circuit Block Diagram

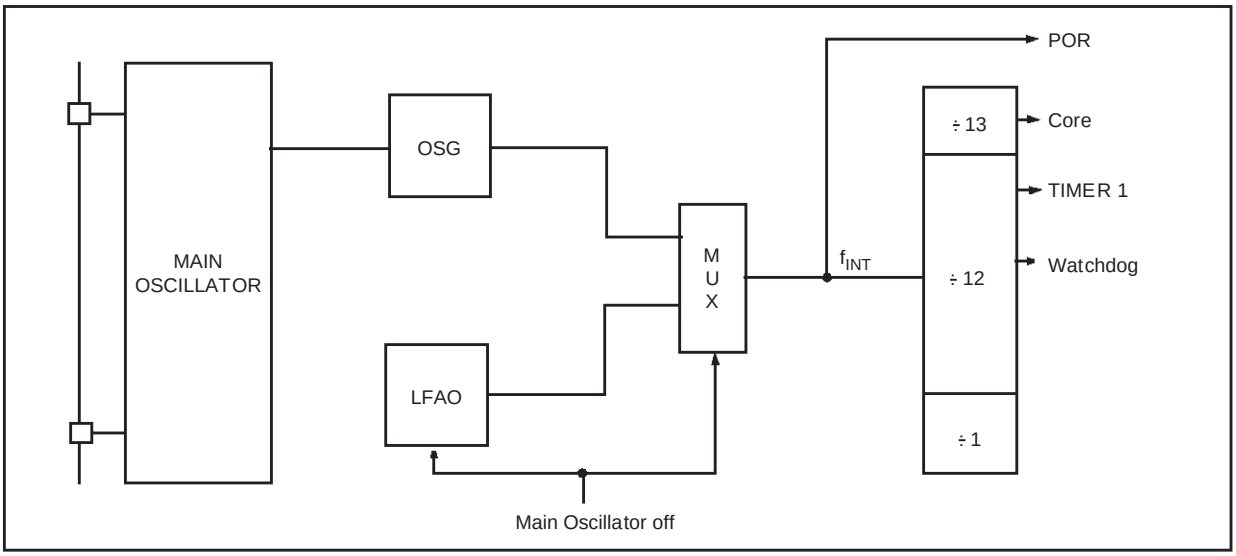
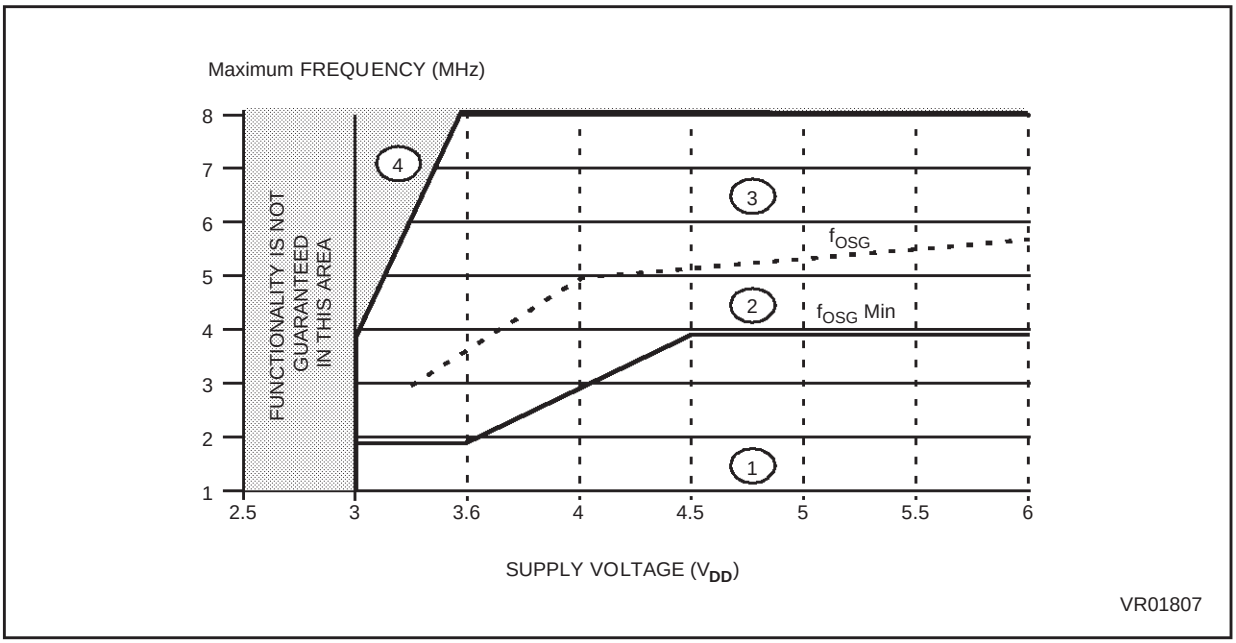


Figure 12. Maximum Operating Frequency (f_{MAX}) versus Supply Voltage (V_{DD})



Notes:

1. In this area, operation is guaranteed at the quartz crystal frequency.

2. When the OSG is disabled, operation in this area is guaranteed at the crystal frequency. When the OSG is enabled, operation in this area is guaranteed at a frequency of at least $f_{OSG\ Min.}$

3. When the OSG is disabled, operation in this area is guaranteed at the quartz crystal frequency. When the OSG is enabled, access to this area is prevented. The internal frequency is kept a $f_{OSG.}$

4. When the OSG is disabled, operation in this area is not guaranteed
When the OSG is enabled, access to this area is prevented. The internal frequency is kept at $f_{OSG.}$

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

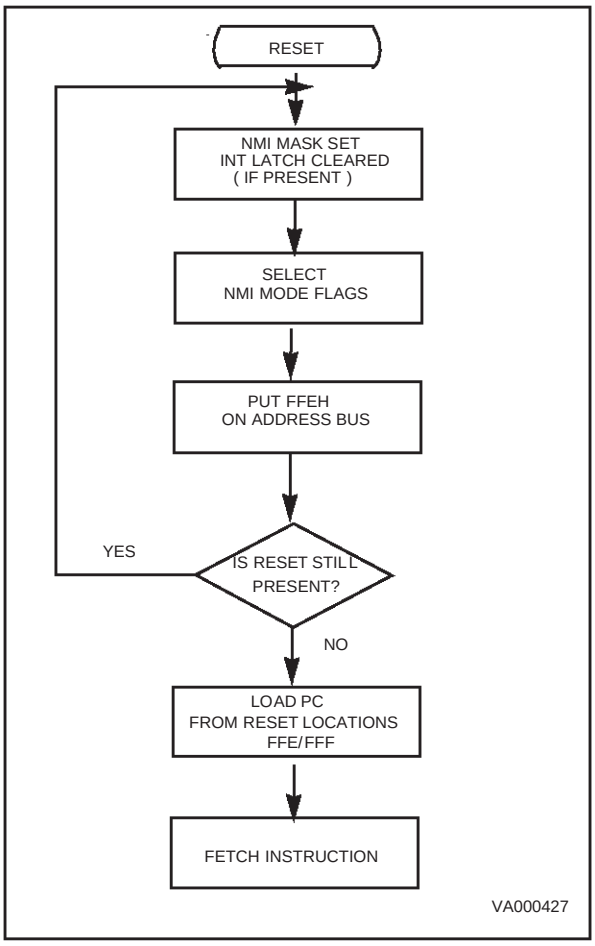
The function of the POR circuit consists in waking up the MCU by detecting around 2V a dynamic (rising edge) variation of the VDD Supply. At the beginning of this sequence, the MCU is configured in the Reset state: all I/O ports are configured as inputs with pull-up resistors and no instruction is executed. 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



RESETS (Cont'd)

3.2.3 Watchdog Reset

The MCU provides a Watchdog timer function in order to ensure graceful recovery from software upsets. If the Watchdog register is not refreshed before an end-of-count condition is reached, the internal reset will be activated. This, amongst other things, resets the watchdog counter.

The MCU restarts just as though the Reset had been generated by the $\overline{\text{RESET}}$ pin, including the built-in stabilisation delay period.

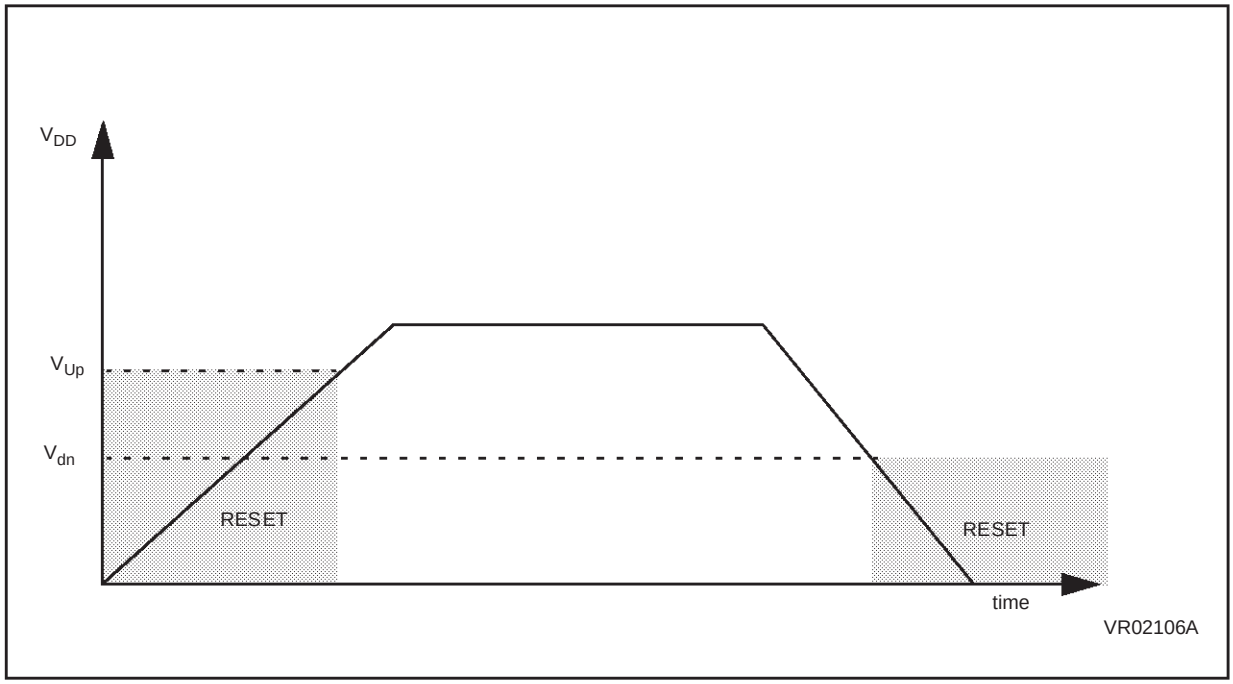
3.2.4 LVD Reset

The on-chip Low Voltage Detector, selectable as user option, features static Reset when supply voltage is below a reference value. Thanks to this feature, external reset circuit can be removed while keeping the application safety. This SAFE RESET is effective as well in Power-on phase as in power supply drop with different reference values, allowing hysteresis effect. Reference value in case of voltage drop has been set lower than the reference value for power-on in order to avoid any parasitic Reset when MCU start's running and sinking current on the supply.

As long as the supply voltage is below the reference value, there is a internal and static RESET command. The MCU can start only when the supply voltage rises over the reference value. Therefore, only two operating mode exist for the MCU: RESET active below the voltage reference, and running mode over the voltage reference as shown on the Figure 14, that represents a power-up, power-down sequence.

Note: When the RESET state is controlled by one of the internal RESET sources (Low Voltage Detector, Watchdog, Power on Reset), the RESET pin is tied to low logic level.

Figure 14. LVD Reset on Power-on and Power-down (Brown-out)



3.2.5 Application Notes

No external resistor is required between V_{DD} and the Reset pin, thanks to the built-in pull-up device.

Direct external connection of the pin $\overline{\text{RESET}}$ to V_{DD} must be avoided in order to ensure safe behaviour of the internal reset sources (AND.Wired structure).

RESETS (Cont'd)

3.2.6 MCU Initialization Sequence

When a reset occurs the stack is reset, the PC is loaded with the address of the Reset Vector (located in program ROM starting at address 0FFEh). A jump to the beginning of the user program must be coded at this address. Following a Reset, the Interrupt flag is automatically set, so that the CPU is in Non Maskable Interrupt mode; this prevents the initialisation routine from being interrupted. The initialisation routine should therefore be terminated by a RETI instruction, in order to revert to normal mode and enable interrupts. If no pending interrupt is present at the end of the initialisation routine, the MCU will continue by processing the instruction immediately following the RETI instruction. If, however, a pending interrupt is present, it will be serviced.

Figure 15. Reset and Interrupt Processing

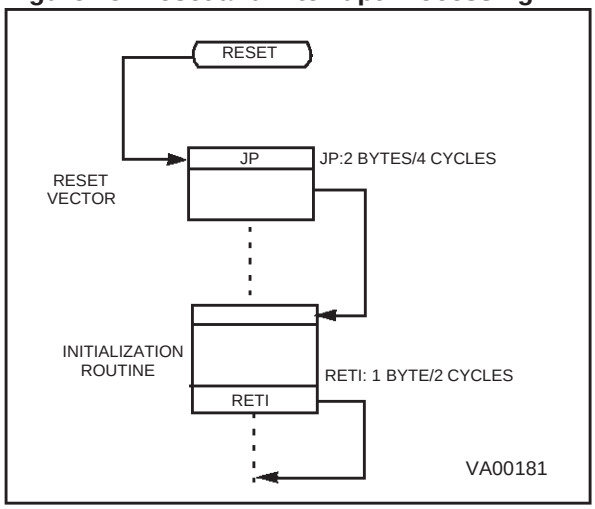
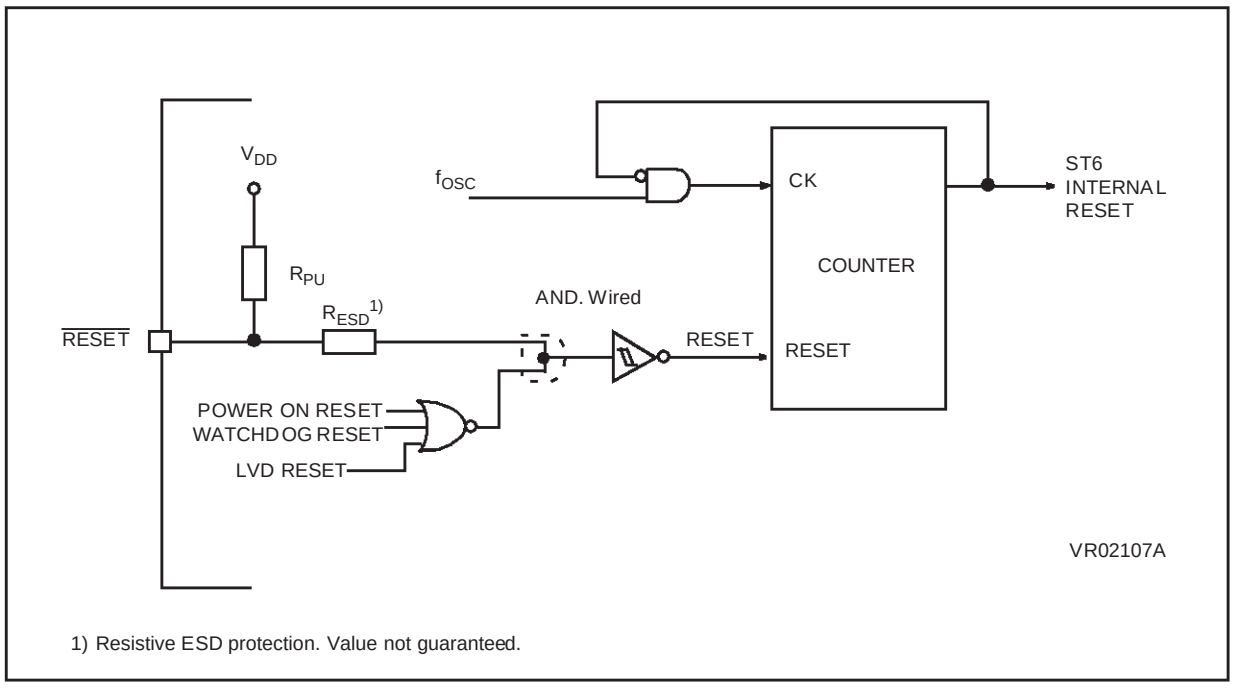


Figure 16. Reset Block Diagram



1) Resistive ESD protection. Value not guaranteed.

RESETS (Cont'd)

Table 5. Register Reset Status

Register	Address(es)	Status	Comment
Oscillator Control Register	0DCh	00h	$f_{INT} = f_{OSC}$; OSG disabled
Port Data Registers	0C0h to 0C1h		I/O are Input with pull-up
Port Direction Register	0C4h to 0C5h		I/O are Input with pull-up
Port Option Register	0CCh to 0CDh		I/O are Input with pull-up
Interrupt Option Register	0C8h		Interrupt disabled
TIMER Status/Control	0D4h		TIMER disabled
X, Y, V, W, Register	080H TO 083H	Undefined	As written if programmed
Accumulator	0FFh		
Data RAM	084h to 0BFh		
Data ROM Window Register	0C9h		
A/D Result Register	0D0h		
TIMER Counter Register	0D3h	FFh	Max count loaded
TIMER Prescaler Register	0D2h	7Fh	
Watchdog Counter Register	0D8h	FEh	
A/D Control Register	0D1h	40h	A/D in Standby (When available)

3.3 DIGITAL WATCHDOG

The digital Watchdog consists of a reloadable downcounter timer which can be used to provide controlled recovery from software upsets.

The Watchdog circuit generates a Reset when the downcounter reaches zero. User software can prevent this reset by reloading the counter, and should therefore be written so that the counter is regularly reloaded while the user program runs correctly. In the event of a software mishap (usually caused by externally generated interference), the user program will no longer behave in its usual fashion and the timer register will thus not be reloaded periodically. Consequently the timer will decrement down to 00h and reset the MCU. In order to maximise the effectiveness of the Watchdog function, user software must be written with this concept in mind.

Watchdog behaviour is governed by two options, known as “WATCHDOG ACTIVATION” (i.e. HARDWARE or SOFTWARE) and “EXTERNAL STOP MODE CONTROL” (see Table 6).

In the SOFTWARE option, the Watchdog is disabled until bit C of the DWDR register has been set. When the Watchdog is disabled, low power Stop mode is available. Once activated, the Watchdog cannot be disabled, except by resetting the MCU.

In the HARDWARE option, the Watchdog is permanently enabled. Since the oscillator will run continuously, low power mode is not available. The STOP instruction is interpreted as a WAIT instruction, and the Watchdog continues to countdown.

However, when the EXTERNAL STOP MODE CONTROL option has been selected low power consumption may be achieved in Stop Mode.

Execution of the STOP instruction is then governed by a secondary function associated with the NMI pin. If a STOP instruction is encountered when the NMI pin is low, it is interpreted as WAIT, as described above. If, however, the STOP instruction is encountered when the NMI pin is high, the Watchdog counter is frozen and the CPU enters STOP mode.

When the MCU exits STOP mode (i.e. when an interrupt is generated), the Watchdog resumes its activity.

Table 6. Recommended Option Choices

Functions Required	Recommended Options
Stop Mode & Watchdog	“EXTERNAL STOP MODE” & “HARDWARE WATCHDOG”
Stop Mode	“SOFTWARE WATCHDOG”
Watchdog	“HARDWARE WATCHDOG”

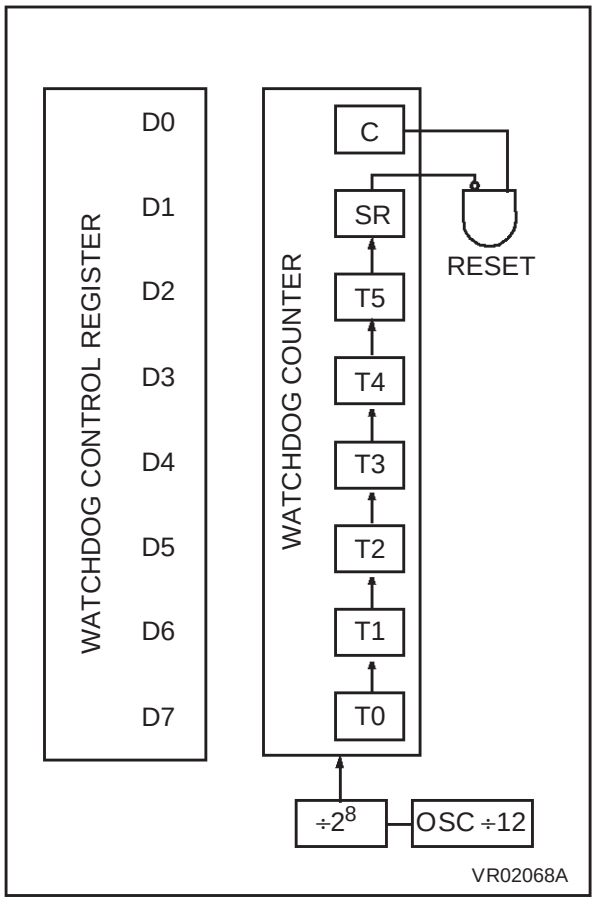
DIGITAL WATCHDOG (Cont'd)

The Watchdog is associated with a Data space register (Digital WatchDog Register, DWDR, location 0D8h) which is described in greater detail in Section 3.3.1 Digital Watchdog Register (DWDR). This register is set to 0FEh on Reset: bit C is cleared to "0", which disables the Watchdog; the timer downcounter bits, T0 to T5, and the SR bit are all set to "1", thus selecting the longest Watchdog timer period. This time period can be set to the user's requirements by setting the appropriate value for bits T0 to T5 in the DWDR register. The SR bit must be set to "1", since it is this bit which generates the Reset signal when it changes to "0"; clearing this bit would generate an immediate Reset.

It should be noted that the order of the bits in the DWDR register is inverted with respect to the associated bits in the down counter: bit 7 of the DWDR register corresponds, in fact, to T0 and bit 2 to T5. The user should bear in mind the fact that these bits are inverted and shifted with respect to the physical counter bits when writing to this register. The relationship between the DWDR register bits and the physical implementation of the Watchdog timer downcounter is illustrated in Figure 17.

Only the 6 most significant bits may be used to define the time period, since it is bit 6 which triggers the Reset when it changes to "0". This offers the user a choice of 64 timed periods ranging from 3,072 to 196,608 clock cycles (with an oscillator frequency of 8MHz, this is equivalent to timer periods ranging from 384µs to 24.576ms).

Figure 17. Watchdog Counter Control



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

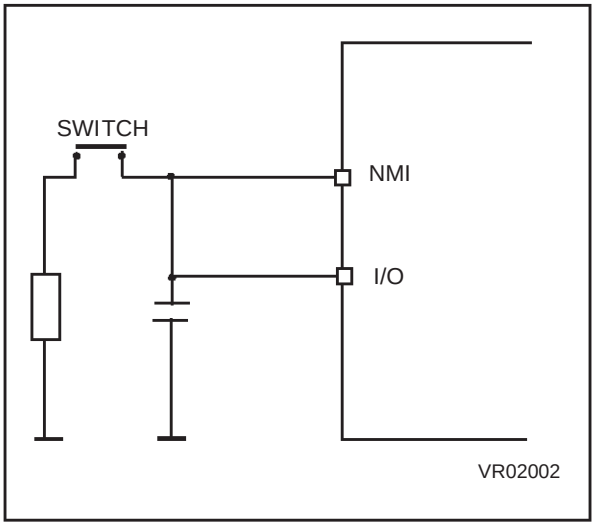
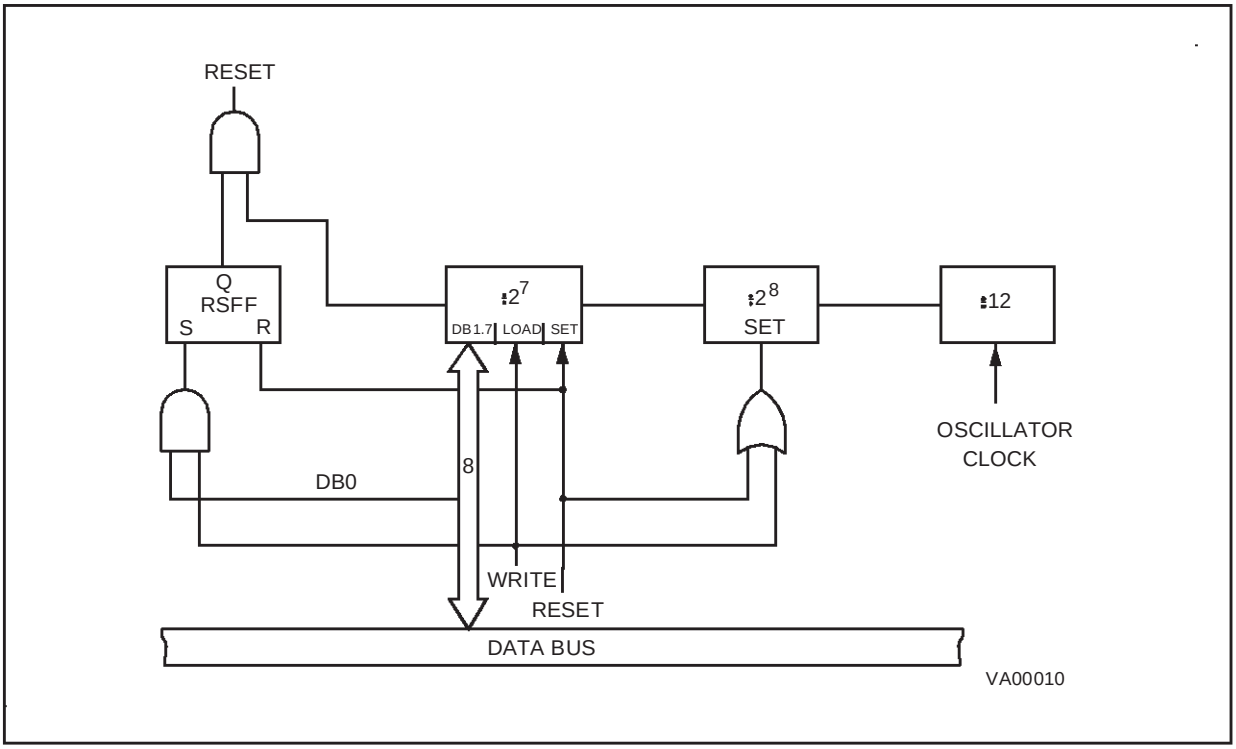


Figure 19. Digital Watchdog Block Diagram



3.4 INTERRUPTS

The CPU can manage four Maskable Interrupt sources, in addition to a Non Maskable Interrupt source (top priority interrupt). Each source is associated with a specific Interrupt Vector which contains a Jump instruction to the associated interrupt service routine. These vectors are located in Program space (see Table 7).

When an interrupt source generates an interrupt request, and interrupt processing is enabled, the PC register is loaded with the address of the interrupt vector (i.e. of the Jump instruction), which then causes a Jump to the relevant interrupt service routine, thus servicing the interrupt.

Interrupt sources are linked to events either on external pins, or on chip peripherals. Several events can be ORed on the same interrupt source, and relevant flags are available to determine which event triggered the interrupt.

The Non Maskable Interrupt request has the highest priority and can interrupt any interrupt routine at any time; the other four interrupts cannot interrupt each other. If more than one interrupt request is pending, these are processed by the processor core according to their priority level: source #1 has the higher priority while source #4 the lower. The priority of each interrupt source is fixed.

Table 7. Interrupt Vector Map

Interrupt Source	Priority	Vector Address
Interrupt source #0	1	(FFCh-FFDh)
Interrupt source #1	2	(FF6h-FF7h)
Interrupt source #2	3	(FF4h-FF5h)
Interrupt source #3	4	(FF2h-FF3h)
Interrupt source #4	5	(FF0h-FF1h)

3.4.1 Interrupt request

All interrupt sources but the Non Maskable Interrupt source can be disabled by setting accordingly the GEN bit of the Interrupt Option Register (IOR). This GEN bit also defines if an interrupt source, including the Non Maskable Interrupt source, can restart the MCU from STOP/WAIT modes.

Interrupt request from the Non Maskable Interrupt source #0 is latched by a flip flop which is automatically reset by the core at the beginning of the non-maskable interrupt service routine.

Interrupt request from source #1 can be configured either as edge or level sensitive by setting accordingly the LES bit of the Interrupt Option Register (IOR).

Interrupt request from source #2 are always edge sensitive. The edge polarity can be configured by setting accordingly the ESB bit of the Interrupt Option Register (IOR).

Interrupt request from sources #3 & #4 are level sensitive.

In edge sensitive mode, a latch is set when a edge occurs on the interrupt source line and is cleared when the associated interrupt routine is started. So, the occurrence of an interrupt can be stored, until completion of the running interrupt routine before being processed. If several interrupt requests occurs before completion of the running interrupt routine, only the first request is stored.

Storage of interrupt requests is not available in level sensitive mode. To be taken into account, the low level must be present on the interrupt pin when the MCU samples the line after instruction execution.

At the end of every instruction, the MCU tests the interrupt lines: if there is an interrupt request the next instruction is not executed and the appropriate interrupt service routine is executed instead.

Table 8. Interrupt Option Register Description

GEN	SET	Enable all interrupts
	CLEARED	Disable all interrupts
ESB	SET	Rising edge mode on interrupt source #2
	CLEARED	Falling edge mode on interrupt source #2
LES	SET	Level-sensitive mode on interrupt source #1
	CLEARED	Falling edge mode on interrupt source #1
OTHERS	NOT USED	

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

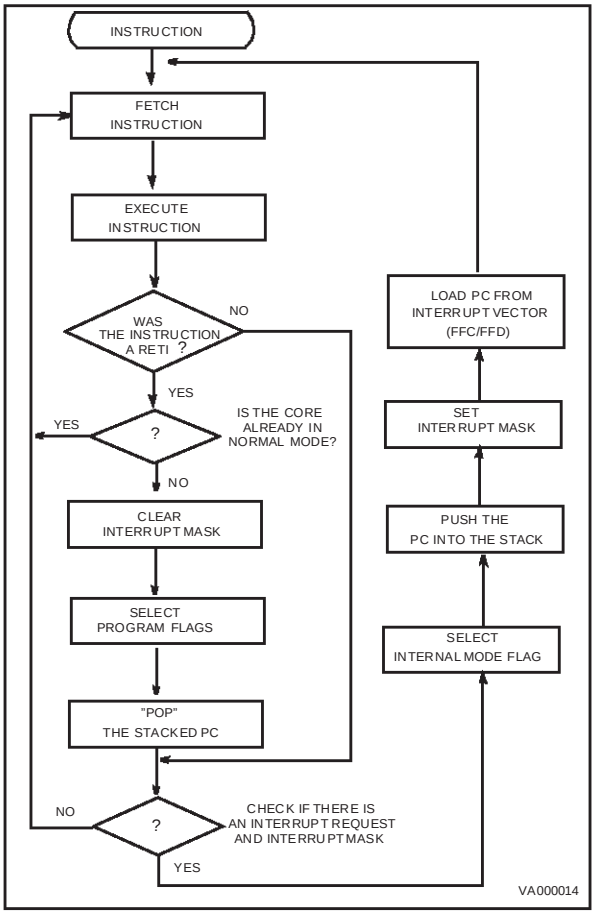
– User selected registers are saved within the interrupt service routine (normally on a software stack).
– The source of the interrupt is found by polling the interrupt flags (if more than one source is associated with the same vector).
– The interrupt is serviced.
– Return from interrupt (RETI)

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 ST62T08C/T09C/T10C/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.

POWER SAVING MODE (Cont'd)

3.5.3 Exit from WAIT and STOP Modes

The following paragraphs describe how the MCU exits from WAIT and STOP modes, when an interrupt occurs (not a Reset). It should be noted that the restart sequence depends on the original state of the MCU (normal, interrupt or non-maskable interrupt mode) prior to entering WAIT or STOP mode, as well as on the interrupt type.

Interrupts do not affect the oscillator selection, consequently, when the LFAO is used, the user program must manage oscillator selection as soon as normal RUN mode is resumed.

3.5.3.1 Normal Mode

If the MCU was in the main routine when the WAIT or STOP instruction was executed, exit from Stop or Wait mode will occur as soon as an interrupt occurs; the related interrupt routine is executed and, on completion, the instruction which follows the STOP or WAIT instruction is then executed, providing no other interrupts are pending.

3.5.3.2 Non Maskable Interrupt Mode

If the STOP or WAIT instruction has been executed during execution of the non-maskable interrupt routine, the MCU exits from the Stop or Wait mode as soon as an interrupt occurs: the instruction which follows the STOP or WAIT instruction is executed, and the MCU remains in non-maskable interrupt mode, even if another interrupt has been generated.

3.5.3.3 Normal Interrupt Mode

If the MCU was in interrupt mode before the STOP or WAIT instruction was executed, it exits from STOP or WAIT mode as soon as an interrupt occurs. Nevertheless, two cases must be considered:

– If the interrupt is a normal one, the interrupt routine in which the WAIT or STOP mode was entered will be completed, starting with the execution of the instruction which follows the STOP or the WAIT instruction, and the MCU is still in the interrupt mode. At the end of this routine pending interrupts will be serviced in accordance with their priority.

– In the event of a non-maskable interrupt, the non-maskable interrupt service routine is processed first, then the routine in which the WAIT or STOP mode was entered will be completed by executing the instruction following the STOP or WAIT instruction. The MCU remains in normal interrupt mode.

Notes:

To achieve the lowest power consumption during RUN or WAIT modes, the user program must take care of:

– configuring unused I/Os as inputs without pull-up (these should be externally tied to well defined logic levels);

– placing all peripherals in their power down modes before entering STOP mode;

– selecting the Low Frequency Auxiliary Oscillator (provided this runs at a lower frequency than the main oscillator).

When the hardware activated Watchdog is selected, or when the software Watchdog is enabled, the STOP instruction is disabled and a WAIT instruction will be executed in its place.

If all interrupt sources are disabled (GEN low), the MCU can only be restarted by a Reset. Although setting GEN low does not mask the NMI as an interrupt, it will stop it generating a wake-up signal.

The WAIT and STOP instructions are not executed if an enabled interrupt request is pending.

4 ON-CHIP PERIPHERALS

4.1 I/O PORTS

The MCU features Input/Output lines which may be individually programmed as any of the following input or output configurations:

– Input without pull-up or interrupt
– Input with pull-up and interrupt
– Input with pull-up, but without interrupt
– Analog input
– Push-pull output
– Open drain output

The lines are organised as bytewise Ports.

Each port is associated with 3 registers in Data space. Each bit of these registers is associated with a particular line (for instance, bits 0 of Port A Data, Direction and Option registers are associated with the PA0 line of Port A).

The DATA registers (DRx), are used to read the voltage level values of the lines which have been configured as inputs, or to write the logic value of the signal to be output on the lines configured as outputs. The port data registers can be read to get the effective logic levels of the pins, but they can be also written by user software, in conjunction with the related option registers, to select the different input mode options.

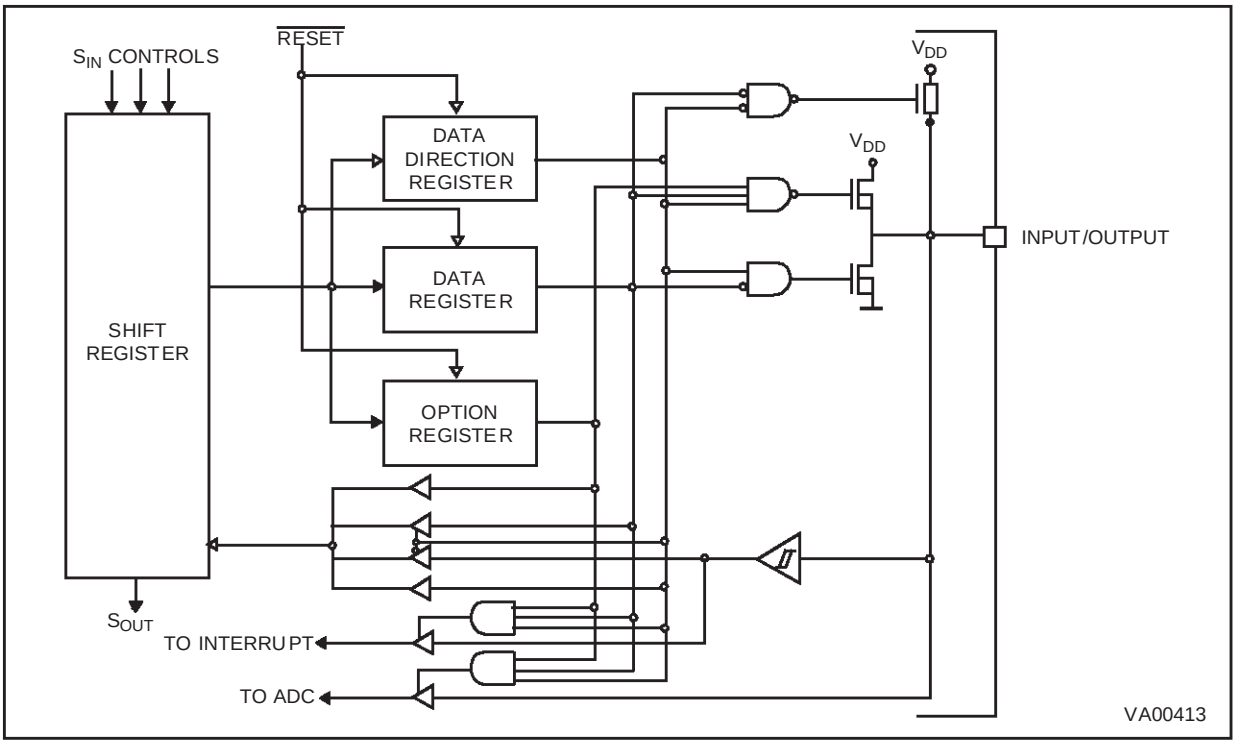
Single-bit operations on I/O registers are possible but care is necessary because reading in input mode is done from I/O pins while writing will directly affect the Port data register causing an undesired change of the input configuration.

The Data Direction registers (DDRx) allow the data direction (input or output) of each pin to be set.

The Option registers (ORx) are used to select the different port options available both in input and in output mode.

All I/O registers can be read or written to just as any other RAM location in Data space, so no extra RAM cells are needed for port data storage and manipulation. During MCU initialization, all I/O registers are cleared and the input mode with pull-ups and no interrupt generation is selected for all the pins, thus avoiding pin conflicts.

Figure 22. I/O Port Block Diagram



I/O PORTS (Cont'd)

4.1.1 Operating Modes

Each pin may be individually programmed as input or output with various configurations.

This is achieved by writing the relevant bit in the Data (DR), Data Direction (DDR) and Option registers (OR). Table 10 illustrates the various port configurations which can be selected by user software.

4.1.1.1 Input Options

Pull-up, High Impedance Option. All input lines can be individually programmed with or without an internal pull-up by programming the OR and DR registers accordingly. If the pull-up option is not selected, the input pin will be in the high-impedance state.

4.1.1.2 Interrupt Options

All input lines can be individually connected by software to the interrupt system by programming the OR and DR registers accordingly. The interrupt trigger modes (falling edge, rising edge and low level) can be configured by software as described in the Interrupt Chapter for each port.

4.1.1.3 Analog Input Options

Some pins can be configured as analog inputs by programming the OR and DR registers accordingly. These analog inputs are connected to the on-chip 8-bit Analog to Digital Converter. *ONLY ONE* pin should be programmed as an analog input at any time, since by selecting more than one input simultaneously their pins will be effectively shorted.

Table 10. I/O Port Option Selection

DDR	OR	DR	Mode	Option
0	0	0	Input	With pull-up, no interrupt
0	0	1	Input	No pull-up, no interrupt
0	1	0	Input	With pull-up and with interrupt
0	1	1	Input	Analog input (when available)
1	0	X	Output	Open-drain output (20mA sink when available)
1	1	X	Output	Push-pull output (20mA sink when available)

Note: X = Don't care

I/O PORTS (Cont'd)

4.1.2 Safe I/O State Switching Sequence

Switching the I/O ports from one state to another should be done in a sequence which ensures that no unwanted side effects can occur. The recommended safe transitions are illustrated in Figure 23. All other transitions are potentially risky and should be avoided when changing the I/O operating mode, as it is most likely that undesirable side-effects will be experienced, such as spurious interrupt generation or two pins shorted together by the analog multiplexer.

Single bit instructions (SET, RES, INC and DEC) should be used with great caution on Ports Data registers, since these instructions make an implicit read and write back of the entire register. In port input mode, however, the data register reads from the input pins directly, and not from the data register latches. Since data register information in input mode is used to set the characteristics of the input pin (interrupt, pull-up, analog input), these may be unintentionally reprogrammed depending on the state of the input pins. As a general rule, it is better to limit the use of single bit instructions on data registers to when the whole (8-bit) port is in output mode. In the case of inputs or of mixed inputs and outputs, it is advisable to keep a copy of the data register in RAM. Single bit instructions may then be used on the RAM copy, after which the whole copy register can be written to the port data register:

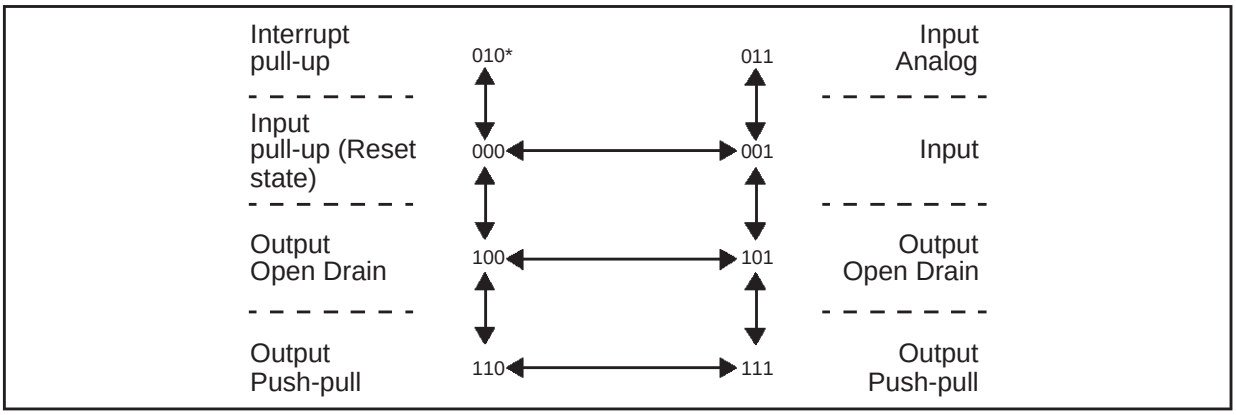
```
SET bit, datacopy
LD a, datacopy
LD DRA, a
```

Warning: Care must also be taken to not use instructions that act on a whole port register (INC, DEC, or read operations) when all 8 bits are not available on the device. Unavailable bits must be masked by software (AND instruction).

The WAIT and STOP instructions allow the ST62xx to be used in situations where low power consumption is needed. The lowest power consumption is achieved by configuring I/Os in input mode with well-defined logic levels.

The user must take care not to switch outputs with heavy loads during the conversion of one of the analog inputs in order to avoid any disturbance to the conversion.

Figure 23. Diagram showing Safe I/O State Transitions



Note *. xxx = DDR, OR, DR Bits respectively

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

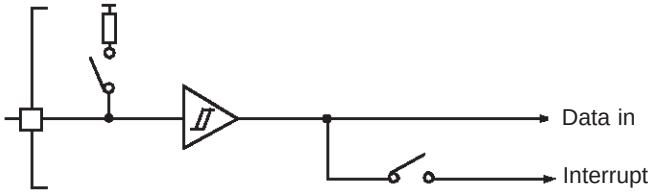
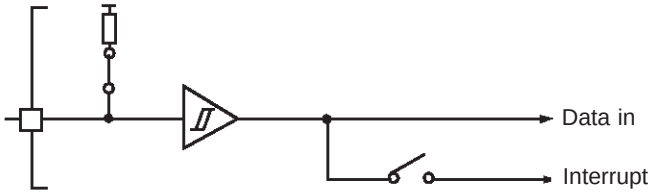
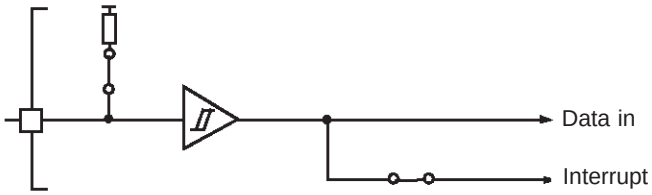
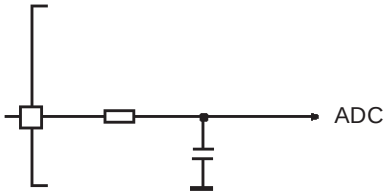
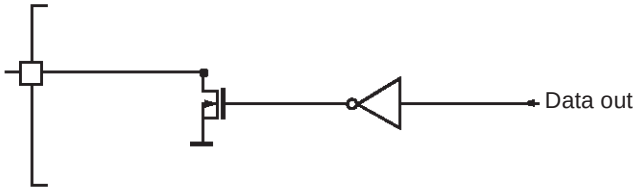
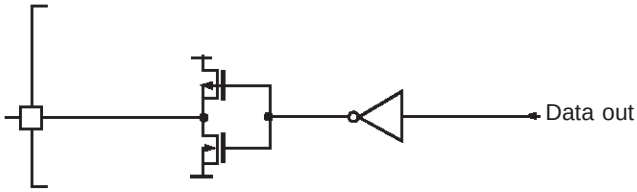
7							0
Px7	Px6	Px5	Px4	Px3	Px2	Px1	Px0

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

Note: X = Don't care

I/O PORTS (Cont'd)

Table 11. I/O Port Option Selections

MODE	AVAILABLE ON(1)	SCHEMATIC
Input	PA0-PA3 PB0-PB7	
Input with pull up	PA0-PA3 PB0-PB7	
Input with pull up with interrupt	PA0-PA3 PB0-PB7	
Analog Input	PB0-PB3 (ST62T10C,T20C,E20C) PB4-PB7 (All except ST62T08C)	
Open drain output 5mA Open drain output 20mA	PB0-PB7 PA0-PA3	
Push-pull output 5mA Push-pull output 20mA	PB0-PB7 PA0-PA3	

Note 1. Provided the correct configuration has been selected.

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.

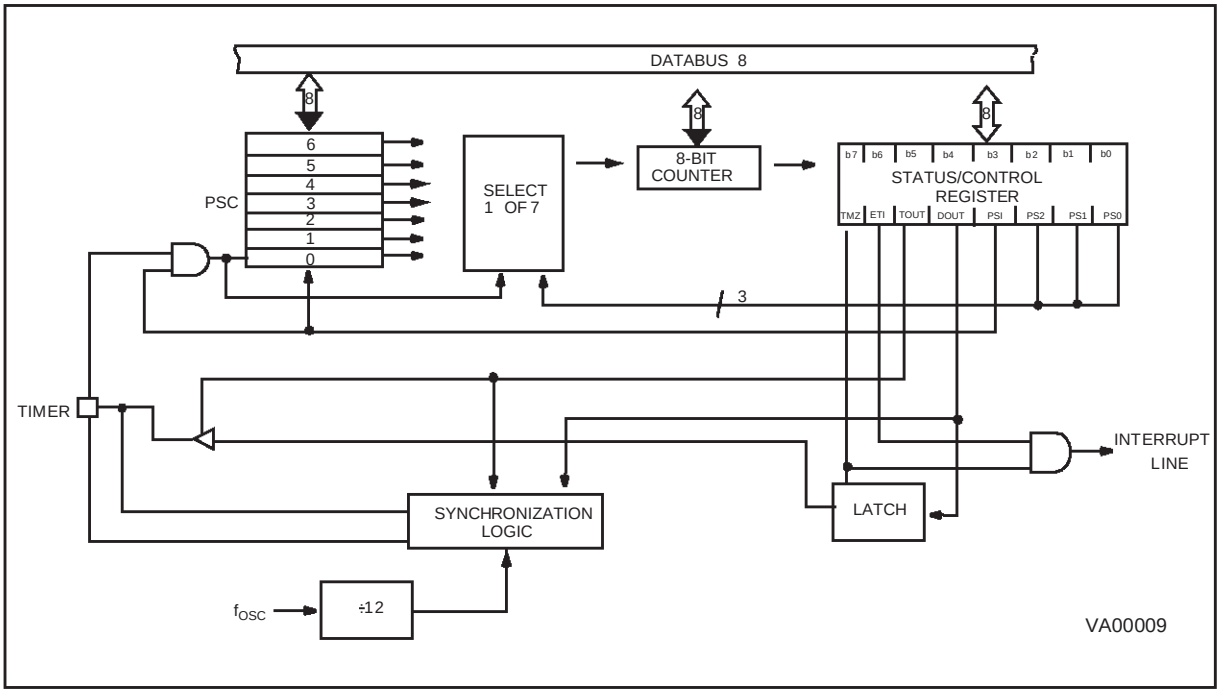
Figure 24 shows the Timer Block Diagram. The external TIMER pin is available to the user. The content of the 8-bit counter can be read/written in the Timer/Counter register, TCR, while the state of the 7-bit prescaler can be read in the PSC register. The control logic device is managed in the TSCR register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



TIMER (Cont'd)

4.2.1 Timer Operating Modes

There are three operating modes, which are selected by the TOUT and DOUT bits (see TSCR register). These three modes correspond to the two clocks which can be connected to the 7-bit prescaler ($f_{INT} \div 12$ or TIMER pin signal), and to the output mode.

4.2.1.1 Gated Mode

(TOUT = "0", DOUT = "1")

In this mode the prescaler is decremented by the Timer clock input ($f_{INT} \div 12$), but ONLY when the signal on the TIMER pin is held high (allowing pulse width measurement). This mode is selected by clearing the TOUT bit in the TSCR register to "0" (i.e. as input) and setting the DOUT bit to "1".

4.2.1.2 Event Counter Mode

(TOUT = "0", DOUT = "0")

In this mode, the TIMER pin is the input clock of the prescaler which is decremented on the rising edge.

4.2.1.3 Output Mode

(TOUT = "1", DOUT = data out)

The TIMER pin is connected to the DOUT latch, hence the Timer prescaler is clocked by the prescaler clock input ($f_{INT} \div 12$).

The user can select the desired prescaler division ratio through the PS2, PS1, PS0 bits. When the TCR count reaches 0, it sets the TMZ bit in the TSCR. The TMZ bit can be tested under program control to perform a timer function whenever it goes high. The low-to-high TMZ bit transition is used to latch the DOUT bit of the TSCR and transfer it to the TIMER pin. This operating mode allows external signal generation on the TIMER pin.

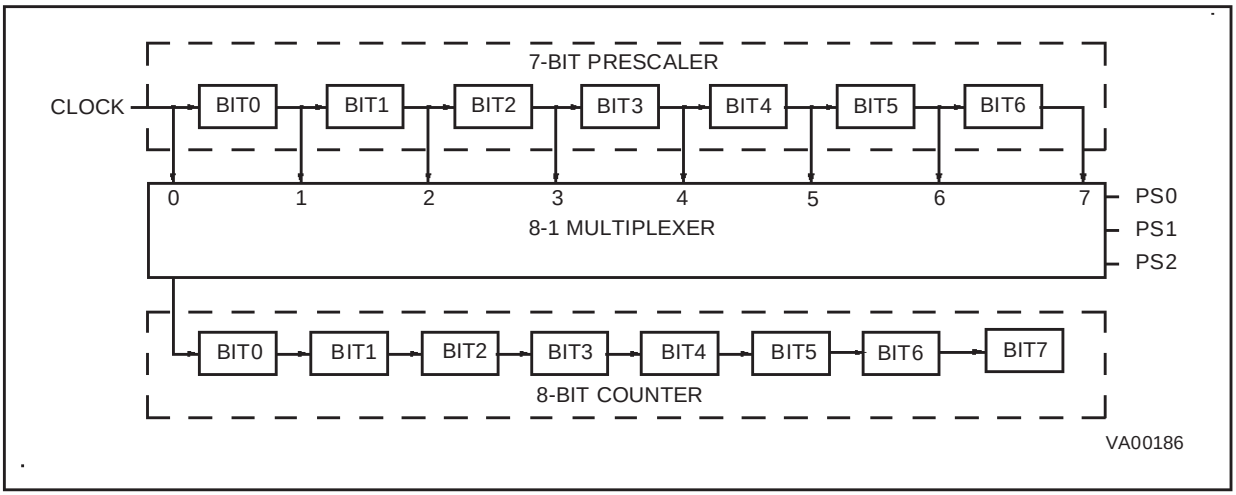
Table 12. Timer Operating Modes

TOUT	DOUT	Timer Pin	Timer Function
0	0	Input	Event Counter
0	1	Input	Gated Input
1	0	Output	Output "0"
1	1	Output	Output "1"

4.2.2 Timer Interrupt

When the counter register decrements to zero with the ETI (Enable Timer Interrupt) bit set to one, an interrupt request is generated as described in the Interrupt Chapter. When the counter decrements to zero, the TMZ bit in the TSCR register is set to one.

Figure 25. Timer Working Principle



TIMER (Cont'd)

4.2.3 Application Notes

The user can select the presence of an on-chip pull-up on the TIMER pin as option.

TMZ is set when the counter reaches zero; however, it may also be set by writing 00h in the TCR register or by setting bit 7 of the TSCR register. The TMZ bit must be cleared by user software when servicing the timer interrupt to avoid undesired interrupts when leaving the interrupt service routine. After reset, the 8-bit counter register is loaded with 0FFh, while the 7-bit prescaler is loaded with 07Fh, and the TSCR register is cleared. This means that the Timer is stopped (PSI="0") and the timer interrupt is disabled.

If the Timer is programmed in output mode, the DOUT bit is transferred to the TIMER pin when TMZ is set to one (by software or due to counter decrement). When TMZ is high, the latch is transparent and DOUT is copied to the timer pin. When TMZ goes low, DOUT is latched.

A write to the TCR register will predominate over the 8-bit counter decrement to 00h function, i.e. if a write and a TCR register decrement to 00h occur simultaneously, the write will take precedence, and the TMZ bit is not set until the 8-bit counter reaches 00h again. The values of the TCR and the PSC registers can be read accurately at any time.

4.2.4 Timer Registers

Timer Status Control Register (TSCR)

Address: 0D4h — Read/Write

7							0
TMZ	ETI	TOUT	DOUT	PSI	PS2	PS1	PS0

Bit 7 = **TMZ**: *Timer Zero bit*

A low-to-high transition indicates that the timer count register has decrement to zero. This bit must be cleared by user software before starting a new count.

Bit 6 = **ETI**: *Enable Timer Interrupt*

When set, enables the timer interrupt request (vector #3). If ETI=0 the timer interrupt is disabled. If ETI=1 and TMZ=1 an interrupt request is generated.

Bit 5 = **TOUT**: *Timers Output Control*

When low, this bit selects the input mode for the TIMER pin. When high the output mode is selected.

Bit 4 = **DOUT**: *Data Output*

Data sent to the timer output when TMZ is set high (output mode only). Input mode selection (input mode only).

Bit 3 = **PSI**: *Prescaler Initialize Bit*

Used to initialize the prescaler and inhibit its counting. When PSI="0" the prescaler is set to 7Fh and the counter is inhibited. When PSI="1" the prescaler is enabled to count downwards. As long as PSI="0" both counter and prescaler are not running.

Bit 2, 1, 0 = **PS2, PS1, PS0**: *Prescaler Mux. Select.* These bits select the division ratio of the prescaler register.

Table 13. Prescaler Division Factors

PS2	PS1	PS0	Divided by
0	0	0	1
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Timer Counter Register TCR

Address: 0D3h — Read/Write

7							0
D7	D6	D5	D4	D3	D2	D1	D0

Bit 7-0 = **D7-D0**: *Counter Bits.*

Prescaler Register PSC

Address: 0D2h — Read/Write

7							0
D7	D6	D5	D4	D3	D2	D1	D0

Bit 7 = **D7**: Always read as "0".

Bit 6-0 = **D6-D0**: Prescaler Bits.

4.3 A/D CONVERTER (ADC)

The A/D converter peripheral is an 8-bit analog to digital converter with analog inputs as alternate I/O functions (the number of which is device dependent), offering 8-bit resolution with a typical conversion time of 70us (at an oscillator clock frequency of 8MHz).

The ADC converts the input voltage by a process of successive approximations, using a clock frequency derived from the oscillator with a division factor of twelve. With an oscillator clock frequency less than 1.2MHz, conversion accuracy is decreased.

Selection of the input pin is done by configuring the related I/O line as an analog input via the Option and Data registers (refer to I/O ports description for additional information). Only one I/O line must be configured as an analog input at any time. The user must avoid any situation in which more than one I/O pin is selected as an analog input simultaneously, to avoid device malfunction.

The ADC uses two registers in the data space: the ADC data conversion register, ADR, which stores the conversion result, and the ADC control register, ADCR, used to program the ADC functions.

A conversion is started by writing a "1" to the Start bit (STA) in the ADC control register. This automatically clears (resets to "0") the End Of Conversion Bit (EOC). When a conversion is complete, the EOC bit is automatically set to "1", in order to flag that conversion is complete and that the data in the ADC data conversion register is valid. Each conversion has to be separately initiated by writing to the STA bit.

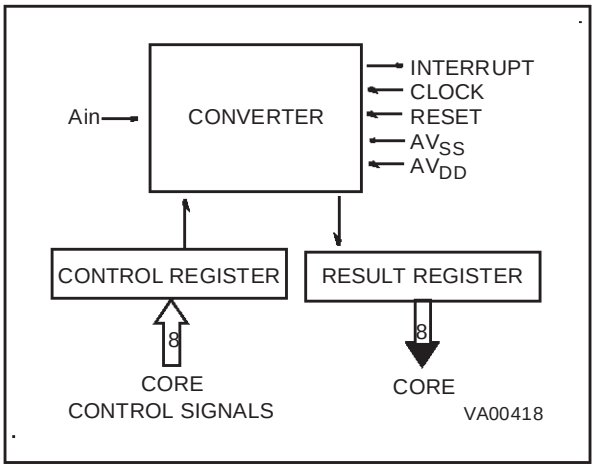
The STA bit is continuously scanned so that, if the user sets it to "1" while a previous conversion is in progress, a new conversion is started before completing the previous one. The start bit (STA) is a write only bit, any attempt to read it will show a logical "0".

The A/D converter features a maskable interrupt associated with the end of conversion. This interrupt is associated with interrupt vector #4 and occurs when the EOC bit is set (i.e. when a conversion is completed). The interrupt is masked using the EAI (interrupt mask) bit in the control register.

The power consumption of the device can be reduced by turning off the ADC peripheral. This is done by setting the PDS bit in the ADC control register to "0". If PDS="1", the A/D is powered and enabled for conversion. This bit must be set at least one instruction before the beginning of the conversion to allow stabilisation of the A/D converter. This action is also needed before entering WAIT mode, since the A/D comparator is not automatically disabled in WAIT mode.

During Reset, any conversion in progress is stopped, the control register is reset to 40h and the ADC interrupt is masked (EAI=0).

Figure 26. ADC Block Diagram



4.3.1 Application Notes

The A/D converter does not feature a sample and hold circuit. The analog voltage to be measured should therefore be stable during the entire conversion cycle. Voltage variation should not exceed ±1/2 LSB for the optimum conversion accuracy. A low pass filter may be used at the analog input pins to reduce input voltage variation during conversion.

When selected as an analog channel, the input pin is internally connected to a capacitor C_{ad} of typically 12pF. For maximum accuracy, this capacitor must be fully charged at the beginning of conversion. In the worst case, conversion starts one instruction (6.5 µs) after the channel has been selected. In worst case conditions, the impedance, ASI, of the analog voltage source is calculated using the following formula:

$$6.5\mu s = 9 \times C_{ad} \times ASI$$

(capacitor charged to over 99.9%), i.e. 30 kΩ including a 50% guardband. ASI can be higher if C_{ad} has been charged for a longer period by adding instructions before the start of conversion (adding more than 26 CPU cycles is pointless).