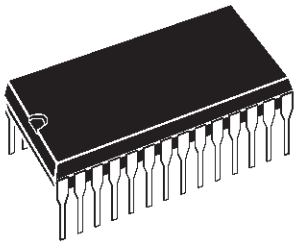
SGS-THOMSON MICROELECTRONICS

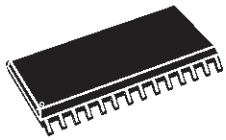
# ST62T55B
# ST62T65B/E65B

## 8-BIT OTP/EPROM MCUs WITH A/D CONVERTER, AUTO-RELOAD TIMER, EEPROM AND SPI

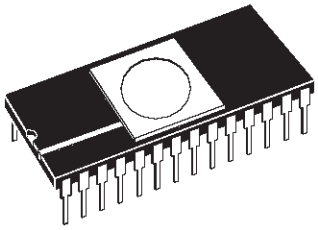
- 3.0 to 6.0V Supply Operating Range
- 8 MHz Maximum Clock Frequency
- -40 to +125°C Operating Temperature Range
- Run, Wait and Stop Modes
- 5 Interrupt Vectors
- Look-up Table capability in Program Memory
- Data Storage in Program Memory: User selectable size
- Data RAM: 128 bytes
- Data EEPROM: 128 bytes (none on ST62T55B)
- User Programmable Options
- 21 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
- 8 I/O lines can sink up to 20mA to drive LEDs or TRIACs directly
- 8-bit Timer/Counter with 7-bit programmable prescaler
- 8-bit Auto-reload Timer with 7-bit programmable prescaler (AR Timer)
- Digital Watchdog
- 8-bit A/D Converter with 13 analog inputs
- 8-bit Synchronous Peripheral Interface (SPI)
- On-chip Clock oscillator can be driven by Quartz Crystal Ceramic resonator or RC network
- User configurable Power-on Reset
- One external Non-Maskable Interrupt
- ST626x-EMU2 Emulation and Development System (connects to an MS-DOS PC via a parallel port)



**PDIP28**



**PSO28**



**CDIP28W**

(See end of Datasheet for Ordering Information)

**DEVICE SUMMARY**

| DEVICE | OTP (Bytes) | EPROM (Bytes) | EEPROM |
|---|---|---|---|
| ST62T55B | 3884 | - | - |
| ST62T65B | 3884 | - | 128 |
| ST62E65B | | 3884 | 128 |

Rev. 2.4

# 1 GENERAL DESCRIPTION

## 1.1 INTRODUCTION

The ST62T55B, ST62T65B and ST62E65B 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.

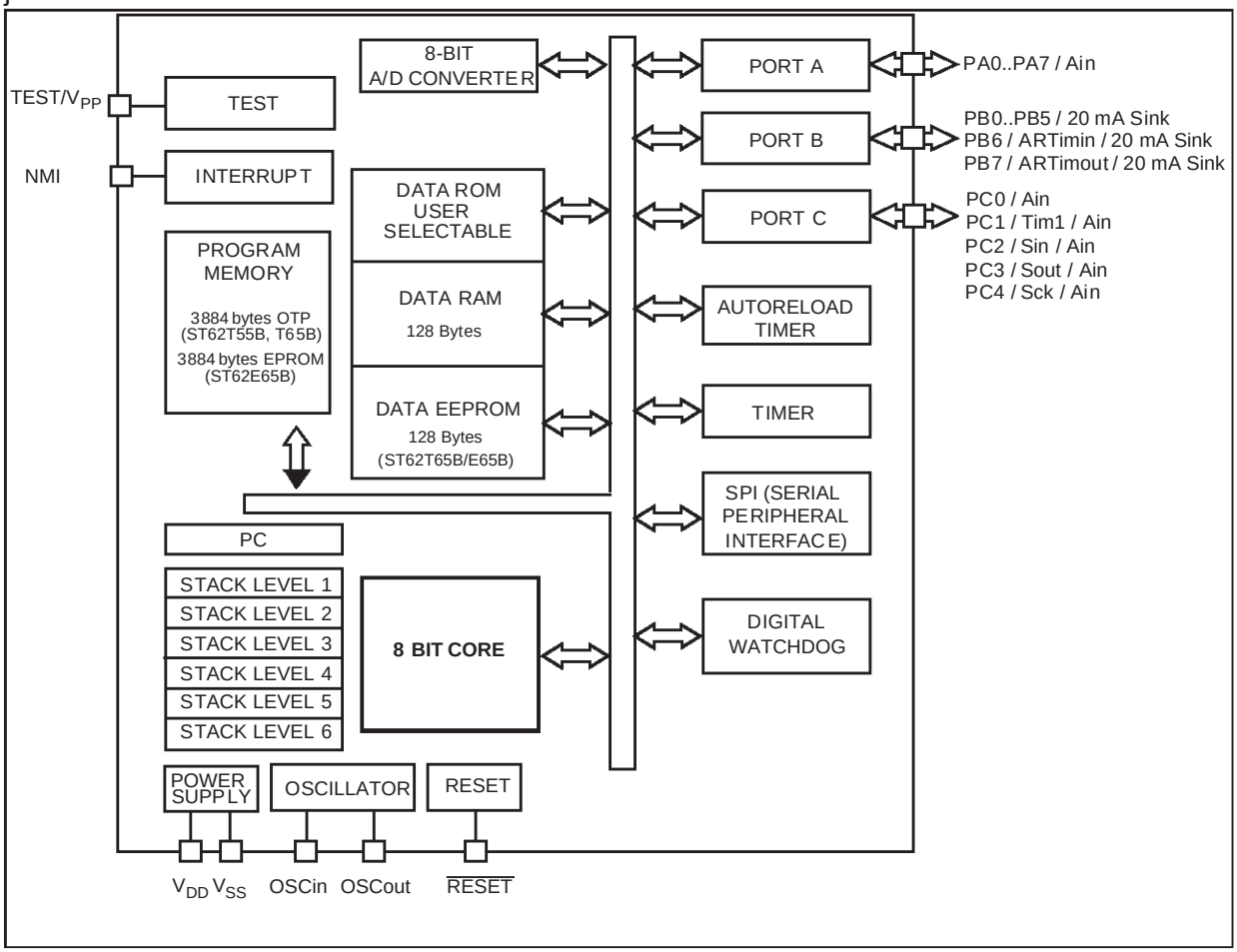
The ST62E65B is the erasable EPROM version of the ST62T65B device, which may be used to emulate the ST62T55B and ST62T65B device, as well as the respective ST6255B and ST6265B ROM devices.

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 byte of theOTP/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 Auto-Reload Timer, EEPROM data capability (except ST62T55B), a serial port communication interface, an 8-bit A/D Converter with 13 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**



### 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.

**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/OTP 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. It is provided with an on-chip pullup resistor and Schmitt trigger characteristics.

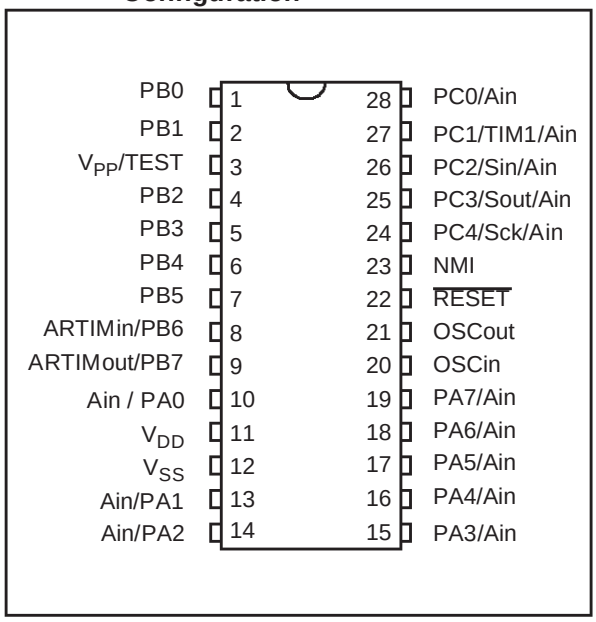
**PA0-PA7.** These 8 lines are organized as one I/O port (A). Each line may be configured under software control as inputs with or without internal pull-up resistors, interrupt generating inputs with pull-up resistors, open-drain or push-pull outputs, analog inputs for the A/D converter.

**PB0-PB5.** These 6 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-PB5 can also sink 20mA for direct LED driving.

**PB6/ARTIMin, PB7/ARTIMout** These pins are either Port B I/O bits or the Input and Output pins of the AR TIMER. To be used as timer input function PB6 has to be programmed as input with or without pull-up. A dedicated bit in the AR TIMER Mode Control Register sets PB7 as timer output function. PB6-PB7 can also sink 20mA for direct LED driving.

**PC0-PC4**. These 5 lines are organized as one I/O port (C). Each line may be configured under software control as input with or without internal pull-up resistor, interrupt generating input with pull-up resistor, analog input for the A/D converter, open-drain or push-pull output.
PC1 can also be used as Timer I/O bit while PC2-PC4 can also be used as respectively Data in, Data out and Clock I/O pins for the on-chip SPI to carry the synchronous serial I/O signals.

**Figure 2. ST62T55B, T65B, E65B Pin Configuration**



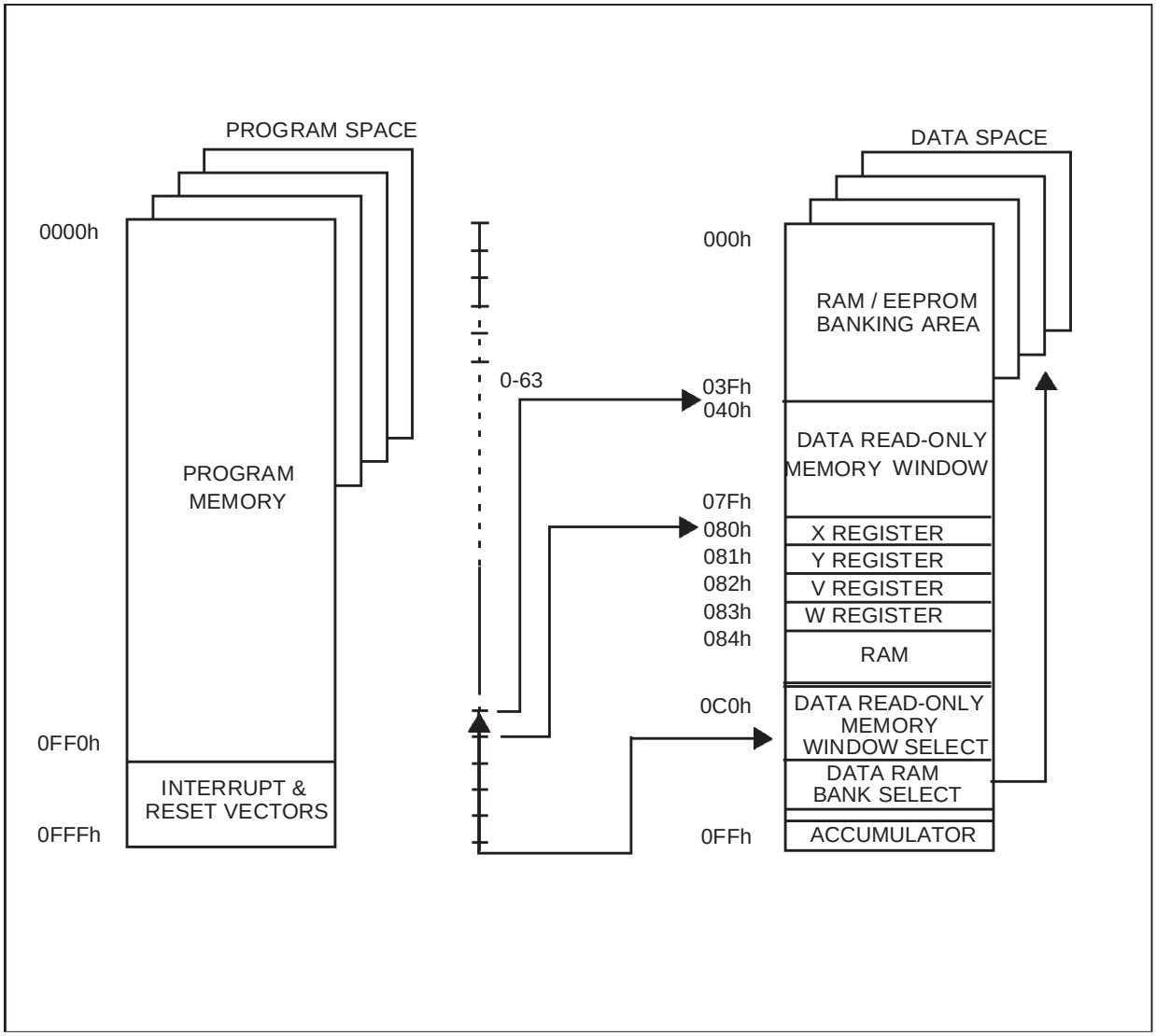
## 1.3 MEMORY MAP

### 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).

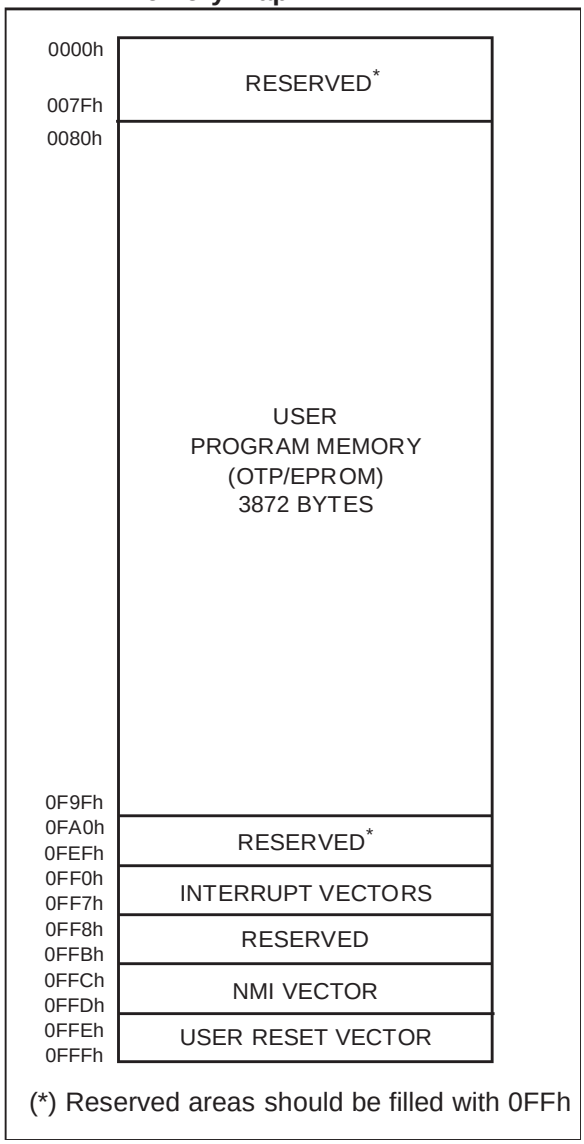
#### 1.3.2.1 Program Memory Protection

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 SGS-THOMSON, to gain access to the OTP contents. Returned parts with a protection set can therefore not be accepted.

**Figure 4.ST62T55B/T65B/E65B 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/EEPROM

In ST62T55B, ST62T65B and ST62E65B 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).

Additional RAM and EEPROM pages can also be addressed using banks of 64 bytes located between addresses 00h and 3Fh.

### 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. Additional RAM/EEPROM Banks**

| Device | RAM | EEPROM |
|---|---|---|
| ST62T55BB | 1 x 64 bytes | - |
| ST62T65B/E65B | 1 x 64 bytes | 2 x 64 bytes |

**Table 2. ST62T55B, ST62T65B and ST62E65B Data Memory Space**

| | |
|---|---|
| RAM and EEPROM | 000h<br>03Fh |
| DATA ROM WINDOW AREA | 040h<br>07Fh |
| X REGISTER | 080h |
| Y REGISTER | 081h |
| V REGISTER | 082h |
| W REGISTER | 083h |
| DATA RAM 60 BYTES | 084h<br>0BFh |
| PORT A DATA REGISTER | 0C0h |
| PORT B DATA REGISTER | 0C1h |
| PORT C DATA REGISTER | 0C2h |
| RESERVED | 0C3h |
| PORT A DIRECTION REGISTER | 0C4h |
| PORT B DIRECTION REGISTER | 0C5h |
| PORT C DIRECTION REGISTER | 0C6h |
| RESERVED | 0C7h |
| INTERRUPT OPTION REGISTER | 0C8h* |
| DATA ROM WINDOW REGISTER | 0C9h* |
| RESERVED | 0CAh<br>0CBh |
| PORT A OPTION REGISTER | 0CCh |
| PORT B OPTION REGISTER | 0CDh |
| PORT C OPTION REGISTER | 0CEh |
| RESERVED | 0CFh |
| A/D DATA REGISTER | 0D0h |
| A/D CONTROL REGISTER | 0D1h |
| TIMER PRESCALER REGISTER | 0D2h |
| TIMER COUNTER REGISTER | 0D3h |
| TIMER STATUS CONTROL REGISTER | 0D4h |
| AR TIMER MODE CONTROL REGISTER | 0D5h |
| AR TIMER STATUS/ CONTROL REGISTER1 | 0D6h |
| AR TIMER STATUS/ CONTROL REGISTER2 | 0D7h |
| WATCHDOG REGISTER | 0D8h |
| AR TIMER RELOAD/CAPTURE REGISTER | 0D9h |
| AR TIMER COMPARE REGISTER | 0DAh |
| AR TIMER LOAD REGISTER | 0DBh |
| OSCILLATOR CONTROL REGISTER | 0DCh* |
| MISCELLANEOUS | 0DDh |
| RESERVED | 0DEh<br>0DFh |
| SPI DATA REGISTER | 0E0h |
| SPI DIVIDER REGISTER | 0E1h |
| SPI MODE REGISTER | 0E2h |
| RESERVED | 0E3h<br>0E7h |
| DATA RAM/EEPROM REGISTER | 0E8h* |
| RESERVED | 0E9h |
| EEPROM CONTROL REGISTER | 0EAh |
| RESERVED | 0EBh<br>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 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 |

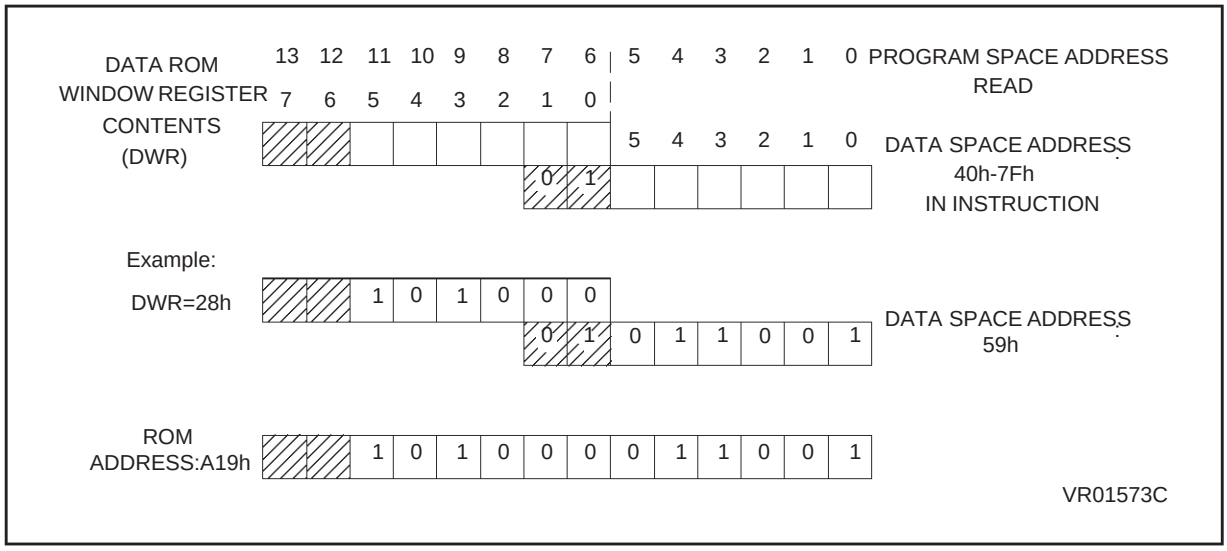
Bits 6, 7 = Not used.

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**



**MEMORY MAP** (Cont'd)

### 1.3.6 Data RAM/EEPROM Bank Register (DRBR)

Address: E8h — Write only

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | DRBR 4 | - | - | DRBR 1 | DRBR 0 |

Bit 7-5 = These bits are not used

Bit 4 - **DRBR4**. This bit, when set, selects RAM Page 2.

Bit 3-2 - Reserved. These bits are not used.

Bit 1 - **DRBR1**. This bit, when set, selects EEPROM Page 1.

Bit 0 - **DRBR0**. This bit, when set, selects EEPROM Page 0.

The selection of the bank is made by programming the Data RAM Bank Switch register (DRBR register) located at address E8h of the Data Space according to Table 1. No more than one bank should be set at a time.

The DRBR register can be addressed like a RAM Data Space at the address E8h; nevertheless it is a write only register that cannot be accessed with single-bit operations. This register is used to select the desired 64-byte RAM/EEPROM bank of the Data Space. The number of banks has to be loaded in the DRBR register and the instruction has to point to the selected location as if it was in bank 0 (from 00h address to 3Fh address).

This register is not cleared during the MCU initialization, therefore it must be written before the first access to the Data Space bank region. Refer to the Data Space description for additional information. The DRBR register is not modified when an interrupt or a subroutine occurs.

**Notes** :

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

In DRBR Register, only 1 bit must be set. Otherwise two or more pages are enabled in parallel, producing errors.

**Table 3. Data RAM Bank Register Set-up**

| DRBR | ST62T55B | ST62T65B/E65B |
|---|---|---|
| 00 | None | None |
| 01 | Not Available | EEPROM Page 0 |
| 02 | Not Available | EEPROM Page 1 |
| 08 | Not Available | Not Available |
| 10h | RAM Page 2 | RAM Page 2 |
| other | Reserved | Reserved |

**MEMORY MAP** (Cont'd)

### 1.3.7 EEPROM Description

EEPROM memory is located in 64-byte pages in data space. This memory may be used by the user program for non-volatile data storage.

Data space from 00h to 3Fh is paged as described in Table 4. EEPROM locations are accessed directly by addressing these paged sections of data space.

The EEPROM does not require dedicated instructions for read or write access. Once selected via the Data RAM Bank Register, the active EEPROM page is controlled by the EEPROM Control Register (EECTL), which is described below.

Bit E20FF of the EECTL register must be reset prior to any write or read access to the EEPROM. If no bank has been selected, or if E2OFF is set, any access is meaningless.

Programming must be enabled by setting the E2ENA bit of the EECTL register.

The E2BUSY bit of the EECTL register is set when the EEPROM is performing a programming cycle. Any access to the EEPROM when E2BUSY is set is meaningless.

Provided E2OFF and E2BUSY are reset, an EEPROM location is read just like any other data location, also in terms of access time.

Writing to the EEPROM may be carried out in two modes: Byte Mode (BMODE) and Parallel Mode (PMODE). In BMODE, one byte is accessed at a time, while in PMODE up to 8 bytes in the same row are programmed simultaneously (with consequent speed and power consumption advantages, the latter being particularly important in battery powered circuits).

**General Notes**:

Data should be written directly to the intended address in EEPROM space. There is no buffer memory between data RAM and the EEPROM space.

When the EEPROM is busy (E2BUSY = "1") EECTL cannot be accessed in write mode, it is only possible to read the status of E2BUSY. This implies that as long as the EEPROM is busy, it is not possible to change the status of the EEPROM Control Register. EECTL bits 4 and 5 are reserved and must never be set.

Care is required when dealing with the EECTL register, as some bits are write only. For this reason, the EECTL contents must not be altered while executing an interrupt service routine.

If it is impossible to avoid writing to this register within an interrupt service routine, an image of the register must be saved in a RAM location, and each time the program writes to EECTL it must also write to the image register. The image register must be written to first so that, if an interrupt occurs between the two instructions, the EECTL will not be affected.

**Table 4. . Row Arrangement for Parallel Writing of EEPROM Locations**

| Byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | Dataspace addresses. Banks 0 and 1. |
|---|---|---|---|---|---|---|---|---|---|
| ROW7 | | | | | | | | | 38h-3Fh |
| ROW6 | | | | | | | | | 30h-37h |
| ROW5 | | | | | | | | | 28h-2Fh |
| ROW4 | | | | | | | | | 20h-27h |
| ROW3 | | | | | | | | | 18h-1Fh |
| ROW2 | | | | | | | | | 10h-17h |
| ROW1 | | | | | | | | | 08h-0Fh |
| ROW0 | | | | | | | | | 00h-07h |

Up to 8 bytes in each row may be programmed simultaneously in Parallel Write mode.

The number of available 64-byte banks (1 or 2) is device dependent.

**MEMORY MAP** (Cont'd)

**Additional Notes on Parallel Mode:**

If the user wishes to perform parallel programming, the first step should be to set the E2PAR2 bit. From this time on, the EEPROM will be addressed in write mode, the ROW address will be latched and it will be possible to change it only at the end of the programming cycle, or by resetting E2PAR2 without programming the EEPROM. After the ROW address is latched, the MCU can only "see" the selected EEPROM row and any attempt to write or read other rows will produce errors.

The EEPROM should not be read while E2PAR2 is set.

As soon as the E2PAR2 bit is set, the 8 volatile ROW latches are cleared. From this moment on, the user can load data in all or in part of the ROW. Setting E2PAR1 will modify the EEPROM registers corresponding to the ROW latches accessed after E2PAR2. For example, if the software sets E2PAR2 and accesses the EEPROM by writing to addresses 18h, 1Ah and 1Bh, and then sets E2PAR1, these three registers will be modified simultaneously; the remaining bytes in the row will be unaffected.

Note that E2PAR2 is internally reset at the end of the programming cycle. This implies that the user must set the E2PAR2 bit between two parallel programming cycles. Note that if the user tries to set E2PAR1 while E2PAR2 is not set, there will be no programming cycle and the E2PAR1 bit will be unaffected. Consequently, the E2PAR1 bit cannot be set if E2ENA is low. The E2PAR1 bit can be set by the user, only if the E2ENA and E2PAR2 bits are also set.

**EEPROM Control Register (EECTL)**

Address: EAh — Read/Write

Reset status: 00h

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| D7 | E2OFF | D5 | D4 | E2PAR1 | E2PAR2 | E2BUSY | E2ENA |

Bit 7 = **D7**: *Unused.*

Bit 6 = **E2OFF**: *Stand-by Enable Bit.* WRITE ONLY. If this bit is set the EEPROM is disabled (any access will be meaningless) and the power consumption of the EEPROM is reduced to its lowest value.

Bit 5-4 = **D5-D4**: *Reserved.* MUST be kept reset.

Bit 3 = **E2PAR1**: *Parallel Start Bit.* WRITE ONLY. Once in Parallel Mode, as soon as the user software sets the E2PAR1 bit, parallel writing of the 8 adjacent registers will start. This bit is internally reset at the end of the programming procedure. Note that less than 8 bytes can be written if required, the undefined bytes being unaffected by the parallel programming cycle; this is explained in greater detail in the Additional Notes on Parallel Mode overleaf.

Bit 2 = **E2PAR2**: *Parallel Mode En. Bit.* WRITE ONLY. This bit must be set by the user program in order to perform parallel programming. If E2PAR2 is set and the parallel start bit (E2PAR1) is reset, up to 8 adjacent bytes can be written simultaneously. These 8 adjacent bytes are considered as a row, whose address lines A7, A6, A5, A4, A3 are fixed while A2, A1 and A0 are the changing bits, as illustrated in 4. E2PAR2 is automatically reset at the end of any parallel programming procedure. It can be reset by the user software before starting the programming procedure, thus leaving the EEPROM registers unchanged.

Bit 1 = **E2BUSY**: *EEPROM Busy Bit.* READ ONLY. This bit is automatically set by the EEPROM control logic when the EEPROM is in programming mode. The user program should test it before any EEPROM read or write operation; any attempt to access the EEPROM while the busy bit is set will be aborted and the writing procedure in progress will be completed.

Bit 0 = **E2ENA**: *EEPROM Enable Bit.* WRITE ONLY. This bit enables programming of the EEPROM cells. It must be set before any write to the EEPROM register. Any attempt to write to the EEPROM when E2ENA is low is meaningless and will not trigger a write cycle.

## 1.4 PROGRAMMING MODES

### 1.4.1 Option Byte

The Option Byte allows 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 byte is located in a non-user map. No address has to be specified.

**EPROM Code Option Byte**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| PRO-TECT | EXTC-NTL | PB2-3 PULL | PB0-1 PULL | WDACT | DELAY | OSCIL | - |

**PROTECT**. 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. No programming equipment is able to gain access to the user program. When this bit is low, the user program can be read.

**EXTCNTL**. This bit selects the External STOP Mode capability. When EXTCNTL is high, pin NMI controls if the STOP mode can be accessed when the watchdog is active. In addition, PB0 is forced as open drain output. When EXTCNTL is low, the STOP instruction is processed as a WAIT as soon as the watchdog is active.

**PB2-3 PULL**. When set this bit removes pull-up at reset on PB2-PB3 pins. When cleared PB2-PB3 pins have an internal pull-up resistor at reset.

**PB0-1 PULL**. When set this bit removes pull-up at reset on PB0-PB1 pins. When cleared PB0-PB1 pins have an internal pull-up resistor at reset.

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

**DELAY**. This bit enables the selection of the delay internally generated after pin RESET is released. When DELAY is low, the delay is 2048 cycles of the oscillator, it is of 32768 cycles when DELAY is high.

**OSCIL**. 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.

**D0**. Reserved. Must be cleared to zero.

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

### 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 ST62T65B/E65B is described in the User Manual of the EPROM Programming Board.

The MCUs can be programmed with the ST62E6xB EPROM programming tools available from SGS-THOMSON.

**Table 5. ST6255B, ST62T65B/E65B 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 SGS-THOMSON (ST62E6X-EPB or ST626X-KIT).

### 1.4.3 EEPROM Data Memory

EEPROM data pages are supplied in the virgin state FFh. Partial or total programming of EEPROM data memory can be performed either through the application software, or through an external programmer. Any SGS-THOMSON tool used for the program memory (OTP/EPROM) can also be used to program the EEPROM data memory.

**PROGRAMMING MODES**(Cont'd)

### 1.4.4 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 15W-sec/cm$^2$. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with 12000µW/cm$^2$ power rating. The ST62E65B should be placed within 2.5cm (1Inch) of the lamp tubes during erasure.

# 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 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.

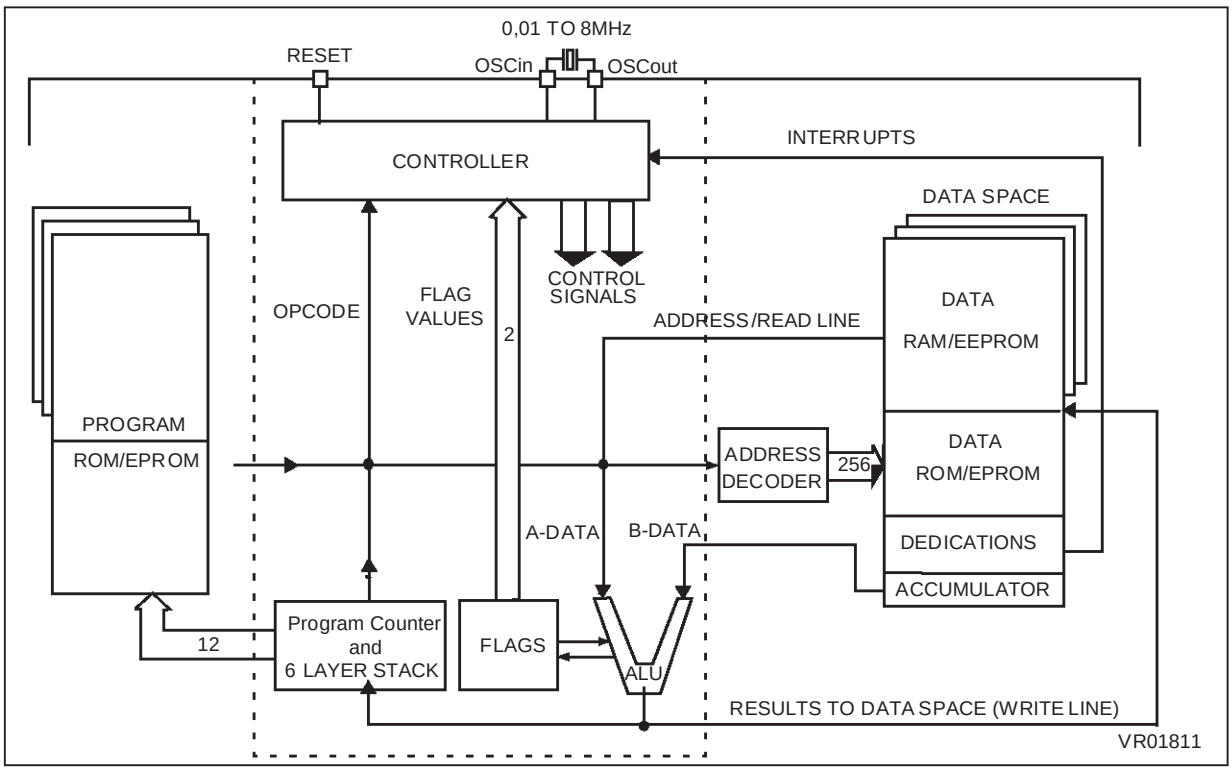
**Accumulator (A).** The accumulator is an 8-bit general purpose register used in all arithmetic calculations, logical operations, and data manipulations. The accumulator can be addressed in Data space as a RAM location at address FFh. Thus the ST6 can manipulate the accumulator just like any other register 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) instruction PC=Jump address
- CALL instruction PC= Call address
- Relative Branch Instruction PC= PC +/- offset
- Interrupt PC=Interrupt vector
- Reset PC= Reset vector
- RET & RETI instructions PC= Pop (stack)
- Normal instruction PC= 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.

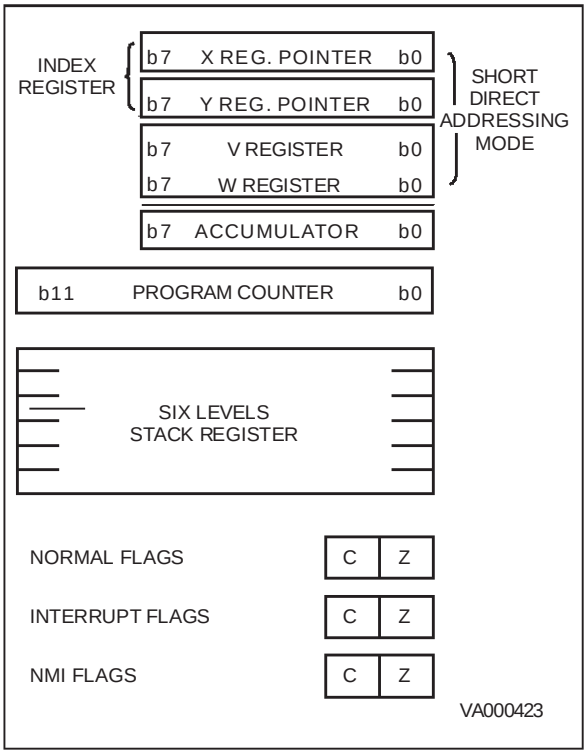
The Carry flag is set when a carry or a borrow occurs during arithmetic operations; otherwise it is cleared. The Carry flag is also set to the value of the bit tested in a bit test instruction; it also participates in the rotate left instruction.

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 ($R_{NET}$).

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

A programmable divider is provided in order to adjust the internal clock of the MCU to the best power consumption and performance trade-off.

The internal MCU clock frequency ($f_{NT}$) drives directly the AR TIMER while it 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 9.

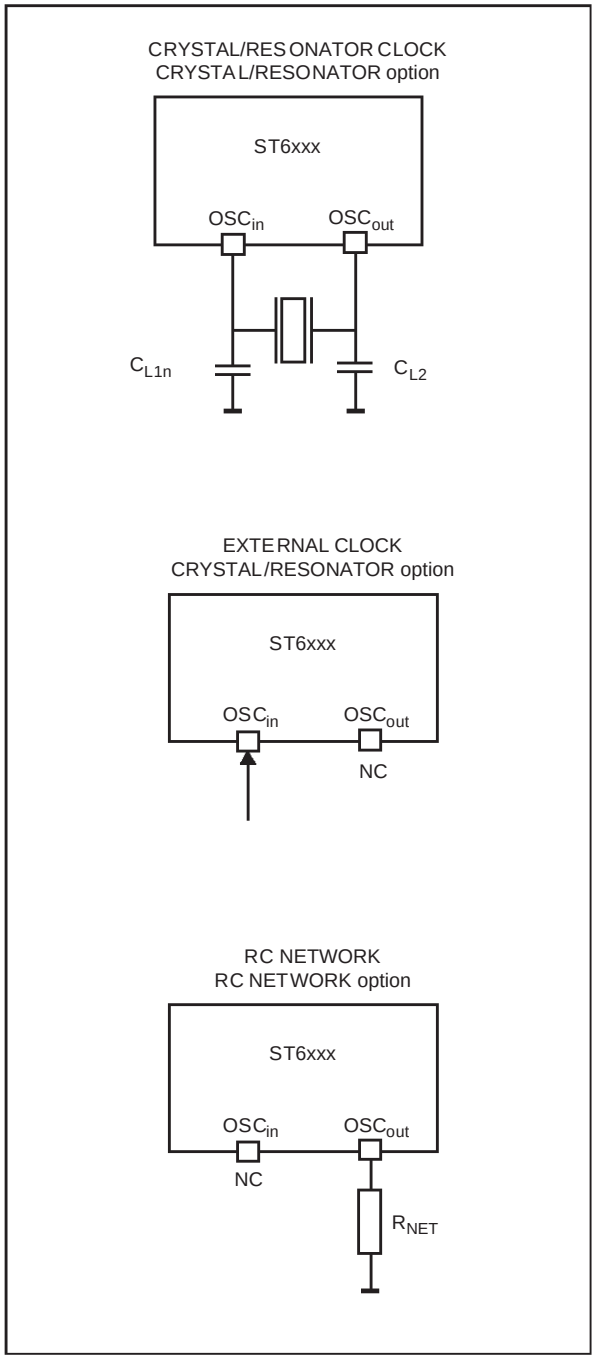
With an 8 MHz oscillator frequency, the fastest machine cycle is therefore 1.625µs.

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.

**Figure 8. Oscillator Configurations**



**CLOCK SYSTEM** (Cont'd)

**Oscillator Control Registers**

Address: DCh — Write only

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | OSCR 3 | OSCR 2 | RS1 | RS0 |

Bit 7-4. These bits are not used.

Bit 3. Reserved. Cleared at Reset. THIS BIT MUST BE SET TO 1 BY USER PROGRAM to achieve lowest power consumption.

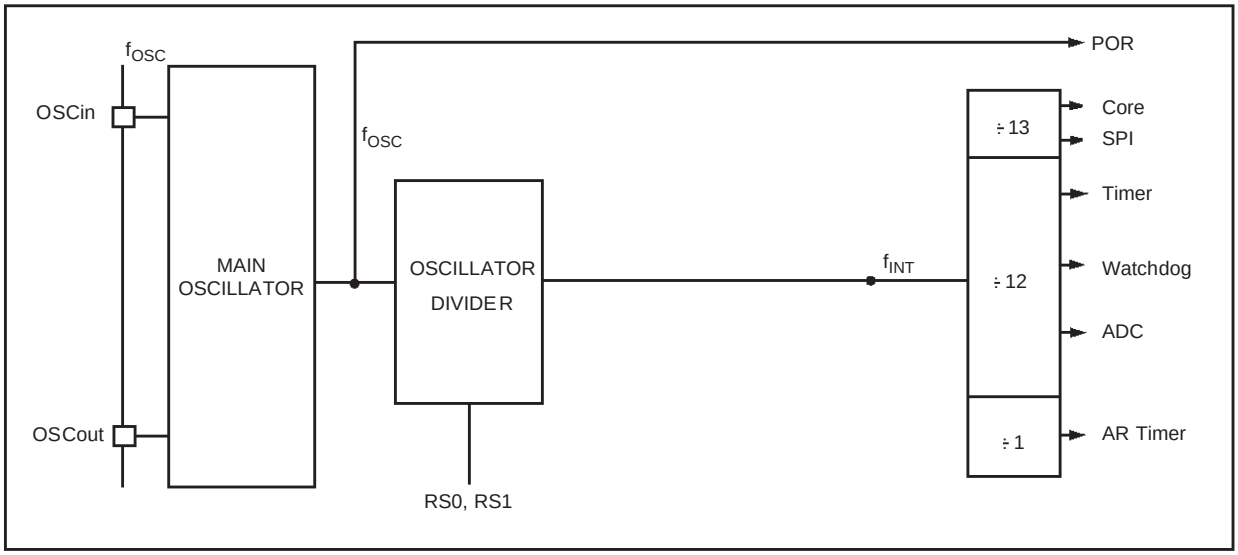
Bit 2. Reserved. Must be kept low.

RS1-RS0. These bits select the division ratio of the Oscillator Divider in order to generate the internal frequency. The following selctions are available:

| RS1 | RS0 | Division Ratio |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 2 |
| 1 | 0 | 4 |
| 1 | 1 | 4 |

**Note**: Care is required when handling the OSCR register as some bits are write only. For this reason, it is not allowed to change the OSCR contents while executing interrupt service routine, as the service routine cannot save and then restore its previous content. If it is impossible to avoid the writing of this register in interrupt service routine, an image of this register must be saved in a RAM location, and each time the program writes to OSCR it must write also to the image register. The image register must be written first, so if an interrupt occurs between the two instructions the OSCR is not affected.

**Figure 9. Clock Circuit Block Diagram**



### 3.2 RESETS

The MCU can be reset in three ways:

– by the external Reset input being pulled low;

– by Power-on Reset;

– by the digital Watchdog peripheral timing out.

#### 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 at an appropriate stage during the power-on sequence. 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.

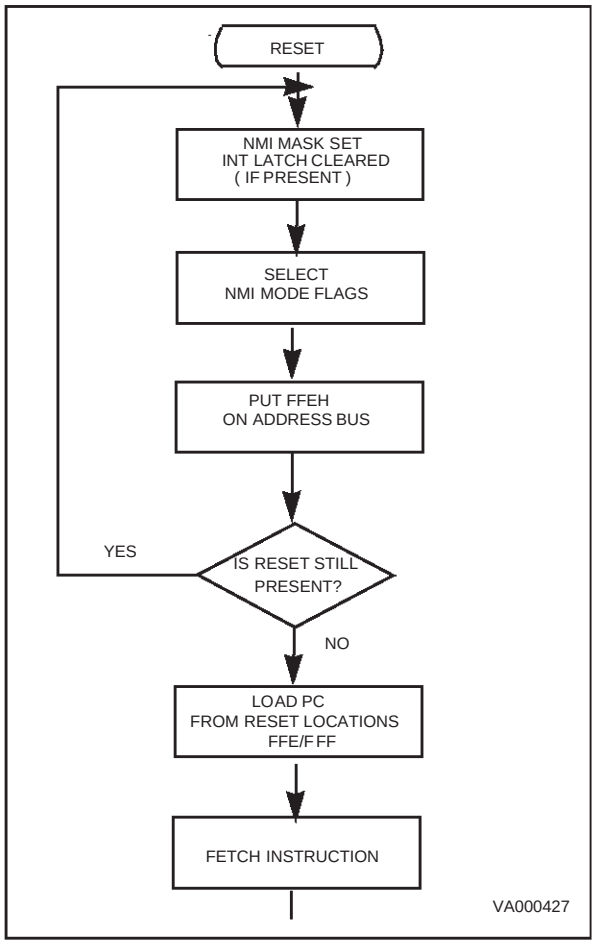
The internal delay is generated by an on-chip counter. The internal reset line is released 2048 internal clock cycles after release of the external reset.

**Notes:**

To ensure correct start-up, the user should take care that the reset signal is not released before the $V_{DD}$ level is sufficient to allow MCU operation at the chosen frequency (see Recommended Operating Conditions).

A proper reset signal for a slow rising $V_{DD}$ supply can generally be provided by an external RC network connected to the $\overline{\text{RESET}}$ pin.

**Figure 10. . 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 Application Notes

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

The POR circuit operates dynamically, in that it triggers MCU initialization on detecting the rising edge of $V_{DD}$. The typical threshold is in the region of 2 volts, but the actual value of the detected threshold depends on the way in which $V_{DD}$ rises.

The POR circuit is *NOT* designed to supervise static, or slowly rising or falling $V_{DD}$.

### 3.2.5 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 11. . Reset and Interrupt Processing**

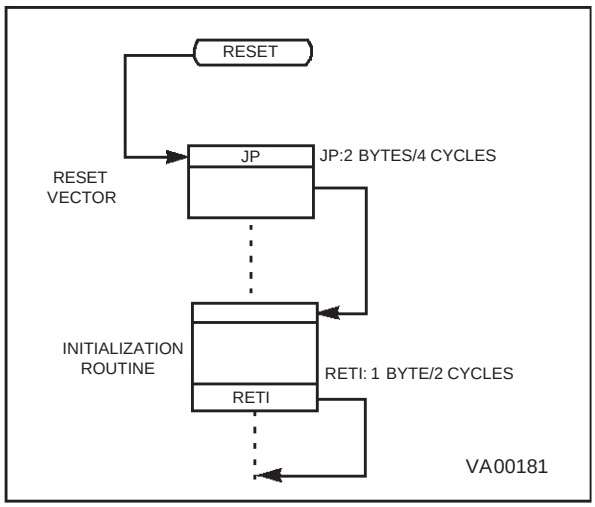
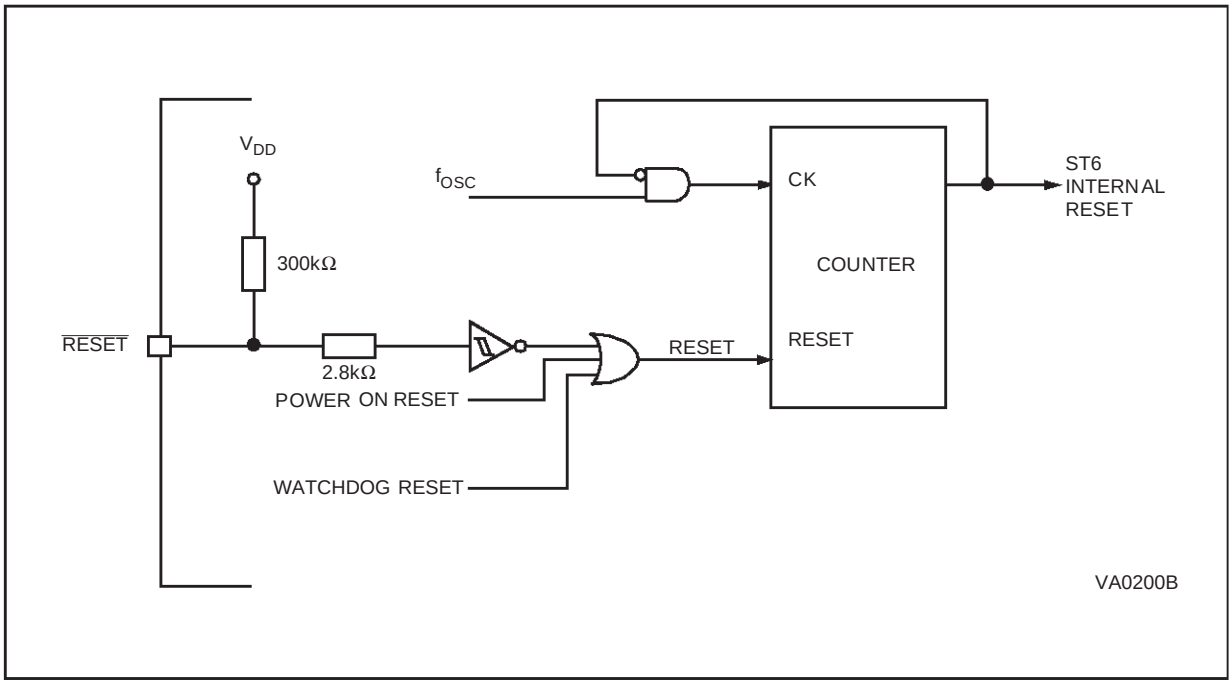



**Figure 12. . Reset Block Diagram**



**RESETS** (Cont'd)

**Table 6. Register Reset Status**

| Register | Address(es) | Status | Comment |
|---|---|---|---|
| Oscillator Control Register | 0DCh | 00h | $f_{INT} = f_{OSC}$; user must set bit3 to 1 |
| EEPROM Control Register | 0EAh | 00h | EEPROM disabled (if available) |
| Port Data Registers | 0C0h to 0C2h | 00h | I/O are Input with pull-up |
| Port Direction Register | 0C4h to 0C6h | 00h | I/O are Input with pull-up |
| Port Option Register | 0CCh to 0CEh | 00h | I/O are Input with pull-up |
| Interrupt Option Register | 0C8h | 00h | Interrupt disabled |
| TIMER Status/Control | 0D4h | 00h | TIMER disabled |
| AR TIMER Mode Control Register | 0D5h | 00h | AR TIMER stopped |
| AR TIMER Status/Control 1 Register | 0D6h | 02h | |
| AR TIMER Status/Control 2 Register | 0D7h | 00h | |
| AR TIMER Compare Register | 0DAh | 00h | |
| AR TIMER Load Register | 0DBh | 00h | |
| Miscellaneous Register | 0DDh | 00h | SPI Output not connected to PC3 |
| SPI Registers | 0E0h to 0E2h | 00h | SPI disabled |
| X, Y, V, W, Register | 080H TO 083H | | |
| Accumulator | 0FFh | | |
| Data RAM | 084h to 0BFh | | |
| Data RAM Page REgister | 0E8h | | |
| Data ROM Window Register | 0C9h | Undefined | |
| EEPROM | 00h to 03Fh | | As written if programmed |
| A/D Result Register | 0D0h | | |
| AR TIMER Load Register | 0DBh | | |
| AR TIMER Reload/Capture Register | 0D9h | | |
| TIMER Counter Register | 0D3h | FFh | |
| TIMER Prescaler Register | 0D2h | 7Fh | Max count loaded |
| Watchdog Counter Register | 0D8h | FEh | |
| A/D Control Register | 0D1h | 40h | A/D in Standby |

### 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 7).

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 7. Recommended Option Choices**

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

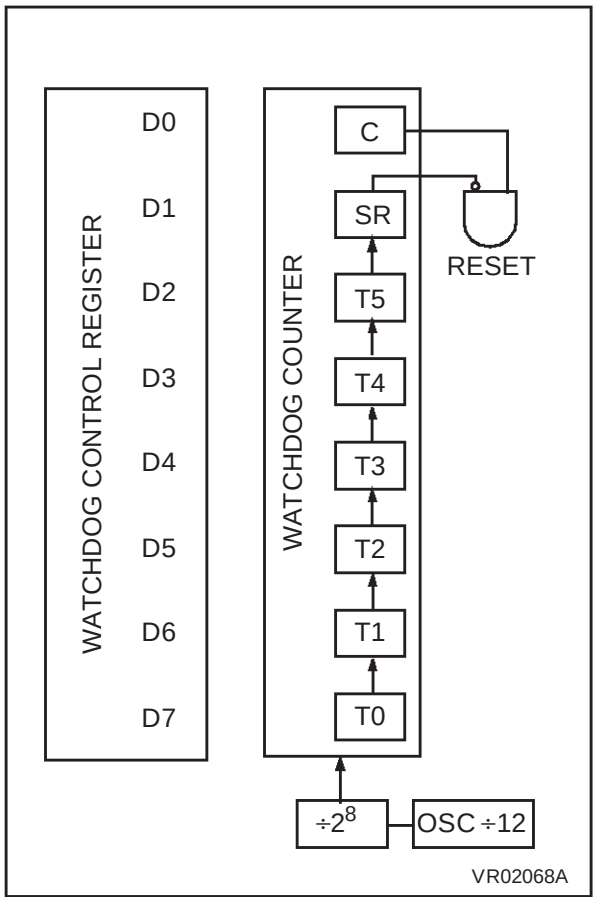
**DIGITAL WATCHDOG** (Cont'd)

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 in13.

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 13. 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 14) 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 14. A typical circuit making use of the EXERNAL STOP MODE CONTROL feature**

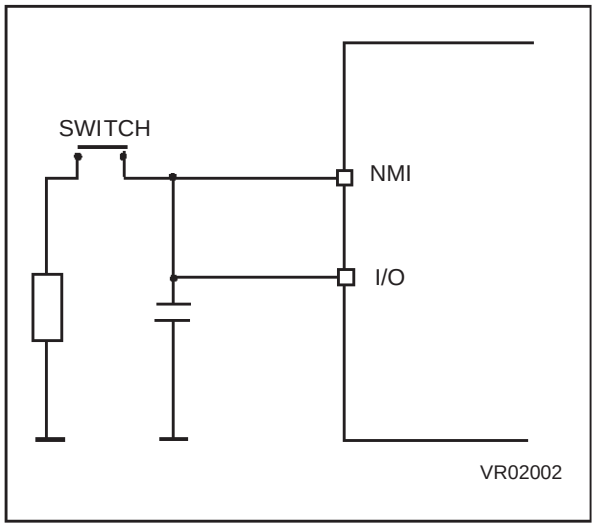
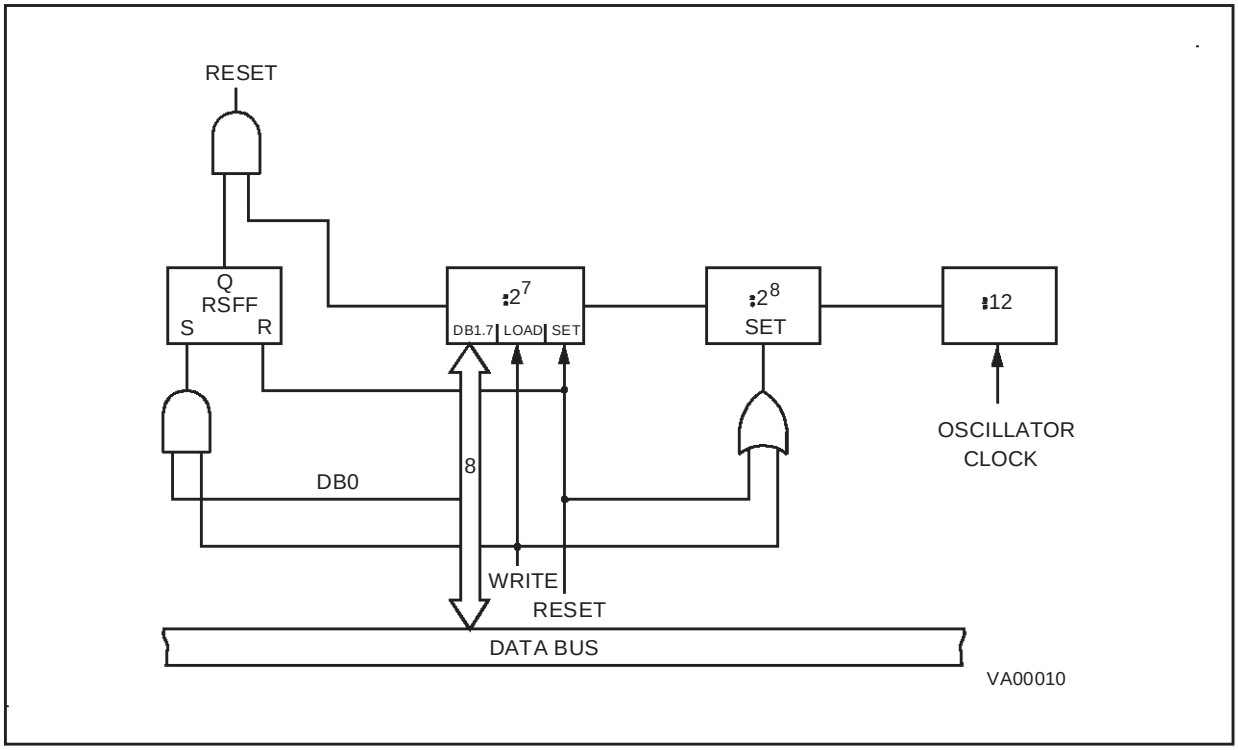



**Figure 15. Digital Watchdog Block Diagram**



### 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 8).

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 8. 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 9. 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 | |

**IINTERRUPTS** (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***

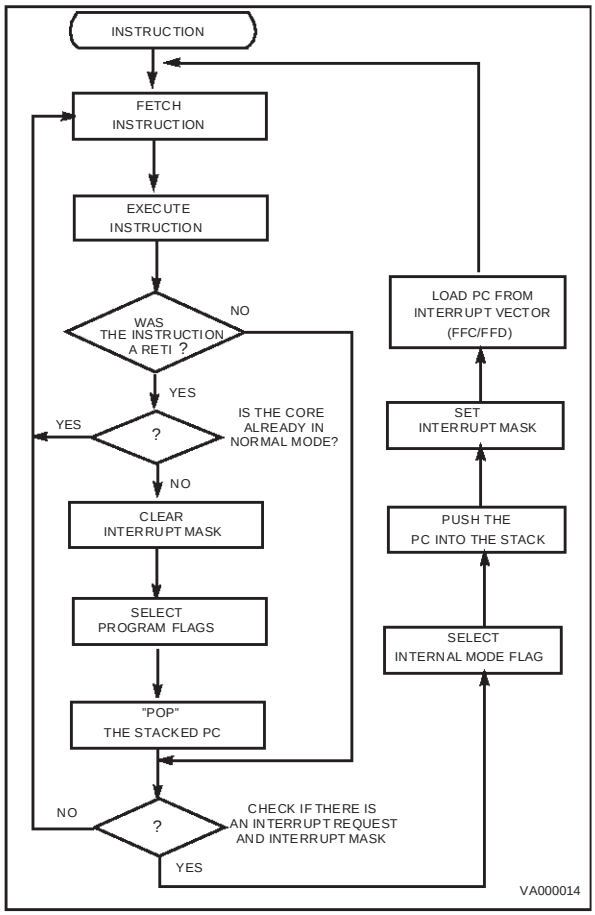
– User selected registers are saved within the interrupt service routine (normally on a software stack).
– The source of the interrupt is found by polling the interrupt flags (if more than one source is associated with the same vector).
– The interrupt is serviced.
– Return from interrupt (RETI)

***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 16. Interrupt Processing Flow Chart**



**IINTERRUPTS** (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

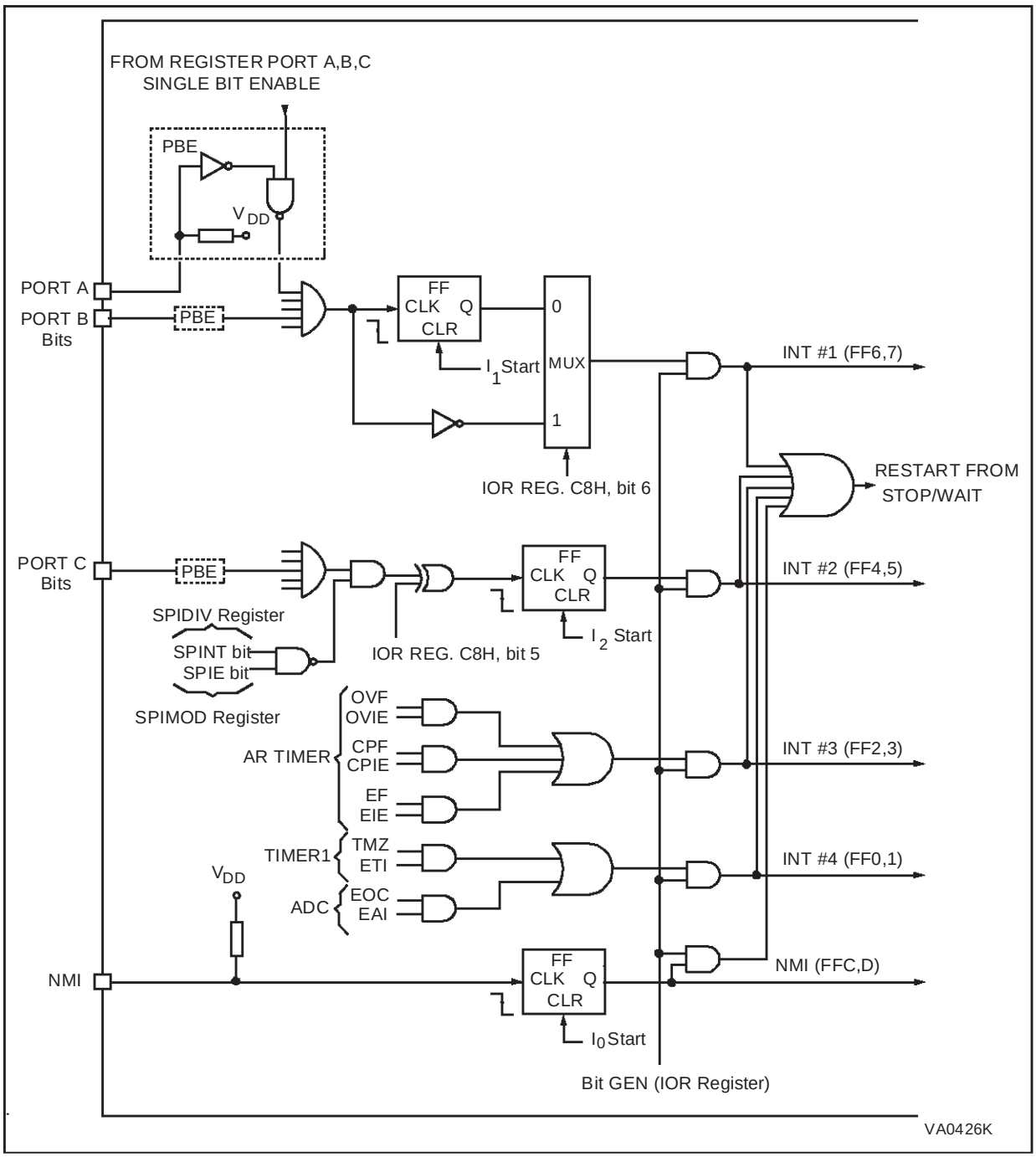
Interrupt sources available on these MCUs are summarized in the Table 10 with associated mask bit to enable/disable the interrupt request.

**Table 10. 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 | TSCR1 | D4h | ETI | TMZ: TIMER Overflow | Vector 4 |
| A/D CONVERTER | ADCR | D1h | EAI | EOC: End of Conversion | Vector 4 |
| AR TIMER | ARMC | D5h | OVIE<br>CPIE<br>EIE | OVF: AR TIMER Overflow<br>CPF: Successful compare<br>EF: Active edge on ARTIMin | Vector 3 |
| SPI | SPIMOD | E2h | SPIE | SPRUN: End of Transmission | Vector 2 |
| Port PAn | ORPA-DRPA | C0h-C4h | ORPAn-DRPAn | PAn pin | Vector 1 |
| Port PBn | ORPB-DRPB | C1h-C5h | ORPBn-DRPBn | PBn pin | Vector 1 |
| Port PCn | ORPC-DRPC | C2h-C6h | ORPCn-DRPCn | PCn pin | Vector 2 |

**INTERRUPTS** (Cont'd)

**Figure 17. Interrupt Block Diagram**



### 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.

#### 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 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 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.

#### 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;

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.

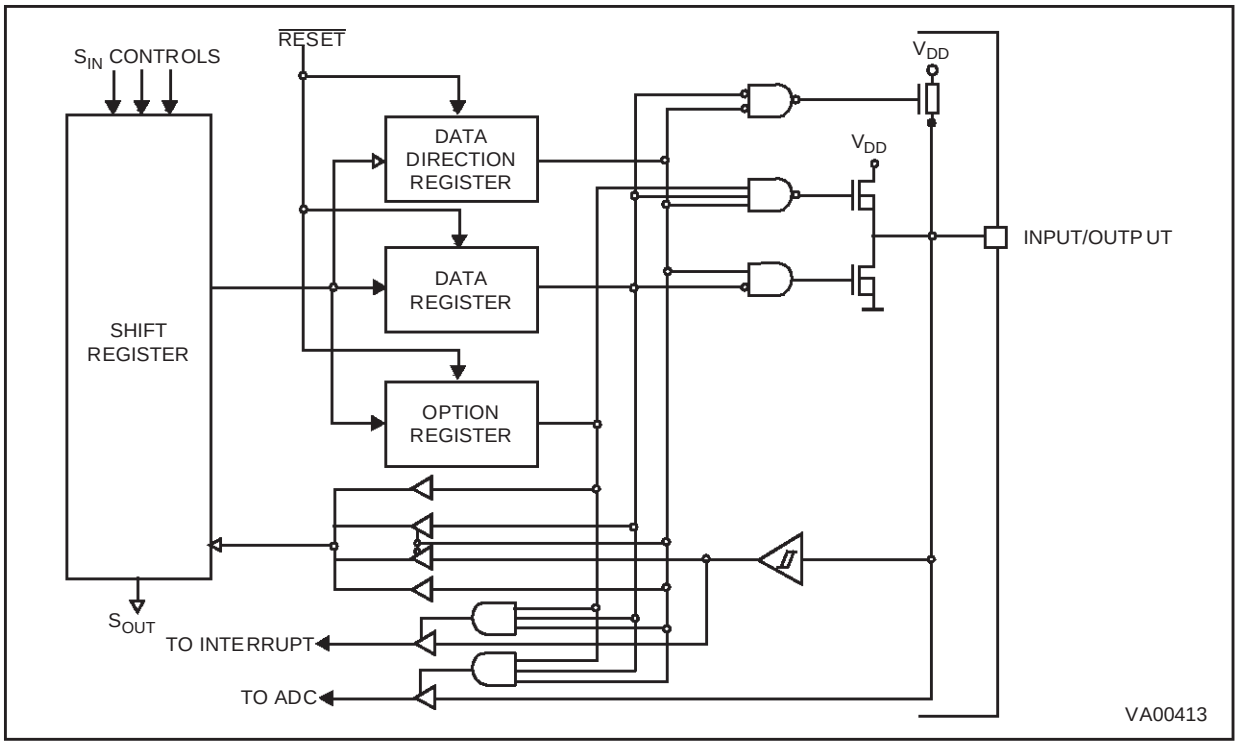
Single-bit operations on I/O registers are possible but care is necessary because reading in input mode is done from I/O pins while writing will directly affect the Port data register causing an undesired change of the input configuration.

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 18. 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 11 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 11. 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 19. 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:

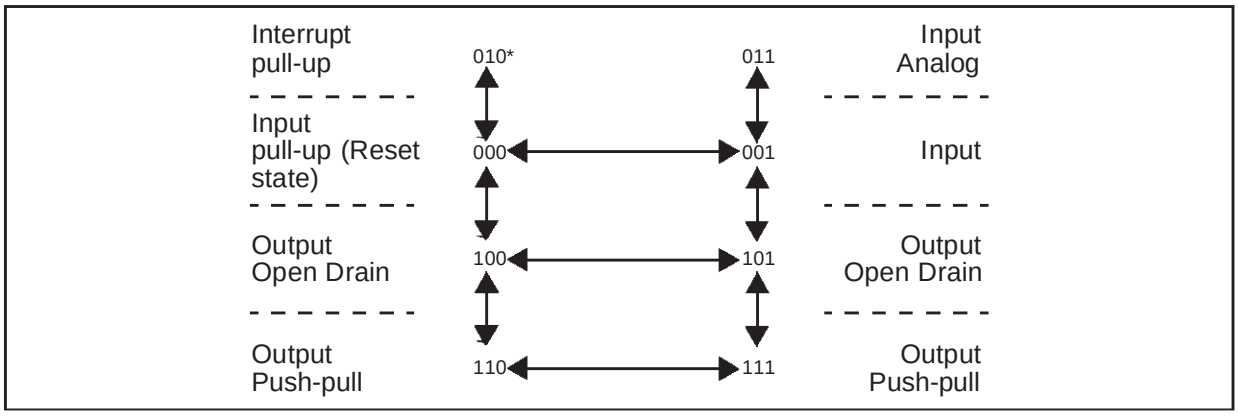
```
SET bit, datacopy
LD a, datacopy
LD DRA, a
```

**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.

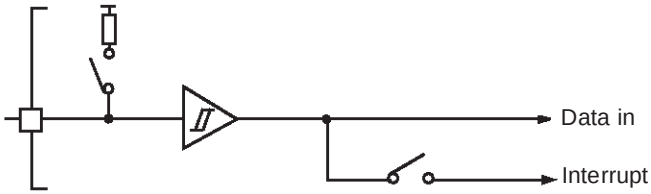
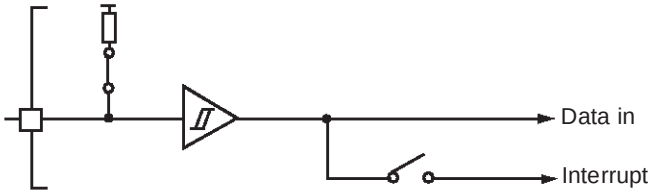
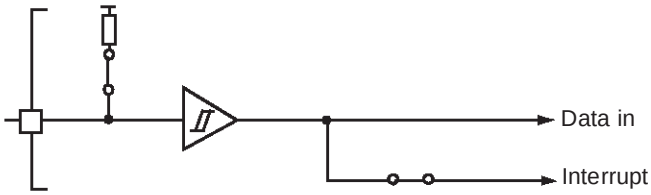
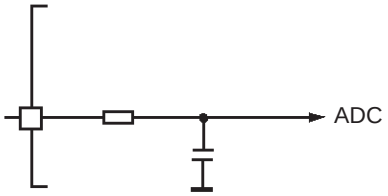
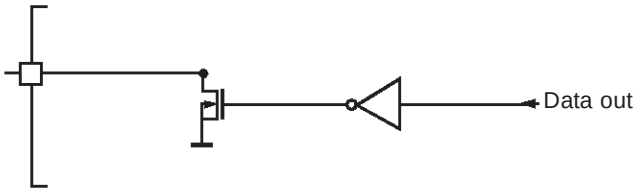
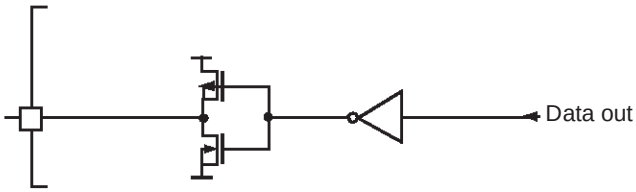
**Figure 19. Diagram showing Safe I/O State Transitions**




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

**I/O PORTS** (Cont'd)

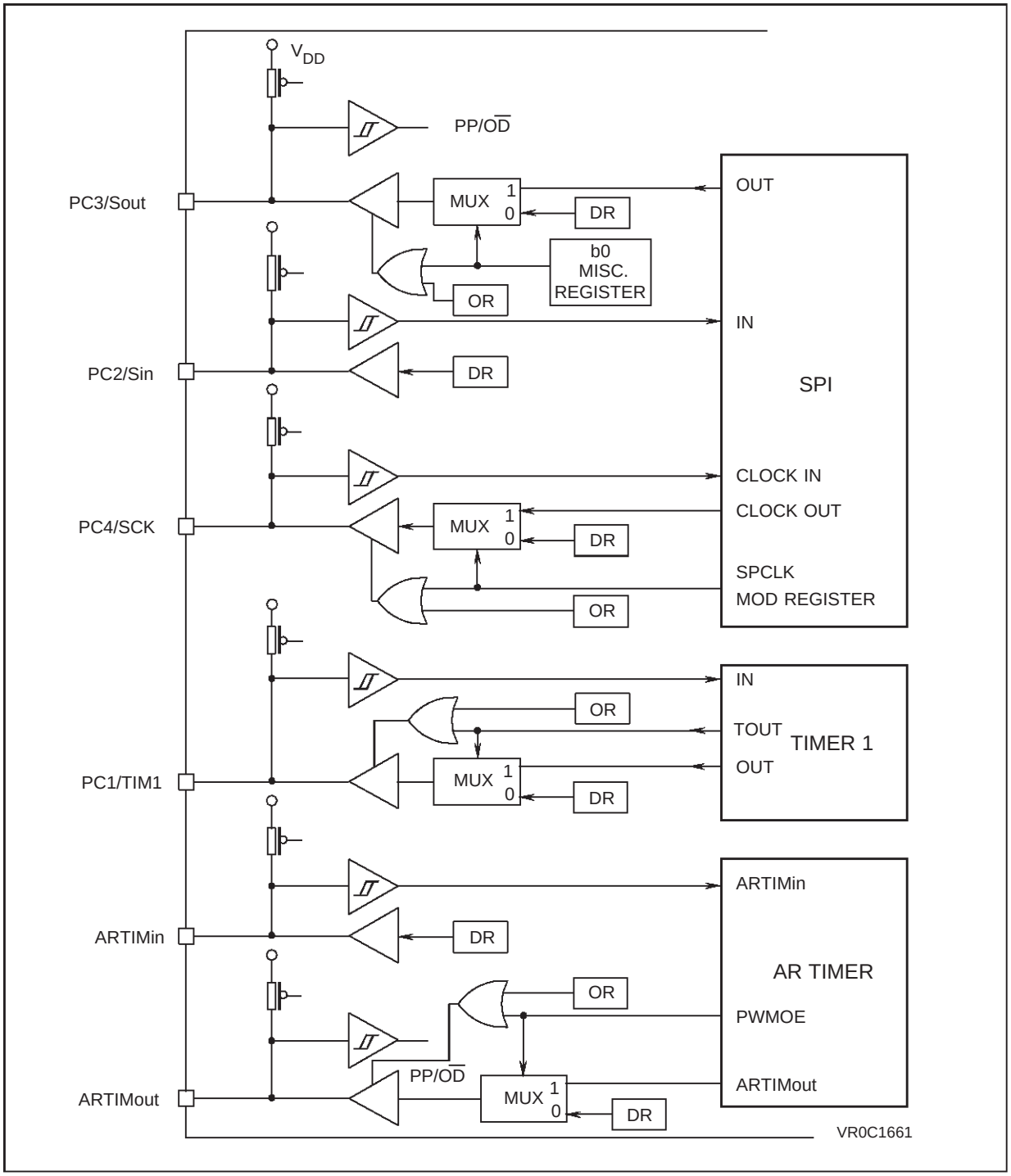
**Table 12. I/O Port Option Selections**

| MODE | AVAILABLE ON(1) | SCHEMATIC |
|---|---|---|
| Input | PA0-PA7<br>PB0-PB5, PB6-PB7<br>PC0-PC4 |  |
| Input<br>with pull up | PA0-PA7<br>PB0-PB5, PB6-PB7<br>PC0-PC4 |  |
| Input<br>with pull up<br>with interrupt | PA0-PA7<br>PB0-PB5, PB6-PB7<br>PC0-PC4 |  |
| Analog Input | PA0-PA7<br>PC0-PC4 |  |
| Open drain output<br>5mA<br><br>Open drain output<br>20mA | PA0-PA7<br>PC0-PC4<br><br>PB0-PB5, PB6-PB7 |  |
| Push-pull output<br>5mA<br><br>Push-pull output<br>20mA | PA0-PA7<br>PC0-PC4<br><br>PB0-PB5, PB6-PB7 |  |

**Note 1**. Provided the correct configuration has been selected.

**I/O PORTS** (Cont'd)

**Figure 20. Peripheral Interface Configuration of SPI, Timer 1 and AR Timer**



### 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.

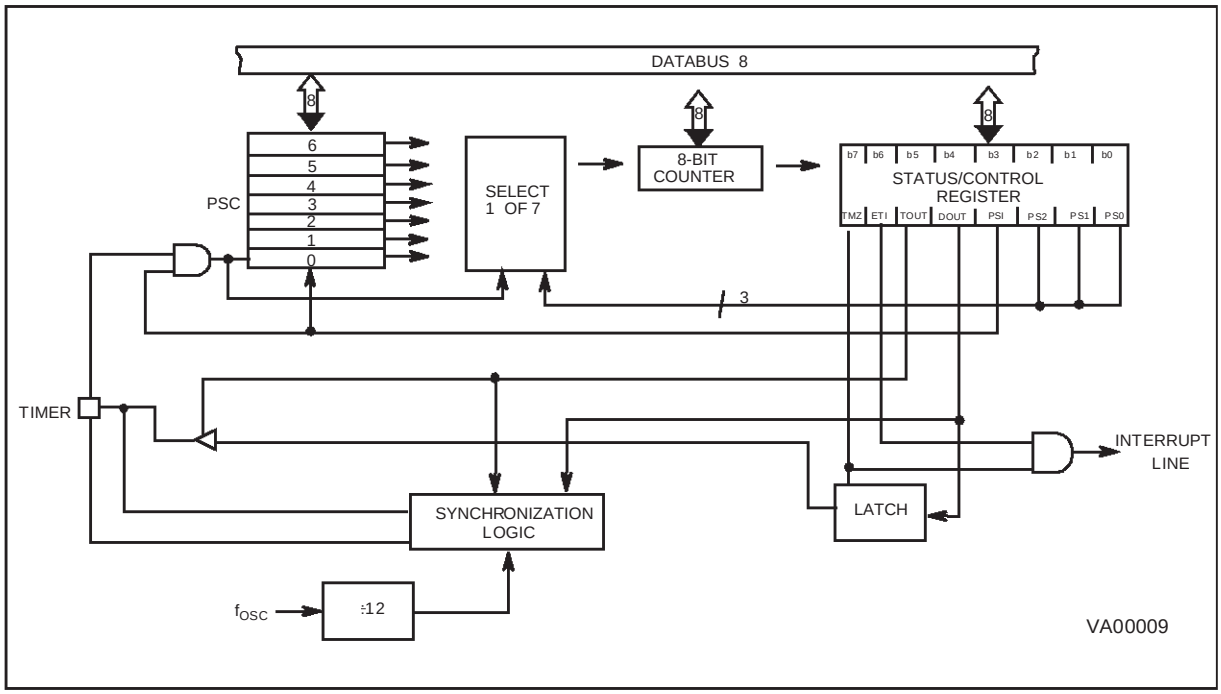
21 shows the Timer Block Diagram. The external TIMER pin is available to the user. The content of the 8-bit counter can be read/written in the Timer/Counter register, TCR, while the state of the 7-bit prescaler can be read in the PSC register. The control logic device is managed in the TSCR register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

22 illustrates the Timer’s working principle.

**Figure 21. 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.

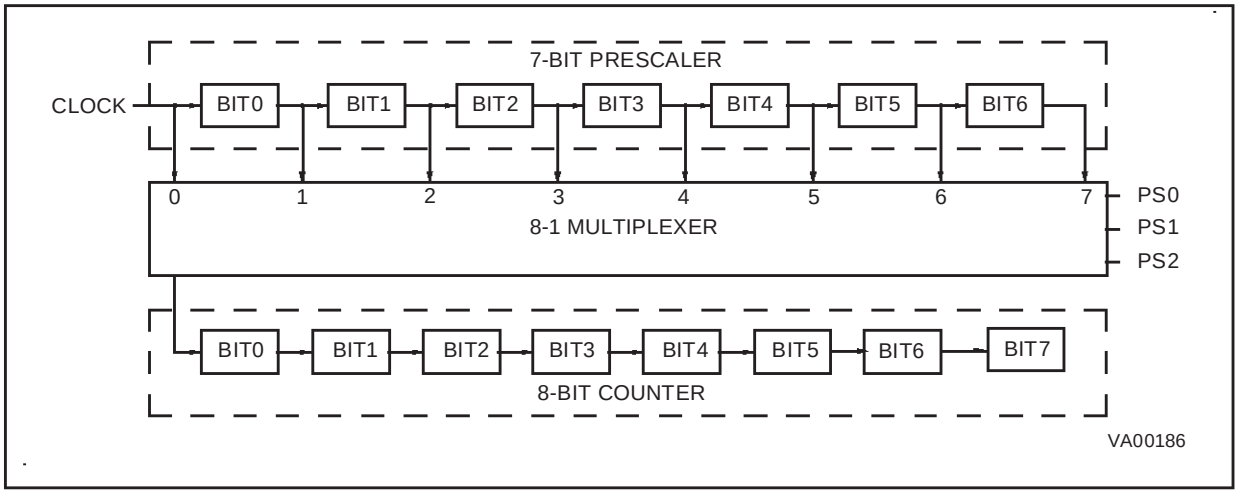
**Table 13. Timer Operating Modes**

| 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 22. 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 14. 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 AUTO-RELOAD TIMER

The Auto-Reload Timer (AR Timer) on-chip peripheral consists of an 8-bit timer/counter with compare and capture/reload capabilities and of a 7-bit prescaler with a clock multiplexer, enabling the clock input to be selected as $f_{INT}$, $f_{INT/3}$ or an external clock source. A Mode Control Register, ARMC, two Status Control Registers, ARSC0 and ARSC1, an output pin, ARTIMout, and an input pin, ARTIMin, allow the Auto-Reload Timer to be used in 4 modes:

– Auto-reload (PWM generation),

– Output compare and reload on external event (PLL),

– Input capture and output compare for time measurement.

– Input capture and output compare for period measurement.

The AR Timer can be used to wake the MCU from WAIT mode either with an internal or with an external clock. It also can be used to wake the MCU from STOP mode, if used with an external clock signal connected to the ARTIMin pin. A Load register allows the program to read and write the counter on the fly.

#### 4.3.1 AR Timer Description

The AR COUNTER is an 8-bit up-counter incremented on the input clock's rising edge. The counter is loaded from the ReLoad/Capture Register, ARRC, for auto-reload or capture operations, as well as for initialization. Direct access to the AR counter is not possible; however, by reading or writing the ARLR load register, it is possible to read or write the counter's contents on the fly.

The AR Timer's input clock can be either the internal clock (from the Oscillator Divider), the internal clock divided by 3, or the clock signal connected to the ARTIMin pin. Selection between these clock sources is effected by suitably programming bits CC0-CC1 of the ARSC1 register. The output of the AR Multiplexer feeds the 7-bit programmable AR Prescaler, ARPSC, which selects one of the 8 available taps of the prescaler, as defined by PSC0-PSC2 in the AR Mode Control Register. Thus the division factor of the prescaler can be set to 2n (where n = 0, 1,..7).

The clock input to the AR counter is enabled by the TEN (Timer Enable) bit in the ARMC register. When TEN is reset, the AR counter is stopped and the prescaler and counter contents are frozen. When TEN is set, the AR counter runs at the rate of the selected clock source. The counter is cleared on system reset.

The AR counter may also be initialized by writing to the ARLR load register, which also causes an immediate copy of the value to be placed in the AR counter, regardless of whether the counter is running or not. Initialization of the counter, by either method, will also clear the ARPSC register, whereupon counting will start from a known value.

#### 4.3.2 Timer Operating Modes

Four different operating modes are available for the AR Timer:

**Auto-reload Mode with PWM Generation.** This mode allows a Pulse Width Modulated signal to be generated on the ARTIMout pin with minimum Core processing overhead.

The free running 8-bit counter is fed by the prescaler's output, and is incremented on every rising edge of the clock signal.

When a counter overflow occurs, the counter is automatically reloaded with the contents of the Reload/Capture Register, ARCC, and ARTIMout is set. When the counter reaches the value contained in the compare register (ARCP), ARTIMout is reset.

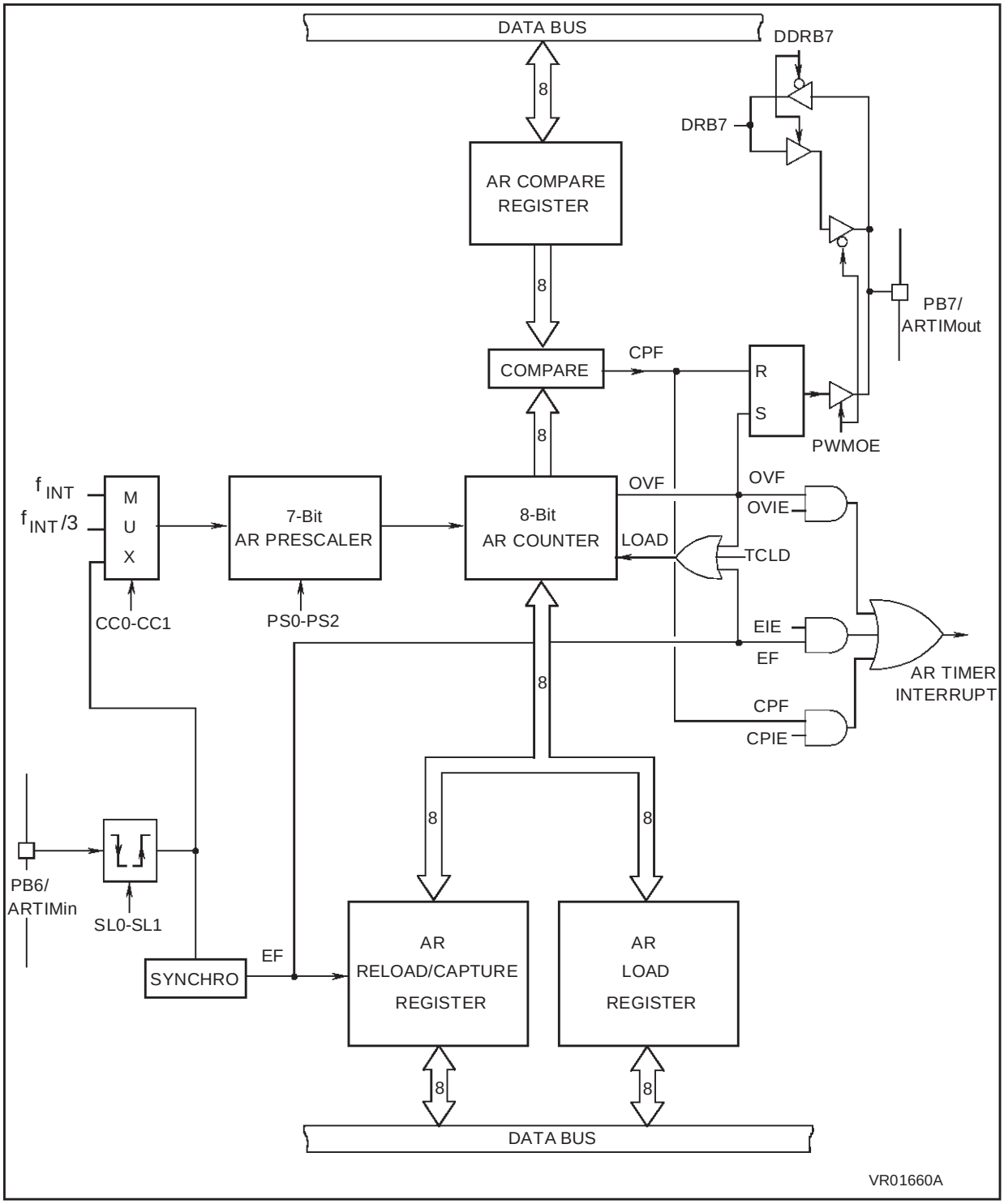
On overflow, the OVF flag of the ARSC0 register is set and an overflow interrupt request is generated if the overflow interrupt enable bit, OVIE, in the Mode Control Register (ARMC), is set. The OVF flag must be reset by the user software.

When the counter reaches the compare value, the CPF flag of the ARSC0 register is set and a compare interrupt request is generated, if the Compare Interrupt enable bit, CPIE, in the Mode Control Register (ARMC), is set. The interrupt service routine may then adjust the PWM period by loading a new value into ARCP. The CPF flag must be reset by user software.

The PWM signal is generated on the ARTIMout pin (refer to the Block Diagram). The frequency of this signal is controlled by the prescaler setting and by the auto-reload value present in the Reload/Capture register, ARRC. The duty cycle of the PWM signal is controlled by the Compare Register, ARCP.

**AUTO-RELOAD TIMER** (Cont'd)

**Figure 23. . AR Timer Block Diagram**



**AUTO-RELOAD TIMER** (Cont'd)

It should be noted that the reload values will also affect the value and the resolution of the duty cycle of PWM output signal. To obtain a signal on ARTIMout, the contents of the ARCP register must be greater than the contents of the ARRC register.

The maximum available resolution for the ARTIMout duty cycle is:

Resolution = 1/[255-(ARRC)]

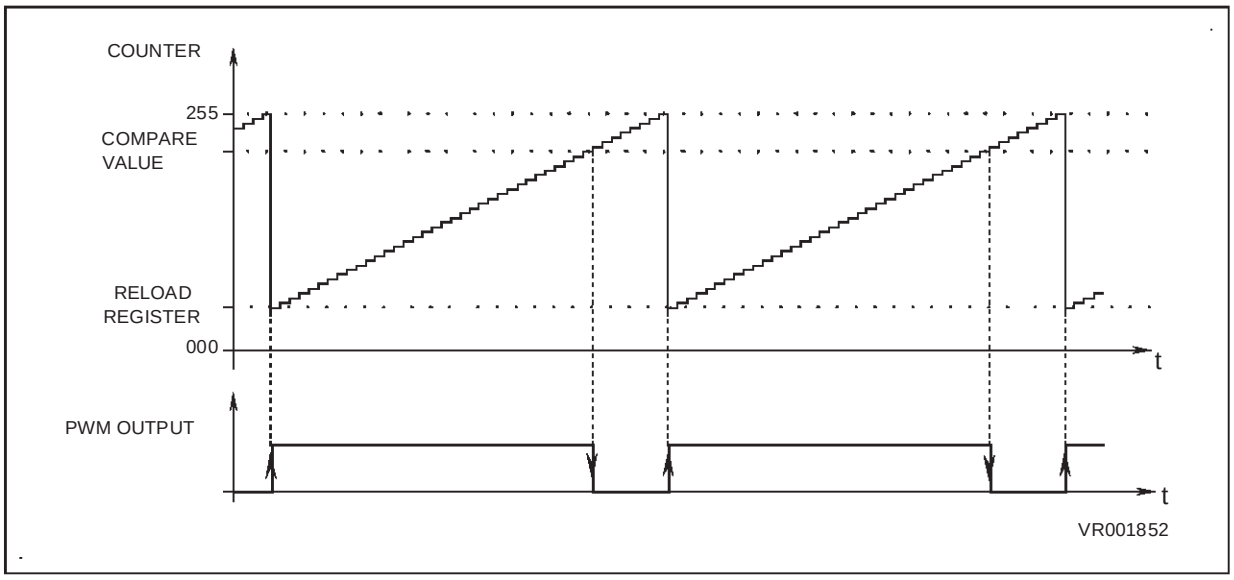
Where ARRC is the content of the Reload/Capture register. The compare value loaded in the Compare Register, ARCP, must be in the range from (ARRC) to 255.

The ARTC counter is initialized by writing to the ARRC register and by then setting the TCLD (Timer Load) and the TEN (Timer Clock Enable) bits in the Mode Control register, ARMC.

Enabling and selection of the clock source is controlled by the CC0, CC1, SL0 and SL1 bits in the Status Control Register, ARSC1. The prescaler division ratio is selected by the PS0, PS1 and PS2 bits in the ARSC1 register.

In Auto-reload Mode, any of the three available clock sources can be selected: Internal Clock, Internal Clock divided by 3 or the clock signal present on the ARTIMin pin.

**Figure 24. . Auto-reload Timer PWM Function**



**AUTO-RELOAD TIMER** (Cont'd)

**Capture Mode with PWM Generation** In this mode, the AR counter operates as a free running 8-bit counter fed by the prescaler output. The counter is incremented on every clock rising edge.

An 8-bit capture operation from the counter to the ARRC register is performed on every active edge on the ARTIMin pin, when enabled by Edge Control bits SL0, SL1 in the ARSC1 register. At the same time, the External Flag, EF, in the ARSC0 register is set and an external interrupt request is generated if the External Interrupt Enable bit, EIE, in the ARMC register, is set. The EF flag must be reset by user software.

Each ARTC overflow sets ARTIMout, while a match between the counter and ARCP (Compare Register) resets ARTIMout and sets the compare flag, CPF. A compare interrupt request is generated if the related compare interrupt enable bit, CPIE, is set. A PWM signal is generated on ARTIMout. The CPF flag must be reset by user software.

The frequency of the generated signal is determined by the prescaler setting. The duty cycle is determined by the ARCP register.

Initialization and reading of the counter are identical to the auto-reload mode (see previous description).

Enabling and selection of clock sources is controlled by the CC0 and CC1 bits in the AR Status Control Register, ARSC1.

The prescaler division ratio is selected by programming the PS0, PS1 and PS2 bits in the ARSC1 Register.

In Capture mode, the allowed clock sources are the internal clock and the internal clock divided by 3; the external ARTIMin input pin should not be used as a clock source.

**Capture Mode with Reset of counter and prescaler, and PWM Generation.** This mode is identical to the previous one, with the difference that a capture condition also resets the counter and the prescaler, thus allowing easy measurement of the time between two captures (for input period measurement on the ARTIMin pin).

**Load on External Input** The counter operates as a free running 8-bit counter fed by the prescaler. the count is incremented on every clock rising edge.

Each counter overflow sets the ARTIMout pin. A match between the counter and ARCP (Compare Register) resets the ARTIMout pin and sets the compare flag, CPF. A compare interrupt request is generated if the related compare interrupt enable bit, CPIE, is set. A PWM signal is generated on ARTIMout. The CPF flag must be reset by user software.

Initialization of the counter is as described in the previous paragraph. In addition, if the external ARTIMin input is enabled, an active edge on the input pin will copy the contents of the ARRC register into the counter, whether the counter is running or not.

**Notes**:

The allowed AR Timer clock sources are the following:

| AR Timer Mode | Clock Sources |
|---|---|
| Auto-reload mode | $f_{INT}$, $f_{INT/3}$, ARTIMin |
| Capture mode | $f_{INT}$, $f_{INT/3}$ |
| Capture/Reset mode | $f_{INT}$, $f_{INT/3}$ |
| External Load mode | $f_{INT}$, $f_{INT/3}$ |

The clock frequency should not be modified while the counter is counting, since the counter may be set to an unpredictable value. For instance, the multiplexer setting should not be modified while the counter is counting.

Loading of the counter by any means (by auto-reload, through ARLR, ARRC or by the Core) resets the prescaler at the same time.

Care should be taken when both the Capture interrupt and the Overflow interrupt are used. Capture and overflow are asynchronous. If the capture occurs when the Overflow Interrupt Flag, OVF, is high (between counter overflow and the flag being reset by software, in the interrupt routine), the External Interrupt Flag, EF, may be cleared simultaneusly without the interrupt being taken into account.

The solution consists in resetting the OVF flag by writing 06h in the ARSC0 register. The value of EF is not affected by this operation. If an interrupt has occured, it will be processed when the MCU exits from the interrupt routine (the second interrupt is latched).

**AUTO-RELOAD TIMER** (Cont'd)

### 4.3.3 AR Timer Registers

**AR Mode Control Register (ARMC)**

Address: D5h — Read/Write

Reset status: 00h

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| TCLD | TEN | PWMOE | EIE | CPIE | OVIE | ARMC1 | ARMC0 |

The AR Mode Control Register ARMC is used to program the different operating modes of the AR Timer, to enable the clock and to initialize the counter. It can be read and written to by the Core and it is cleared on system reset (the AR Timer is disabled).

Bit 7 = **TLCD**: *Timer Load Bit.* This bit, when set, will cause the contents of ARRC register to be loaded into the counter and the contents of the prescaler register, ARPSC, are cleared in order to initialize the timer before starting to count. This bit is write-only and any attempt to read it will yield a logical zero.

Bit 6 = **TEN***: Timer Clock Enable.* This bit, when set, allows the timer to count. When cleared, it will stop the timer and freeze ARPSC and ARTSC.

Bit 5 = **PWMOE**: *PWM Output Enable.* This bit, when set, enables the PWM output on the ARTIMout pin. When reset, the PWM output is disabled.

Bit 4 = **EIE**: *External Interrupt Enable.* This bit, when set, enables the external interrupt request. When reset, the external interrupt request is masked. If EIE is set and the related flag, EF, in the ARSC0 register is also set, an interrupt request is generated.

Bit 3 = **CPIE**: *Compare Interrupt Enable.* This bit, when set, enables the compare interrupt request. If CPIE is reset, the compare interrupt request is masked. If CPIE is set and the related flag, CPF, in the ARSC0 register is also set, an interrupt request is generated.

Bit 2 = **OVIE**: *Overflow Interrupt.* This bit, when set, enables the overflow interrupt request. If OVIE is reset, the compare interrupt request is masked. If OVIE is set and the related flag, OVF in the ARSC0 register is also set, an interrupt request is generated.

Bit 1-0 = **ARMC1-ARMC0**: *Mode Control Bits 1-0* These are the operating mode control bits. The following bit combinations will select the various operating modes:

| ARMC1 | ARMC0 | Operating Mode |
|---|---|---|
| 0 | 0 | Auto-reload Mode |
| 0 | 1 | Capture Mode |
| 1 | 0 | Capture Mode with Reset of ARTC and ARPSC |
| 1 | 1 | Load on External Edge Mode |

**AR Timer Status/Control Registers ARSC0 & ARSC1.** These registers contain the AR Timer status information bits and also allow the programming of clock sources, active edge and prescaler multiplexer setting.

ARSC0 register bits 0,1 and 2 contain the interrupt flags of the AR Timer. These bits are read normally. Each one may be reset by software. Writing a one does not affect the bit value.

**AR Status Control Register 0 (ARSC0)**

Address: D6h — Read/Clear

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| D7 | D6 | D5 | D4 | D3 | EF | CPF | OVF |

Bits 7-3 = **D7-D3**: *Unused*

Bit 2 = **EF**: *External Interrupt Flag.* This bit is set by any active edge on the external ARTIMin input pin. The flag is cleared by writing a zero to the EF bit.

Bit 1 = **CPF**: *Compare Interrupt Flag.* This bit is set if the contents of the counter and the ARCP register are equal. The flag is cleared by writing a zero to the CPF bit.

Bit 0 = **OVF**: *Overflow Interrupt Flag.* This bit is set by a transition of the counter from FFh to 00h (overflow). The flag is cleared by writing a zero to the OVF bit.

**AUTO-RELOAD TIMER** (Cont'd)

**AR Status Control Register 1(ARSC1)**

Address: D7h — Read/Write

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| PS2 | PS1 | PS0 | D4 | SL1 | SL0 | CC1 | CC0 |

Bist 7-5 = **PS2-PS0**: *Prescaler Division Selection Bits 2-0.* These bits determine the Prescaler division ratio. The prescaler itself is not affected by these bits. The prescaler division ratio is listed in the following table:

**Table 15. . Prescaler Division Ratio Selection**

| PS2 | PS1 | PS0 | ARPSC Division Ratio |
|---|---|---|---|
| 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 |

Bit 4 = **D4**: *Reserved.* Must be kept reset.

Bit 3-2 = **SL1-SL0**: *Timer Input Edge Control Bits 1-0.* These bits control the edge function of the Timer input pin for external synchronization. If bit SL0 is reset, edge detection is disabled; if set edge detection is enabled. If bit SL1 is reset, the AR Timer input pin is rising edge sensitive; if set, it is falling edge sensitive.

| SL1 | SL0 | Edge Detection |
|---|---|---|
| X | 0 | Disabled |
| 0 | 1 | Rising Edge |
| 1 | 1 | Falling Edge |

Bit 1-0 = **CC1-CC0**: *Clock Source Select Bit 1-0.* These bits select the clock source for the AR Timer through the AR Multiplexer. The programming of the clock sources is explained in the followingTable 16:

**Table 16. . Clock Source Selection.**

| CC1 | CC0 | Clock Source |
|---|---|---|
| 0 | 0 | $F_{int}$ |
| 0 | 1 | $F_{int}$ Divided by 3 |
| 1 | 0 | ARTIMin Input Clock |
| 1 | 1 | Reserved |

**AR Load Register ARLR** The ARLR load register is used to read or write the ARTC counter register "on the fly" (while it is counting). The ARLR register is not affected by system reset.

**AR Load Register (ARLR)**

Address: DBh — Read/Write

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

Bit 7-0 = **D7-D0**: *Load Register Data Bits.* These are the load register data bits.

**AR Reload/Capture Register**. The ARRC reload/capture register is used to hold the auto-reload value which is automatically loaded into the counter when overflow occurs.

**AR Reload/Capture (ARRC)**

Address: D9h — Read/Write

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

Bit 7-0 = **D7-D0**: *Reload/Capture Data Bits* These are the Reload/Capture register data bits.

**AR Compare Register**. The CP compare register is used to hold the compare value for the compare function.

**AR Compare Register (ARCP)**

Address: DAh — Read/Write

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

Bit 7-0 = **D7-D0**: *Compare Data Bits* These are the Compare register data bits.

### 4.4 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.

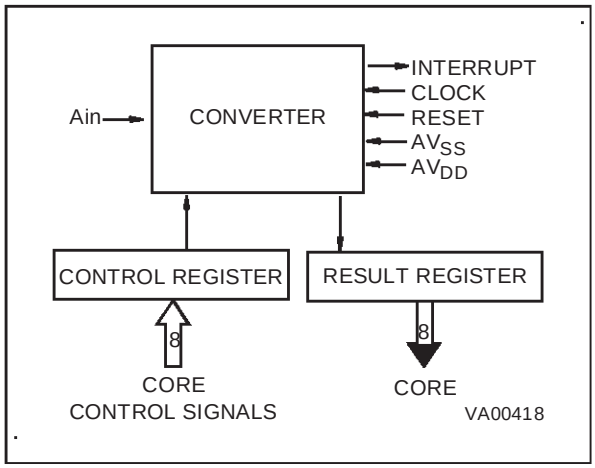
The STA bit is continuously scanned so that, if the user sets it to “1” while a previous conversion is in progress, a new conversion is started before completing the previous one. The start bit (STA) is a write only bit, any attempt to read it will show a logical “0”.

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 25. ADC Block Diagram**




#### 4.4.1 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).

**A/D CONVERTER** (Cont'd)

Since the ADC is on the same chip as the microprocessor, the user should not switch heavily loaded output signals during conversion, if high precision is required. Such switching will affect the supply voltages used as analog references.

The accuracy of the conversion depends on the quality of the power supplies ($V_{DD}$ and $V_{SS}$). The user must take special care to ensure a well regulated reference voltage is present on the $V_{DD}$ and $V_{SS}$ pins (power supply voltage variations must be less than 5V/ms). This implies, in particular, that a suitable decoupling capacitor is used at the $V_{DD}$ pin.

The converter resolution is given by::

$$\frac{V_{DD} - V_{SS}}{256}$$

*The Input voltage (Ain) which is to be converted must be constant for 1µs before conversion and remain constant during conversion.*

Conversion resolution can be improved if the power supply voltage ($V_{DD}$) to the microcontroller is lowered.

In order to optimise conversion resolution, the user can configure the microcontroller in WAIT mode, because this mode minimises noise disturbances and power supply variations due to output switching. Nevertheless, the WAIT instruction should be executed as soon as possible after the beginning of the conversion, because execution of the WAIT instruction may cause a small variation of the $V_{DD}$ voltage. The negative effect of this variation is minimized at the beginning of the conversion when the converter is less sensitive, rather than at the end of conversion, when the less significant bits are determined.

The best configuration, from an accuracy standpoint, is WAIT mode with the Timer stopped. Indeed, only the ADC peripheral and the oscillator are then still working. The MCU must be woken up from WAIT mode by the ADC interrupt at the end of the conversion. It should be noted that waking up the microcontroller could also be done using the Timer interrupt, but in this case the Timer will be working and the resulting noise could affect conversion accuracy.

**A/D Converter Control Register (ADCR)**

Address: 0D1h — Read/Write

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| EAI | EOC | STA | PDS | D3 | D2 | D1 | D0 |

Bit 7 = **EAI**: *Enable A/D Interrupt.* If this bit is set to "1" the A/D interrupt is enabled, when EAI=0 the interrupt is disabled.

Bit 6 = **EOC**: *End of conversion. Read Only.* This read only bit indicates when a conversion has been completed. This bit is automatically reset to "0" when the STA bit is written. If the user is using the interrupt option then this bit can be used as an interrupt pending bit. Data in the data conversion register are valid only when this bit is set to "1".

Bit 5 = **STA***: Start of Conversion. Write Only.* Writing a "1" to this bit will start a conversion on the selected channel and automatically reset to "0" the EOC bit. If the bit is set again when a conversion is in progress, the present conversion is stopped and a new one will take place. This bit is write only, any attempt to read it will show a logical zero.

Bit 4 = **PDS***: Power Down Selection.* This bit activates the A/D converter if set to "1". Writing a "0" to this bit will put the ADC in power down mode (idle mode).

Bit 3-0 = **D3-D0.** Not used

**A/D Converter Data Register (ADR)**

Address: 0D0h — Read only

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

Bit 7-0 = **D7-D0***: 8 Bit A/D Conversion Result.*

### 4.5 SERIAL PERIPHERAL INTERFACE (SPI)

The SPI peripheral is an optimized synchronous serial interface with programmable transmission modes and master/slave capabilities supporting a wide range of industry standard SPI specifications. The SPI interface may also implement asynchronous data transfer, in which case processor overhead is limited to data transfer from or to the shift register on an interrupt driven basis.

The SPI may be controlled by simple user software to perform serial data exchange with low-cost external memory, or with serially controlled peripherals to drive displays, motors or relays.

The SPI's shift register is simultaneously fed by the Sin pin and feeds the Sout pin, thus transmission and reception are essentially the same process. Suitable setting of the number of bits in the data frame can allow filtering of unwanted leading data bits in the incoming data stream.

The SPI comprises an 8-bit Data/Shift Register, DSR, a Divide register, DIV, a Mode Control Register MOD, and a Miscellaneous register, MISCR.

The SPI may be operated either in Master mode or in Slave mode.

Master mode is defined by the synchronous serial clock being supplied by the MCU, by suitably programming the clock divider (DIV register). Slave mode is defined by the serial clock being supplied externally on the SCK pin by the external Master device.

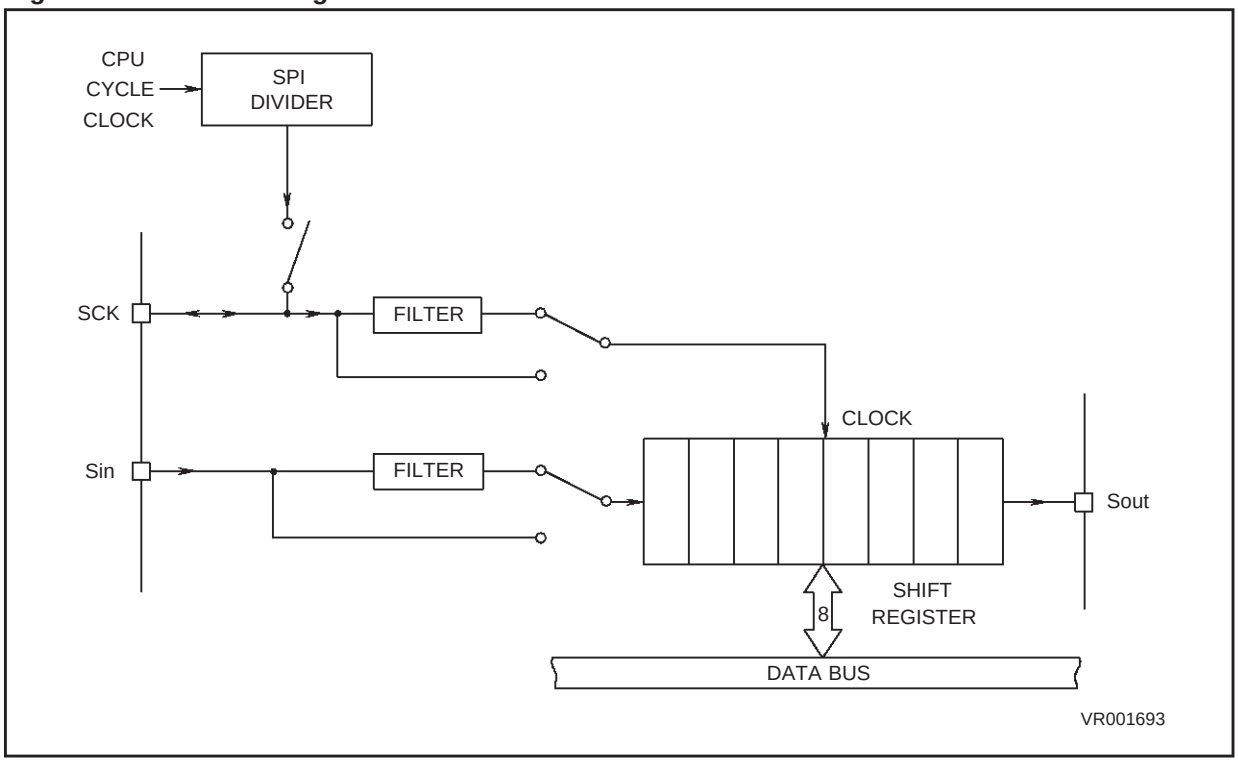
For maximum versatility the SPI may be programmed to sample data either on the rising or on the falling edge of SCK, with or without phase shift (clock Polarity and Phase selection).

The Sin, Sout and SCK signals are connected as alternate I/O pin functions.

For serial input operation, Sin must be configured as an input. For serial output operation, Sout is selected as an output by programming Bit 0 of the Miscellaneous Register: clearing this bit will set the pin as a standard I/O line, while setting the bit will select the Sout function.

An interrupt request may be associated with the end of a transmission or reception cycle; this is defined by programming the number of bits in the data frame and by enabling the interrupt. This request is associated with interrupt vector #2, and can be masked by programming the SPIE bit of the MOD register. Since the SPI interrupt is "ORed" with the port interrupt source, an interrupt flag bit is available in the DIV register allowing discrimination of the interrupt request.

**Figure 26. SPI Block Diagram**



**SERIAL PERIPHERAL INTERFACE SPI**(Cont'd)

### 4.5.1 SPI Registers

**SPI Mode Control Register (MOD)**

Address: E2h — Read/Write

Reset status: 00h

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| SPRUN | SPIE | CPHA | SPCLK | SPIN | SPSTRT | EFILT | CPOL |

The MOD register defines and controls the transmission modes and characteristics.

This register is read/write and all bits are cleared at reset. Setting SPSTRT = 1 and SPIN = 1 is not allowed and must be avoided.

Bit 7 = **SPRUN**: *SPI Run*. This bit is the SPI activity flag. This can be used in either transmit or receive modes; it is automatically cleared by the SPI at the end of a transmission or reception and generates an interrupt request (providing that the SPIE Interrupt Enable bit is set). The Core can stop transmission or reception at any time by resetting the SPRUN bit; this will also generate an interrupt request (providing that the SPIE Interrupt enable bit is set). The SPRUN bit can be used as a start condition parameter, in conjunction with the SPSTRT bit, when an external signal is present on the Sin pin. Note that a rising edge is then necessary to initiate reception; this may require external data inversion.

Bit 6 = **SPIE**: *SPI Interrupt Enable*. This bit is the SPI Interrupt Enable bit. If this bit is set the SPI interrupt (vector #2) is enabled, when SPIE is reset, the interrupt is disabled.

Bit 5 = **CPHA**: *Clock Phase Selection*. This bit selects the clock phase of the clock signal. If this bit is cleared to zero the normal state is selected; in this case Bit 7 of the data frame is present on Sout pin as soon as the SPI Shift Register is loaded. If this bit is set to one the shifted state' is selected; in this case Bit 7 of data frame is present on Sout pin on the first falling edge of Shift Register clock. The polarity relation and the division ratio between Shift Register and SPI base clock are also programmable; refer to DIV register and Timing Diagrams for more information.

Bit 4= **SPCLK**: *Base Clock Selection*

This bit selects the SPI base clock source. It is either the core cycle clock ($f_{INT}$/13) (Master mode) or the signal provided at SCK pin by an external device (slave mode). If SPCLK is low and the SCK pin is configured as input, the slave mode is selected. If SPCLK is high and the SCK pin is configured as output, the master mode is selected. In this case, the phase and polarity of the clock are controlled by CPOL and CPHA.

Bit 3 = **SPIN**: *Input Selection*

This bit enables the transfer of the data input to the Shift Register in receive mode. If this bit is cleared the Shift Register input is 0. If this bit is set, the Shift Register input corresponds to the input signal present on the Sin pin.

Bit 2 = **SPSTRT**: *Start Selection*

This bit selects the transmission or reception start mode. If SPSTRT is cleared, the internal start condition occurs as soon as the SPRUN bit is set. If SPSTRT is set, the internal start signal is the logic "AND" between the SPRUN bit and the external signal present on the Sin pin; in this case transmission will start after the latest of both signals providing that the first signal is still present (note that this implies a rising edge). After the transmission or recetion has been started, it will continue even if the Sin signal is reset.

Bit 1 = **EFILT**: *Enable Filters*

This bit enables/disables the input noise filters on the Sin and SCK inputs. If it is cleared to zero the filters are enabled, if set to one the filters are disabled. These noise filters will eliminate any pulse on Sin and SCK with a pulse width smaller than one to two Core clock periods (depending on the occurrence of the signal edge with respect to the Core clock edge). For example, if the ST6260B/65B runs with an 8MHz crystal, Sin and SCK will be delayed by 125 to 250ns.

Bit 0 = **CPOL**: *Clock Polarity*

This bit controls the relationship between the data on the Sin and Sout pins and SCK. The CPOL bit selects the clock edge which captures data and allows it to change state. It has the greatest impact on the first bit transmitted (the MSB) as it does (or does not) allow a clock transition before the first data capture edge.

Refer to the timing diagrams at the end of this section for additional details. These show the relationship between CPOL, CPHA and SCK, and indicate the active clock edges and strobe times.

**SERIAL PERIPHERAL INTERFACE SPI**(Cont'd)

**SPI DIV Register (DIV)**

Address: E1h — Read/Write

Reset status: 00h

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| SPINT | DOV6 | DIV5 | DIV4 | DIV3 | CD2 | CD1 | CD0 |

The SPIDIV register defines the transmission rate and frame format and contains the interrupt flag.

Bits CD0-CD2, DIV3-DIV6 are read/write while SPINT can be read and cleared only. Write access is not allowed if SPRUN in the MOD register is set.

Bit 7 = **SPINT**: *Interrupt Flag.* It is automatically set to one by the SPI at the end of a transmission or reception and an interrupt request can be generated depending on the state of the interrupt mask bit in the MOD control register. This bit is read-only and must be cleared by user software at the end of the interrupt service routine.

Bit 6-3 = **DIV6-DIV3**: *Burst Mode Bit Clock Period Selection.* Define the number of shift register bits that are transmitted or received in a frame. The available selections are listed in18. The normal maximum setting is 8 bits, since the shift register is 8 bits wide. Note that by setting a greater number of bits, in conjunction with the SPIN bit in the MOD register, unwanted data bits may be filtered from the data stream.

Bit 2-0 = **CD2-CD0**: *Base/Bit Clock Rate Selection*. Define the division ratio between the core clock ($f_{INT}$ divided by 13) and the clock supplied to the Shift Register in Master mode.

**Table 17. Base/Bit Clock Ratio Selection**

| CD2-CD0 | | | Divide Ratio (decimal) |
|---|---|---|---|
| 0 | 0 | 0 | Divide by 1 |
| 0 | 0 | 1 | Divide by 2 |
| 0 | 1 | 0 | Divide by 4 |
| 0 | 1 | 1 | Divide by 8 |
| 1 | 0 | 0 | Divide by 16 |
| 1 | 0 | 1 | Divide by 32 |
| 1 | 1 | 0 | Divide by 64 |
| 1 | 1 | 1 | Divide by 256 |

**Note**: For example, when an 8MHz CPU clock is used, asynchronous operation at 9600 Baud is possible (8MHz/13/64). Other Baud rates are available by proportionally selecting division factors depending on CPU clock frequency.

Data setup time on Sin is typically 250ns min, while data hold time is typically 50ns min.

**Table 18. Burst Mode Bit Clock Periods**

| DIV6-DIV3 | | | | Number of bits sent | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Reserved (not to be used) | |
| 0 | 0 | 0 | 1 | 1 | |
| 0 | 0 | 1 | 0 | 2 | |
| 0 | 0 | 1 | 1 | 3 | |
| 0 | 1 | 0 | 0 | 4 | |
| 0 | 1 | 0 | 1 | 5 | |
| 0 | 1 | 1 | 0 | 6 | |
| 0 | 1 | 1 | 1 | 7 | |
| 1 | 0 | 0 | 0 | 8 | |
| 1 | 0 | 0 | 1 | 9 | Refer to the description of the DIV6-DIV3 bits in the DIV Register |
| 1 | 0 | 1 | 0 | 10 | |
| 1 | 0 | 1 | 1 | 11 | |
| 1 | 1 | 0 | 0 | 12 | |
| 1 | 1 | 0 | 1 | 13 | |
| 1 | 1 | 1 | 0 | 14 | |
| 1 | 1 | 1 | 1 | 15 | |

**SPI Data/Shift Register (SPIDSR)**

Address: E0h — Read/Write

Reset status: XXh

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

SPIDSR is read/write, however write access is not allowed if the SPRUN bit of Mode Control register is set to one.

Data is sampled into SPDSR on the SCK edge determined by the CPOL and CPHA bits. The affect of these setting is shown in the following diagrams.

The Shift Register transmits and receives the Most Significant Bit first.

Bit 7-0 = **DSR7-DSR0**: *Data Bits.* These are the SPI shift register data bits.

**Miscellaneous Register (MISCR)**

Address: DDh — Write only

Reset status: xxxxxxxb

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | D0 |

Bit 7-1 = **D7-D1**: *Reserved.*

Bit 0 = **D0**: *Bit 0.* This bit, when set, selects the Sout pin as the SPI output line. When this bit is cleared, Sout acts as a standard I/O line.

**SERIAL PERIPHERAL INTERFACE SPI**(Cont'd)

### 4.6 SPI Timing Diagrams

**Figure 27. CPOL = 0 Clock Polarity Normal, CPHA = 0 Phase Selection Normal**

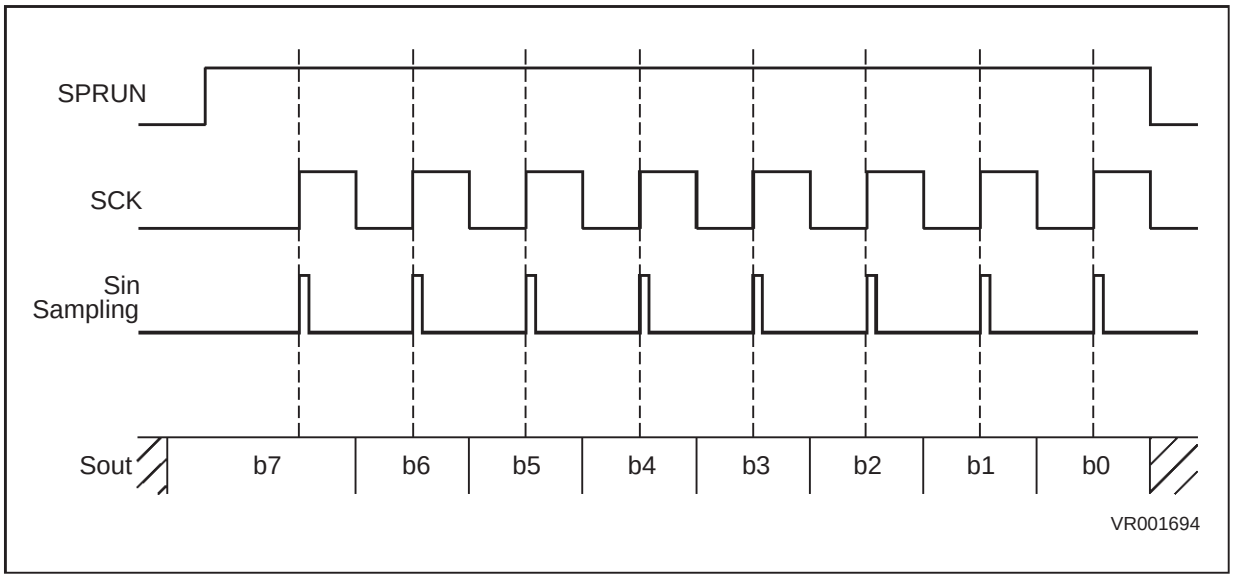
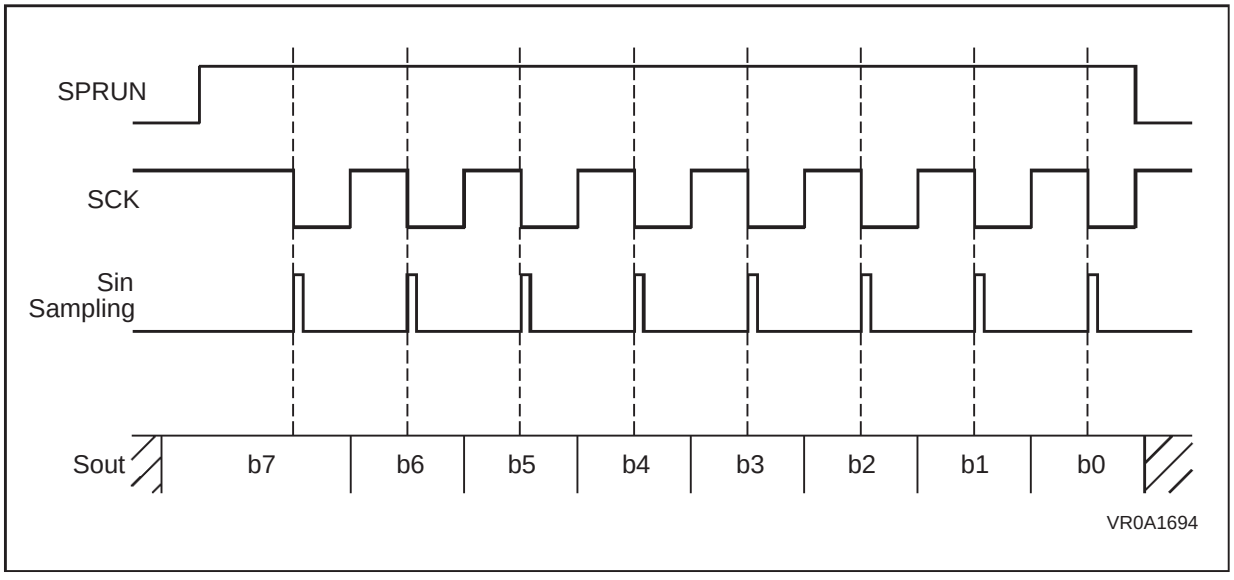



**Figure 28. CPOL = 1 Clock Polarity Inverted, CPHA = 0 Phase Selection Normal**



**SERIAL PERIPHERAL INTERFACE SPI**(Cont'd)

**Figure 29. CPOL = 0 Clock Polarity Normal, CPHA = 1 Phase Selection Shifted**

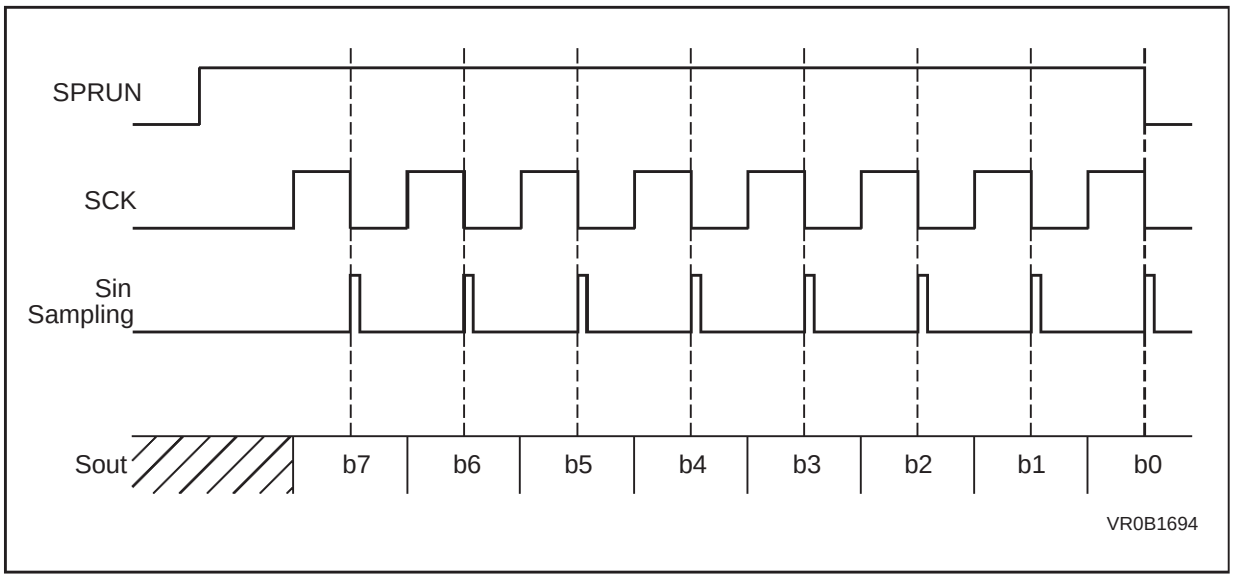
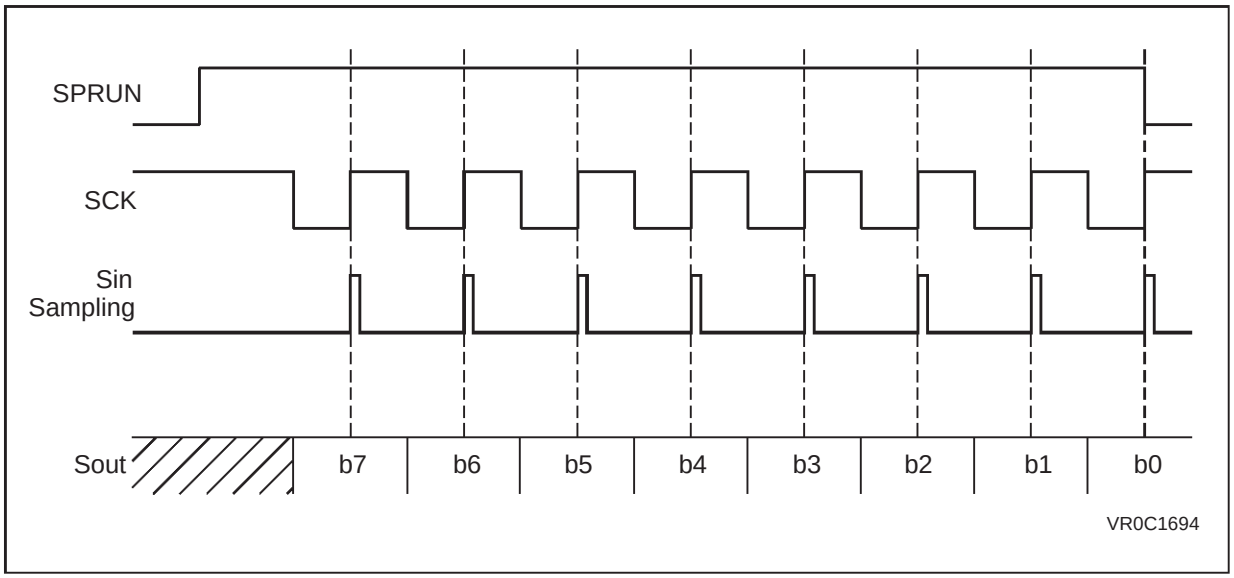



**Figure 30. CPOL = 1 Clock Polarity Inverted, CPHA = 1 Phase Selection Shifted**



# 5 SOFTWARE

## 5.1 ST6 ARCHITECTURE

The ST6 software has been designed to fully use the hardware in the most efficient way possible while keeping byte usage to a minimum; in short, to provide byte efficient programming capability. The ST6 core has the ability to set or clear any register or RAM location bit of the Data space with a single instruction. Furthermore, the program may branch to a selected address depending on the status of any bit of the Data space. The carry bit is stored with the value of the bit when the SET or RES instruction is processed.

## 5.2 ADDRESSING MODES

The ST6 core offers nine addressing modes, which are described in the following paragraphs. Three different address spaces are available: Program space, Data space, and Stack space. Program space contains the instructions which are to be executed, plus the data for immediate mode instructions. Data space contains the Accumulator, the X,Y,V and W registers, peripheral and Input/Output registers, the RAM locations and Data ROM locations (for storage of tables and constants). Stack space contains six 12-bit RAM cells used to stack the return addresses for subroutines and interrupts.

**Immediate**. In the immediate addressing mode, the operand of the instruction follows the opcode location. As the operand is a ROM byte, the immediate addressing mode is used to access constants which do not change during program execution (e.g., a constant used to initialize a loop counter).

**Direct**. In the direct addressing mode, the address of the byte which is processed by the instruction is stored in the location which follows the opcode. Direct addressing allows the user to directly address the 256 bytes in Data Space memory with a single two-byte instruction.

**Short Direct**. The core can address the four RAM registers X,Y,V,W (locations 80h, 81h, 82h, 83h) in the short-direct addressing mode. In this case, the instruction is only one byte and the selection of the location to be processed is contained in the opcode. Short direct addressing is a subset of the direct addressing mode. (Note that 80h and 81h are also indirect registers).

**Extended**. In the extended addressing mode, the 12-bit address needed to define the instruction is obtained by concatenating the four less significant bits of the opcode with the byte following the opcode. The instructions (JP, CALL) which use the extended addressing mode are able to branch to any address of the 4K bytes Program space.

An extended addressing mode instruction is two-byte long.

**Program Counter Relative**. The relative addressing mode is only used in conditional branch instructions. The instruction is used to perform a test and, if the condition is true, a branch with a span of -15 to +16 locations around the address of the relative instruction. If the condition is not true, the instruction which follows the relative instruction is executed. The relative addressing mode instruction is one-byte long. The opcode is obtained in adding the three most significant bits which characterize the kind of the test, one bit which determines whether the branch is a forward (when it is 0) or backward (when it is 1) branch and the four less significant bits which give the span of the branch (0h to Fh) which must be added or subtracted to the address of the relative instruction to obtain the address of the branch.

**Bit Direct**. In the bit direct addressing mode, the bit to be set or cleared is part of the opcode, and the byte following the opcode points to the address of the byte in which the specified bit must be set or cleared. Thus, any bit in the 256 locations of Data space memory can be set or cleared.

**Bit Test & Branch**. The bit test and branch addressing mode is a combination of direct addressing and relative addressing. The bit test and branch instruction is three-byte long. The bit identification and the tested condition are included in the opcode byte. The address of the byte to be tested follows immediately the opcode in the Program space. The third byte is the jump displacement, which is in the range of -127 to +128. This displacement can be determined using a label, which is converted by the assembler.

**Indirect**. In the indirect addressing mode, the byte processed by the register-indirect instruction is at the address pointed by the content of one of the indirect registers, X or Y (80h,81h). The indirect register is selected by the bit 4 of the opcode. A register indirect instruction is one byte long.

**Inherent**. In the inherent addressing mode, all the information necessary to execute the instruction is contained in the opcode. These instructions are one byte long.

### 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 19. 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 20. 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

**INSTRUCTION SET** (Cont'd)

**Conditional Branch**. The branch instructions achieve a branch in the program when the selected condition is met.

**Bit Manipulation Instructions**. These instructions can handle any bit in data space memory. One group either sets or clears. The other group (see Conditional Branch) performs the bit test branch operations.

**Control Instructions**. The control instructions control the MCU operations during program execution.

**Jump and Call.** These two instructions are used to perform long (12-bit) jumps or subroutines call inside the whole program space.

**Table 21. Conditional Branch Instructions**

| Instruction | Branch If | Bytes | Cycles | Flags | |
|---|---|---|---|---|---|
| | | | | Z | C |
| JRC e | C = 1 | 1 | 2 | * | * |
| JRNC e | C = 0 | 1 | 2 | * | * |
| JRZ e | Z = 1 | 1 | 2 | * | * |
| JRNZ e | Z = 0 | 1 | 2 | * | * |
| JRR b, rr, ee | Bit = 0 | 3 | 5 | * | Δ |
| JRS b, rr, ee | Bit = 1 | 3 | 5 | * | Δ |

**Notes:**
b. 3-bit address
e. 5 bit signed displacement in the range -15 to +16<F128M>
ee. 8 bit signed displacement in the range -126 to +129
rr. Data space register
Δ. Affected. The tested bit is shifted into carry.
*. Not Affected

**Table 22. Bit Manipulation Instructions**

| Instruction | Addressing Mode | Bytes | Cycles | Flags | |
|---|---|---|---|---|---|
| | | | | Z | C |
| SET b,rr | Bit Direct | 2 | 4 | * | * |
| RES b,rr | Bit Direct | 2 | 4 | * | * |

**Notes:**
b. 3-bit address;
rr. Data space register;
*. Not<M> Affected

**Table 23. Control Instructions**

| Instruction | Addressing Mode | Bytes | Cycles | Flags | |
|---|---|---|---|---|---|
| | | | | Z | C |
| NOP | Inherent | 1 | 2 | * | * |
| RET | Inherent | 1 | 2 | * | * |
| RETI | Inherent | 1 | 2 | Δ | Δ |
| STOP (1) | Inherent | 1 | 2 | * | * |
| WAIT | Inherent | 1 | 2 | * | * |

**Notes:**
1. This instruction is deactivated<N>and a WAIT is automatically executed instead of a STOP if the watchdog function is selected.
Δ. Affected
*. Not Affected

**Table 24. Jump & Call Instructions**

| Instruction | Addressing Mode | Bytes | Cycles | Flags | |
|---|---|---|---|---|---|
| | | | | Z | C |
| CALL abc | Extended | 2 | 4 | * | * |
| JP abc | Extended | 2 | 4 | * | * |

**Notes:**
abc. 12-bit address;
*. Not Affected

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

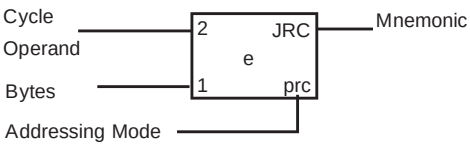
| LOW / HI | 0 0000 | 1 0001 | 2 0010 | 3 0011 | 4 0100 | 5 0101 | 6 0110 | 7 0111 | LOW / HI |
|---|---|---|---|---|---|---|---|---|---|
| 0 0000 | 2 JRNZ e 1 pcr | 4 CALL abc 2 ext | 2 JRNC e 1 pcr | 5 JRR b0,rr,ee 3 bt | 2 JRZ e 1 pcr | # | 2 JRC e 1 prc | 4 LD a,(x) 1 ind | 0 0000 |
| 1 0001 | 2 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 |



## Opcode Map Summary (Continued)

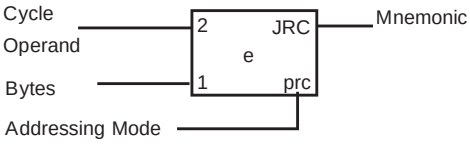
| LOW / HI | 8<br>1000 | 9<br>1001 | A<br>1010 | B<br>1011 | C<br>1100 | D<br>1101 | E<br>1110 | F<br>1111 | LOW / HI |
|---|---|---|---|---|---|---|---|---|---|
| **0**<br>**0000** | 2 JRNZ<br>e<br>1 pcr | 4 JP<br>abc<br>2 ext | 2 JRNC<br>e<br>1 pcr | 4 RES<br>b0,rr<br>2 b.d | 2 JRZ<br>e<br>1 pcr | 4 LDI<br>rr,nn<br>3 imm | 2 JRC<br>e<br>1 prc | 4 LD<br>a,(y)<br>1 ind | **0**<br>**0000** |
| **1**<br>**0001** | 2 JRNZ<br>e<br>1 pcr | 4 JP<br>abc<br>2 ext | 2 JRNC<br>e<br>1 pcr | 4 SET<br>b0,rr<br>2 b.d | 2 JRZ<br>e<br>1 pcr | 4 DEC<br>x<br>1 sd | 2 JRC<br>e<br>1 prc | 4 LD<br>a,rr<br>2 dir | **1**<br>**0001** |
| **2**<br>**0010** | 2 JRNZ<br>e<br>1 pcr | 4 JP<br>abc<br>2 ext | 2 JRNC<br>e<br>1 pcr | 4 RES<br>b4,rr<br>2 b.d | 2 JRZ<br>e<br>1 pcr | 4 COM<br>a | 2 JRC<br>e<br>1 prc | 4 CP<br>a,(y)<br>1 ind | **2**<br>**0010** |
| **3**<br>**0011** | 2 JRNZ<br>e<br>1 pcr | 4 JP<br>abc<br>2 ext | 2 JRNC<br>e<br>1 pcr | 4 SET<br>b4,rr<br>2 b.d | 2 JRZ<br>e<br>1 pcr | 4 LD<br>x,a<br>1 sd | 2 JRC<br>e<br>1 prc | 4 CP<br>a,rr<br>2 dir | **3**<br>**0011** |
| **4**<br>**0100** | 2 JRNZ<br>e<br>1 pcr | 4 JP<br>abc<br>2 ext | 2 JRNC<br>e<br>1 pcr | 4 RES<br>b2,rr<br>2 b.d | 2 JRZ<br>e<br>1 pcr | 2 RETI<br>1 inh | 2 JRC<br>e<br>1 prc | 4 ADD<br>a,(y)<br>1 ind | **4**<br>**0100** |
| **5**<br>**0101** | 2 JRNZ<br>e<br>1 pcr | 4 JP<br>abc<br>2 ext | 2 JRNC<br>e<br>1 pcr | 4 SET<br>b2,rr<br>2 b.d | 2 JRZ<br>e<br>1 pcr | 4 DEC<br>y<br>1 sd | 2 JRC<br>e<br>1 prc | 4 ADD<br>a,rr<br>2 dir | **5**<br>**0101** |
| **6**<br>**0110** | 2 JRNZ<br>e<br>1 pcr | 4 JP<br>abc<br>2 ext | 2 JRNC<br>e<br>1 pcr | 4 RES<br>b6,rr<br>2 b.d | 2 JRZ<br>e<br>1 pcr | 2 STOP<br>1 inh | 2 JRC<br>e<br>1 prc | 4 INC<br>(y)<br>1 ind | **6**<br>**0110** |
| **7**<br>**0111** | 2 JRNZ<br>e<br>1 pcr | 4 JP<br>abc<br>2 ext | 2 JRNC<br>e<br>1 pcr | 4 SET<br>b6,rr<br>2 b.d | 2 JRZ<br>e<br>1 pcr | 4 LD<br>y,a<br>1 sd | 2 JRC<br>e<br>1 prc | 4 INC<br>rr<br>2 dir | **7**<br>**0111** |
| **8**<br>**1000** | 2 JRNZ<br>e<br>1 pcr | 4 JP<br>abc<br>2 ext | 2 JRNC<br>e<br>1 pcr | 4 RES<br>b1,rr<br>2 b.d | 2 JRZ<br>e<br>1 pcr | # | 2 JRC<br>e<br>1 prc | 4 LD<br>(y),a<br>1 ind | **8**<br>**1000** |
| **9**<br>**1001** | 2 RNZ<br>e<br>1 pcr | 4 JP<br>abc<br>2 ext | 2 JRNC<br>e<br>1 pcr | 4 SET<br>b1,rr<br>2 b.d | 2 JRZ<br>e<br>1 pcr | 4 DEC<br>v<br>1 sd | 2 JRC<br>e<br>1 prc | 4 LD<br>rr,a<br>2 dir | **9**<br>**1001** |
| **A**<br>**1010** | 2 JRNZ<br>e<br>1 pcr | 4 JP<br>abc<br>2 ext | 2 JRNC<br>e<br>1 pcr | 4 RES<br>b5,rr<br>2 b.d | 2 JRZ<br>e<br>1 pcr | 4 RCL<br>a<br>1 inh | 2 JRC<br>e<br>1 prc | 4 AND<br>a,(y)<br>1 ind | **A**<br>**1010** |
| **B**<br>**1011** | 2 JRNZ<br>e<br>1 pcr | 4 JP<br>abc<br>2 ext | 2 JRNC<br>e<br>1 pcr | 4 SET<br>b5,rr<br>2 b.d | 2 JRZ<br>e<br>1 pcr | 4 LD<br>v,a<br>1 sd | 2 JRC<br>e<br>1 prc | 4 AND<br>a,rr<br>2 dir | **B**<br>**1011** |
| **C**<br>**1100** | 2 JRNZ<br>e<br>1 pcr | 4 JP<br>abc<br>2 ext | 2 JRNC<br>e<br>1 pcr | 4 RES<br>b3,rr<br>2 b.d | 2 JRZ<br>e<br>1 pcr | 2 RET<br>1 inh | 2 JRC<br>e<br>1 prc | 4 SUB<br>a,(y)<br>1 ind | **C**<br>**1100** |
| **D**<br>**1101** | 2 JRNZ<br>e<br>1 pcr | 4 JP<br>abc<br>2 ext | 2 JRNC<br>e<br>1 pcr | 4 SET<br>b3,rr<br>2 b.d | 2 JRZ<br>e<br>1 pcr | 4 DEC<br>w<br>1 sd | 2 JRC<br>e<br>1 prc | 4 SUB<br>a,rr<br>2 dir | **D**<br>**1101** |
| **E**<br>**1110** | 2 JRNZ<br>e<br>1 pcr | 4 JP<br>abc<br>2 ext | 2 JRNC<br>e<br>1 pcr | 4 RES<br>b7,rr<br>2 b.d | 2 JRZ<br>e<br>1 pcr | 2 WAIT<br>1 inh | 2 JRC<br>e<br>1 prc | 4 DEC<br>(y)<br>1 ind | **E**<br>**1110** |
| **F**<br>**1111** | 2 JRNZ<br>e<br>1 pcr | 4 JP<br>abc<br>2 ext | 2 JRNC<br>e<br>1 pcr | 4 SET<br>b7,rr<br>2 b.d | 2 JRZ<br>e<br>1 pcr | 4 LD<br>w,a<br>1 sd | 2 JRC<br>e<br>1 prc | 4 DEC<br>rr<br>2 dir | **F**<br>**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 |
| $I_O$ | Current Drain per Pin Excluding $V_{DD}$, $V_{SS}$ | ±10 | mA |
| $IV_{DD}$ | Total Current into $V_{DD}$ (source) | 50 | mA |
| $IV_{SS}$ | Total Current out of $V_{SS}$ (sink) | 50 | 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.

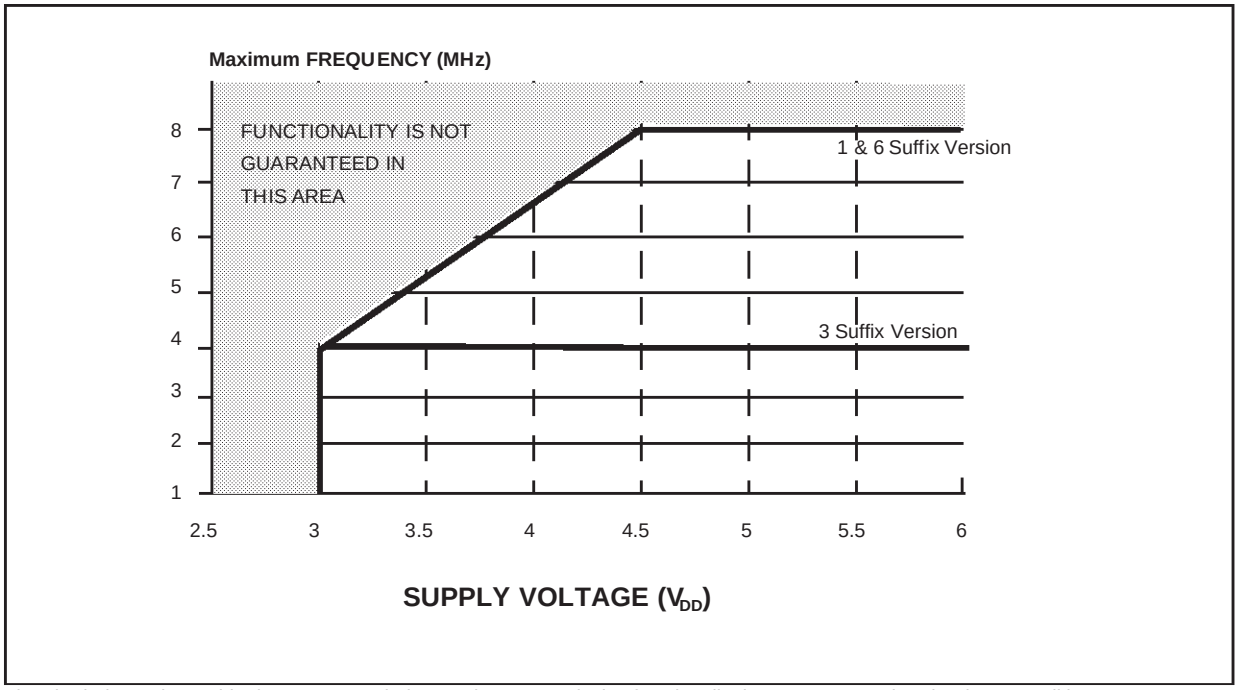
## 6.2 RECOMMENDED OPERATING CONDITIONS

| Symbol | Parameter | Test Conditions | Value | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min. | Typ. | Max. | |
| $T_A$ | Operating Temperature | 6 Suffix Version<br>1 Suffix Version<br>3 Suffix Version | -40<br>0<br>-40 | | 85<br>70<br>125 | °C |
| $V_{DD}$ | Operating Supply Voltage | $f_{OSC}$ = 2MHz<br>fosc= 8MHz | 3.0<br>4.5 | | 6.0<br>6.0 | V |
| $f_{OSC}$ | Oscillator Frequency[2)] | $V_{DD}$ = 3V<br>$V_{DD}$ = 4.5V, 1 & 6 Suffix<br>$V_{DD}$ = 4.5V, 3 Suffix | 0<br>0<br>0 | | 4.0<br>8.0<br>4.0 | MHz |
| $I_{INJ+}$ | Pin Injection Current (positive) | $V_{DD}$ = 4.5 to 5.5V | | | +5 | mA |
| $I_{INJ-}$ | Pin Injection Current (negative) | $V_{DD}$ = 4.5 to 5.5V | | | -5 | mA |

**Notes**:

1. Care must be taken in case of negative current injection, where adapted impedance must be respected on analog sources to not affect the A/D conversion. For a -1mA injection, a maximum 10 KΩ is recommended.
2. An oscillator frequency above 1MHz is recommended for reliable A/D results.

**Figure 31. Maximum Operating FREQUENCY (Fmax) Versus SUPPLY VOLTAGE ($V_{DD}$)**




The shaded area is outside the recommended operating range; device functionality is not guaranteed under these conditions.

### 6.3 DC ELECTRICAL CHARACTERISTICS

($T_A$ = -40 to +125°C unless otherwise specified)

| Symbol | Parameter | Test Conditions | Value | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min. | Typ. | Max. | |
| $V_{IL}$ | Input Low Level Voltage<br>All Input pins | | | | $V_{DD}$ x 0.3 | V |
| $V_{IH}$ | Input High Level Voltage<br>All Input pins | | $V_{DD}$ x 0.7 | | | V |
| $V_{Hys}$ | Hysteresis Voltage (1)<br>All Input pins | $V_{DD}$= 5V<br>$V_{DD}$= 3V | 0.2<br>0.2 | | | V |
| $V_{OL}$ | Low Level Output Voltage<br>All Output pins | $V_{DD}$= 5.0V; $I_{OL}$ = +10µA<br>$V_{DD}$= 5.0V; $I_{OL}$ = + 3mA | | | 0.1<br>0.8 | V |
| | Low Level Output Voltage<br>20 mA Sink I/O pins | $V_{DD}$= 5.0V; $I_{OL}$ = +10µA<br>$V_{DD}$= 5.0V; $I_{OL}$ = +7mA<br>$V_{DD}$= 5.0V; $I_{OL}$ = +15mA | | | 0.1<br>0.8<br>1.3 | |
| $V_{OH}$ | High Level Output Voltage<br>All Output pins | $V_{DD}$= 5.0V; $I_{OH}$ = -10µA<br>$V_{DD}$= 5.0V; $I_{OH}$ = -3.0mA | 4.9<br>3.5 | | | V |
| $R_{PU}$ | Pull-up Resistance | All Input pins | 40 | 100 | 200 | KΩ |
| | | RESET pin | 150 | 350 | 900 | |
| $I_{IL}$<br>$I_{IH}$ | Input Leakage Current<br>All Input pins but RESET | $V_{IN}$ = $V_{SS}$ (No Pull-Up configured)<br>$V_{IN}$ = $V_{DD}$ | | 0.1 | 1.0 | µA |
| | Input Leakage Current<br>RESET pin | $V_{IN}$ = $V_{SS}$<br>$V_{IN}$ = $V_{DD}$ | -8 | -16 | -30<br>10 | |
| $I_{DD}$ | Supply Current in RESET Mode | $V_{RESET}$=$V_{SS}$<br>$f_{OSC}$=8MHz | | | 3.5 | mA |
| | Supply Current in RUN Mode (2) | $V_{DD}$=5.0V $f_{INT}$=8MHz, $T_A$ < 85°C | | | 6.6 | mA |
| | Supply Current in WAIT Mode (3) | $V_{DD}$=5.0V $f_{INT}$=8MHz, $T_A$ < 85°C | | | 1.5 | mA |
| | Supply Current in STOP Mode (3) | $I_{LOAD}$=0mA<br>$V_{DD}$=5.0V | | | 20 | µA |

**Notes:**

(1) Hysteresis voltage between switching levels

(2) All peripherals running

(3) All peripherals in stand-by

### DC ELECTRICAL CHARACTERISTICS (Cont'd)

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

| Symbol | Parameter | Test Conditions | Value | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min. | Typ. | Max. | |
| $V_{OL}$ | Low Level Output Voltage<br>All Output pins | $V_{DD}$= 5.0V; $I_{OL}$ = +10µA<br>$V_{DD}$= 5.0V; $I_{OL}$ = + 5mA | | | 0.1<br>0.8 | V |
| | Low Level Output Voltage<br>20 mA Sink I/O pins | $V_{DD}$= 5.0V; $I_{OL}$ = +10µA<br>$V_{DD}$= 5.0V; $I_{OL}$ = +10mA<br>$V_{DD}$= 5.0V; $I_{OL}$ = +20mA | | | 0.1<br>0.8<br>1.3 | |
| $V_{OH}$ | High Level Output Voltage<br>All Output pins | $V_{DD}$= 5.0V; $I_{OH}$ = -10µA<br>$V_{DD}$= 5.0V; $I_{OH}$ = -5.0mA | 4.9<br>3.5 | | | V |
| $I_{DD}$ | Supply Current in STOP Mode (3) | $I_{LOAD}$=0mA<br>$V_{DD}$=5.0V | | | 10 | µA |

## 6.4 AC ELECTRICAL CHARACTERISTICS

($T_A$ = -40 to +125°C unless otherwise specified)

| Symbol | Parameter | Test Conditions | Value | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min. | Typ. | Max. | |
| $t_{REC}$ | Supply Recovery Time (1) | | 100 | | | ms |
| $T_{WR}$ | Minimum Pulse Width ($V_{DD}$ = 5V)<br>RESET pin<br>NMI pin | | <br>100<br>100 | | | ns |
| $T_{WEE}$ | EEPROM Write Time | $T_A$ = 25°C<br>$T_A$ = 85°C<br>$T_A$ = 125°C | | 5<br>10<br>20 | 10<br>20<br>30 | ms |
| Endurance (2) | EEPROM WRITE/ERASE Cycle | $Q_A$ $L_{OT}$ Acceptance | 300,000 | 1 million | | cycles |
| Retention | EEPROM Data Retention | $T_A$ = 55°C | 10 | | | years |
| $C_{IN}$ | Input Capacitance | All Inputs Pins | | | 10 | pF |
| $C_{OUT}$ | Output Capacitance | All Outputs Pins | | | 10 | pF |

**Notes**:
1. Period for which $V_{DD}$ has to be connected at 0V to allow internal Reset function at next power-up.
2. Sampled but not tested

## 6.5 A/D CONVERTER CHARACTERISTICS

($T_A$ = -40 to +125°C unless otherwise specified)

| Symbol | Parameter | Test Conditions | Value | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min. | Typ. | Max. | |
| Res | Resolution | | | 8 | | Bit |
| $A_{TOT}$ | Total Accuracy (1) (2) | $f_{OSC}$ > 1.2MHz<br>$f_{OSC}$ > 32kHz | | | ±2<br>±4 | LSB |
| $t_C$ | Conversion Time | $f_{OSC}$ = 8MHz, $T_A$ < 85°C<br>$f_{OSC}$ = 4MHz | | 70<br>140 | | µs |
| ZIR | Zero Input Reading | Conversion result when $V_{IN}$ = $V_{SS}$ | 00 | | | Hex |
| FSR | Full Scale Reading | Conversion result when $V_{IN}$ = $V_{DD}$ | | | FF | Hex |
| $AD_I$ | Analog Input Current During Conversion | $V_{DD}$= 4.5V | | | 1.0 | µA |
| $AC_{IN}$ | Analog Input Capacitance | | | 2 | 5 | pF |

**Notes**:
1. Noise at $AV_{DD}$, $AV_{SS}$ <10mV
2. With oscillator frequencies less than 1MHz, the A/D Converter accuracy is decreased.

### 6.6 TIMER CHARACTERISTICS

($T_A$ = -40 to +125°C unless otherwise specified)

| Symbol | Parameter | Test Conditions | Value | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min. | Typ. | Max. | |
| $f_{IN}$ | Input Frequency on TIMER Pin | | | | | MHz |
| $t_W$ | Pulse Width at TIMER Pin | $V_{DD}$ = 3.0V<br>$V_{DD}$ >4.5V | 1<br>125 | | | µs<br>ns |

### 6.7 SPI CHARACTERISTICS

($T_A$ = -40 to +125°C unless otherwise specified)

| Symbol | Parameter | Test Conditions | Value | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min. | Typ. | Max. | |
| $F_{CL}$ | Clock Frequency | Applied on Scl | | | 500 | kHz |
| $t_{SU}$ | Set-up Time | Applied on Sin | | 250 | | ns |
| $t_h$ | Hold Time | Applied onSin | | 50 | | ns |

### 6.8 ARTIMER ELECTRICAL CHARACTERISTICS

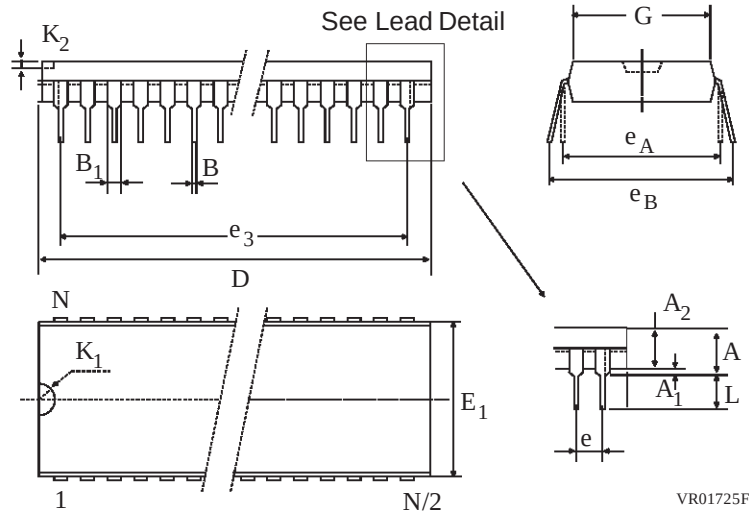
($T_A$ = -40 to +125°C unless otherwise specified)

| Symbol | Parameter | Test Conditions | Value | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min | Typ | Max | |
| $f_{IN}$ | Input Frequency on ARTIMin Pin | RUN and WAIT Modes | | | | MHz |
| | | STOP mode | | | 2 | |

# 7 GENERAL INFORMATION

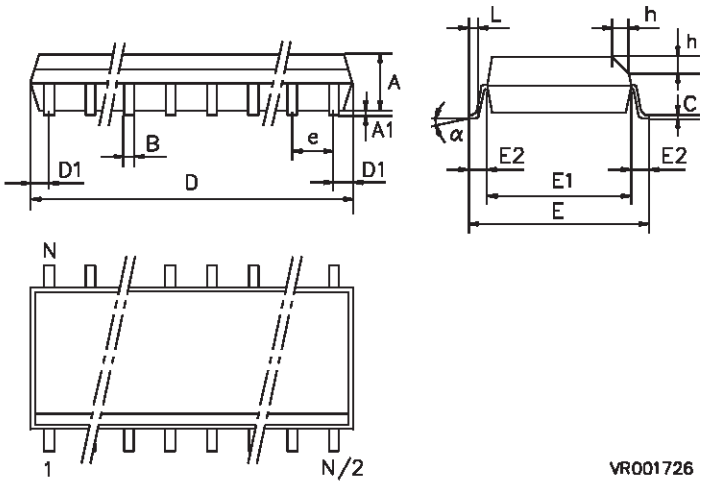
## 7.1 PACKAGE MECHANICAL DATA

**Figure 32. 28-Pin Plastic Dual In-Line Package, 600-mil Width**




| Dim. | mm | | | inches | | |
|---|---|---|---|---|---|---|
| | Min | Typ | Max | Min | Typ | Max |
| A | | 4.445 | | | 0.175 | |
| A1 | | 0.63 | | | 0.025 | |
| A2 | - | - | - | - | - | - |
| B | | 0.45 | | | 0.018 | |
| B1 | | 1.27 | | | 0.050 | |
| C | 0.23 | | 0.310 | 0.009 | | 0.012 |
| D | | | 37.34 | | | 1.470 |
| e3 | | 33.02 | | | 1.300 | |
| E1 | | | 14.10 | | | 0.555 |
| e | | 2.54 | | | 0.100 | |
| eA | | | | | | |
| eB | | 15.20 | 16.68 | | 0.598 | 0.657 |
| G | | | | | | |
| K1 | - | - | - | - | - | - |
| K2 | - | - | - | - | - | - |
| L | | 3.30 | | | 0.130 | |
| | Number of Pins | | | | | |
| N | 28 | | | | | |

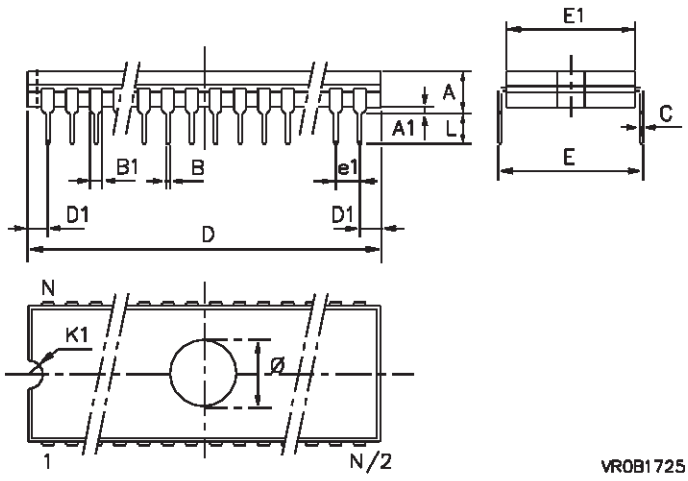
**Figure 33. 28-Pin Plastic Small Outline Package, 300-mil Width**




| Dim. | mm | | | inches | | |
|---|---|---|---|---|---|---|
| | Min | Typ | Max | Min | Typ | Max |
| A | 2.35 | | 2.65 | 0.093 | | 0.104 |
| A1 | 0.10 | | 0.30 | 0.004 | | 0.012 |
| B | 0.33 | | 0.51 | 0.013 | | 0.020 |
| C | 0.23 | | 0.32 | 0.009 | | 0.013 |
| D | 17.70 | | 18.10 | 0.697 | | 0.713 |
| D1 | - | - | - | - | - | - |
| E | 10.00 | | 10.65 | 0.394 | | 0.419 |
| E1 | 7.40 | | 7.60 | 0.291 | | 0.299 |
| E2 | - | - | - | - | - | - |
| e | | 1.27 | | | 0.050 | |
| h | 0.25 | | 0.75 | 0.010 | | 0.030 |
| L | 0.40 | | 1.27 | 0.016 | | 0.050 |
| ∝ | 0° | | 8° | | | |
| | Number of Pins | | | | | |
| N | 28 | | | | | |

**PACKAGE MECHANICAL DATA** (Cont'd)

**Figure 34. 28-Ceramic Dual In Line Package, 600-mil Width**




| Dim. | mm | | | inches | | |
|---|---|---|---|---|---|---|
| | Min | Typ | Max | Min | Typ | Max |
| A | | | 5.71 | | | 0.225 |
| A1 | 0.50 | | 1.78 | 0.020 | | 0.070 |
| B | 0.40 | | 0.55 | 0.016 | | 0.022 |
| B1 | 1.17 | | 1.42 | 0.046 | | 0.056 |
| C | 0.22 | | 0.31 | 0.009 | | 0.012 |
| D | | | 38.10 | | | 1.50 |
| D1 | 1.52 | | 2.49 | 0.060 | | 0.098 |
| E | - | - | - | - | - | - |
| E1 | 13.05 | | 13.36 | 0.514 | | 0.526 |
| L | 3.00 | | | 0.118 | | |
| e1 | 2.29 | | 2.79 | 0.090 | | 0.110 |
| ∅ | 6.86 | | 7.36 | 0.270 | | 0.290 |
| | Number of Pins | | | | | |
| N | 28 | | | | | |

## 7.2 .ORDERING INFORMATION

**Table 25. OTP/EPROM VERSION ORDERING INFORMATION**

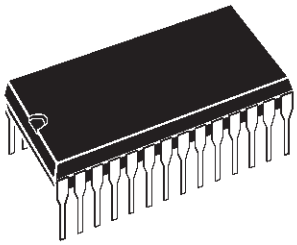
| Sales Type | Program Memory (Bytes) | EEPROM (Bytes) | Temperature Range | Package |
|---|---|---|---|---|
| ST62E65BF1 | 3884 (EPROM) | 128 | 0 to +70°C | CDIP20 |
| ST62T55BB6<br>ST62T55BB3 | 3884 (OTP) | None | -40 to + 85°C<br>-40 to + 125°C | PDIP28 |
| ST62T55BM6<br>ST62T55BM3 | | | -40 to + 85°C<br>-40 to + 125°C | PSO28 |
| ST62T65BB6<br>ST62T65BB3 | 3884 (OTP) | 128 | -40 to + 85°C<br>-40 to + 125°C | PDIP28 |
| ST62T65BM6<br>ST62T65BM3 | | | -40 to + 85°C<br>-40 to + 125°C | PSO28 |

SGS-THOMSON MICROELECTRONICS

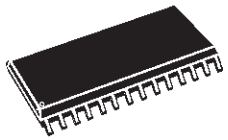
# ST62P55B
# ST62P65B

## 8-BIT FASTROM MCUs WITH A/D CONVERTER, AUTO-RELOAD TIMER, EEPROM AND SPI

- 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: 128 bytes
- Data EEPROM: 128 bytes (none on ST6255B)
- 21 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
- 8 I/O lines can sink up to 20mA to drive LEDs or TRIACs directly
- 8-bit Timer/Counter with 7-bit programmable prescaler
- 8-bit Auto-reload Timer with 7-bit programmable prescaler (AR Timer)
- Digital Watchdog
- 8-bit A/D Converter with 13 analog inputs
- 8-bit Synchronous Peripheral Interface (SPI)
- On-chip Clock oscillator can be driven by Quartz Crystal Ceramic resonator or RC network
- User configurable Power-on Reset
- One external Non-Maskable Interrupt
- ST626x-EMU2 Emulation and Development System (connects to an MS-DOS PC via a parallel port)



**PDIP28**



**PSO28**

(See end of Datasheet for Ordering Information)

**DEVICE SUMMARY**

| DEVICE | ROM (Bytes) | EEPROM |
|---|---|---|
| ST62P55B | 3884 | - |
| ST62P65B | 3884 | 128 |

# 1 GENERAL DESCRIPTION

## 1.1 INTRODUCTION

The ST62P55B and ST62P65B are the **F**actory **A**dvanced **S**ervice **T**echnique ROM (FASTROM) versions of ST62T55B and ST62T65B OTP devices.

They offer the same functionality as OTP devices, selecting as FASTROM options the options defined in the programmable option byte of the OTP version.

## 1.2 ORDERING INFORMATION

The following section deals with the procedure for transfer of customer codes to SGS-THOMSON.

### 1.2.1 Transfer of Customer Code

Customer code is made up of the ROM contents and the list of the selected FASTROM options. The ROM contents are to be sent on diskette, or by electronic means, with the hexadecimal file generated by the development tool. All unused bytes must be set to FFh.

The selected options are communicated to SGS-THOMSON using the correctly filled OPTION LIST appended.

### 1.2.2 Listing Generation and Verification

When SGS-THOMSON receives the user's ROM contents, a computer listing is generated from it. This listing refers exactly to the ROM contents and options which will be used to produce the specified MCU. The listing is then returned to the customer who must thoroughly check, complete, sign and return it to SGS-THOMSON. The signed listing forms a part of the contractual agreement for the production of the specific customer MCU.

The SGS-THOMSON Sales Organization will be pleased to provide detailed information on contractual points.

**Table 1. ROM Memory Map for ST62P55B/P65B**

| 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 |

**Table 2. FASTROM version Ordering Information**

| Sales Type | ROM | EEPROM (Bytes) | Temperature Range | Package |
|---|---|---|---|---|
| ST62P55BB1/XXX<br>ST62P55BB6/XXX<br>ST62P55BB3/XXX (*) | 3884 Bytes | None | 0 to +70°C<br>-40 to + 85°C<br>-40 to + 125°C | PDIP28 |
| ST62P55BM1/XXX<br>ST62P55BM6/XXX<br>ST62P55BM3/XXX (*) | | | 0 to +70°C<br>-40 to + 85°C<br>-40 to + 125°C | PSO28 |
| ST62P65BB1/XXX<br>ST62P65BB6/XXX<br>ST62P65BB3/XXX (*) | | 128 | 0 to +70°C<br>-40 to + 85°C<br>-40 to + 125°C | PDIP28 |
| ST62P65BM1/XXX<br>ST62P65BM6/XXX<br>ST62P65BM3/XXX (*) | | | 0 to +70°C<br>-40 to + 85°C<br>-40 to + 125°C | PSO28 |

(*) Advanced information

**ST62P55B and ST62P65B FASTROMMICROCONTROLLER OPTION LIST**

Customer . . . . . .
Address . . . . . .
. . . . . .
Contact . . . . . .
Phone No . . . . . .
Reference . . . . . .

SGS-THOMSON Microelectronics references

Device: [ ] ST62P55B [ ] ST62P65B
Package: [ ] Dual in Line Plastic [ ] Small Outline Plastic with condionning:
[ ] Standard (Stick)
[ ] Tape & Reel
Temperature Range: [ ] 0°C to + 70°C [ ] - 40°C to + 85°C

Oscillator Source Selection: [ ] Crystal Quartz/Ceramic resonator (Default)
[ ] RC Network
Watchdog Selection: [ ] Software Activation (STOP mode available)
[ ] Hardware Activation (no STOP mode)
Power on Reset Delay [ ] 32768 cycle delay
[ ] 2048 cycle delay
PB0..PB1 Pull-Up at RESET [ ] Enabled
[ ] Disabled
PB2..PB3 Pull-Up at RESET [ ] Enabled
[ ] Disabled
Readout Protection: [ ] Standard
[ ] Enabled
External STOP Mode Control [ ] Enabled
[ ] Disabled (Default)

Comments : Supply Operating Range in the application:
Oscillator Fequency in the application:

Notes . . . . . .
Signature . . . . . .
Date . . . . . .

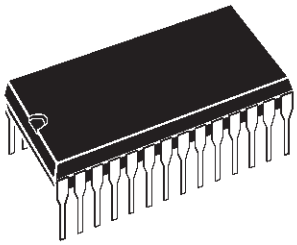
**Notes:**

SGS-THOMSON MICROELECTRONICS

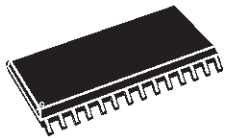
# ST6255B
# ST6265B

## 8-BIT ROM MCUs WITH A/D CONVERTER, AUTO-RELOAD TIMER, EEPROM AND SPI

- 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: 128 bytes
- Data EEPROM: 128 bytes (none on ST6255B)
- 21 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
- 8 I/O lines can sink up to 20mA to drive LEDs or TRIACs directly
- 8-bit Timer/Counter with 7-bit programmable prescaler
- 8-bit Auto-reload Timer with 7-bit programmable prescaler (AR Timer)
- Digital Watchdog
- 8-bit A/D Converter with 13 analog inputs
- 8-bit Synchronous Peripheral Interface (SPI)
- On-chip Clock oscillator can be driven by Quartz Crystal Ceramic resonator or RC network
- User configurable Power-on Reset
- One external Non-Maskable Interrupt
- ST626x-EMU2 Emulation and Development System (connects to an MS-DOS PC via a parallel port)



**PDIP28**



**PSO28**

(See end of Datasheet for Ordering Information)

**DEVICE SUMMARY**

| DEVICE | FASTROM (Bytes) | EEPROM |
|---|---|---|
| ST6255B | 3884 | - |
| ST6265B | 3884 | 128 |

# 1 GENERAL DESCRIPTION

## 1.1 INTRODUCTION

The ST6255B and ST6265B is mask programmed ROM version of ST62T55B and ST62T65B OTP devices.

They offer the same functionality as OTP devices, selecting as ROM options the options defined in the programmable option byte of the OTP version.

## 1.2 ROM READOUT PROTECTION

If the ROM READOUT PROTECTION option is selected, a protection fuse can be blown to prevent any access to the program memory content.

In case the user wants to blow this fuse, high voltage must be applied on the TEST pin.

**Figure 1. Programming wave form**

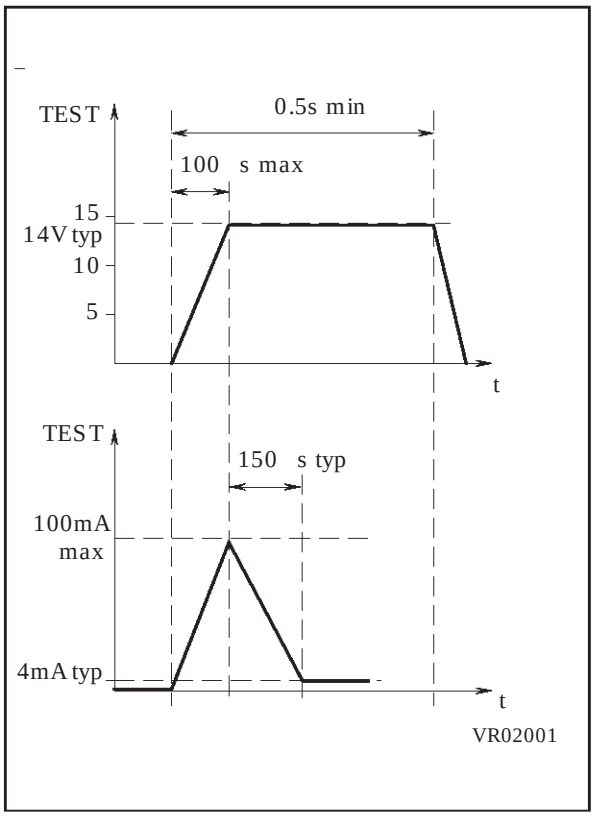
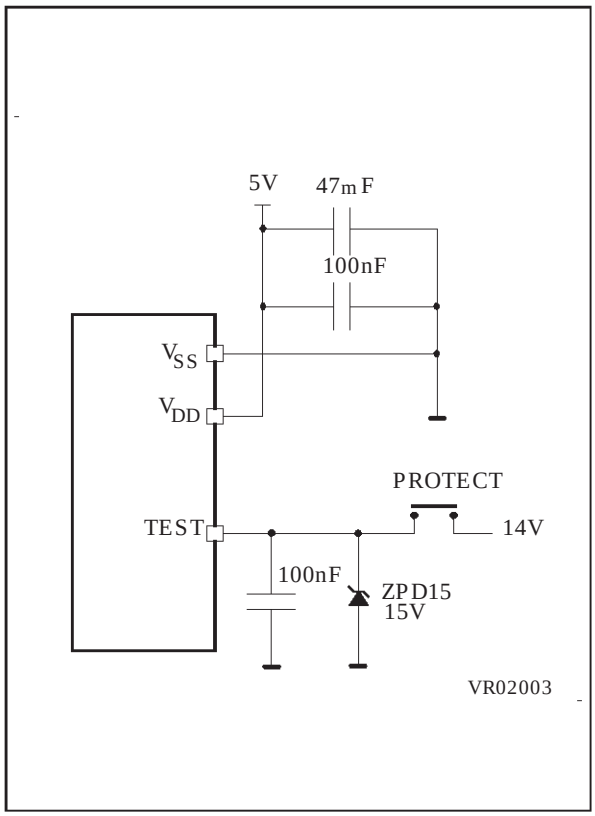



**Figure 2. Programming Circuit**




Note: ZPD15 is used for overvoltage protection

**ST6255B and ST6265B MICROCONTROLLER OPTION LIST**

Customer . . . . . .
Address . . . . . .
. . . . . .
Contact . . . . . .
Phone No . . . . . .
Reference . . . . . .

SGS-THOMSON Microelectronics references

Device: [ ] ST6255B [ ] ST6265B

Package: [ ] Dual in Line Plastic [ ] Small Outline Plastic with condionning:
[ ] Standard (Stick)
[ ] Tape & Reel

Temperature Range: [ ] 0°C to + 70°C [ ] - 40°C to + 85°C

Special Marking: [ ] No [ ] Yes "_ _ _ _ _ _ _ _ _ _ _ _ "

Authorized characters are letters, digits, '.', '-', '/' and spaces only.

Maximum character count: DIP28: 10
SO28: 8

Oscillator Source Selection: [ ] Crystal Quartz/Ceramic resonator (Default)
[ ] RC Network

Watchdog Selection: [ ] Software Activation (STOP mode available)
[ ] Hardware Activation (no STOP mode)

Power on Reset Delay [ ] 32768 cycle delay
[ ] 2048 cycle delay

PB0..PB1 Pull-Up at RESET [ ] Enabled
[ ] Disabled

PB2..PB3 Pull-Up at RESET [ ] Enabled
[ ] Disabled

ROM Readout Protection: [ ] Standard (Fuse cannot be blown)
[ ] Enabled (Fuse can be blown by the customer)
Note: No part is delivered with protected ROM.
The fuse must be blown for protection to be effective.

External STOP Mode Control [ ] Enabled
[ ] Disabled (Default)

Comments : Supply Operating Range in the application:
Oscillator Fequency in the application:

Notes . . . . . .
Signature . . . . . .
Date . . . . . .

### 1.3 ORDERING INFORMATION

The following section deals with the procedure for transfer of customer codes to SGS-THOMSON.

#### 1.3.1 Transfer of Customer Code

Customer code is made up of the ROM contents and the list of the selected mask options. The ROM contents are to be sent on diskette, or by electronic means, with the hexadecimal file generated by the development tool. All unused bytes must be set to FFh.

The selected mask options are communicated to SGS-THOMSON using the correctly filled OPTION LIST appended.

#### 1.3.2 Listing Generation and Verification

When SGS-THOMSON receives the user's ROM contents, a computer listing is generated from it. This listing refers exactly to the mask which will be used to produce the specified MCU. The listing is then returned to the customer who must thoroughly check, complete, sign and return it to SGS-THOMSON. The signed listing forms a part of the contractual agreement for the creation of the specific customer mask.

The SGS-THOMSON Sales Organization will be pleased to provide detailed information on contractual points.

**Table 1. ROM Memory Map for ST6255B/65B**

| 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 |

**Table 2. ROM version Ordering Information**

| Sales Type | ROM | EEPROM (Bytes) | Temperature Range | Package |
|---|---|---|---|---|
| ST6255BB1/XXX<br>ST6255BB6/XXX<br>ST6255BB3/XXX | 3884 Bytes | None | 0 to +70°C<br>-40 to + 85°C<br>-40 to + 125°C | PDIP28 |
| ST6255BM1/XXX<br>ST6255BM6/XXX<br>ST6255BM3/XXX | | | 0 to +70°C<br>-40 to + 85°C<br>-40 to + 125°C | PSO28 |
| ST6265BB1/XXX<br>ST6265BB6/XXX<br>ST6265BB3/XXX | | 128 | 0 to +70°C<br>-40 to + 85°C<br>-40 to + 125°C | PDIP28 |
| ST6265BM1/XXX<br>ST6265BM6/XXX<br>ST6265BM3/XXX | | | 0 to +70°C<br>-40 to + 85°C<br>-40 to + 125°C | PSO28 |

Information furnished is believed to be accurate and reliable. However, SGS-THOMSON Microelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of SGS-THOMSON Microelectronics. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. SGS-THOMSON Microelectronics products are not authorized for use as critical components in life support devices or systems without the express written approval of SGS-THOMSON Microelectronics.

©1998 SGS-THOMSON Microelectronics - All rights reserved.

Purchase of $I^2C$ Components by SGS-THOMSON Microelectronics conveys a license under the Philips $I^2C$ Patent. Rights to use these components in an $I^2C$ system is granted provided that the system conforms to the $I^2C$ Standard Specification as defined by Philips.

SGS-THOMSON Microelectronics Group of Companies
Australia - Brazil - Canada - China - France - Germany - Italy - Japan - Korea - Malaysia - Malta - Morocco - The Netherlands - Singapore Spain - Sweden - Switzerland - Taiwan - Thailand - United Kingdom - U.S.A.