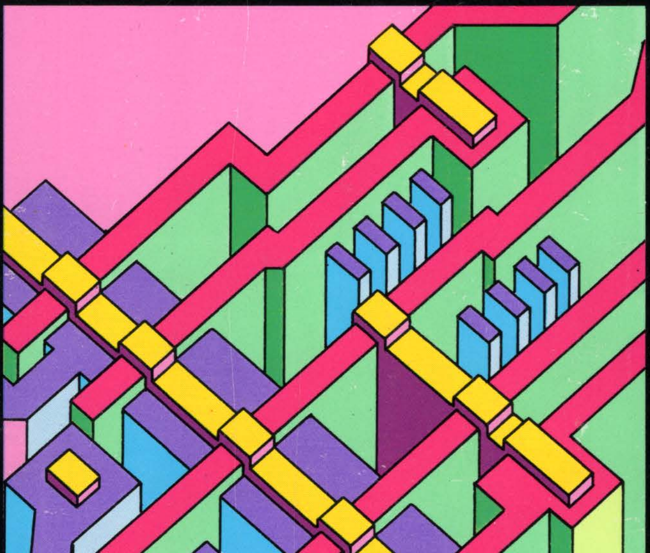
ST6210/ST6215
ST6220/ST6225

USER MANUAL



SGS-THOMSON
MICROELECTRONICS

10027

ST6210/15/20/25

USER MANUAL

JUNE 1991

USE IN LIFE SUPPORT DEVICES OR SYSTEMS MUST BE EXPRESSLY AUTHORIZED.

SGS-THOMSON PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF SGS-THOMSON Microelectronics.
As used herein :

1. Life support devices or systems are those which (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided with the product, can be reasonably expected to result in significant injury to the user.

2. A critical component is any component of a life support device or system whose failure to perform can reasonably be expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

TABLE OF CONTENTS

DATA SHEETS

GENERAL INDEX

DATA SHEETS

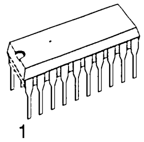
SGS-THOMSON
MICROELECTRONICS

ST6210,E10-ST6215,E15
ST6220,E20-ST6225,E25

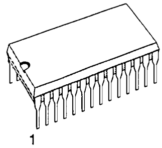
8 BIT HCMOS MCUs WITH A/D CONVERTER

- 8 bit Architecture
- Static HCMOS Operation
- 3.0 to 6.0V Supply Operating Range
- 8 MHz Clock
- Run, Wait & Stop Modes
- 5 different interrupt vectors
- Look-up table capability in ROM
- User ROM: 1828 bytes (ST6210,15)
- User ROM: 3876 bytes (ST6220,25)
- Reserved ROM: 220 bytes
- Data ROM: User selectable size (in program ROM)
- Data RAM: 64 bytes
- PDIP20, PSO20 (ST6210,20) packages
- PDIP28, PSO28 (ST6215,25) packages
- 12/20 fully software programmable I/O as:
  - Input with pull-up resistor
  - Input without Pull-up resistor
  - Input with interrupt generation
  - Open-drain or push-pull outputs
  - Analog Inputs
- 4 I/O lines can sink up to 10mA for direct led driving
- 8 bit counter with a 7 bit programmable prescaler (Timer)
- Digital Watchdog/Timer
- 8 bit A/D Converter with up to 8 (ST6210, ST6220) and up to 16 (ST6215, ST6225) analog inputs
- One external not maskable interrupt
- On-chip clock oscillator
- Power-on Reset
- Byte efficient instruction set
- Bit test and jump instructions
- Wait, Stop and Bit Manipulation instructions
- True LIFO 6 level stack
- 9 powerful addressing modes
- The accumulator, the X, Y, V & W registers, the port and peripherals data/control registers are addressed in the Data Space as RAM locations
- The development tool of the ST621X, ST622X microcontrollers consists of the ST621X-EMU emulation and development system connected via a standard RS232 serial line to an MS-DOS Personal Computer

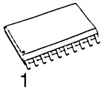
Device Summary page 4/45



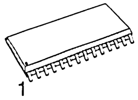
PDIP20



PDIP28



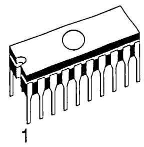
PSO20



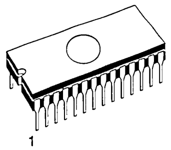
PSO28

(Ordering Information at the end of the datasheet)

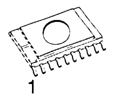
EPROM PACKAGES



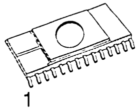
FDIP20W



FDIP28W



CSO20W



CSO28W

ST62E10, E15, E20 and E25 are the EPROM versions
ST62T10, T15, T20 and T25 are the OTP versions

Figure 1. ST6210,62E10, ST6220,62E20 Pin Configuration

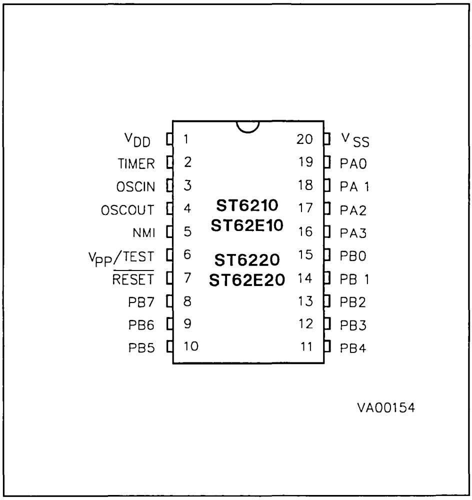


Figure 2. ST6215,62E15, ST6225,62E25 Pin Configuration

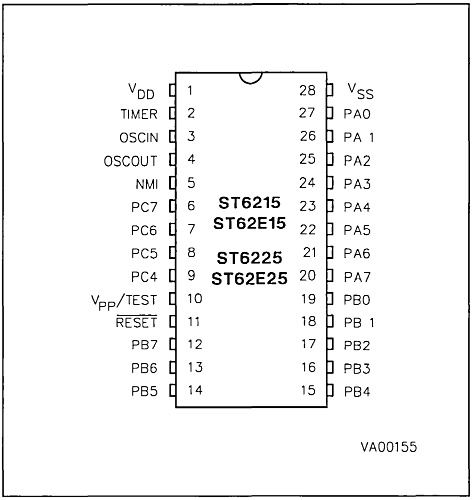
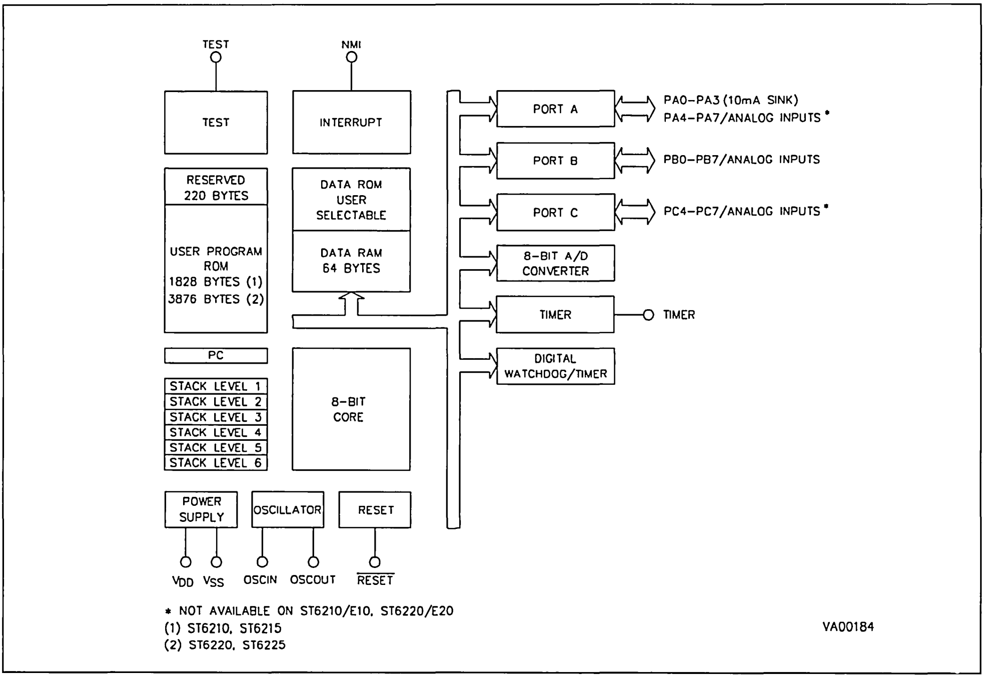


Figure 3. ST6210,15,20,25 Block Diagram



* NOT AVAILABLE ON ST6210/E10, ST6220/E20
(1) ST6210, ST6215
(2) ST6220, ST6225

GENERAL DESCRIPTION

The ST6210,E10, ST6215,E15, ST6220,E20 and ST6225,E25 microcontrollers are members of the 8-bit HCMOS ST62XX family, a series of devices oriented to low-medium complexity applications. All ST62XX members are based on a building block approach: a common core is surrounded by a combination of on-chip peripherals (macrocells). The macrocells of the ST6210,E10, ST6215,E15, ST6220,E20 and ST6225,E25 are: the Timer peripheral that includes an 8-bit counter with a 7-bit software programmable prescaler (Timer), the 8-bit A/D Converter with up to 8 (ST6210,E10, ST6220,E20) and up to 16 (ST6215,E15, ST6225,E25) analog inputs (A/D inputs are alternate functions of I/O pins), the digital Watchdog/Timer (DWD). Thanks to these peripherals these devices are well suited for automotive, appliance and industrial applications. The ST62E10, ST62E15, ST62E20 and ST62E25 EPROM versions are available for prototypes and low-volume production; also OTP versions are available (see ordering information at the end of the datasheet for additional information). The only difference between ST6210,15 and ST6220,25 is the program memory size that is 2K bytes for the ST6210,15 and 4K bytes for the ST6220,25.

PIN DESCRIPTION

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

OSCIN and OSCOUT. These pins are internally connected with the on-chip oscillator circuit. A quartz crystal, a ceramic resonator or an external clock signal can be connected between these two pins in order to allow the correct operation of the MCU with various stability/cost trade-offs. The OSCIN pin is the input pin, the OSCOUT pin is the output pin. Refer to ON-CHIP CLOCK OSCILLATOR description for additional information.

$\overline{\text{RESET}}$. The active low $\overline{\text{RESET}}$ pin is used to restart the microcontroller to the beginning of its program. Refer to RESET and WATCHDOG description for additional information.

TEST/V_{PP}. The TEST pin is used to place the MCU into special operating mode. If TEST is held at V_{SS} the MCU enters the normal operating mode. If TEST is held at V_{DD} the test operating mode is automatically selected (the user should connect this pin to V_{SS} for normal operation). If this pin is connected to a +12.5V level during the reset phase the EPROM programming mode is entered (EPROM/OTP versions only).Refer to TEST mode description for additional information.

NMI. The NMI pin provides the capability for asynchronous applying an external not maskable interrupt to the MCU. The NMI is falling edge sensitive. On ST6210,15 and ST6220,25 the user can select as ROM mask option (see option list at the end of the datasheet) the availability of an on-chip pull-up at NMI pin. On EPROM/OTP versions this pull-up is not available and should be provided externally. Refer to INTERRUPT description for additional information.

TIMER. This is the timer I/O pin. In input mode it is connected to the prescaler and acts as external timer clock or as control gate for the internal timer clock. In the output mode the timer pin outputs the data bit when a time-out occurs. On ST6210,15 and ST6220,25 the user can select as ROM mask option (see option list at the end of the datasheet) the availability of an on-chip pull-up at TIMER pin. On EPROM/OTP versions this pull-up is not available and should be provided externally. Refer to TIMER description for additional information.

PA0-PA3,PA4-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. PA0-PA3 can also sink 10mA for direct led driving while PA4-PA7 can be programmed as analog inputs for the A/D converter. (*) PA4-PA7 are not available on ST6210,E10, ST6220,E20.

PB0-PB7. These 8 lines are organized as one I/O port (B). Each line may be configured under software control as inputs with or without internal pull-up resistors, interrupt generating inputs with pull-up resistors, open-drain or push-pull outputs and as analog inputs for the A/D converter.

PIN DESCRIPTION (Continued)

PC4-PC7(*). These 4 lines are organized as one I/O port (C). 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 and as analog inputs for the A/D converter.(*) PC4-PC7 are not available on ST6210,E10, ST6220,E20.

DEVICE	ROM (Bytes)	EPROM (Bytes)	I/O Pins
ST6210	2K		12
ST6215	2K		20
ST6220	4K		12
ST6225	4K		20
ST62E10		2K	12
ST62E15		2K	20
ST62E20		4K	12
ST62E25		4K	20

ST62E10,E15 and ST62E20,E25 EPROM VERSIONS DESCRIPTION

The ST62E10,E15 and ST62E20,E25 are the EPROM versions of the ST6210,15 and ST6220,25 products. These products are used during the development of the application, the pre-production or the production of small volume. OTP versions with the same characteristics are also available. They include an EPROM memory instead of the ROM memory of the ST6210,15 and ST6220,25 products, and so the program and the constants of the program can be easily modified by the user thanks to the EPROM version programmer.

From a user's point of view, the ST62E10,E15 and ST62E20,E25 products have exactly the same software and hardware features than those of the ROM versions, except the fact that the EPROM version has a special mode defined during the Reset sequence (see PIN DESCRIPTION). This mode is used to configure the product in the EPROM/OTP programmer mode in which a 12.5V voltage has to be applied on the TEST pin in order to program the EPROM memory. The programmer of the ST62E10,E15 and ST62E20,E25 products is also described in the User' manual of the EPROM programmer. In addition on EPROM/OTP versions at TIMER and NMI pins is not available the on-chip pull-up resistor that can be selected by the user (see option list at the end of the datasheet) to be connected or not in masked devices as ROM option. On masked devices the watchdog activation is also selected as ROM option while for EPROM/OTP versions four different part numbers are available (see ordering information at the end of the datasheet).

ST62XX CORE

The Core of the ST62XX Family is implemented independently from the I/O or memory configuration. Consequently, it can be treated as an independent central processor communicating with I/O and memory via internal addresses, data, and control busses. The in-core communication is arranged as shown in Figure 4; the controller being externally linked to both the reset and the oscillator, while the core is linked to the dedicated on-chip macrocells peripherals via the serial data bus and indirectly for interrupt purposes through the control registers. The user could also appreciate the performances of ST62XX Core by referring to Registers and SOFTWARE descriptions.

Registers

The ST62XX Family Core has six registers and three pairs of flags available to the programmer. They are shown in Figure 5 and are explained in the following paragraphs.

Figure 5. ST62XX Core Programming Model

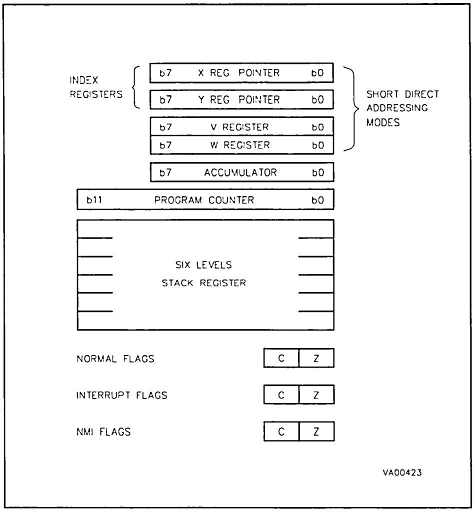
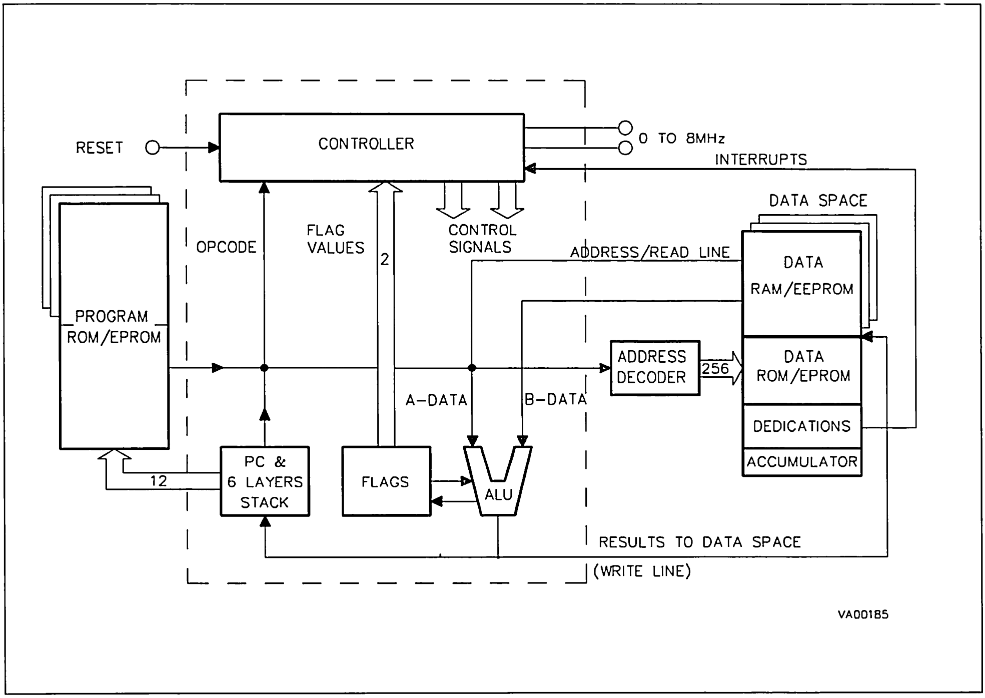


Figure 4. ST62XX Core Block Diagram



ST62XX CORE (Continued)

Accumulator (A). The accumulator is an 8-bit general purpose register used in all arithmetic calculations, logical operations, and data manipulations. The accumulator is addressed in the data space as RAM location at the FFH address. Accordingly, the ST62XX instruction set can use the accumulator as any other register of the data space.

Indirect Registers (X, Y). These two indirect registers are used as pointers to the memory locations in the data space. They are used in the register-indirect addressing mode. These registers can be addressed in the data space as RAM locations at the 80H (X) and 81H (Y) addresses. They can also be accessed with the direct, short direct, or bit direct addressing modes. Accordingly, the ST62XX 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 one byte in short direct addressing mode . These registers can be addressed in the data space as RAM locations at the 82H (V) and 83H (W) addresses. They can also be accessed with the direct and bit direct addressing modes. Accordingly, the ST62XX 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 that contains the address of the next ROM location to be processed by the Core. This ROM location may be an opcode, an operand, or an address of operand. The 12-bit length allows the direct addressing of 4096 bytes in the program space. Nevertheless, if the program space contains more than 4096 locations, the further program space can be addressed by using the Program Bank Switch register. The PC value is increment, after it is read for the address of the current instruction, by sending it through the one bit ALU, so giving the address of the next byte in the program. To execute relative jumps the PC and the offset are shifted through the ALU, where they will be added, and the result is 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 instructions	PC= PC ± offset
– Interrupt	PC= Interrupt vector
– Reset	PC= Reset vector
– Test mode	PC=Test mode vector (1)
– RET & RETI instructions . .	PC= Pop (stack)
– Normal instruction	PC= PC + 1

Note: 1. Not available for the user.

Flags (C, Z)

The ST62XX Core includes three pairs of flags that correspond to3 different modes: 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, one pair is used during the interrupt mode (CI, ZI) and one is used during the not-maskable interrupt mode (CNMI, ZNMI).

The ST62XX Core uses the pair of flags that corresponds to the actual mode: as soon as an interrupt (resp. a Non-Maskable-Interrupt) is generated, the ST62XX Core uses the interrupt flags (resp. the NMI flags) instead of the normal flags. When the RETI instruction is executed, the normal flags (resp. the interrupt flags) are restored if the MCU was in the normal mode (resp. in the interrupt mode) before the interrupt. Should be observed that each flag set can only be addressed in its own routine (Not-maskable interrupt, normal interrupt or main routine). The interrupt flags are not cleared during the context switching and so, they remain in the state they were at the exit of the last routine switching.

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, and 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.

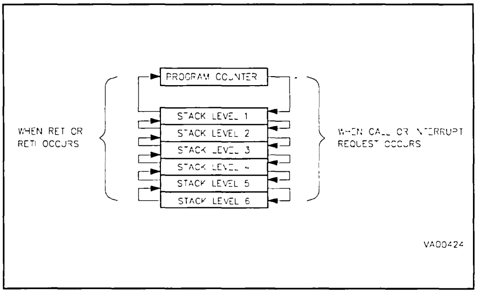
The switching between these three sets is automatically performed when an NMI, an interrupt or a RETI instructions occur. As the NMI mode is automatically selected after the reset of the MCU, the ST62XX Core uses at first the NMI flags. Refer to INTERRUPT DESCRIPTION for additional information.

Stack

The ST62XX Core includes true LIFO hardware stack that 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 is shifted into the next level while the content of the PC is shifted into the first level (the value of the sixth level will be lost).

ST62XX CORE (Continued)

Figure 6. Stack Operation



When 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. These two operating modes are described in Figure 6. Since the accumulator, as all other data space registers, is not stored in this stack the handling of this registers shall be performed inside the subroutine. The stack pointer will remain in its deepest position, if more than 6 calls or interrupts are executed, so that 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. Data ROM Window Register

RDW
Data Window register
(C9H, Write Only)

D7	D6	D5	D4	D3	D2	D1	D0

The RDW register can be addressed like a RAM location in the Data Space at the C9H address, nevertheless it is write-only register that can not be accessed with single-bit operations. This register is used to move up and down the 64-byte read-only data window (from the 40H address to 7FH address of the Data Space) along the ROM memory of the MCU by step of 64 bytes.
The effective address of the byte to be read as a data in the ROM memory is obtained by the concatenation of the 6 less significant bits of the address given in the instruction (as less significant bits) and the content of the RDW register (as most significant bits). Refer to the Data Space description for additional information.

MEMORY SPACES

The MCUs operate in three different memory spaces: Program Space, Data Space, and Stack Space; they are described in Figure 8, Figure 9, Figure 10 and Figure 11.

Figure 8. ST6210,62E10, ST6215,62E15 Program Space

b7 b0	
NOT IMPLEMENTED	0000H 07FFH
RESERVED	0800H 087FH
USER PROGRAM ROM 1828 BYTES	0880H 0F9FH
RESERVED	0FA0H 0FEFH
INTERRUPT VECTOR #4 A/D INTERRUPT	0FF0H 0FF1H
INTERRUPT VECTOR #3 TIMER INTERRUPT	0FF2H 0FF3H
INTERRUPT VECTOR #2 PORT B & C INTERRUPT	0FF4H 0FF5H
INTERRUPT VECTOR #1 PORT A INTERRUPT	0FF6H 0FF7H
RESERVED	0FF8H 0FFBH
INTERRUPT VECTOR #0 NMI INTERRUPT	0FFCH 0FFDH
USER RESET VECTOR	0FFEH 0FFFH

On EPROM versions there are no reserved areas. These reserved bytes are present on ROM/OTP versions.

MEMORY SPACES (Continued)

Figure 9. ST6210,62E10, ST6215,62E15 Data Space

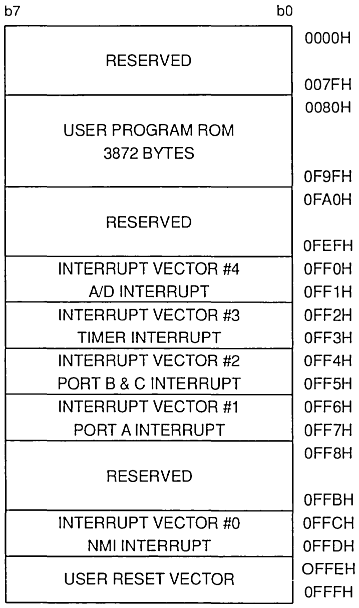
b7 b0	
NOT IMPLEMENTED	000H 03FH
DATA ROM/EPROM WINDOW 64 BYTE	040H 07FH
X REGISTER	080H
Y REGISTER	081H
V REGISTER	082H
W REGISTER	083H
DATA RAM 60 BYTES	084H 0BFH
PORT A DATA REGISTER	0C0H
PORT B DATA REGISTER	0C1H
PORT C DATA REGISTER	0C2H
RESERVED	0C3H
PORT A DIRECTION REGISTER	0C4H
PORT B DIRECTION REGISTER	0C5H
PORT C DIRECTION REGISTER	0C6H
RESERVED	0C7H
INTERRUPT OPTION REGISTER	0C8H
DATA ROM WINDOW REGISTER	0C9H
RESERVED	0CAH 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 PSC REGISTER	0D2H
TIMER DATA REGISTER	0D3H
TIMER TSCR REGISTER	0D4H
RESERVED	0D5H 0D7H
WATCHDOG REGISTER	0D8H
RESERVED	0D9H 0FEH
ACCUMULATOR	0FFH

Figure 10. ST6220,62E20, ST6225,62E25 Data Space

b7 b0	
NOT IMPLEMENTED	000H 03FH
DATA ROM/EPROM WINDOW 64-BYTE	040H 07FH
X REGISTER	080H
Y REGISTER	081H
V REGISTER	082H
W REGISTER	083H
DATA RAM 60 BYTES	084H 0BFH
PORT A DATA REGISTER	0C0H
PORT B DATA REGISTER	0C1H
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 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 PSC REGISTER	0D2H
TIMER DATA REGISTER	0D3H
TIMER TSCR REGISTER	0D4H
RESERVED	0D5H 0D7H
WATCHDOG REGISTER	0D8H
RESERVED	0D9H 0FEH
ACCUMULATOR	0FFH

MEMORY SPACES (Continued)

Figure 11. ST6220,62E20, ST6225,62E25 Program Space



b7 b0	
RESERVED	0000H 007FH
USER PROGRAM ROM 3872 BYTES	0080H 0F9FH
RESERVED	0FA0H 0FEFH
INTERRUPT VECTOR #4 A/D INTERRUPT	0FF0H 0FF1H
INTERRUPT VECTOR #3 TIMER INTERRUPT	0FF2H 0FF3H
INTERRUPT VECTOR #2 PORT B & C INTERRUPT	0FF4H 0FF5H
INTERRUPT VECTOR #1 PORT A INTERRUPT	0FF6H 0FF7H
RESERVED	0FF8H 0FFBH
INTERRUPT VECTOR #0 NMI INTERRUPT	0FFCH 0FFDH
USER RESET VECTOR	OFFEH 0FFFH

On EPROM versions there are no reserved areas. These reserved bytes are present on ROM/OTP versions.

Program Space

The program space is physically implemented in the ROM memory and includes all the instructions that are to be executed, as well as the data required for the immediate addressing mode instructions, the reserved test area and user vectors. It is addressed thanks to the 12-bit Program Counter register (PC register) and so, the ST62XX Core can directly address up to 4K bytes of Program Space. Nevertheless, the Program Space can be extended by the addition of 2-Kbyte ROM banks.

The ST6210,E10, ST6215,E15, ST6220,E20 and ST6225,E25 products include 2048,4096 bytes of ROM available for the user and this space can be directly addressed by the program counter and for this reason the Program Bank Switch (PBS) register has not been implemented on these devices. Nevertheless, the data ROM window register (RDW) has been implemented and all the locations of the ROM memory can be addressed by the user as read-only data in the data space.

Data Space

The instruction set of the ST62XX Core operates on a specific space, named Data Space that contains all the data necessary for the processing of the program. The Data Space allows the addressing of RAM memory, ST62XX Core/peripheral registers, and read-only data such as constants and the look-up tables.

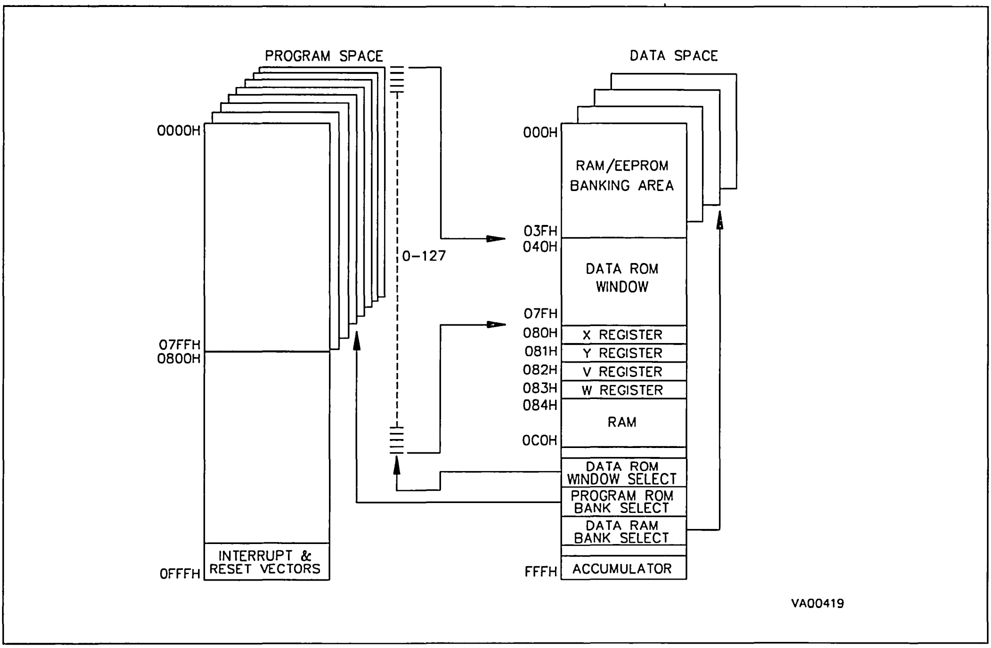
All the read-only data are physically implemented in the ROM memory in which the Program Space is also implemented. The ROM memory contains consequently the program to be executed, the constants and the look-up tables needed for the program.

The locations of Data Space in which the different constants and look-up tables are addressed by the ST62XX Core can be considered as being a 64-byte window through which it is possible to access to the read-only data stored in the ROM memory (see Figure 12).

This window is located from the 40H address to the 7FH address in the Data space and allows the direct reading of the bytes from the 000H address to the 03FH address in the ROM memory. All the bytes of the ROM memory can be used to store either instructions or read-only data. Indeed, the window can be moved by step of 64 bytes along the ROM memory in writing the appropriate code in the Data ROM Window register (RDW register). The effective address of the byte to be read as a data in the ROM memory is obtained by the concatenation of the 6 less significant bits of the address in the Data Space (as less significant bits) and the content of the RDW register (as most significant bits). For instance, if the user wants to read the ROM location at the A19H address, the RDW register has to be loaded with 28H and the address to be pointed to in the Data Space has to be 59H.

MEMORY SPACES (Continued)

Figure 12. ST62XX Memory Addressing Description Diagram



The RAM memory can be also extended by the addition of 64 bytes RAM banks addressed as being located between the 00H and 7FH addresses.

In the ST6210,E10, ST6215,E15, ST6220,E20 and ST6225,E25 products 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 (RDW register).

As the data space is less than 256 bytes the ST62XX Core can directly address this area and the Data Bank Switch register (DBS) has not been implemented.

Stack Space

The stack space consists of six 12 bit registers that are used for stacking subroutine and interrupt return addresses plus the current program counter register.

TEST MODE

The test mode can be entered by connecting the TEST pin to an high logic level when reset is active; this action enables the factory test mode. The user is recommended to avoid this situation for normal operation.

INTERRUPT

The ST62XX Core can manage 4 different maskable interrupt sources, plus one non-maskable interrupt source (top priority level interrupt). Each source is associated with a particular interrupt vector that contains a Jump instruction to the related interrupt service routine. Each vector is located in the Program Space at a particular address (see Table 1).

When a source provides an interrupt request, and the request processing is also enabled by the ST62XX Core, then the PC register is loaded with the address of the interrupt vector (i.e. of the Jump instruction).

Finally, the PC is loaded with the address of the Jump instruction and the interrupt routine is processed. The ST6210,E10, ST6215,E15, ST6220,E20 and ST6225,E25 microcontrollers have six different interrupt sources associated to different interrupt vectors as it is described in Table 1.

INTERRUPT(Continued)

Table 1. Interrupt Vector/Source Relationship

Interrupt Source	Associated Vector	Vector Address
NMI pin	Interrupt vector #0 (NMI)	(FFCH, FFDH)
Port A pins	Interrupt vector #1	(FF6H, FF7H)
Port B pins	Interrupt vector #2	(FF4H, FF5H)
Port C pins	Interrupt vector #2	(FF4H, FF5H)
TIMER peripheral	Interrupt vector #3	(FF3H, FF2H)
ADC peripheral	Interrupt vector #4	(FF0H, FF1H)

Interrupt Vectors Description

The ST62XX Core includes 5 different interrupt vectors in order to branch to 5 different interrupt routines in the static page of the Program Space (the interrupt start address should be located in the static page) :

- The interrupt vector associated with the non-maskable interrupt source is named interrupt vector #0. It is located at the (FFCH,FFDH) addresses in the Program Space. On ST6210,E10, ST6215,E15, ST6220,E20 and ST6225,E25 this vector is associated with the external falling edge sensitive interrupt pin.
- The interrupt vector located at the (FF6H, FF7H) addresses is named interrupt vector #1. It is associated with Port A pins that can be programmed by software either in the falling edge detection mode or in the low level sensitive detection mode according to the code loaded in the Interrupt Option Register (IOR).
- The interrupt vector located at the (FF4H, FF5H) addresses is named interrupt vector #2. It is associated with Port B and C pins that can be programmed by software either in the falling edge detection mode or in the positive edge detection mode according to the code loaded in the Interrupt Option Register (IOR).
- The two interrupt vectors located respectively at the (FF3H, FF2H) addresses and the (FF1H, FF0H) addresses are respectively named interrupt vector #3 and #4. Vector #3 is associated to the TIMER peripheral while vector #4 to the A/D converter peripheral. All the on-chip peripherals (refers to TIMER & A/D descriptions for further details) have an interrupt request flag bit (TMZ for timer, EOC for A/D), this bit is set to one when the device wants to generate an interrupt request and a mask bit (ETI for timer, EAI for A/D) that must be set to one to allow the transfer of the flag bit to the Core.

Interrupt Priority

The non-maskable interrupt request has the highest priority and can interrupt any other interrupt routines at any time, nevertheless the four other interrupts can not interrupt each other. If more than one interrupt request are pending, they are processed by the ST62XX Core according to their priority level: vector #1 has the higher priority while vector #4 the lower.
The priority of each interrupt source is hardware fixed.

Figure 13. Interrupt Option Register

IOR
Interrupt Option Register
(C8H, Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

The Interrupt Option Register (IOR register, location C8H) is used to enable/disable the individual interrupt sources and to select the operating mode of the external interrupt inputs. This register can be addressed in the Data Space as RAM location at the C8H address, nevertheless it is write-only register that can not be accessed with single-bit operations. The operating modes of the external interrupt inputs associated to interrupt vectors #1 and #2 are selected through bits 4 and 5 of the IOR register. Table 2 summarizes the use of the IOR register:

Table 2. Interrupt Option Register Description

IOR4	SET	Enable all the interrupts of the product
	CLEARED	Disable all the interrupts of the product
IOR5	SET	Rising edge mode on interrupt input (#2)
	CLEARED	Falling edge mode on interrupt input (#2)
IOR6	SET	Level-sensitive mode on interrupt input (#1)
	CLEARED	Falling edge mode on interrupt input (#1)

External Interrupts Operating Modes

The NMI interrupt is associated to the external interrupt pin of the ST6210,E10, ST6215,E15, ST6202,E20 and ST6225,E25 devices. This pin is falling edge sensitive and the interrupt pin signal is latched by a flip-flop which is automatically reset by the Core at the beginning of the non-maskable

INTERRUPT (Continued)

interrupt service routine. A schmitt trigger is present on NMI pin.
The two interrupt sources associated with the falling/rising edge mode of the external interrupt pins (Ports A-vector 1, Ports B and C-vector 2) are connected to two internal latches. Each latch is set when a falling/rising edge occurs during the processing of a the first one, will be processed as soon as the first one has been finished (if there is not a more priority interrupt request). If more then one interrupt occurs during the processing of the first one, these other interrupt requests will be lost.
The storage of the interrupt requests is not available in the level sensitive detection mode. To be taken into account, the low level must be present on the interrupt pin when the Core samples the line after the execution of the instructions.
During the end of each instruction the Core tests the interrupt lines and if there is an interrupt request the next instruction is not executed and the related interrupt routine is executed.

Note

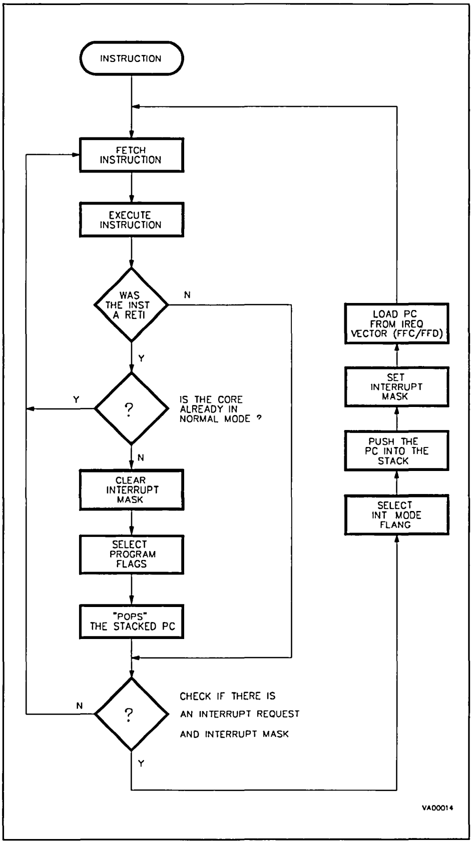
On ST6210,15 and ST6220,25 the user can select as ROM mask option (see option list at the end of the datasheet) the availability of an on-chip pull-up at NMI pin. On ST62E10,E15, ST62E20,E25 this pull-up is not available and should be provided externally.

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 does not know about the context and the time at which it occurred. As a result the user should save all the data space registers which will be used inside the interrupt routines. There are separate sets of processor flags for normal, interrupt and non-maskable interrupt modes which are automatically switched and so these do not need to be saved.

The following list summarizes the interrupt procedure:

- Interrupt detection
- The flags C and Z of the main routine are exchanged with the flags C and Z of the interrupt routine (resp. the NMI flags)
- The value of the PC is stored in the first level of the stack
- The normal interrupt lines are inhibited (NMI still active)
- The edge flip-flop is reset
- The related interrupt vector is loaded in the PC.

Figure 14. Interrupt Processing Flow Chart

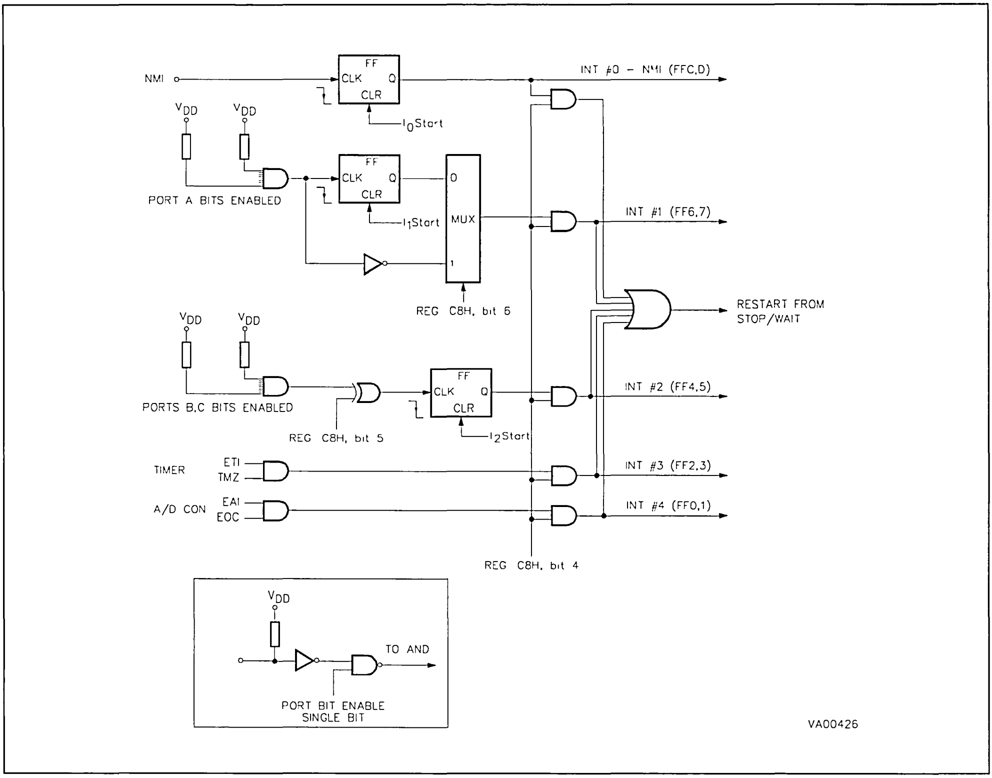


- User selected registers are saved inside the interrupt service routine (normally on a software stack)
- The source of the interrupt is found by polling (if more than one source is associated to the same vector)
- Interrupt servicing
- Return from interrupt (RETI)
- Automatically the ST62XX core switches back to the normal flags (resp the interrupt flags) and pops the previous PC value from the stack

The interrupt routine begins usually by the identification of the device that has generated the interrupt request (by polling).
The user should save the registers which are used inside the interrupt routine (that holds relevant data) into a software stack.

INTERRUPT (Continued)

Figure 15. Interrupt Circuit Diagram



After the RETI instruction execution, the Core carries out the previous actions and the main routine can continue.

Interrupt Request and Mask Bits

Interrupt Option Register, IOR Location 0C8H

- IOR4. If this bit is set all the ST6210,E10, ST6215,E15, ST6220,E20 and ST6225,E25 interrupts are enabled, if reset all the interrupt are disabled (including the NMI).
- IOR5. If this bit is set all the inputs lines associated to interrupt vector #2 are rising edge sensitive, if reset they are falling edge sensitive.
- IOR6. If this bit is set all the inputs lines associated to interrupt vector #1 are low level sensitive, if reset they are falling edge sensitive.

All other bits into this register are not used.

Timer Peripheral, TSCR register location D4H

- **TMZ bit**. Low-to-high transition indicates that the timer count register has decrement to zero. This means that an interrupt request can be generated in relation to the state of ETI bit.
- **ETI bit**. This bit, when set,enables the timer interrupt request.

A/D Converter Peripheral, ADCR register location D1H

- **EOC bit**. This read only bit indicates when a conversion has been completed going to one. An interrupt request can be generated in relation to the state of EAI bit.
- **EAI bit**. This bit, when set, enables the A/D converter interrupt request.

RESET

The ST6210,E10, ST6215,E15, ST6220,E20 and ST6225,E25 MCUs can be reset in three ways: by the external reset input (RESET) tied low, by

RESET(Continued)

power-on reset and by the digital watchdog/timer peripheral (see DIGITAL WATCHDOG/TIMER descrip- tion).

RESET Input

The RESET pin can be connected to a device of the application board in order to restart the MCU during its operation. The activation of the Reset pin may occurs in the RUN, WAIT or STOP mode. This input has to be used to reset the MCU internal state and provide a correct start-up procedure. The pin is active low and have a schmitt trigger input. The internal reset signal is generated by adding a delay to the external signal. Therefore even short pulses at the reset pin will be accepted. This feature is valid providing that V_{DD} have finished its rising phase and oscillator is correctly running (normal RUN or WAIT modes).

If the Reset activation occurs in the RUN or Wait mode, the MCU is configured in the Reset mode as long as the signal of the RESET pin is low. The processing of the program is stopped (in RUN mode only) and the Input/Outputs are in the High-impedance state. As soon as the level on the Reset pin becomes high, the initialization sequence is executed. Refer to the MCU initialization sequence for additional information.

If a Reset pin activation occurs in the STOP mode, all the inputs/outputs are configured in the High-impedance state and the oscillator starts as long as the level on the RESET pin remains low. When the level of the RESET pin becomes high, a delay is generated by the ST62XX Core to wait that the oscillator becomes completely stabilized.

Then, the initialization sequence is started. Refer to the MCU initialization sequence below for additional information.

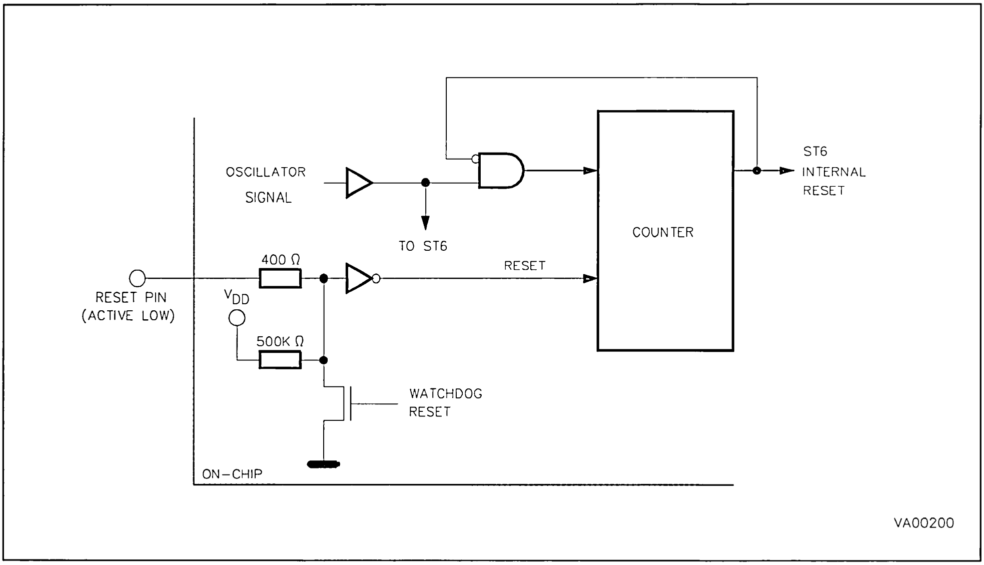
Power-on Reset

The function of the POR consists in waking up the MCU during the power-on sequence. At the beginning of this sequence, the MCU is configured in the Reset state: every Input/Output port is configured in the input mode (High-impedance state) and no instruction is executed. When the power supply voltage becomes sufficient, the oscillator starts to operate, nevertheless the ST62XX Core generates a delay to allow the oscillator to be completely stabilized before the execution of the first instruction. Then,

The initialization sequence is executed. Refer to the MCU initialization sequence for additional information. As long as the reset pin is kept at low level (refer to Electric Characteristics description) the processor remains in reset state. The reset will be released after the voltage at the reset pin reaches the related high level.

To have correct MCU start-up the user should take care that the reset input does not change to the high level before the V_{DD} level is sufficient to allow MCU operation at the chosen frequency (see Recommended Operating Conditions paragraph).

Figure 16. Reset Circuit



RESET (Continued)

Watchdog Reset

The ST6210,E10, ST6215,E15, ST6220,E20 and ST6225,E25 provides an on-chip watchdog/timer function in order to provide a graceful recovery from a software upset. If the watchdog register is not refreshed preventing the end-of-count being reached, an internal circuit pulls down the reset pin. The MCU will enter the reset state as soon as the voltage at reset pin reaches the related low level. This also resets the watchdog which subsequently turns off the pull-down and activates the pull-up device at the reset pin. This causes the positive transition at the reset pin and terminates the reset state.

Application Notes

An external resistor between V_{DD} and reset pin is not required because an internal pull-up device is provided. If the user prefers, for any reason, to add an external pull-up resistor its value must not be less than 30KΩ. If the value is lower than 30KΩ the on-chip watchdog pull-down transistor might not be able to pull-down the reset pin resulting in an external deactivation of the watchdog function.

Figure 17. Reset & Interrupt Processing Flow Chart

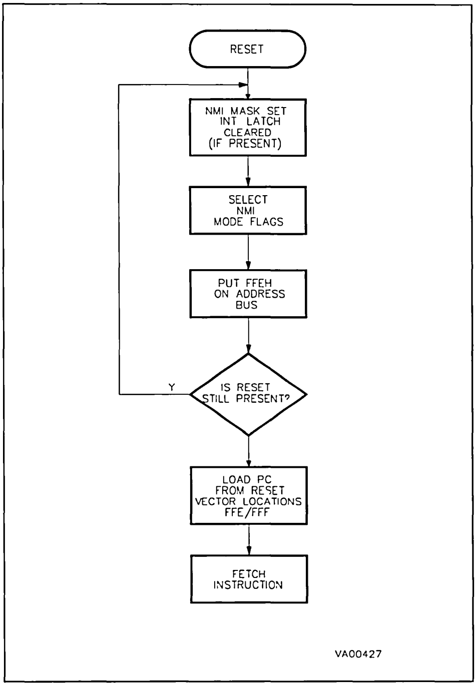
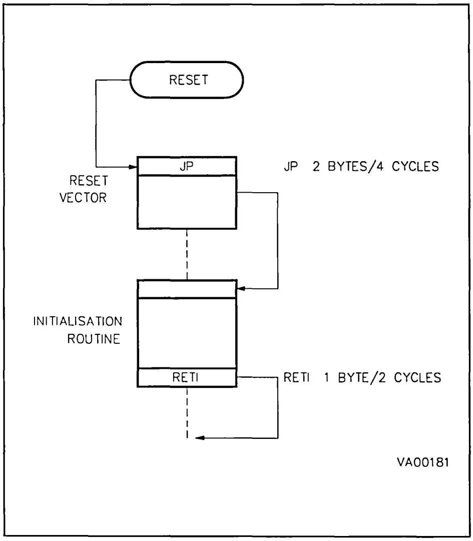


Figure 18. Restart Initialization Program Flow Chart



The POR device operates in a dynamic manner in the way that it brings about the initialization of the MCU when it detects a dynamic rising edge of the V_{DD} voltage. The typical detected threshold is about 2 volts, but the actual value of the detected threshold depends on the way in which the V_{DD} voltage rises up. The POR device *DOES NOT* allow the supervision of a static rising or falling edge of the V_{DD} voltage.

MCU Initialization Sequence

When a reset occurs the stack is reset to program counter, the PC is loaded with the address of the reset vector (located in the program ROM at addresses FFEH & FFFH). A jump instruction to the beginning of the program has to be written into these locations.After a reset a NMI is automatically activated so that the Core is in non-maskable interrupt mode to prevent false or ghost interrupts during the restart phase. Therefore the restart routine should be terminated by a RETI instruction to switch to normal mode and enable interrupts. If no pending interrupt is present at the end of the reset routine the device will continue with the instruction after the RETI; otherwise the pending interrupt will be serviced

WAIT & STOP MODES

The STOP and WAIT modes have been implemented in the ST62XX Core in order to reduce the consumption of the product when the latter has

WAIT & STOP MODES (Continued)

no instruction to execute. These two modes are described in the following paragraphs

WAIT Mode

The configuration of the MCU in the WAIT mode occurs as soon as the WAIT instruction is executed. The microcontroller can also be considered as being in a "software frozen" state where the Core stops processing the instructions of the routine,,the contents of the RAM locations and peripheral registers are saved as long as the power supply voltage is higher than the RAM retention voltage but where the peripherals are still working. The WAIT mode is used when the user wants to reduce the consumption of the MCU when it is in idle, while not loosing count of time or monitoring of external events. The oscillator is not stopped in order to provide clock signal to the peripherals. The timer counting may be enabled (writing the PSI bit in TSCR register) and the timer interrupt may be also enabled before entering the WAIT mode; this allows the WAIT mode to be left when timer interrupt occurs. The above explanation related to the timers applies also to the A/D converter. If the exit from the WAIT mode is performed with a general RESET (either from the activation of the external pin or by watchdog reset) the MCU will enter a normal reset procedure as described in the RESET chapter. If an interrupt is generated during WAIT mode the MCU behavior depends on the state of the ST62XX Core before the initialization of the WAIT sequence, but also of the kind of the interrupt request that is generated. This case will be described in the following paragraphs. In any case, the ST62XX Core does not generate any delay after the occurrence of the interrupt because the oscillator clock is still available.

STOP Mode

If the Watchdog is disabled the 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 contents of the RAM locations and peripheral registers are saved as long as the power supply voltage is higher than the RAM retention voltage, and the ST62XX Core waits the occurrence of an external interrupt request or Reset activation to output from the STOP state.If the exit from the STOP mode is performed with a general RESET (by the activation of the external pin) the MCU will enter a normal reset procedure as described in the RESET chapter. The case of an interrupt depends on the state of the ST62XX Core before the initialization of the STOP sequence and also of the kind of the interrupt request that is generated.

This case will be described in the following paragraphs. In any case, the ST62XX Core generates a delay after the occurrence of the interrupt request in order to wait the complete stabilization of the oscillator before the execution of the first instruction.

Exit from WAIT and STOP Modes

The following paragraphs describe the output procedure of the ST62XX Core from WAIT and STOP modes when an interrupt occurs. It must be noted that the restart sequence depends on the original state of the MCU (normal, interrupt or non-maskable interrupt mode) before the start of the WAIT or STOP sequence, but also of the type of the interrupt request that is generated.

Normal Mode. If the ST62XX Core was in the main routine when the WAIT or STOP instruction has been executed, the ST62XX Core outputs from the stop or wait mode as soon as any interrupt occurs; the related interrupt routine is executed and at the end of the interrupt service routine the instruction that follows the Stop or the Wait instruction is executed if no other interrupts are pending.

Not Maskable Interrupt Mode. If the STOP or WAIT instruction has been executed during the execution of the non-maskable interrupt routine, the ST62XX Core outputs from the stop or wait mode as soon as any interrupt occurs: the instruction that follows the Stop or the Wait instruction is executed and the ST62XX Core is still in the non-maskable interrupt mode even if an other interrupt has been generated.

Normal Interrupt Mode. If the ST62XX Core was in the interrupt mode before the initialization of the STOP or WAIT sequence, it outputs from the stop or wait mode as soon as any interrupt occurs. Nevertheless, two cases have to be considered:

-If the interrupt is a normal interrupt, the interrupt routine in which the wait or stop was entered will be completed with the execution of the instruction that follows the STOP or the WAIT and the ST6 Core is still in the interrupt mode. At the end of this routing pending interrupts will be serviced in accordance to their priority.

-If the interrupt is a non-maskable interrupt, the non-maskable routine is processed at first. Then, the routine in which the wait or stop was entered will be completed with the execution of the instruction that follows the STOP or the WAIT and the ST6 Core is still in the normal interrupt mode.

Note

To reach the lowest power consumption the user software must put the A/D converter in its power down mode by clearing the PDS bit in the A/D control register before entering the STOP instruction.

WAIT & STOP MODES (Continued)

If all the interrupt sources are disabled, the restart of the MCU can only be done by a Reset activation. The Wait and Stop instructions are not executed if an enabled interrupt request is pending.

ON-CHIP CLOCK OSCILLATOR

The internal oscillator circuit is designed to require a minimum of external components. A crystal, a ceramic resonator, or an external signal (provided to the OSCIN pin) may be used to generate a system clock with various stability/cost tradeoffs. The different clock generator options connection methods are shown in Figure 19, crystal specifications and suggested PC board layouts are given in Figure 20 and Figure 21.

One machine cycle takes 13 oscillator pulses; 12 clock pulses are needed to increment the PC while and additional 13th pulse is needed to stabilize the internal latches during memory addressing. This means that with a clock frequency of 8MHz the machine cycle is 1.625µs. The crystal oscillator start-up time is a function of many variables: crystal parameters (especially RS), oscillator load capacitance (CL), IC parameters, ambient temperature, and supply voltage. It must be observed that the

Figure 19. Clock Generator Connections

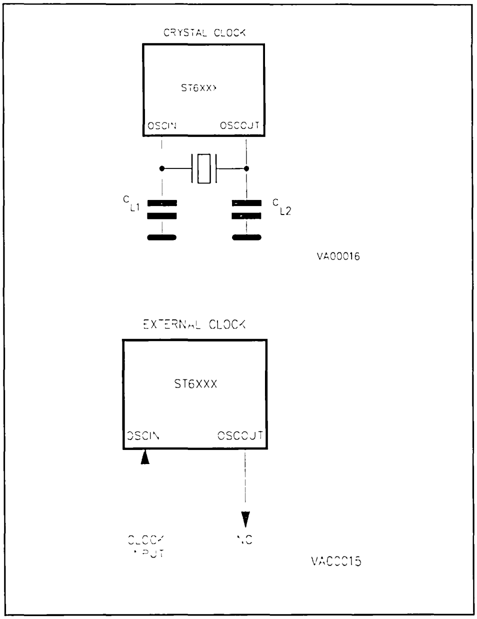
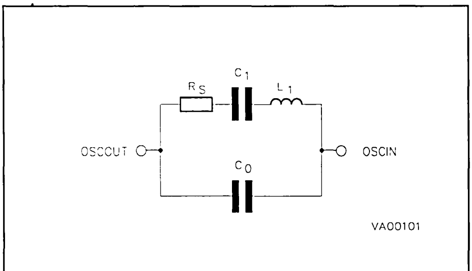
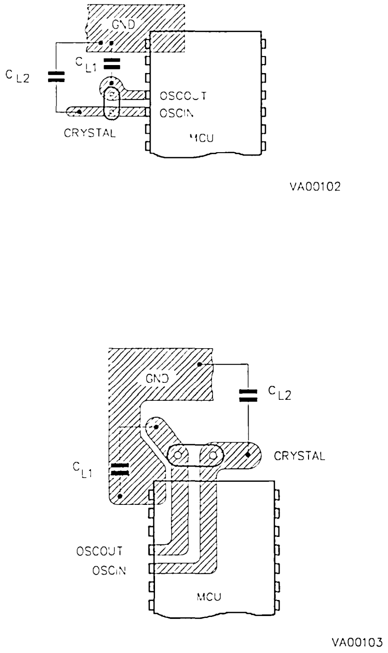


Figure 20. Crystal Parameters



Cristal Parameters AT-Cut Parallel Resonance Crystal
C_0 = Parallel Resonance Capacitance

Figure 21. PC Board Layout Examples



ON-CHIP CLOCK OSCILLATOR (Continued)

crystal or ceramic leads and circuit connections must be as short as possible. Typical values for CL1, CL2 are 15-22pF for a 4/8MHz crystal. The oscillator output frequency is internally divided by 13 to produce the machine cycle and by 12 to produce the Timer, the Watchdog and the A/D peripheral clock. A machine cycle is the smallest unit needed to execute any operation (i.e., increment the program counter). An instruction may need two, four, or five byte cycles to be executed.

Table 3. Instruction Timing with 8MHz Clock

Instruction Type	Cycles	Execution Time
Branch if bit set/reset	5 Cycles	8.125µs
Branch & subroutine branch Bit Manipulation Load instructions Arithmetic & Logic	4 Cycles 4 Cycles 4 Cycles 4 Cycles	6.50µs 6.50µs 6.50µs 6.50µs
Conditional branch Program control	2 Cycles 2 Cycles	3.25µs 3.25µs

INPUT/OUTPUT PORTS

The ST6210,E10, ST6220,E20 and ST6215,E15, ST6225,E25 microcontrollers have respectively 12 and 20 Input/Output lines that can be individually programmed either in the input mode or the output mode with the following options that can be selected by software:

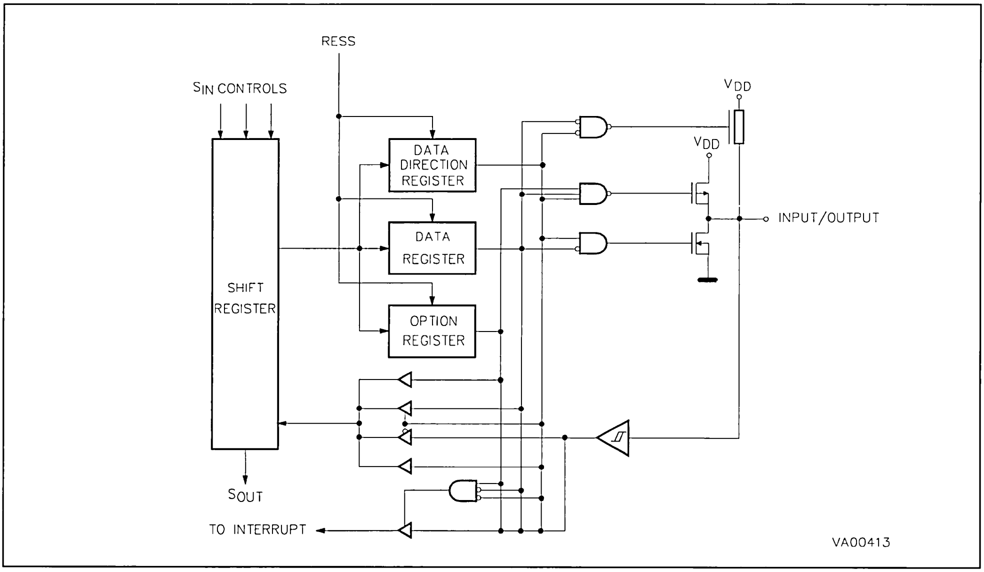
- Input without pull-up and without interrupt
- Input with pull-up and with interrupt
- Input with pull-up without interrupt
- Analog input
- Push-pull output
- Open drain NMOS output

The input mode allows to configure the lines in the high impedance state. The lines are organized in three ports (port A,B,C).

The ports occupies 9 registers in the data space. Each bit of these registers is associated with a particular line (for instance, the bits 0 of the Port A Data, Direction and Option registers are associated with the PA0 line of Port A).

There being three DATA registers (DRA, DRB, DRC), that are used to read the voltage level values of the lines programmed in the input mode, or to write the logic value of the signal to be output on the lines configured in the output mode. The port

Figure 22. I/O Port Block Diagram



INPUT/OUTPUT PORTS (Continued)

data registers can be read to get the effective logic levels of the pins, but they can be also written by the 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 changes of the input configuration.

The three Data Direction registers (DDRA, DDRB, DDRB) allow the selection of the direction of each pin (input or output).

The three Option registers (ORPA, ORPB, ORPC) are used to select the different port options that are available both in input and in output mode.

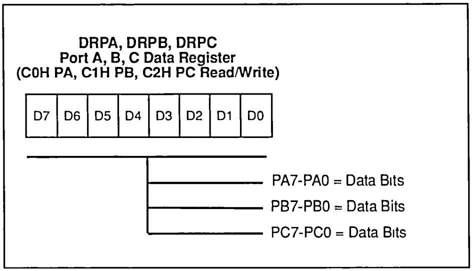
All the I/O registers can be read or written as any other RAM location of the data space, so no extra RAM cell is needed for port data storing and manipulation. During the initialization of the MCU, all the I/O registers are cleared and the input mode with pull-up/no-interrupt is selected on all the pins, thus avoiding pin conflicts.

I/O Pin Programming

Each pin can be individually programmed as input or output with different input and output configurations.

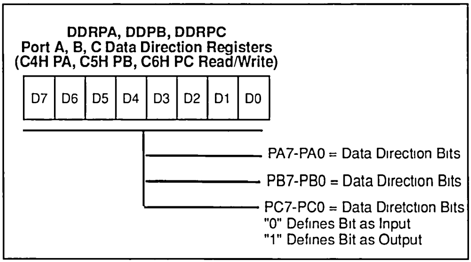
This is achieved by writing to the relevant bit in the data (DR), data direction register (DDR) and option registers (OR). Table 4 shows all the port configurations that can be selected by the user software.

Figure 23. I/O Port Data Registers



Notes:
1. For complete coding explanation refer to Table 4
2 PA4-PA7 and PC4-PC7 are not available on ST6210,E10, ST6220/E20
3. PC0-PC3 are not available as pins

Figure 24. I/O Port Data Direction Registers


DDRPA, DDPB, DDRPC
Port A, B, C Data Direction Registers
(C4H PA, C5H PB, C6H PC Read/Write)
D7
D6
D5
D4
D3
D2
D1
D0
PA7-PA0 = Data Direction Bits
PB7-PB0 = Data Direction Bits
PC7-PC0 = Data Diretction Bits
"0" Defines Bit as Input
"1" Defines Bit as Output

Notes:
1. For complete coding explanation refer to Table 4
2. PA4-PA7 and PC4-PC7 are not available on ST6210,E10, ST6220,E20. They should be programmed in output mode.
3 PC0-PC3 are not available as pins They should be programmed in output mode

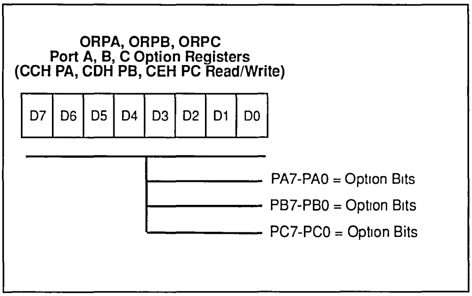
Table 4. I/O Port Options Selection

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, with interrupt
0	1	1	Input	No pull-up, no interrupt (for the PA0-PA3 pins).
			Input	Analog input (for the PA4-PA7, PB0-PB7, PC4-PC7 pin)
1	0	X	Output	10mA sink Open-drain output (for the PA0-PA3 pins)
			Output	5mA sink Open-drain output (for the PA4-PA7, PB0-PB7, PC4-PC7 pins)
1	1	X	Output	Push-pull output

Notes:
X. Means don't care.
1. PA4-PA7 and PC4-PC7 are not available on ST6210,E10, ST6220,E20.

INPUT/OUTPUT PORTS (Continued)

Figure 25. I/O Port Data Option Registers



Notes:
1 For complete coding explanation refer to Table 4
2 PA4-PA7 and PC4-PC7 are not available on ST6210,E10, ST6220,E20.
3 PC0-PC3 are not available as pins

Input Option Description

Pull-up, High Impedance Option. All the input lines can be individually programmed with or without an internal pull-up according to the codes programmed in the OR and DR registers (see table 4). If the pull-up option is not selected, the input pin is in the high-impedance state.

Interrupt Option. All the input lines can be individually connected by software to the interrupt lines of the ST62XX Core according to the codes programmed in the OR and DR registers (see table 4). The pins of Port A are AND-connected to the interrupt associated to the vector #1. The pins of Port B & C are AND-connected to the interrupt associated to the vector #2. The interrupt modes (falling edge sensitive, rising edge sensitive, low level sensitive) can be selected by software for each port thanks to the IOR register (refer to INTERRUPT PROCESSING chapter for additional information).

Analog Input Option. The sixteen PA4-PA7, PB0-PB7, PC4-PC7 pins can be configured to be analog inputs according to the codes programmed in the OR and DR registers (see table 4). These analog inputs are connected to the on-chip 8-bit Analog to Digital Converter. *ONLY ONE* pin should be programmed as analog input at a time, otherwise the selected inputs will be shorted.

Note

Single bit SET and RES instructions should be used very carefully with Port A, B and C data registers becaues these instructions do an implicite read and write back of the whole addressed register byte. In port input mode however data register address reads from input pins, not from data register latches and data register information in input mode is used to set characteristics of the input pin (interrupt, pull-up, analog input), therefore these characteristics may be unintentionally reprogrammed depending on the state of input pins. As general rule is better to use SET and RES instructions on data register only when the whole port is in output mode. If input or mixed configuration is needed it is recommended to keep a copy of the data register in RAM. On this copy it is possible to use single bit instructions, then the copy register could be written into the port data register.

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

The WAIT and STOP instructions allow the ST6210,E10, ST6215,E15, ST6220,E20 and ST6225,E25 to be used in situations where low power consumption is needed. This can only be achieved however if the I/O pins either are programmed as inputs with well defined logic levels or have no power consuming resistive loads in output mode. The unavailable I/O lines on ST6210,E10 and ST6220,E20 should be programmed in output mode.

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

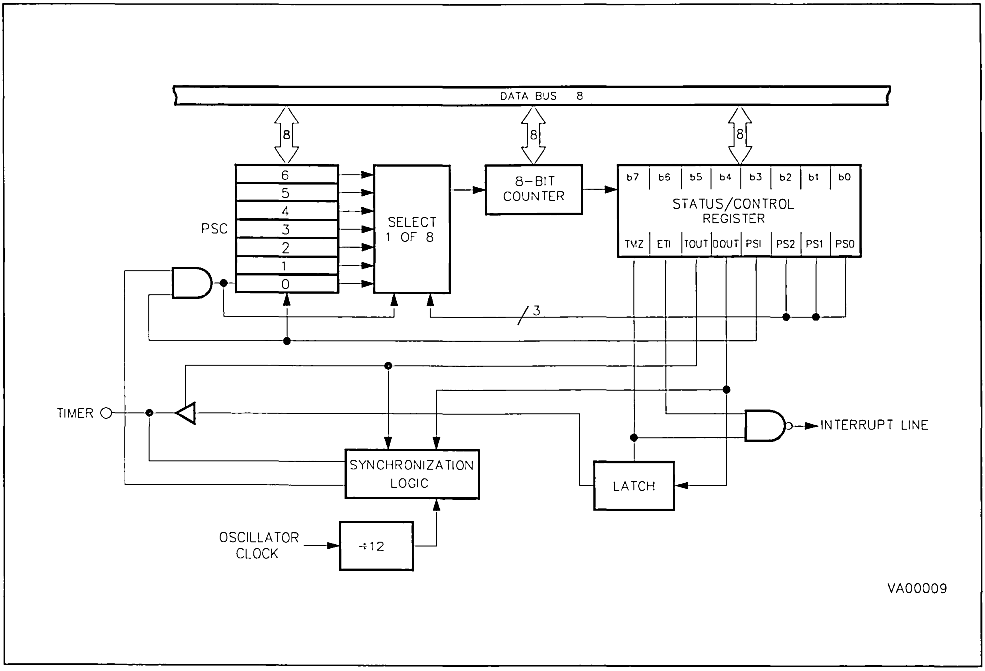
TIMER

The ST6210,E10, ST6215,E15, ST6220,E20 and ST6225,E25 offer one on-chip Timer peripheral consisting of an 8-bit counter with a 7-bit programmable prescaler, thus giving a maximum count of 2^{15}, and a control logic that allows configuring the peripheral in three operating modes. Figure 28 shows the timer block diagram. This timer has the external TIMER pin available for the user. The content of the 8-bit counter can be read/written in the Timer/Counter register TCR that can be addressed in the data space as RAM location at the D3H address. The state of the 7-bit prescaler can be read in the PSC register at the D2H address. The control logic device can be managed thanks to the TSCR register (D4H address) as it is described in the following paragraphs.

The 8-bit counter is decrement 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 one. If the ETI (enable timer interrupt) bit in the TSCR is also set to one an interrupt request, associated to interrupt vector #3, is generated. The interrupt of the timer can be used to exit the MCU from the WAIT mode.

TIMER (Continued)

Figure 26. Timer Peripheral Block Diagram



The prescaler decrements on rising edge. The prescaler input can be the oscillator frequency divided by 12 or an external clock at TIMER pin. Depending on the division factor programmed by PS2/PS1/PS0 (see table 6) bits·in the TSCR, the clock input of the timer/counter register is multiplexed to different sources. On division factor 1, the clock input of the prescaler is also that of timer/counter; on factor 2, bit 0 of prescaler register is connected to the clock input of TCR.

This bit changes its state with the half frequency of prescaler clock input. On factor 4, bit 1 of PSC is connected to clock input of TCR, and so on. On division factor 128, the MSB bit 6 of PSC is connected to clock input of TCR. The prescaler initialize bit (PSI) in the TSCR register must be set to one to allow the prescaler (and hence the counter) to start. If it is cleared to zero then all of the prescaler bits are set to one and the counter is inhibited from counting. The prescaler can be given any value between 0 and 7FH by writing to the D2H address, if bit PSI in the TSCR register is set to one. The tap of the prescaler is selected using the PS2/PS1/PS0 bits in the control register. Figure 27 shows the timer working principle.

Timer Operating Modes

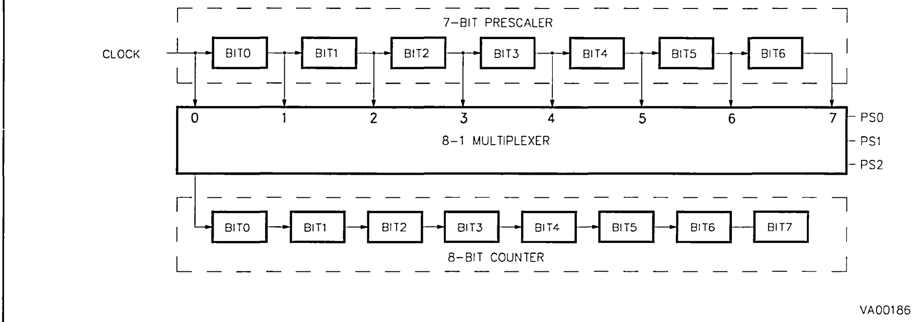
There are three operating modes of the Timer peripheral. The choice of the modes can be done thanks to the bits TOUT and DOUT (see TSCR register). These three modes correspond to the two clock frequencies that can be connected on the 7-bit prescaler (TOSC/12 or TIMER pin signal) and to the output mode.

Gated Mode (TOUT = 0, DOUT = 1). In this mode the prescaler is decremented by the timer clock input (oscillator divided by 12) but ONLY when the signal at TIMER pin is held high (giving a pulse width measurement potential). This mode is selected by having the TOUT bit in TSCR register cleared to 0 (i.e. as input) and DOUT bit set to 1.

Clock Input Mode (TOUT = 0, DOUT = 0). In this mode the TIMER pin is an input and the prescaler is decrement by rising edge events arriving there. The maximum input frequency that can be applied to the external pin in this mode is 1/4 of the oscillator

TIMER (Continued)

Figure 27. Timer Working Principle



frequency when the processor is running but can be higher when the WAIT mode is entered (This is due to the need for synchronization with the Core, this not being necessary during WAITing).

Output Mode (TOUT = 1, DOUT = data out). The timer pin (TIMER) is connected to the DOUT latch. Therefore the timer prescaler is clocked by the prescaler clock input (OSC/12).

The user can select the desired prescaler division ratio through the PS2/PS1/PS0 bits. When 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 provides it for TIMER pin. This operating mode allows an external signal generation on the TIMER pin.

The Table 5 summarizes the TIMER operating modes.

Table 5. Timer Operating Modes

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

Timer Interrupt

When the counter register decrements to zero and the software controlled ETI (enable timer interrupt) bit is set to one then an interrupt request associated to interrupt vector #3 is generated. When the counter decrements to zero also the TMZ bit in the TSCR register is set to one.

Note

On ST6210,15, ST6220,25 the user can select as ROM mask option (see option list at the end of the datasheet) the availability of an on-chip pull-up at TIMER pin. On ST62E10,E15 and ST62E20,E25 this pull-up is not available and should be provided externally.

TMZ is set when the counter reaches 00H ; however, it may be set by writing 00H in the TCR register or setting the bit 7 of the TSCR register. 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 to FFH while the 7-bit prescaler is loaded to 7FH , and the TSCR register is cleared which means that timer is stopped (PSI=0) and timer interrupt disabled.

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

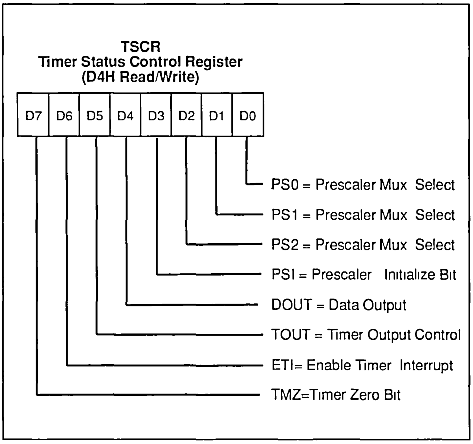
TIMER (Continued)

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.

Timer Registers

Figure 28. Timer Status Control Register


TSCR
Timer Status Control Register
(D4H Read/Write)
D7
D6
D5
D4
D3
D2
D1
D0
PS0 = Prescaler Mux Select
PS1 = Prescaler Mux Select
PS2 = Prescaler Mux Select
PSI = Prescaler Initialize Bit
DOUT = Data Output
TOUT = Timer Output Control
ETI= Enable Timer Interrupt
TMZ=Timer Zero Bit

TMZ. Low-to-high transition indicates that the timer count register has decrement to zero. This bit must be cleared by user software before to start with a new count.

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

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

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

PSI. Used to initialize the prescaler and inhibit its counting while 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.

PSn. These bits select the division ratio of the prescaler register. (see Table 6)

Table 6. Prescaler Division Factors

PS2	PS1	PSO	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

Figure 29. Timer Counter Register

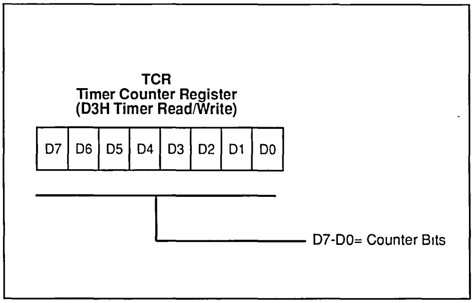
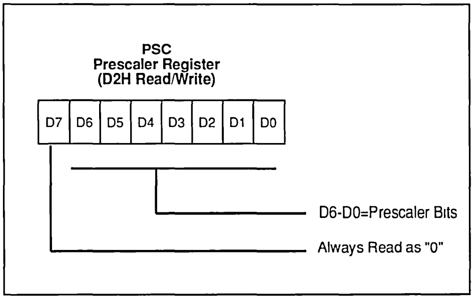


Figure 30. Prescaler Register



DIGITAL WATCHDOG/TIMER

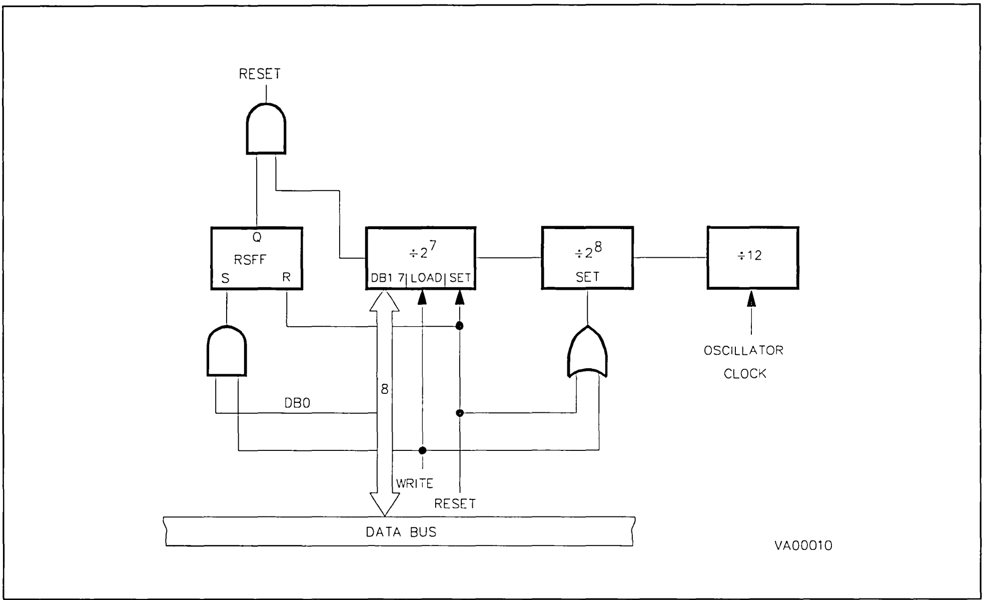
The digital watchdog/timer of the ST6210,E10, ST6215,E15, ST6220,E20 and ST6225,E25 devices consists of a down counter that can be used to provide a controlled recovery from a software upset.

On ST6210,E10, ST6215,E15, ST6220,E20 and ST6225,E25 the watchdog activation (hardware or software) is user selectable; on masked devices the watchdog activation can be selected as ROM option while for EPROM/OTP versions different part numbers are available (see ordering information at the end of the datasheet). If the hardware option is selected the watchdog is automatically initialized after reset so that this function does not need to be activated by the user program. As the watchdog function is always activated this down counter can't be used as a timer. In case of software option the watchdog activation can be controlled by the user software so that the watchdog can be used as a simple 7-bit timer for general purpose counting.

The watchdog is using one data space register (DWDR location D8H). The watchdog register is set to FEH on reset and immediately starts to count down, requiring no software start if the hardware option has been selected. The watchdog time can be programmed using the 6 MSbits in the watchdog register, this gives the possibility to generate a reset in a time between 3072 to 196608 clock cycles in 64 possible steps. (With a clock frequency of 8MHz, this means from 384µs to 24.576ms). The check time can be set differently for different routines within the general program. The reset is prevented if the register 'is reloaded with the desired value before bits 2-7 decrement from all zeros to all ones.If the software option is selected the timer option (watchdog deactivated) there are 7 available counter bits. This is because when the cell is used as watchdog function, bit 1 of the register is used for managing the watchdog.
If the watchdog is active (either by hardware or software activation) the STOP instruction is deactivated and a WAIT instruction is automatically executed instead of a STOP.Bit 1 of the watchdog register (set to one at reset) can be used to generate a software reset if cleared to zero.

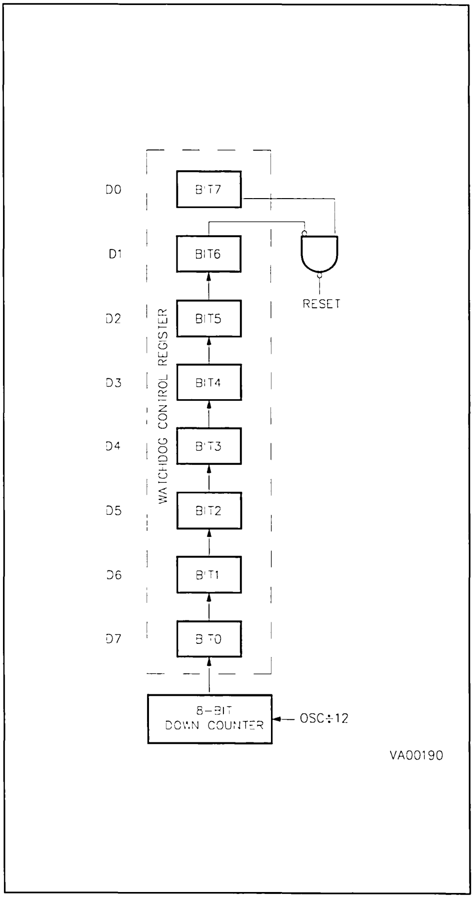
If the software option is selected, after a reset, the Watchdog/timer is in off-state. The watchdog should be activated inside the Reset restart routine by writing a "1" in watchdog/timer register bit 0 (this is automatically done in the hardware activated option). Bit one of this register must be set to one before to program bit zero as otherwise a Reset will be immediately generated when bit 0 is set. This allows the user to generate a reset by software (bit 0=1, bit 1=0). Once bit 0 is set, it can not be cleared by software without generating a Reset.
Figure 31 shows the watchdog block diagram while Figure 32 shows its working principle.

Figure 31. Digital Watchdog/Timer Block Diagram



DIGITAL WATCHDOG/TIMER (Continued)

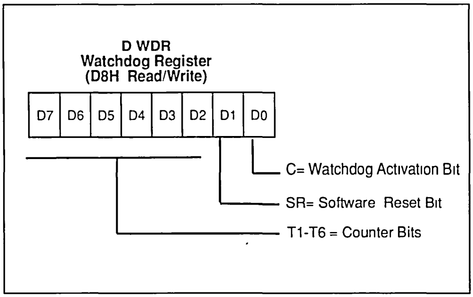
Figure 32. Digital Watchdog Timer Working Principle



Note

In many applications the user may need to synchronize the RESET with the external circuitry and so when the watchdog initiates the ST6210,E10, ST6215,E15 reset the external RESET pin will be held low until the on-chip reset circuit ensures the good start-up condition (see RESET description for additional information). This time is at least 50ns.

Figure 33. Watchdog Register



C. This is the watchdog activation bit. This bit is hardware set to one if hardware option is selected and the user can't change the value of this bit (the watchdog is always active). When the software option is selected, if this bit is set to one the watchdog function will be activated. When cleared to zero it allows the use of the counter as a 7-bit timer.

SR. This bit is set to one during the reset and will generate a software reset if cleared to zero. When C=0 (watchdog disabled, software option only) it is the MSB of the 7-bit timer.

T1-T6. These are the watchdog counter bits. It should be noted that D7 (T1) is the LSB of the counter and D2 (T6) is the MSB of the counter, these bits are in the opposite order to normal.

Application Notes

The hardware activation option is very useful when the external circuitry may inject noises on the reset pin, where there is an unstable supply voltage, or RF influence or other similar phenomena.If the watchdog software activation is selected and the watchdog is not used during power-on reset external noise may cause the undesired activation of the watchdog with a generation of an unexpected reset. To avoid this risk two, two additional instructions, that check the state of the watchdog and eventually reset the chip, are needed within the first 27 instructions after the reset. These instructions are:

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

These instructions should be executed at the very beginning of the customer program.

If the watchdog is used (both hardware or software activated), during power-on reset the watchdog register may be set to a low value, that could give a reset after 28 instructions earliest. To avoid undesired resets, the watchdog must be set to the desired value within the first 27 instructions, the best is to put at the very beginning.

DIGITAL WATCHDOG/TIMER (Continued)

Alternatively the normal legal state can be checked with the following short routine:

```
ldi a, 0FEH
and a, WD
cpi a, 0FEH
jrz #+3
ldi WD, 0FDH
```

This sequence is recommended for security applications, where possible stack confusion error loops must be avoided and the watchdog must only be refreshed after extensive checks.

8-BIT A/D CONVERTER

The ST6210,E10, ST6215,E15, ST6220,E20 and ST6225,E25 A/D converter is an 8-bit analog to digital converter with 8 (PB0-PB7 on ST6210,E10, ST6220,E20) or 16 (PA4-PA7, PB0-PB7, PC4-PC7 on ST6215,E15, ST6225,E25) analog inputs (as alternate functions of I/O lines) offering 8-bit resolution with + 1/2 bit of linearity and a conversion time of 70µs (clock frequency of 8MHz). The A/D peripheral converts 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, the A/D converter accuracy is decreased.

The selection of the pin signal that has to be converted is done by configuring the related I/O line as analog input through the I/O ports option and data registers (refer to I/O ports description for additional information). It must be underlined that only one I/O line can be configured as analog input at a time. The user must avoid the situation in which more than one I/O is selected to be analog input to avoid malfunction of the chip. The ADC uses two registers in the data space: the ADC data conversion register which stores the conversion result and the ADC control register 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 has been finished this 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 are valid. Each conversion has to be separately initiated by writing to the STA bit.The STA bit is continually being scanned so that if the user sets it to 1 while a previous conversion is in progress then a new conversion is started before the previous one has been completed.The start bit (STA) is a write only bit, any attempt to read it will show a logical 0.The A/D converter has a maskable interrupt associated to the end of conversion. This interrupt is associated to the 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 converter can resolve the input voltage with an accuracy of:

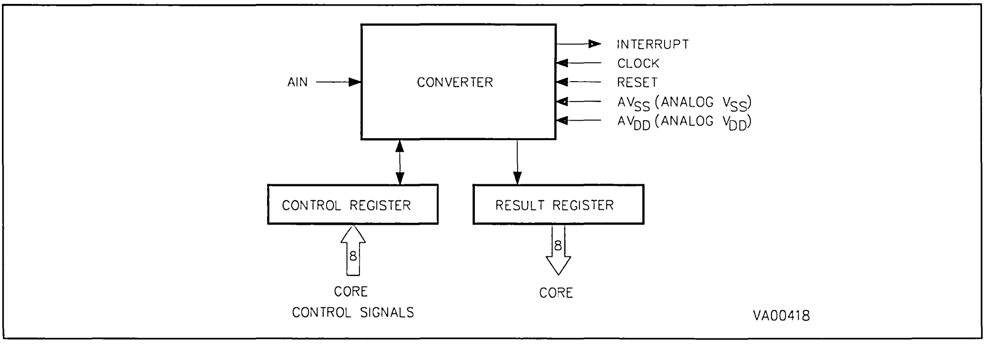
$$\frac{V_{DD} - V_{SS}}{256}$$

So if operating with a supply voltage of 5V the resolution is about 20mV. *The Input voltage (Ain) which has to be converted must be constant for 1µs before conversion and remain constant during the conversion.*

If the user wishes to reduce the power consumption of the devices by turning off the ADC peripheral this can be done using the PDS bit in the ADC control register.If PDS=1 the A/D is supplied and enabled for conversion, if PDS=0 the A/D is in power-down mode with no power consumption. This bit must be set at least one instruction before the beginning of the conversion to allow the stabilization of the A/D converter.*This action is needed also before entering the STOP instruction as the A/D comparator is not automatically disabled by the STOP mode*

During reset any conversion in progress is stopped, the control register is reset to all zeros and the A/D interrupt is masked (EAI=0). Figure 34 shows the A/D Converter block diagram.

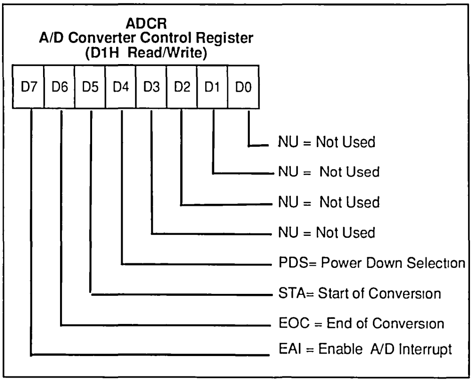
Figure 34. A/D Converter Block Diagram



8-BIT A/D CONVERTER (Continued)

A/D Converter Registers

Figure 35. A/D Converter Control Register



EAI. If this bit is set to one the A/D interrupt (vector #4) is enabled, when EAI=0 the interrupt is disabled.

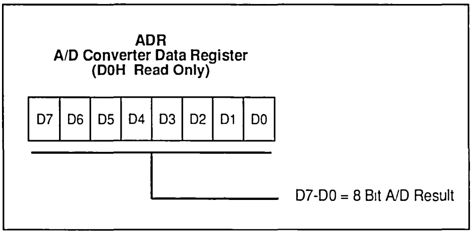
EOC. This read only bit indicates when a conversion has been completed. This bit is automatically reset to zero when the STA bit is written. If the user is using the interrupt option then this bit can be used as an interrupt pending bit. Data in the data conversion register are valid only when this bit is set to one.

STA. Writing a '1' in this bit will start a conversion on the selected channel and automatically reset to zero 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.

PDS. This bit activates the A/D converter if set to 1. Writing a zero into this bit will put the ADC in power down mode (idle mode).

D3-D0. Not used

Figure 36. A/D Converter Data Register



D7-D0. These are the conversion result bits; the register is read only and stores the result of the last conversion. The contents of this register are valid only when EOC bit in the ADCR register is set to one (end-of-conversion).

A/D Converter Specifications

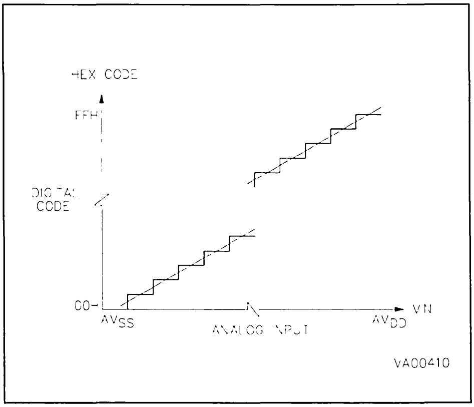
This paragraph deals with a short description of main A/D converter specifications and parameters (see Electrical Characteristics for detailed values and test conditions).

Resolution. This parameter represents the input voltage change needed to cause the output of an A/D to change between one code and the next adjacent code. An A/D converter with "n" switches can convert 1 part in 2^n. As a result the least significant increment, usually called least significant bit (LSB), is 2-n while the most significant bit (MSB) has a weight of 2-1. The ST6210,E10, ST6215,E15, ST6220,E20 and ST6225,E25 A/D converter, that has an 8-bit resolution, may convert 1 part in 28 (1 part in 256). The resolution is a design dependent parameter and is not representative of the A/D converter accuracy or linearity.

Quantizing Error. This error represents the maximum deviation from the straight line transfer function of a perfect A/D converter. As the A/D converter gives in a digital finite code an analog input, only an infinite resolution will give a quantizing error equal to zero. The ST6210,E10, ST6215,E15, ST6220,E20 and ST6225,E25 A/D converter can guarantee a maximum quantizing error of +1/2 LSB.

Non Linearity. This parameter describes the deviation of the A/D from a linear transfer curve (represented in Figure 37). The non-linearity error (often specified has linearity error) is not comprehensive of the quantization, zero or scale errors. A specification of ±1/2 LSB implies error in addition to quantization error. Non linearity errors are usually expressed in % of FS or fractional LSB. See Figure 38.

Figure 37. Ideal A/D Converter Transfer Characteristics



8-BIT CONVERTER (Continued)

Full Scale Error. This parameter is the deviation of actual input voltage from the design expected value for a full scale output code (for ST6210,E10, ST6215,E15, ST6220,E20 and ST6225,E25 full scale=FFH). This error is related to variations in the reference voltage (V$_{DD}$ and V$_{SS}$ in ST6210,E10, ST6215/E15, ST6220,E20 and ST6225,E25), amplifier gain, etc.(see Figure 39).

Zero Offset. This parameter represents the mean input voltage needed to obtain the zero code output (See Figure 40). This kind of error is usually related to A/D amplifier or comparator input voltage or current offset and can be adjusted with an external potentiometer. Zero offset is usually expressed in % of FS or fractional LSB.

Figure 38. Non Linearity Conversioln Error

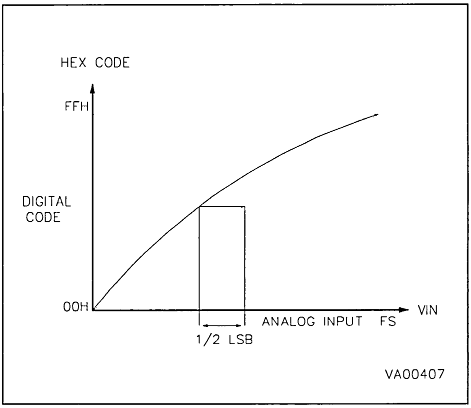


Figure 39. Full Scale Conversion Error

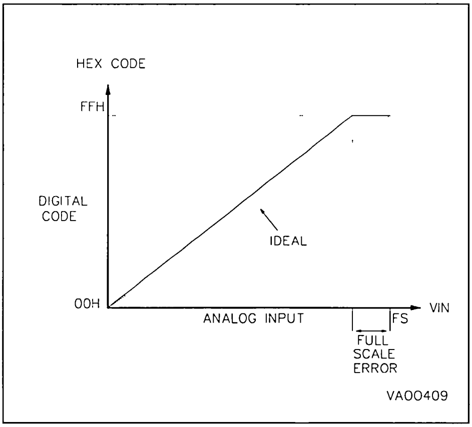
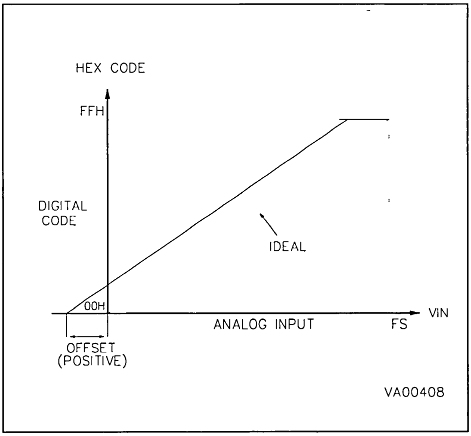


Figure 40. Zero Offset Conversion Error



Note

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 needed. This is because such switching will affect the supply voltages which are used for comparisons.

Low pass filters should be used at the analog input pins. For 70µs - 8 bit conversions the impedance of the analog voltage sources should be less than 30kOhm while the impedance of the reference voltage should not exceed 2kOhm.

The analog voltage applied on the input pins must be present at less 1µs before the beginning of the conversion and all along the conversion. The variation of the analog input voltage must not exceed 10mV during the conversion time to obtain the best resolution.

The accuracy of the conversion depends on the quality of the power supply voltages (V$_{DD}$ and V$_{SS}$). So, the user must specially take care of applying regulated reference voltage on the V$_{DD}$ and V$_{SS}$ pins (the variation of the power supply voltage must be inferior to 5V/ms).

It must be observed that the more accurate measurements are obtained on the PC4-PC7 pins, but in any way, no pin must be switched during the conversion to avoid any noise disturbance.

The resolution of the conversion can be improved if the power supply voltage (V$_{DD}$) of the microcontroller becomes lower. For instance, if V$_{DD}$ = 3V, a 15mV resolution can be guaranteed.

8-BIT A/D CONVERTER (Continued)

In order to optimize the resolution of the conversion, the user can configure the microcontroller in the WAIT mode because this mode allows the minimization of the noise disturbances and the variations of the power supply voltages due to output the switching of the outputs. Nevertheless, it must be take care of executing the WAIT instruction as soon as possible after the beginning of the conversion because the execution of the WAIT instruction may provide a small variation of the V_{DD} voltage (the negative effect of this variation is minimized at the beginning of the conversion because the latter is less sensitive than the end of the conversion when the less significant bits are determined). The best configuration from a accuracy point of view is the WAIT mode with the Timer stopped. Indeed, only the ADC peripheral and the oscillator are still working. The MCU has to be wake-up from the WAIT mode by the interrupt of the ADC peripheral at the end of the conversion. It must be noticed that the wake-up of the microcontroller could be done also with the interrupt of the TIMER, but in this case, the Timer is working and some noise could disturb the converter in terms of accuracy.

SOFTWARE DESCRIPTION

The ST62XX 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 ST62XX 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.

Addressing Modes

The ST62XX Core has nine addressing modes which are described in the following paragraphs. The ST62XX Core uses three different address spaces : 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 four 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 that is processed by the instruction is stored in the location that 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) that 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 that 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 that characterize the kind of the test, one bit that determines whether the branch is a forward (when it is 0) or backward (when it is 1) branch and the four less significant bits that give the span of the branch (0H to FH) that 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.

SOFTWARE DESCRIPTION (Continued)

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 -126 to +129. 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.

Instruction Set

The ST62XX Core has a set of 40 basic instructions. When these instructions are combined with nine addressing modes, 244 usable opcodes can be obtained. They can be divided into six different types:load/store, arithmetic/logic, conditional branch, control instructions, jump/call, bit manipulation. The following paragraphs describe the different types.
All the instructions within 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 an immediate data.
See Table 7.

Table 7. 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

SOFTWARE DESCRIPTION (Continued)

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. See Table 8.

Table 8. 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	*
AND A, (Y)	Indirect	1	4	Δ	*
AND A, rr	Direct	2	4	Δ	*
ANDI A, #N	Immediate	2	4	Δ	*
CLR A	Short Direct	2	4	Δ	Δ
CLR rr	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 Registers
#. Immediate data (stored in ROM memory)
rr. Data space register

Δ Affected
*. Not Affected

SOFTWARE DESCRIPTION (Continued)

Conditional Branch. The branch instructions achieve a branch in the program when the selected condition is met. See Table 9.

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. See Table 10.

Control Instructions. The control instructions control the MCU operations during program execution. See Table 11.

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

Table 9. 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
ee. 8 bit signed displacement in the range -126 to +129
rr. Data space register
Δ Affected
*. Not Affected

Table 10. 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 Affected

Table 11. 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 on ST6010,12,13,14 and a WAIT is automatically executed instead of a STOP (the hardware activated watchdog is present)
Δ Affected
*. Not Affected

Table 12. 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

SOFTWARE DESCRIPTION (Continued)

Opcode Map Summary. The following table contains an opcode map for the instructions used on the MCU.

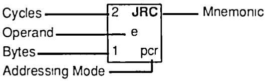
Low Hi	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111	Low Hi
0 0000	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRR b0,rr,ee 3 bt	2 JRZ e 1 pcr	#	2 JRC e 1 prc	4 LD a,(x) 1 ind	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b0,rr 2 b d	2 JRZ e 1 pcr	4 LDI rr,nn 3 imm	2 JRC e 1 pcr	4 LD a,(y) 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	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b0,rr 2 b d	2 JRZ e 1 pcr	4 DEC x 1 sd	2 JRC e 1 pcr	4 LD a,rr 2 dir	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 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b4,rr 2 b d	2 JRZ e 1 pcr	4 COM a 1 inh	2 JRC e 1 pcr	4 CP a,(y) 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	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b4,rr 2 b d	2 JRZ e 1 pcr	4 LD x,a 1 sd	2 JRC e 1 pcr	4 CP a,rr 2 dir	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	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b2,rr 2 b d	2 JRZ e 1 pcr	2 RETI 1 inh	2 JRC e 1 pcr	4 ADD a,(y) 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	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b2,rr 2 b d	2 JRZ e 1 pcr	4 DEC y 1 sd	2 JRC e 1 pcr	4 ADD a,rr 2 dir	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	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b6,rr 2 b d	2 JRZ e 1 pcr	2 STOP 1 inh	2 JRC e 1 pcr	4 INC (y) 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	#	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b6,rr 2 b d	2 JRZ e 1 pcr	4 LD y,a 1 sd	2 JRC e 1 pcr	4 INC rr 2 dir	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	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b1,rr 2 b d	2 JRZ e 1 pcr	#	2 JRC e 1 pcr	4 LD (y),a 1 ind	8 1000
9 1001	2 JRNZ 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	#	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b1,rr 2 b d	2 JRZ e 1 pcr	4 DEC v 1 sd	2 JRC e 1 pcr	4 LD rr,a 2 dir	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	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b5,rr 2 b d	2 JRZ e 1 pcr	4 RLC a 1 inh	2 JRC e 1 pcr	4 AND a,(y) 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	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b5,rr 2 b d	2 JRZ e 1 pcr	4 LD v,a 1 sd	2 JRC e 1 pcr	4 AND a,rr 2 dir	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	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b3,rr 2 b d	2 JRZ e 1 pcr	2 RET 1 inh	2 JRC e 1 pcr	4 SUB a,(y) 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	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b3,rr 2 b d	2 JRZ e 1 pcr	4 DEC w 1 sd	2 JRC e 1 pcr	4 SUB a,rr 2 dir	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	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b7,rr 2 b d	2 JRZ e 1 pcr	2 WAIT 1 inh	2 JRC e 1 pcr	4 DEC (y) 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	#	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b7,rr 2 b d	2 JRZ e 1 pcr	4 LD w,a 1 sd	2 JRC e 1 pcr	4 DEC rr 2 dir	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



ABSOLUTE MAXIMUM RATINGS

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

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

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

$$Tj = T_A + PD \times RthJA$$

where: T_A = Ambient Temperature.
RthJA = Package thermal resistance (junction-to ambient).
PD = Pint + Pport.
Pint = I_{DD} x V_{DD} (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	V
V_O	Output Voltage	V_{SS} – 0.3 to V_{DD} + 0.3	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
T_j	Junction Temperature	150	°C
T_{STG}	Storage Temperature	– 60 to 150	°C

Note: 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

THERMAL CHARACTERISTIC

Symbol	Parameter	Test Conditions	Value			Unit
			Min.	Typ.	Max.	
RthJA	Thermal Resistance	Plastic DIP 20 Plastic DIP 28 Plastic SO 20 Plastic SO 28			60 55 80 75	°C/W

RECOMMENDED OPERATING CONDITIONS

Symbol	Parameter	Test Conditions	Value			Unit
			Min.	Typ.	Max.	
T_A (4)	Operating Temperature	1 Version 6 Version	0 – 40		70 85	°C
V_{DD}	Operating Supply Voltage		3		6	V
f_{OSC}	Oscillator Frequency	V_{DD} = 4.5 to 6.0V	0.01		8	MHz
f_{OSC}	Oscillator Frequency	V_{DD} = 3.5V	0.01		2	MHz
f_{OSC}	Oscillator Frequency	V_{DD} = 3.0V	0 01		1	MHz
AV_{DD}(1) AV_{SS}	Analog Supply Voltage	$V_{SS} \leq AV_{SS} < AV_{DD} \leq V_{DD}$	V_{SS}		V_{DD}	V
$I_{inj}+$	Pin Injection Current (positive) Digital Input (2)	V_{DD} = 4.5 to 5.5V T_A = –40 to 85°C			+ 5	mA
$I_{inj}-$	Pin Injection Current (negative) Digital Input (2)	V_{DD} = 4.5 to 5.5V T_A = –40 to 85°C			– 5	mA
$I_{inj}+$	Pin Injection Current (positive) Analog Inputs	V_{DD} = 4.5 to 5.5V T_A = –40 to 85°C			+ 5	mA
$I_{inj}-$	Pin Injection Current (negative) Analog Inputs	V_{DD} = 4.5 to 5.5V T_A = –40 to 85°C			note 3,	

Notes:

1. An oscillator frequency above 1MHz is recommended for reliable A/D results
2 A current of ± 5mA can be forced on each pin of the digital section without affecting the functional behaviour of the device. For a positive current injected into one pin, a part of this current (~ 10%) can be expected to flow from the neighbouring pins. A current of – 5mA can be forced on one input of the analog section at a time (or – 2.5mA for all inputs at a time) without affecting the conversion.
3 If a total current of – 1mA is flowing into the single analog channel or if the total current flowing into all the analog inputs is of 1mA, all the conversion is resulting shifted of + 1 LSB. If a total positive current of + 5mA is flowing into the single analog channel or if the total current flowing into all the analog inputs is of 5mA, all the conversion is resulting shifted of + 2 LSB.
4. EPROM write operation only at 25°C

DC ELECTRICAL CHARACTERISTICS
(T_A= – 40 to + 85˚C unless otherwise specified)

Symbol	Parameter	Test Conditions	Value			Unit
			Min.	Typ.	Max.	
V_{IL}	Input Low Level Voltalge	RESET Pin V_{DD}= 3V V_{DD}= 5V			1 1.6	V
V_{IH}	Input High Level Voltage	RESET Pin V_{DD}= 3V V_{DD}= 5V	2 3.4			V
I_{IL} I_{IH}	Input Leakage Current	RESET Pin Vin= V_{DD} (1) Vin= V_{DD} (2) Vin= V_{SS}	– 8	– 16	10 1 –30	µA mA µA
V_{IL}	Input Low Level Voltage	NMI,TIMER V_{DD}=3V V_{DD}=5V			0.3x V_{DD}	V
V_{IH}	Input High Level Voltage	NMI,TIMER V_{DD}= 3V V_{DD}= 5V	0.7x V_{DD}			V
V_{OL}	Low Level Output Voltage	TIMER, I_{OL}= 5.0mA V_{DD}= 5.0V			0.2x V_{DD}	V
V_{OH}	High Level Output Voltage	TIMER, I_{OL}= – 5.0mA V_{DD}= 5.0V	0.65x V_{DD}			V
I_{IL} I_{IH}	Input Leakage Current	TIMER, Vin= V_{DD} or V_{SS} V_{DD}= 3.0V V_{DD}= 5.5V		0.1 0.1	1.0 1.0	µA
I_{DD}	Supply Current RESET Mode	V_{RESET} = V_{SS} fOSC = 8Mhz			3.5	mA
I_{DD}	Supply Current RUN Mode	fOSC= 8MHz, ILoad= 0mA V_{DD}= 5.0V			3.5	mA
I_{DD}	Supply Current WAIT Mode	fOSC= 8MHz, (4) ILoad= 0mA V_{DD}= 5.0V			1.5	mA
I_{DD}	Supply Current STOP Mode (3)	ILoad= 0mA V_{DD}= 5.0V			10	µA.

Notes:
1 No Watchdog Reset activated
2. Reset generated by Watchdog
3. When the watchdog function is activated the STOP instruction is deactivated WAIT instruction is automatically executed.
4. Timer and A/D in OFF state

I/O Ports

Symbol	Parameter	Test Conditions	Value			Unit
			Min.	Typ.	Max.	
V_{IL}	Input Low Level Voltage	I/O Pins			0.3x V_{DD}	V
V_{IH}	Input High Level Voltage	I/O Pins	0.7x V_{DD}			V
V_{OL}	Low Level Output Voltage	I/O Pins, I_O= 10µA (sink) V_{DD}= 3.0V V_{DD}= 5.0V			0.1 0.1	V
V_{OL}	Low Level Output Voltage	I/O Pins, I_{OL}= 5.0mA V_{DD}= 5.0V			0.8	V
V_{OL}	Low Level Output Voltage, PA0-PA3 Only	I/O Pins, I_{OL} = 10mA V_{DD} = 5.0V			0.8	V
V_{OH}	High Level Output Voltage	I/O Pins, I_O= – 10µA(sink) V_{DD}= 3.0V V_{DD}= 5.0V	2.9 4.9			V
V_{OH}	High Level Output Voltage	I/O Pins, I_{OL}= – 5.0mA V_{DD}= 5.0V	3.5			V
I_{IL} I_{IH}	Input Leakage Current	I/O Pins, Vin= V_{DD} or V_{SS} V_{DD}= 3.0V V_{DD}= 5.5V		0.1 0.1	1.0 1.0	µA
R_{PU}	Pull-up Resistor	I/O Pins Vin= 0V	50	100	200	KΩ

AC ELECTRICAL CHARACTERISTICS

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

Symbol	Parameter	Test Conditions	Value			Unit
			Min.	Typ.	Max.	
f_{OSC}	Oscillator Frequency	V_{DD} = 3.0V V_{DD} = 4.5V V_{DD} = 5.5V	0.01 0.01 0.01		1 8 8	MHz
t_{ILH}	Interrupt Pin Maximum Pulse Widht	V_{DD} = 3.0V V_{DD} = 4.5V V_{DD} = 5.5V			no limit	µs
t_{SU}	Oscillator Start-up Time			10	100	ms
t_{SR}	Supply Rise Time		0.01		100	ms
t_{REC}	Supply Recovery Time		100			ms
T_W	Minimum Pulse Width	NMI Pin V_{DD}=5V	10			µs
T_W	Minimum Pulse Width	RESET Pin	tcyc			second
C_{IN}	Input Capacitance	All Inputs Pins			10	pF
C_{OUT}	Output Capacitance	All outputs Pins			10	pF

A/D CONVERTER CHARACTERISTICS

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

Symbol	Parameter	Test Conditions	Value			Unit
			Min.	Typ.	Max.	
Res	Resolution		8	8	8	Bit
Lin	Non Linearity	Max deviation from the best straight line			± 1/2	LSB
Qe	Quantization Error	Uncertainly due to converter resolution			± 1/2	LSB
ZO	Zero Offset	$V_{in} = AV_{SS}$			1	LSB
FSO	Full Scale Offset	$V_{in} = AV_{DD}$			1	LSB
t_C(1)	Conversion Time	f_{OSC} = 8MHz		70		µs
V_{AN}	Conversion Range		AV_{SS}		AV_{DD}	V
ZIR	Zero Input Reading	Conversion result when $V_{in} = AV_{SS}$	00			Hex
FSR	Full Scale Reading	Conversion result when $V_{in} = AV_{DD}$			FF	Hex
AV_{SS}(2) AV_{DD}	Analog Reference		V_{SS}		V_{DD}	V
AD_I	Analog Input Current During Conversion	V_{DD}= 4.5V			1.0	µA
AC_{IN}(3)	Analog Input Capacitance				2	pF
ASI	Analog Source Impedance				30	KΩ
SSI	Analog Reference Supply Impedence				2	KΩ

Notes:

1. With oscillator frequencies less than 1MHz, the A/D Converter accuracy is decreased. It is recommended not to use the A/D with fOSC < 1Mhz.
2 In ST6210,E10, ST6215,E15, ST6220,E20 and ST6225,E25 AV_{SS} and AV_{DD} are internally connected to digital V_{SS} and V_{DD}
3 Excluding Pad Capacitance.

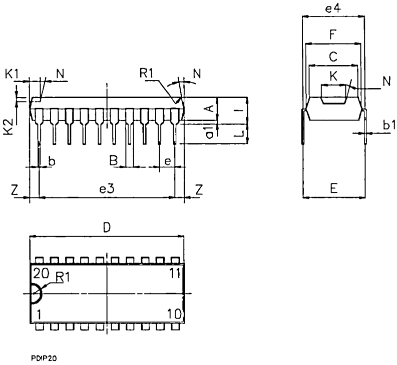
TIMER CHARACTERISTICS

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

Symbol	Parameter	Test Conditions	Value			Unit
			Min.	Typ.	Max.	
t_{RES}	Resolution		$\frac{12}{f_{OSC}}$			second
f_{IN}	Input Frequency on TIMER Pin	V_{DD} = 3.0V V_{DD} = 4.5V			$\frac{f_{OSC}}{4}$	MHz
t_W	Pulse Width at TIMER Pin	V_{DD} = 3.0V V_{DD} = 4.5V V_{DD} = 5.5V	1 125 125			µs ns ns

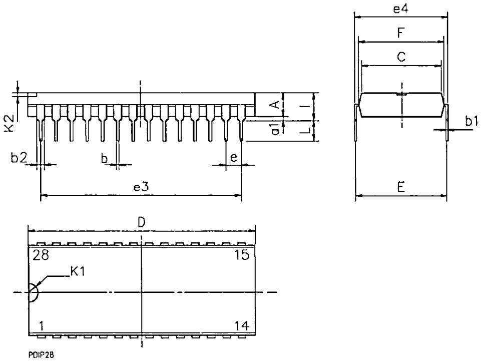
PACKAGES MECHANICAL DATA

Figure 43. 20-Pin Dual in Line Plastic (B), 300-Mil Width



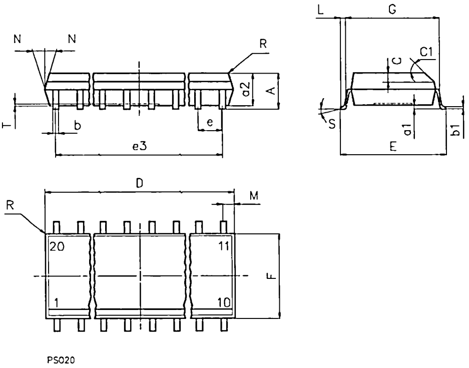
Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A	–	–	–	–	–	–
a1	0 254			0 010		
B	1 39		1 65	0 054		0 064
b		0 45			0 017	
b1		0 25			0 009	
C	–	–	–	–	–	–
D			25 4			1 000
E		8 5			0 334	
e		2 54			0 100	
e3		22 86			0 900	
e4	–	–	–	–	–	–
F			7 1			0 279
I			3 93			0 154
L		3 3			0 129	
Z			1 34			0 052

Figure 44. 28-Pin Dual in Line Plastic (B), 600-Mil Widht



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A	–	–	–	–	–	–
a1		0 63			0 025	
b		0 45			0 018	
b1	0 23		0 31	0 009		0 012
b2		1 27			0 050	
C	–	–	–	–	–	–
D			37 34			1 470
E	15 20		16 68	0 598		0 657
e		2 54			0 100	
e3		33 02			1 300	
e4	–	–	–	–	–	–
F			14 10			0 555
I		4 45			0 175	
L		3 30			0 130	

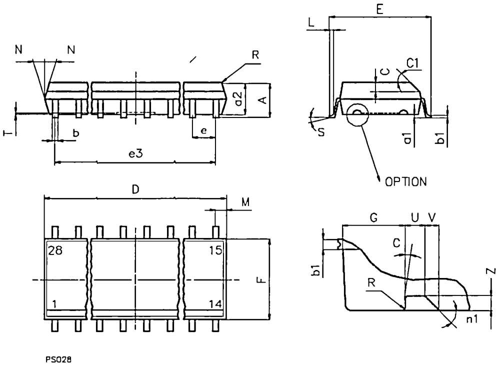
Figure 45. 20-Lead Small Outline Plastic (M), 300-Mil Body Width



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A			2 65			0 104
a1	0 10		0 20	0 003		0 007
a2			2 45			0 096
b	0 35		0 49	0 013		0 019
b1	0 23		0 32	0 009		0 012
C		0 50			0 019	
c1		45 °			45 °	
D	12 60		13 00	0 496		0 511
E	10 00		10 65	0 393		0 419
e		1 27			0 050	
e3		11 43			0 450	
F	7 40		7 60	0 291		0 299
G	8 80		9 15	0 346		0 360
L	0 50		1 27	0 019		0 050
M			0 75			0 029
S			8 °			8 °

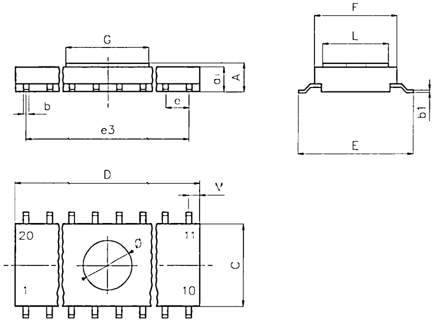
PACKAGES MECHANICAL DATA (Continued)

Figure 46. 28-Lead Small Outline Plastic (M), 300-Mil Body Widht



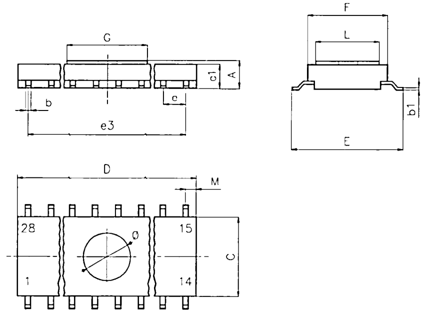
Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A			2 65			0 104
a1	0 10		0 30	0 004		0 011
a2	–	–	–	–	–	–
b	0.35		0.49	0 013		0 019
b1	0 23		0 32	0 009		0 012
C		0 50			0 019	
c1		45 °			45 °	
D	17 70		18 10	0 696		0 712
E	10 00		10 65	0 393		0 419
e		1 27			0 050	
e3	–	–	–	–	–	–
F	7 40		7 60	0 291		0 299
G	–	–	–	–	–	–
L	0 40		1 27	0 015		0 050
M	–	–	–	–	–	–
S			8 °			8 °

Figure 47. 20-Pin Ceramic Small Outline



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A			3 68			0 145
a1	1 80		2 50	0 071		0 098
B	0 39		0 47	0 015		0 019
b1	0 17		0 23	0 007		0 009
C		7 45			0 293	
D	12 60		13 00	0 496		0 512
E	10 00		10 65	0 394		0 419
e		1 27			0 050	
e3		11 43			0 450	
F	8 30		9 20	0 327		0 362
G	9 50		9 85	0 374		0 388
/L	6 50		6 70	0 256		0 264
M			0 80			0 031
			4 48			0 176

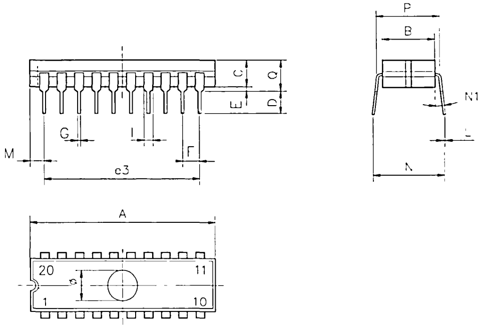
Figure 48. 28-Pin Ceramic Small Outline



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A			3 68			0 145
a1	1 80		2 50	0 071		0 098
B	0 39		0 47	0 015		0 019
b1	0 17		0 23	0 007		0 009
C		7 45			0 293	
D	17 50		18.50	0 689		0.728
E	10 00		10.65	0 394		0 419
e		1 27			0 050	
e3		16 51			0 650	
F	8 30		9 20	0.327		0 362
G	9 50		9.85	0 374		0 388
/L	6 50		6 70	0 256		0 264
M			0 80			0 031
			4 48			0 176

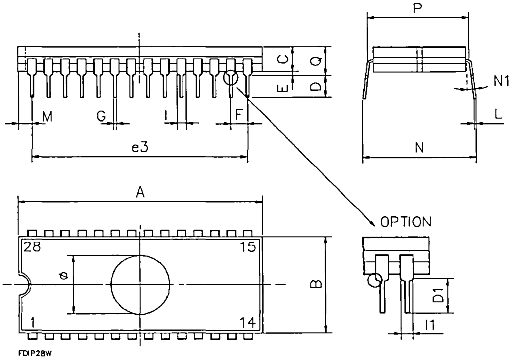
PACKAGES MECHANICAL DATA (Continued)

Figure 49. 20-Lead Freat Seal Ceramic Dual in Line Package, 300-Mil Body Width



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A			26 92			1 059
B			7 80			0 307
C	–	–	–	–	–	–
D	3 40			0 134		
E	0 50		1 78	0 020		0 070
e3		22 86			0 90	
F	2 29		2 79	0 090		0 110
G	0 40		0 55	0 016		0 022
I	1 27		1 52	0 050		0 060
L	0 22		0 31	0 009		0 012
M	0 51		1 27	0 020		0 050
N	–	–	–	–	–	–
N1	0 °		10 °	0 °		10 °
P	7 90		8 13	0 311		0 320
Q			5 71			0 225
∅	4 24		4 39	0 167		0 173

Figure 50. 28-Lead Freat Seal Ceramic Dual in Line Package, 600-Mil Body Widht



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A			38 10			1 500
B	13 05		13 36	514		526
C	3 90		5 08	154		199
D	3 00			118		
E	0 50		1 78	020		070
e3		33 02			1 300	
F	2 29		2 79	090		.110
G	0 40		0 55	016		027
I	1 17		1 42	046		.056
L	0 22		0 31	009		012
M	1 52		2 49	060		098
N						
N1	0°		10°	0°		10°
P	15 40		15 80	606		622
Q			5 71			225
Ø	6 86		7 36	270		290

ORDERING INFORMATION

The following chapter deals with the procedure for transfer the Program/Data ROM codes to SGS-THOMSON.

Communication of the ROM content. To communicate the contents of Program/Data ROM memories to SGS-THOMSON, the customer has to send one 2764 EPROM that must be programmed as follows:

ST6210,ST6215 (2K ROM Devices)

0000H-087FH Reserved (Should be filled with FFH)

0880H-0F9FH User program
0FA0H-0FEFH Reserved (Should be filled with FFH)
0FF0H-0FF7H Interrupt Vectors
0FF8H-0FFBH Reserved (Should be filled with FFH)
0FFCH-0FFDH NMI Interrupt Vector
0FFEH-0FFFH Reset Vector

ST6220,ST6225 (4K ROM Devices)

0000H-007FH Reserved (Should be filled with FFH)
0080H-0F9FH User program

0FA0H-0FEFH Reserved (Should be filled with FFH)
0FF0H-0FF7H Interrupt Vectors
0FF8H-0FFBH Reserved (Should be filled with FFH)
0FFCH-0FFDH NMI Interrupt Vector
0FFEH-0FFFH Reset Vector

The above reserved areas are the same also in OTP devices.

The ROM code must be generated with ST6 Assembler.

All unused bytes must be set FFH. For shipment to SGS-THOMSON the EPROMs should be placed in a conductive IC carrier and packaged carefully.

Listing Generation & Verification. When SGS-THOMSON receives the EPROMs, they are compared and a computer listing is generated from them. This listing refers exactly to the mask that will be used to produce the microcontroller. Then the listing is returned to the customer that must thoroughly check, complete, sign and return it to SGS-THOMSON. SGS-THOMSON will also program one 2764 EPROM from the data file corresponding to the listing to help the customer in its verification. The signed listing constitutes a part of the contractual agreement for the creation of the customer mask. SGS-THOMSON sales organization will provide detailed information on contractual points.

ORDERING INFORMATION (Continued)

ROM DEVICES

Sales Type	ROM x 8	I/O	Additional Features	Temperature Range	Package
ST6210B1 ST6210B6	2K Bytes	12	A/D CONVERTER	0 to +70°C -40 to +85°C	PDIP20
ST6210M1 ST6210M6				0 to +70°C -40 to +85°C	PSO20
ST6215B1 ST6215B6		20		0 to +70°C -40 to +85°C	PDIP28
ST6215M1 ST6215M6				0 to +70°C -40 to +85°C	PSO28
ST6220B1 ST6220B6	4K Bytes	12		0 to +70°C -40 to +85°C	PDIP20
ST6220M1 ST6220M6				0 to +70°C -40 to +85°C	PSO20
ST6225B1 ST6225B6		20		0 to +70°C -40 to +85°C	PDIP28
ST6225M1 ST6225M6				0 to +70°C -40 to +85°C	PSO28

Note: Each ROM content is identifical by 2 alphabetic characters to be added to the sales type.

EPROM DEVICES

Sales Type	ROM x 8	I/O	Additional Features	Temperature Range	Package
ST62E10B1/HWD ST62E10B1/SWD ST62E10M1/HWD ST62E10M1/SWD	2K Bytes	12	EPROM; A/D CONV; HARD WDG EPROM; A/D CONV; SOFT WDG EPROM; A/D CONV; HARD WDG EPROM; A/D CONV; SOFT WDG	0 to +70°C	FDIP20W FDIP20W CSO20W CSO20W
ST62E15B1/HWD ST62E15B1/SWD ST62E15M1/HWD ST62E15M1/SWD		20	EPROM; A/D CONV; HARD WDG EPROM; A/D CONV; SOFT WDG EPROM; A/D CONV; HARD WDG EPROM; A/D CONV; SOFT WDG		FDIP28W FDIP28W CSO28W CSO28W
ST62E20B1/HWD ST62E20B1/SWD ST62E20M1/HWD ST62E20M1/SWD	4K Bytes	12	EPROM; A/D CONV; HARD WDG EPROM; A/D CONV; SOFT WDG EPROM; A/D CONV; HARD WDG EPROM; A/D CONV; SOFT WDG		FDIP20W FDIP20W CSO20W CSO20W
ST62E25B1/HWD ST62E25B1/SWD ST62E25M1/HWD ST62E25M1/SWD		20	EPROM; A/D CONV; HARD WDG EPROM; A/D CONV; SOFT WDG EPROM; A/D CONV; HARD WDG EPROM; A/D CONV; SOFT WDG		FDIP28W FDIP28W CSO28W CSO28W

ORDERING INFORMATION (Continued)

OTP DEVICES

Sales Type	ROM x 8	I/O	Additional Features	Temperature Range	Package
ST62T10B1/HWD ST62T10B6/HWD	2K Bytes	12	OTP; A/D CONV; HARD WDG	0 to +70°C -40 to +85°C	PDIP20
ST62T10B1/SWD ST62T10B6/SWD			OTP; A/D CONV; SOFT WDG	0 to +70°C -40 to +85°C	
ST62T10M1/HWD ST62T10M6/HWD			OTP; A/D CONV; HARD WDG	0 to +70°C -40 to +85°C	PSO20
ST62T10M1/SWD ST62T10M6/SWD			OTP; A/D CONV; SOFT WDG	0 to +70°C -40 to +85°C	
ST62T15B1/HWD ST62T15B6/HWD	2K	20	OTP; A/D CONV; HARD WDG	0 to +70°C -40 to +85°C	PDIP28
ST62T15B1/SWD ST62T15B6/SWD			OTP; A/D CONV; SOFT WDG	0 to +70°C -40 to +85°C	
ST62T15M1/HWD ST62T15M6/HWD			OTP; A/D CONV; HARD WDG	0 to +70°C -40 to +85°C	PSO28
ST62T15M1/SWD ST62T15M6/SWD			OTP; A/D CONV; SOFT WDG	0 to +70°C -40 to +85°C	
ST62T20B1/HWD ST62T20B6/HWD	4K Bytes	12	OTP; A/D CONV; HARD WDG	0 to +70°C -40 to +85°C	PDIP20
ST62T20B1/SWD ST62T20B6/SWD			OTP; A/D CONV; SOFT WDG	0 to +70°C -40 to +85°C	
ST62T20M1/HWD ST62T20M6/HWD			OTP; A/D CONV; HARD WDG	0 to +70°C -40 to +85°C	PSO20
ST62T20M1/SWD ST62T20M6/SWD			OTP; A/D CONV; SOFT WDG	0 to +70°C -40 to +85°C	
ST62T25B1/HWD ST62T25B6/HWD		20	OTP; A/D CONV; HARD WDG	0 to +70°C -40 to +85°C	PDIP28
ST62T25B1/SWD ST62T25B6/SWD			OTP; A/D CONV; SOFT WDG	0 to +70°C -40 to +85°C	
ST62T25M1/HWD ST62T25M6/HWD			OTP; A/D CONV; HARD WDG	0 to +70°C -40 to +85°C	PSO28
ST62T25M1/SWD ST62T25M6/SWD			OTP; A/D CONV; SOFT WDG	0 to +70°C -40 to +85°C	

ST6210, ST6215, ST6220, ST6225 MICROCONTROLLER OPTION LIST

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

SGS-THOMSON Microelectronics references
Device [ ] (d)
Package [ ] (p)
Temperature Range [ ] (t)

For marking one line with 11 characters maximum is possible
Special Marking [ ] (y/n) Line 1 "_ _ _ _ _ _ _ _ _ _ _" (N)

Notes:
[d] 1 = ST6210, 2 = ST6215, 3 = ST6220, 4 = ST6225

[p] B = Dual in Line Plastic, M = Small Outline Plastic,

(t) 1= 0°C to + 70°C, 6 = – 40°C to + 85°C

(N) Letters, digits, '.', '–', '/' and spaces only

Input pull-up selection:
( ) On NMI pin
( ) On TIMER pin

Watchdog Selection:
( ) Hardware Activation
( ) Software Activation

Signature ....
Date ....

ST6-SW

SOFTWARE DEVELOPMENT TOOLS FOR ST6 MCU FAMILY

- Includes:
  - Macro assembler
  - Linker
  - Software simulator
- Runs on MS-DOS systems
- Window based graphic interface
- Extensive symbol manipulation

GENERAL DESCRIPTION

Full software development tools is achieved using the ST6 Software Development Tools consisting of a powerful macro assembler, a linker and a software simulator.

The ST6 Macro assembler accepts a source file written in ST6 assembly language using any text editor package and transforms it in an ST6 executable file.

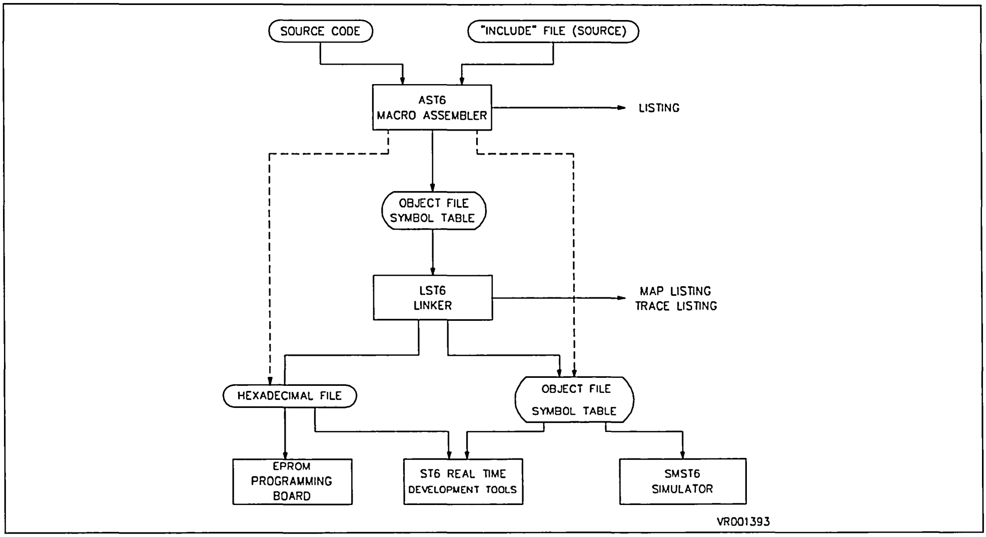
To enable good testability and fast debugging, many application software are made up of several modules, each of them performing an elementary task. Each module is assembled independently of the others, thus producing a number of object files. The ST6 Linker combines these object files into a single executable program. Both object format and hexadecimal format are produced. The hexadecimal file is used to program an EPROM while the object file is used to run the simulator or the debugger.

The ST6 Software Simulator allows the user to debug and execute any executable program written for any member of the ST62/ST63 family of microcontrollers without the aid of additional hardware.

Once debugged with the simulator, the program can be programmed into an EPROM device by using the hexadecimal file and the ST6 programming board. By plugging the EPROM device into the application hardware, simple applications can be debugged without the need of an emulator.

The ST6 Hardware Development Tools are required where high performance debugging is needed.

Figure 1. Development Flow Chart



ST6 ASSEMBLER

- Macro call and conditional assembly
- Extensive symbol manipulation
- Error diagnostics

General Description

The ST6 Macro assembler accepts a source file written in ST6 assembly language and transforms it into an executable file in relocatable object code format. When the whole program is in one file only, the assembler also generates an hexadecimal file (INTEL hex format) ready to be programmed into an EPROM device.

The assembler recognizes the use of section, symbols, macros and conditional assembly directives. In addition, the ST6 Assembler is able to produce detailed assembly listing and symbol cross reference file.

Figure 2. AST6 Directives

.ASCII	Stores in program space a string as a sequence of ASCII codes
.ASCIZ	Same as .ASCII followed by a null character
.BLOCK	Reserves a block of contiguous memory location
.BYTE	Stores successive bytes of data in program space
.DEF	Defines the characteristics of a data space location
.DISPLAY	Displays a string during assembly process
.DP_ON	Segments the data space
.EJECT	Starts a new listing page
.ELSE	Beginning of the alternative part in conditional assembly block
.END	End of source file
.ENDC	End of conditional assembly block
.ENDM	End of a macro definition
.EQU	Assigns the value of an expression to a label
.ERROR	User defined assembly error
.EXTERN	Defines a symbol as external
.IFC	Beginning of conditional assembly block
.INPUT	Includes an additional source file in the present one
.GLOBAL	Defines a symbol as global
.LABEL.W	Initializes Data ROM Window Register
.LABEL.D	Gains access to a label in a Data ROM Window
.LINESIZE	Set listing line length
.LIST	Enables the listing of specified fields of the source file
.MACRO	Beginning of a macro definition
.MEXIT	End of a macro expansion
.NOTRANSMIT	Inhibits symbol transmission to the linker
.ORG	Set current location counter
.PAGE_D	Specifies the page number in data space
.PL	Set listing page length
.PP_ON	Segments the program space in 2K pages
.ROMSIZE	Defines the available ROM size
.SECTION	Provides a logical partitioning of program space
.SET	Same as .EQU, but can be redefined in the source file
.TITLE	Assigns title to the document
.TRANSMIT	Transmits symbol definitions to the linker
.VERS	Defines the target ST6 device
.WARNING	User defined assembly warning
.WINDOW	Defines a continuous relocatable block of program code
.W_ON	Enables the use of the .WINDOW directive
.WORD	Stores successive words of data in program space

ST6 LINKER

- Links up to 32 modules
- Extensive symbol manipulation
- 33 sections (including interrupt vectors)
- Error diagnostics

The ST6 Linker is responsible for combining a number of object files into a single program, associating an absolute address to each section of code, and resolving any external references.

The ST6 Linker produces an hexadecimal file in INTEL format to be down loaded into an EPROM device and an object code file to be used with the simulator. The linker also produces a map file which gives information about the sections, pages, modules and labels. Finally, listing files are produced which update the assembler listings with real addresses of symbols and statements.

This software program allows the user to develop modular programs, which may then be combined and addressed as defined by the user. The flexibility of the ST6 Linker is greatly increased by the use of sections allowing the user to group pieces of software from different modules. The location and the size of each section is user selectable.

ST6 SIMULATOR

- Window based graphic interface
- On line assembler/disassembler
- Supports symbolic debugging
- 128 breakpoints and 128 software traps
- TRACE mode
- I/O and CLOCK simulation

SIMST6 allows the user to debug and execute any program written for any of the current and future members of the ST6 family of microcontrollers, without the aid of additional hardware.

The user specifies the target device, its mapping and the object code file to be used. The simulator functionally duplicates the operation of the ST6 and completely supports the instruction set. I/O channels may be opened, read, and written, in order to simulate the I/O functions of peripherals, while interrupts may be set, and then set pending, in order to simulate the handling of interrupts. The simulator uses the clock frequency assigned by the user, along with the number of clock cycles needed by each instruction to keep track of the real time execution speed.

The ST6 Simulator accepts command lines in both interactive and batch mode.

ORDERING INFORMATION

Sales Type	Description
ST6-SW	ST6 software development tools (includes assembler, linker and emulator)

Note : The ST6 software package is included in all ST6xxx-EMU real time develoment tools.

ST6xxx-EMU

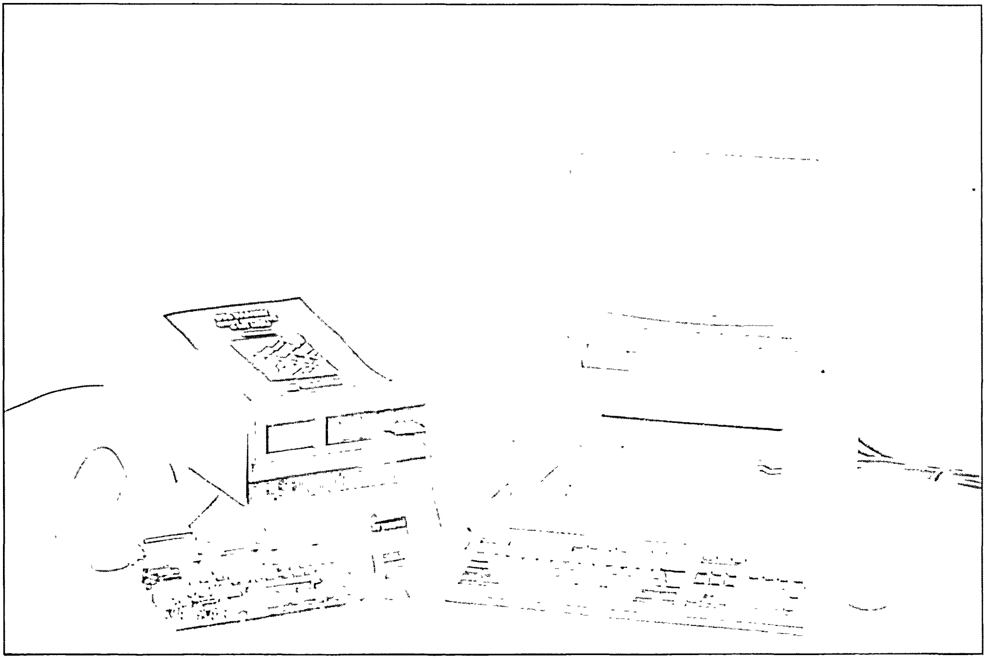
REAL TIME DEVELOPMENT TOOLS FOR ST6 MCU FAMILY

HARDWARE FEATURES

- Supports ST62xx and ST63xx family
- Real time emulation
- 32 KBytes of emulation memory
- Breakpoints on up to 256 events
- Events can be defined on program space, data space and on up to 4 external signals
- 1K of real trace memory
- Tracing of up to 32 bits including 4 external signals

SOFTWARE FEATURES

- Symbolic debugger
- Window based interface
- On line assembler/disassembler
- Log files capable of storing any displayed screen
- Command files able to execute a set of debugger commands



GENERAL DESCRIPTION

The ST6 Real Time Development System is an advanced hardware development system designed and configured to provide comprehensive support for the ST6 family of MCU's.

The mainframe consists of a basic part, common to all ST6 devices, and one (ST62 sub family) or two (ST63 sub family) dedicated board depending of specific device to emulate. Only the dedicated boards have to be changed to emulate a new device within the ST62/ST63 subfamilies.

The software part of the real time emulation tool is the symbolic debugger. It can be run on a PC compatible system and is common to all ST62/ST63 devices. It drives the emulator mainframe through an RS232 channel. The debugger uses a windowed menu driven user interface and enables the user to set the configuration of the emulator.

Once assembled and eventually linked and debugged by using the simulator, the application software is ready to be down loaded into the ST6-EMU. The device probe is connected into the application hardware. The development station will perform a real time emulation of the target device, thus allowing high performance test and debugging of both application hardware and software.

The breakpoints allow user to stop the MCU when the application software reaches selected addresses and/or addresses within a selected ranges and/or on data fetch (or read or write or both) cycles. The user is then able to read and modify any register and memory location. An on line assembler/disassembler is also available to ease the debugging.

The logic analyser can be used when real time emulation is needed. It allows to display the last 1024 cycles. The displayed cycles are either fetch cycles only or fetch cycles and data space accesses. Addresses, data, control/status bits and 4 user signals are displayed using mnemonic and user symbols.

Such a powerful tool enables the user to detect and trap any pattern and thus quickly debug the application. The trapping of random patterns is greatly improved by the capability to quit the emulation session while the emulator continue to run the application software. When the user re-enters the debugger, the emulation session resumes and information about any events of interest will be flashed to the screen in the form of a message.

Log files offer the ability to send any screen display to a text file. In particular, log files are very useful to save the contents of the logic analyser and/or the contents of data registers to be analysed or printed.

Command files can be used to execute a set of debugger commands in order to ease and speed up the emulation session.

A powerful help facility can be called at any time to give additional information about the commands, the processor or the emulator.

When the program is fully debugged, the ST6 EPROM remote programming board can be used to program the emulation device with the INTEL hex format file produced by the linker.

Figure 1. SDBST6 Command Summary

ALL	One Line Assembler
BASE	Change base of numbers
BREAK	Display/set breakpoint
CB	Clear breakpoints
CMP	Compare memory
DL	Display memory in listing ASM form
DM	Display/change memory
DOS	Branch to DOS
DR	Display/change registers
DS	Display symbol table
FM	Fill memory with pattern
GO	Start user program
GRAPH	Return to GRAPHIC interface
HELP	Call HELP utility
HWTEST	Execute diagnostic test
LOAD	Load memory from a file
LCONF	Load data pages configuration
MOVE	Move memory block
NEXT	Single/multi step mode
PM	Display/change paged Data ROM locations
QUIT	Abandon the program
RESET	Reset ST6 core dedications
SAVE	Save memory into a file
SB	Set address breakpoints
SCONF	Save data pages configuration
SEARCH	Search pattern in memory
REM	Put comment in a log file
SET	Set system options
SR	Set register
TRACE	Display traced execution
USE	Execute command file
WR	Display current Working Register set
UPLOAD	Copy ROMulator into HOST

ORDERING INFORMATION

Sales Type	Description
ST621X-EMU	Complete emulator package for ST621X/2X devices (including dedicated board and ST6-SW software package)
ST624X-EMU	Complete emulator package for ST624X devices (including dedicated board and ST6-SW software package)
ST629X-EMU	Complete emulator package for ST629X devices (including dedicated board and ST6-SW software package)
ST63TVS-EMU	Complete emulator package for ST6326, 27, 28, 40, 42, 44, 46, 56, 57 and 58 devices (including dedicated board and ST6-SW software package)
ST638X-EMU	Complete emulator package for ST6385, 86, 87 and 88 devices (including dedicated boards and ST6-SW software package)
ST639X-EMU	Complete emulator package for ST6391, 93, 94, 95, 96 and 99 devices (including dedicated boards and ST6-SW software package)
ST621X-DBE	Separate dedicated board for ST621X devices
ST624X-DBE	Separate dedicated board for ST624X devices
ST629X-DBE	Separate dedicated board for ST629X devices
ST663TVS-DBE	Separate dedicated board for ST63TVS devices
ST638X-DBE	Separate dedicated board for ST638X devices
ST639X-DBE	Separate dedicated board for ST639X devices

Note : The emulator power supply can be adjusted to 220V or 110V

ST62Exx-EPB

EPROM PROGRAMMING BOARD FOR ST62 MCU FAMILY

HARDWARE FEATURES

- Programs the ST62Exx EPROM and OTP MCUs
- Standalone and PC driven modes
- All ST62Exx packages are supported

SOFTWARE FEATURES

- Menu driven software
- S19 or INTEL hex file formats

DESCRIPTION

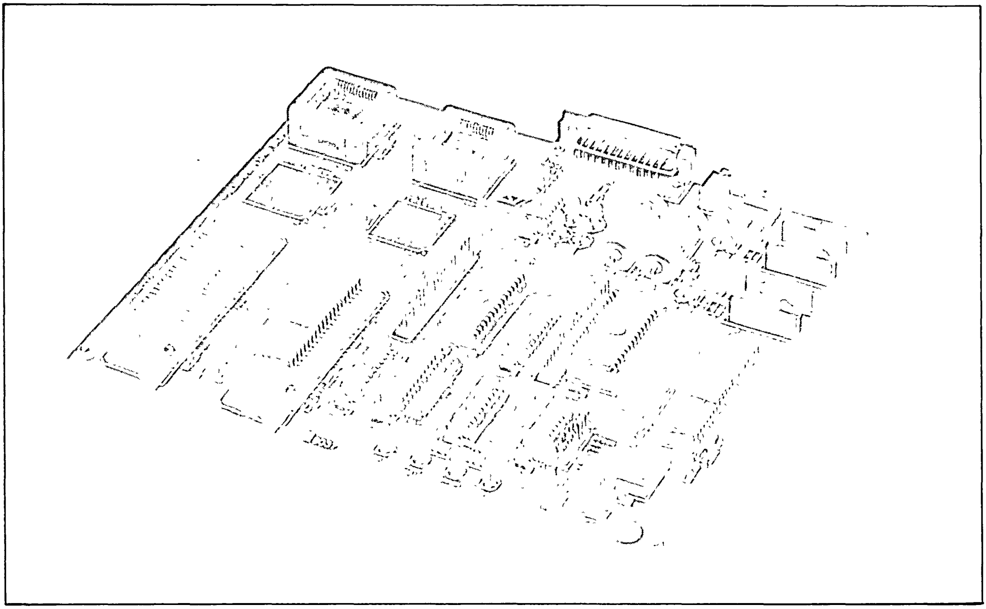
Different programming boards are designed for programming of the various EPROM and OTP devices of the ST62 sub-family. For a particular device, all available packages are supported by the same programming board.

It can run either in standalone or remote mode under control of a DOS compatible PC.

In standalone mode, the microcontrollers can be programmed with a simple key operation directly from a master EPROM device or a master microcontroller. Two colour LEDs indicate the operational pass or fail.

In standalone mode an EPROM memory or a master MCU is plugged into the programming board. The code from the EPROM or the master MCU is read and programmed into the ST62 EPROM or OTP device. Both VERIFY and BLANK CHECK functions are provided.

In remote mode, the programming board is connected to a DOS compatible PC through an RS232 serial channel. Object code in either S19 or INTEL HEX format is read from disk file to program the ST62 EPROM or OTP device. The menu driven software also offers VERIFY, BLANK CHECK, READ MASTER and other utility functions.



ORDERING INFORMATION

Sales Types [1]	Supported Devices	Supported Packages
ST62E1X- EPB/xxx	ST62E10 [2] ST62T10 [2] ST62E15 [2] ST62T15 [2] ST62E20 [2] ST62T20 [2] ST62E25 [2] ST62T25 [2]	DIP20 DIP28 SO20 SO28
ST62E4X-EPB/xxx	ST62E40 ST62E45 ST62T40 ST62T45	QFP52 QFP80
ST62E94-EPB/xxx	ST62E94 ST62T94	DIP28 SO28

Notes :

1 ST62Exx-EPB/110 : 110V Power Supply
ST62Exx-EPB/220 · 220V Power Supply

2. Both /HWD and /SWD options are supported

SGS-THOMSON MICROELECTRONICS

ST62Exx-GANG

GANG PROGRAMMER FOR ST62 MCU FAMILY

HARDWARE FEATURES

- Programs simultaneously up to 10 ST62Exx EPROM and OTP MCUs
- Standalone and PC driven modes
- DIP and SO packages supported

SOFTWARE FEATURES

- Menu driven software
- S19 or INTEL hex file format

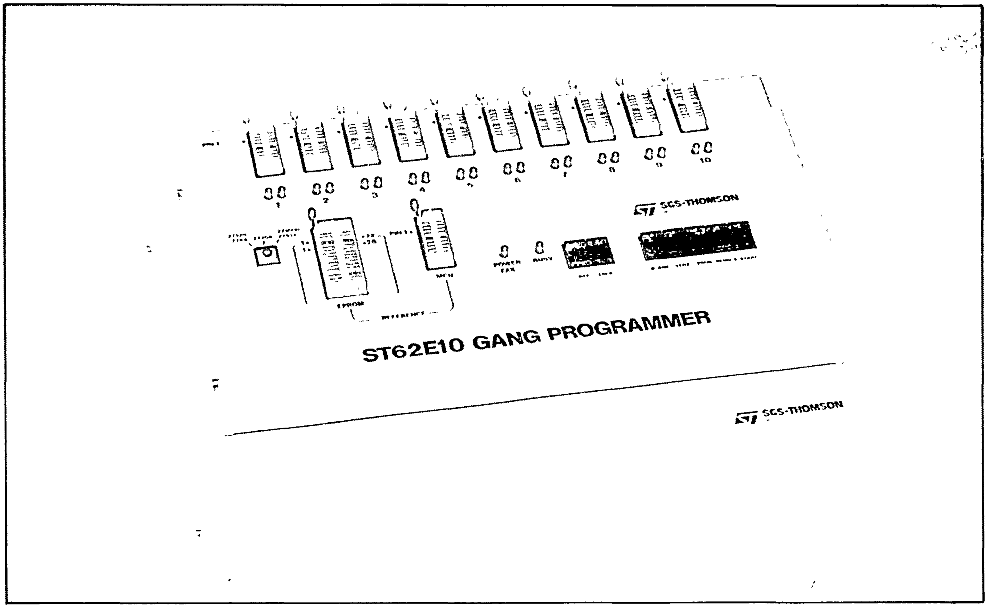
DESCRIPTION

The ST62 gang programmers are designed for programming up to 10 EPROM or OTP devices. It can run either in standalone or remote mode under control of a DOS compatible PC.

In standalone mode, the target ST62 MCUs are programmed with a simple key operation directly from a master EPROM memory or from a master EPROM MCU. Two color LEDs indicate for each target device the operational pass or fail. Both VERIFY and BLANK CHECK functions are provided.

In Remote mode, the gang programmer is connected to a DOS compatible PC through an RS232 serial channel. Object code in either S19 or INTEL HEX format is read from disk files to program the target devices. The menu driven software also offers VERIFY, BLANK CHECK, READ master and other utility functions.

The gang programmer is made up of a two parts, a base unit common to all ST62XX devices and a dedicated package adaptator.



ORDERING INFORMATION

Sales Types	Description	Supported Devices [1]	Supported Packages
ST62E10-GANGDIP	Gang Programmer	ST62E10 ST62T10 ST62E20 ST62T20	DIP20
ST62E10-GANGSO	Gang Programmer	ST62E10 ST62T10 ST62E20 ST62T20	SO20
ST62E15-GANGDIP	Gang Programmer	ST62E15 ST62T15 ST62E25 ST62T25	DIP28
ST62E15-GANGSO	Gang Programmer	ST62E15 ST62T15 ST62E25 ST62T25	SO28

Note 1. Both /HWD and /SWD options are supported

PROGRAMMING MANUAL

SGS-THOMSON
MICROELECTRONICS

ST62 - ST63

PROGRAMMING MANUAL

INTRODUCTION

This manual deals with the description of instruction set and addressing modes of ST62,63 microcontroller series. The manual is divided in two main sections. The first one includes, after a general family description, the addressing modes description. The second one includes the detailed description of ST62,63 instruction set. Here following each instruction is deeply described and are underlined the differences among each ST6 series. 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.

Table 1. ST62,63 Series Core Characteristics

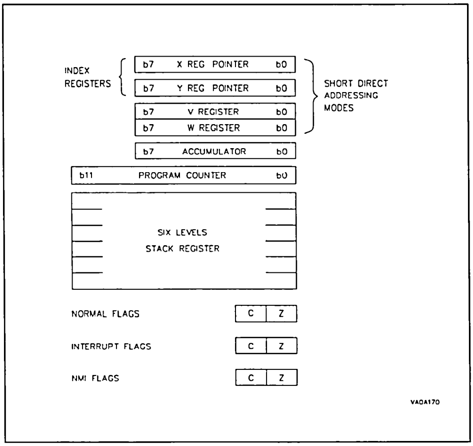
	ST62,63 Series
Stack Levels	6
Interrupt Vectors	5
NMI	YES
Flags Sets	3
Program ROM	2K + 2K• n 20K Max
Data RAM	64 byte • m
Data ROM	64 byte pages in ROM
Carry Flag SUB Instruction	Reset if A > Source
Carry Flag CP Instruction	Set if A < Source

PROGRAMMING MODEL

It is useful at this stage to outline the programming model of the ST62,63 series, by which we mean the available memory spaces, their relation to one another, the interrupt philosophy and so on.

Memory Spaces. The ST6 devices have three different memory spaces: data, program and stack. All addressing modes are memory space specific so there is no need for the user to specify which space is being used as in more complex systems. The stack space, which is used automatically with subroutine and interrupt management for program counter storage, is not accessible to the user.

Figure 1. ST6 Family Programming Model



PROGRAMMING MODEL (Continued)

Figure 2. ST62 Data Space Example

b7 b0	
NOT IMPLEMENTED	000H 03FH
DATA ROM/EPROM WINDOW 64 BYTE	040H 07FH
X REGISTER	080H
Y REGISTER	081H
V REGISTER	082H
W REGISTER	083H
DATA RAM 60 BYTES	084H 0BFH
PORT A DATA REGISTER	0C0H
PORT B DATA REGISTER	0C1H
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 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 PSC REGISTER	0D2H
TIMER DATA REGISTER	0D3H
TIMER TSCR REGISTER	0D4H
RESERVED	0D5H 0D7H
WATCHDOG REGISTER	0D8H
RESERVED	0D9H 0FEH
ACCUMULATOR	0FFH

Figure 3. ST62 Program Memory Example

b7 b0	
NOT IMPLEMENTED	0000H 07FFH
RESERVED	0800H 087FH
USER PROGRAM ROM 1828 BYTES	0880H 0F9FH
RESERVED	0FA0H 0FEFH
INTERRUPT VECTOR #4 A/D INTERRUPT	0FF0H 0FF1H
INTERRUPT VECTOR #3 TIMER INTERRUPT	0FF2H 0FF3H
INTERRUPT VECTOR #2 PORT B & C INTERRUPT	0FF4H 0FF5H
INTERRUPT VECTOR #1 PORT A INTERRUPT	0FF6H 0FF7H
RESERVED	0FF8H 0FFBH
INTERRUPT VECTOR #0 NMI INTERRUPT	0FFCH 0FFDH
USER RESET VECTOR	0FFEH 0FFFH

On EPROM versions there are no reserved areas. These reserved bytes are present on ROM/OTP versions.

Data Memory Space. The following registers in the data space have fixed addresses which are hardware selected so as to decrease access times and reduce addressing requirements and hence program length. The Accumulator is an 8 bit register in location 0FFH. The X, Y, V & W registers have the addresses 80H-83H respectively. These are used for short direct addressing, reducing byte requirements in the program while the first two, X & Y, can also be used as index registers in the indirect addressing mode. These registers are part of the data RAM space. In the ST62 and ST63 for data space ROM a 6 bit (64 bytes addressing) window multiplexing in program ROM is available through a dedicated data ROM banking register.

PROGRAMMING MODEL (Continued)

For data RAM and I/O expansion the lowest 64 bytes of data space (00H-03FH) are paged through a data RAM banking register.

Self-check Interrupt Vector FF8H & FF9H:
jp (self-check interrupt routine)

A jump instruction to the reset and interrupt routines must be written into these locations.

ST62 & ST63 Program Memory Space. The ST62 and ST63 devices can directly address up to 4K bytes (program counter is 12-bit wide). A greater ROM size is obtained by paging the lower 2K of the program ROM through a dedicated banking register located in the data space. The higher 2K of the program ROM can be seen as static and contains the reset, NMI and interrupt vectors at the following fixed locations:

Reset Vector FFEH & FFFH:
jp (reset routine)

NMI Interrupt Vector FFCH & FFDH:
jp (NMI routine)

Non user Vector FFAH & FFBH

Non user Vector FF8H & FF9H

Interrupt #1 Vector FF6H & FF7H jp (Int 1 routine)

Interrupt #2 Vector FF4H & FF5H jp (Int 2 routine)

Interrupt #3 Vector FF2H & FF3H jp (Int 3 routine)

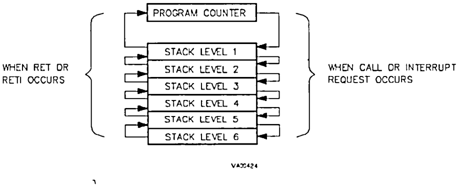
Interrupt #4 Vector FF0H & FF1H jp (Int 4 routine)

Program Counter & Stack Area. The program counter is a twelve bit counter register since it has to cover a direct addressing of 4K byte program memory space. When an interrupt or a subroutine occurs the current PC value is forward "pushed" into a deep LIFO stacking area. On the return from the routine the top (last in) PC value is "popped" out and becomes the current PC value. The ST60/61 series offer a 4-word deep stack for program counter storage during interrupt and sub-routines calls. In the ST62 and ST63 series the stack is 6-word deep.

Status Flags. Three pairs of status flags, each pair consisting of a Zero flag and a Carry flag, are available. In the ST62 and ST63 an additional third set is available. One pair monitors the normal status while the se-cond monitors the state during interrupts; the third flags set monitors the status during Non Maskable interrupt servicing. The switching from one set to another one is automatic as the interrupt requests (or NMI request for ST62,ST63 only) are acknowledged and when the program returns after an interrupt service routine. After reset, NMI set is active, until the first RETI instruction is executed.

ST62 & ST63 Interrupt Description. The ST62 and ST63 devices have 5 user interrupt vectors (plus one vector for testing purposes). Interrupt vector #0 is connected to the not maskable interrupt input of the core. Interrupts from #1 to #4 can be connected to different on-chip and external sources (see individual datasheets for detailed information). All interrupts can be globally disabled through the interrupt option register. After the reset ST62 and ST63 devices are in NMI mode, so no other interrupts can be accepted and the NMI flags set is in use, until the RETI instruction is performed. If an interrupt is detected, a special cycle will be executed, during this cycle the program counter is loaded with the related interrupt vector address. NMI can interrupt other interrupt routines at any time while normal interrupt can't interrupt each other. If more then one interrupt is waiting service, they will be accepted according to their priority. Interrupt #1 has the highest priority while interrupt #4 the lowest. This priority relationship is fixed.

Figure 2. ST62 & ST63 Stack Area

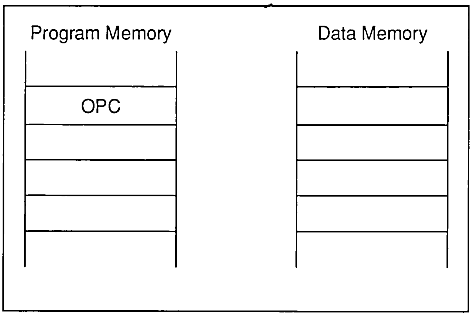


ADDRESSING MODES

The ST6 family gives the user nine addressing modes for access to data locations. Some of these are specifically tailored to particular instruction types or groups while others are designed to reduce program length and operating time by using the hardware facilities such as the X, Y, V & W registers. The data locations can be in either the program memory space or the data memory space when the ST6 is operating due to user software. In addition the ST6 has a stack space for the 12 bit program counter but this is controlled by internal programming and is not accessible by the user. This section will describe all the addressing modes which are provided to the user. The following is the complete list of the ST6 available addressing modes:

- Inherent
- Direct
- Short Direct
- Indirect
- Immediate
- Program Counter Relative
- Extended
- Bit Direct
- Bit Test & Branch

Inherent. For instructions using the inherent addressing mode the opcode contains all the information necessary for execution. All instructions using this mode are **One Byte** instructions.

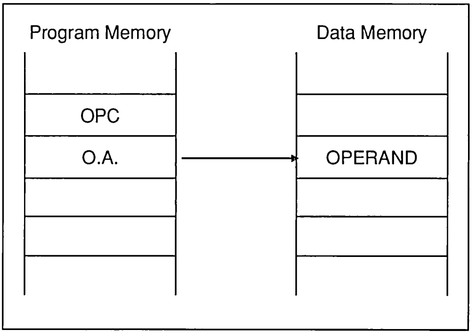


OPC = Opcode

Example:

Instruction	Comments
WAIT	Puts ST6 ınto the low power WAIT mode
STOP	Puts the ST6 into the lowest power mode
RETI	Returns from interrupt. Pops the PC from the PC stack.Sets the normal set of flags

Direct. In the direct addressing mode the address of the data is given by the program memory byte immediately following the opcode. This data location is in the data memory space. All instructions using this mode are **Two Bytes** instructions, lasting **Four Cycles**.



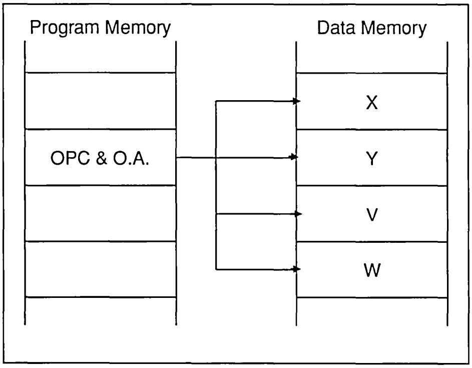
OPC = Opcode
O A = Operand Address

Example:

Instruction	Comments
LD A,0A3H	Loads the accumulator wıth the value found in location A3H in the data space.
SUB A,11H	The value found in locations 11H in the data memory is subtracted from the value in the accumulator.

ADDRESSING MODES (Continued)

Short Direct. ST6 core has four fixed location registers in the data space which may be addressed in a short direct manner. The addresses and names of these registers are 80H (X), 81H (Y), 82H (V) and 83H (W). When using this addressing mode the data is in one of these registers and the address is a part of the opcode. All instructions using this mode are **One Byte** instructions, lasting **Four Cycles.**

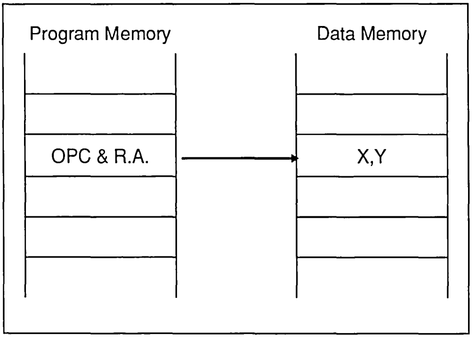


OPC = Opcode
O.A .= Operand Address

Example:

Instruction	Comments
LD A,X	The value of the X register (80H) is loaded into the accumulator.
INC X	The X register is incremented.

Indirect. The indirect mode must use either the X (80H) or Y (81H) register. This register contains the address of the data. The operand is at the data space address pointed to by the content of X or Y registers. All instructions using this mode are **One Byte** instructions, lasting **Four Cycles.**

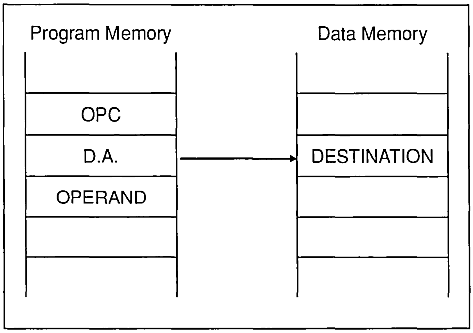


OPC = Opcode
R.A. = Register Address

Example:

Instruction	Comments
LD A,(X)	The value in the registers pointed to by the X register is loaded into the accumulator.
ADD A,(Y)	The value in the register pointed to by the Y register is added to the accumulator value.
INC (Y)	The value in the register pointed to by the Y register is incremented.

Immediate. In the immediate addressing mode the operand is found in the program ROM in a byte which is the last byte of the instruction. This addressing mode can be used for initializing data space registers and supplying constants. Instructions using this mode can be **Two** or **Three Bytes** instructions, lasting **Four Cycles**.



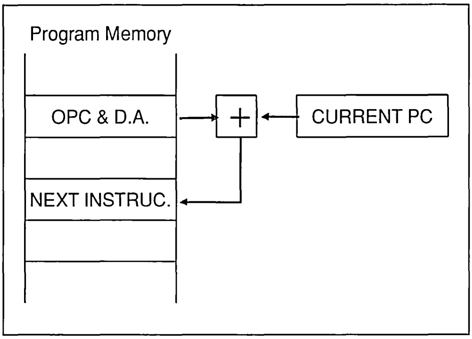
OPC = Opcode
D.A. = Destination Address

ADDRESSING MODES (Continued)

Example:

Instruction	Comments
LDI 34H,DFH	Loads immediate value DFH into data space location 34H.
SUBI A,22H	The immediate value 22H is subtracted from the acc.

Program Counter Relative. This addressing mode is used only with conditional branches within the program. The opcode byte contains the data which is a fixed offset value. This offset is added to the program counter to give the address of the next instruction. The offset can have any value in the range -15 to +16. It is determined by the last five bits of the opcode. All instructions using this mode are **One Byte** Instructions, lasting **Two Cycles**.



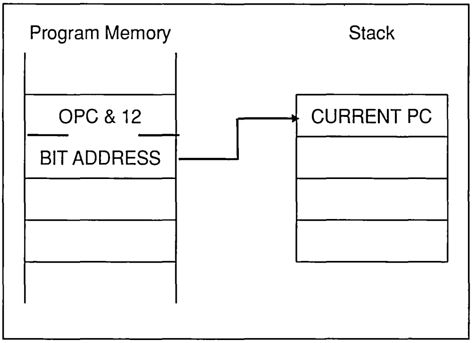
OPC = Opcode
D.A. = Destination Address

Example:

Instruction	Comments
JRC 3	If the carry flag is set then PC = PC+3
JRNZ -7	If the zero flag is not set (i.e the result of a previous instruction is not zero) then PC = PC-7

The relative jump address can be also a label that is automatically handled by the assembler.

Extended. The extended addressing mode is used to make long jumps within the program memory space (4K). The data requires 12 bits and is provided by half of the opcode byte and all of the second byte. All instructions using this mode are **Two Bytes** instructions, lasting **Four Cycles**.



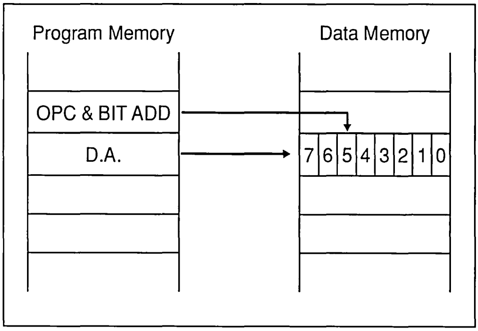
OPC = Opcode

Example:

Instruction	Comments
JP 3FAH	Loads 3FAH into program counter and continues with the instruction at 3FAH.
CALL ROU1	The current PC is pushed onto the stack and PC loaded with the value associated to the ROU1 label

The absolute jump address can be also a label that is automatically handled by the assembler.

Bit Direct. This addressing mode allows the user to set or clear any specified bit in a data memory register. The address of the bit is given in the form: "b,R" where b is the number of the bit and R is the address of the register. The bit is determined by three bits in the opcode and the register address is given by the second byte. All instructions using this mode are **Two Byte** instructions, lasting **Four Cycles**.



OPC = Opcode
D.A. = Destination Address

ADDRESSING MODES (Continued)

Example:

Instruction	Comments
SET 4,A	Sets bit 4 of the accumulator to 1.
RES 0,PORT	Clears bit 0 of PORT register

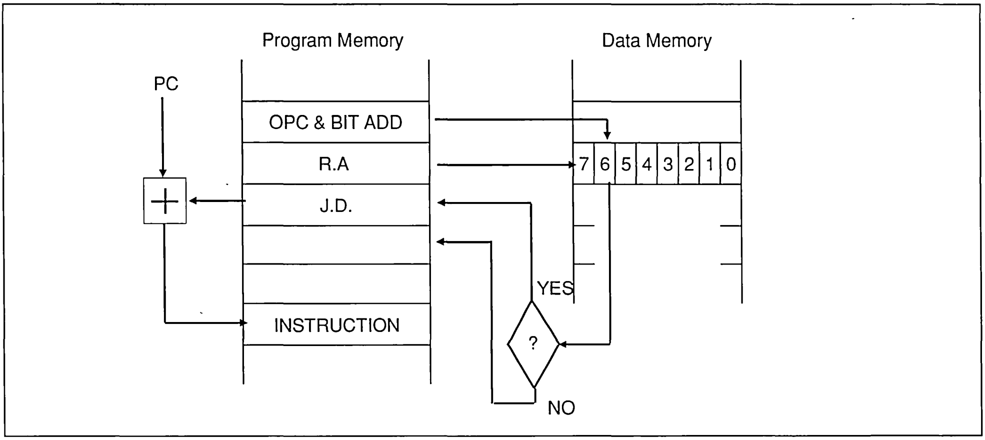
The register address can be associated to a label that is automatically handled by the assembler.

Bit Test & Branch. The bit test addressing mode is used in conditional jump instructions in which the jump depends on the result of a bit test. The opcode specifies the bit to be tested, the byte following the opcode in the register address in data space, and the third byte is the jump displacement, which is in the range -126 to +129. This displacement can be determined using a label, which is converted by the assembler. The state of the tested bit is also copied into the carry flag. All instructions using this mode are **Three Byte** instructions, lasting **Five Cycles**.

Example:

Instruction	Comments
JRS 3,PORT,LAB1	If bit three of data memory register associated to PORT label is set then PC=PC+LAB1 (where LAB1 is the jump displacement associated to a label
JRR 0,0AH,-72	If bit 0 of data memory register OAH is reset to 0 then PC=PC-72.

The register address and the jump displacement can be associated to labels that are automatically handled by the assembler.



OPC = Opcode
R.A. = Relative Address
J.D. = Jump Displacement

ST62 & ST63 INSTRUCTION SET

The ST62,63 instructions can be divided functionally into the following seven groups.

- LOAD AND STORE
- ARITHMETIC AND LOGIC
- CONDITIONAL BRANCH
- JUMP AND CALL
- BIT MANIPULATION
- CONTROL
- IMPLIED

The following summary shows the instructions belonging to each group, the number of operands required for each instructions and the number of machine cycles. The flag behaviour is usually the same for both ST62 and ST63. The only difference is present for CP and SUB instructions as specified in the detailed description.

Table 2. Load & Store Instructions

Instruction	Bytes	Cycles	Flags	
			Z	C
LD	1	4	Δ	*
LD rr	2	4	Δ	*
LDI A	2	4	Δ	*
LDI	3	4	*	*

Notes Δ· Affected
*. Not Affected

Table 3. Arithmetic & Logic Instructions

Instruction	Bytes	Cycles	Flags	
			Z	C
ADD	2	6 4	Δ	Δ
ADD (X,Y)	1	4	Δ	D
ADDI	2	4	Δ	D
AND	2	4	Δ	*
AND (X,Y)	1	4	Δ	*
ANDI	2	4	Δ	*
CLR A	2	4	Δ	D
CLR	3	4	*	*
COM	1	4	Δ	D
CP	2	4	Δ	D
CP (X,Y)	1	4	Δ	D
CPI	2	4	Δ	D
DEC	1	4	Δ	*
DEC A/rr	2	4	Δ	*
INC	1	4	Δ	*
INC A/rr	2	4	Δ	*
RLC	1	4	Δ	D
SLA	2	4	Δ	D
SUB	2	4	Δ	D
SUB (X,Y)	1	4	Δ	D
SUBI	2	4	Δ	D

Notes Δ: Affected
* Not Affected

Table 4. Conditional Branch Insructions

Instruction	Bytes	Cycles	Flags	
			Z	C
JRC	1	2	*	*
JRNC	1	2	*	*
JRR	3	5	*	Δ
JRS	3	5	*	Δ
JRZ	1	2	*	*
JRNZ	1	2	*	*

Notes. Δ. Affected
*· Not Affected

Table 5. Jump & Call Instructions

Instruction	Bytes	Cycles	Flags	
			Z	C
CALL	2	4	*	*
JP	2	4	*	*

Notes Δ· Affected
*: Not Affected

ST62 & ST63 INSTRUCTION SET (Continued)

Table 6. Bit Manipulation Instructions

Instruction	Bytes	Cycles	Flags	
			Z	C
RES	2	4	*	*
SET	2	4	*	*

Notes. Δ: Affected
*: Not Affected

Table 7. Control Instructions

Instruction	Bytes	Cycles	Flags	
			Z	C
NOP	1	2	*	*
RET	1	2	*	*
RETI	1	2	Δ	Δ
STOP	1	2	*	*
WAIT	1	2	*	*

Notes Δ· Affected
* Not Affected

Table 8. Addressing Modes/Instruction Table

Instruction	Inh	Dir	Sh Dir	Ind	Imm	PCR	Ext	Bit Dir	Bit Test	Flags	
										Z	C
ADD		X	X	X						Δ	Δ
AND		X	X	X						Δ	*
CALL							X			*	*
CLR A		X								Δ	D
CLR		X								*	*
COM	X									Δ	Δ
CP		X		X	X					Δ	Δ
DEC		X	X	X						Δ	*
INC		X	X	X						Δ	*
JP							X			*	*
JRC, JRNC						X				*	*
JRZ, JRNZ						X				*	*
JRR, JRS									X	*	Δ
LD, LDI					X					Δ	*
NOP						X				*	*
RES, SET								X		*	*
RET	X									*	*
RETI	X									Δ	Δ
RLC	X									Δ	Δ
SLA	X									Δ	Δ
STOP, WAIT	X									*	*
SUB		X		X	X					Δ	Δ

Notes:

INH.	Inherent, DIR: Direct, SH.DIR· Short Direct,
IND.	Indirect, IMM: Immediate, PCR: Program Counter Relative
EXT.	Extended, BIT DIR: Bit Direct, BIT TEST.. Bit Test
Δ	Affected
* .	Not Affected

ST62 & ST63 INSTRUCTION SET (Continued)

Table 9. ST62,63 Opcode Map

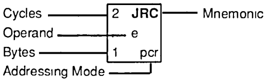
Low Hi	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111	Low Hi
0 0000	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRR b0,rr,ee 3 bt	2 JRZ e 1 pcr	#	2 JRC e 1 prc	4 LD a,(x) 1 ind	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b0,rr 2 b d	2 JRZ e 1 pcr	4 LDI rr,nn 3 imm	2 JRC e 1 pcr	4 LD a,(y) 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	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b0,rr 2 b d	2 JRZ e 1 pcr	4 DEC x 1 sd	2 JRC e 1 pcr	4 LD a,rr 2 dir	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 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b4,rr 2 b d	2 JRZ e 1 pcr	4 COM a 1 inh	2 JRC e 1 pcr	4 CP a,(y) 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	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b4,rr 2 b d	2 JRZ e 1 pcr	4 LD x,a 1 sd	2 JRC e 1 pcr	4 CP a,rr 2 dir	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	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b2,rr 2 b d	2 JRZ e 1 pcr	2 RETI 1 inh	2 JRC e 1 pcr	4 ADD a,(y) 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	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b2,rr 2 b d	2 JRZ e 1 pcr	4 DEC y 1 sd	2 JRC e 1 pcr	4 ADD a,rr 2 dir	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	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b6,rr 2 b d	2 JRZ e 1 pcr	2 STOP 1 inh	2 JRC e 1 pcr	4 INC (y) 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	#	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b6,rr 2 b d	2 JRZ e 1 pcr	4 LD y,a 1 sd	2 JRC e 1 pcr	4 INC rr 2 dir	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	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b1,rr 2 b d	2 JRZ e 1 pcr	#	2 JRC e 1 pcr	4 LD (y),a 1 ind	8 1000
9 1001	2 JRNZ 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	#	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b1,rr 2 b d	2 JRZ e 1 pcr	4 DEC v 1 sd	2 JRC e 1 pcr	4 LD rr,a 2 dir	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	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b5,rr 2 b d	2 JRZ e 1 pcr	4 RLC a 1 inh	2 JRC e 1 pcr	4 AND a,(y) 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	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b5,rr 2 b d	2 JRZ e 1 pcr	4 LD v,a 1 sd	2 JRC e 1 pcr	4 AND a,rr 2 dir	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	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b3,rr 2 b d	2 JRZ e 1 pcr	2 RET 1 inh	2 JRC e 1 pcr	4 SUB a,(y) 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	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b3,rr 2 b d	2 JRZ e 1 pcr	4 DEC w 1 sd	2 JRC e 1 pcr	4 SUB a,rr 2 dir	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	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 RES b7,rr 2 b d	2 JRZ e 1 pcr	2 WAIT 1 inh	2 JRC e 1 pcr	4 DEC (y) 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	#	2 JRNZ e 1 pcr	4 JP abc 2 ext	2 JRNC e 1 pcr	4 SET b7,rr 2 b d	2 JRZ e 1 pcr	4 LD w,a 1 sd	2 JRC e 1 pcr	4 DEC rr 2 dir	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



ST62 & ST63 INSTRUCTION SET (Continued)

Table 10. Instruction Set Cycle-by-Cycle Summary

Instruction	Cycles	Cycles(#)	Address Bus	Data Bus	CPU Activity	Notes
Indirect Addressing Mode						
ADD, AND, CP, DEC, INC, LD, SUB	4	1 2 3 4	Opcode Address(*) Opcode Address +1 Opcode Address +1 Opcode Address +1	Opcode (*) Next Instruction Next Instruction Next Instruction	Decode Opcode Read Operand Address Read Operand Execute Instruction	ROM Data Space not Addressed
ADD, AND, CP, DEC, INC, LD, SUB	4	1 2 3 4	Opcode Address(*) Opcode Address +1 Opcode Address +1 Data Space Rom Add	Opcode (*) Next Instruction Next Instruction Rom Data (#)	Decode Opcode Read Operand Address Read Operand Execute Instruction	ROM Data Space Addressed
Direct Addressing Mode						
ADD, AND, CP, DEC, INC, LD, RES, SET, LSA, SUB, CLR	4	1 2 3 4	Opcode Address(*) Opcode Address +1 Opcode Address +1 (*) Opcode Address +2	Opcode (*) Operand Address Operand Address(*) Next Instruction	Decode Opcode Address Data Space Read Operand Execute Instruction	ROM Data Space not Addressed'
ADD, AND, CP, DEC, INC, LD, RES, SET, LSA, SUB, CLR	4	1 2 3 4	Opcode Address(*) Opcode Address +1 Opcode Address +1 (*) Data Space Rom Add (*)	Opcode (*) Operand Address Operand Address(#) Rom Data (#)	Decode Opcode Address Data Space Read Operand Execute Instruction	ROM Data Space Addressed
Immediate Addressing Mode						
ADDI, ANDI, CPI, LDI, SUBI	4	1 2 3 4	Opcode Address(*) Opcode Address +1 Opcode Address +1(*) Opcode Address +2(*)	Opcode (*) Immediate Operand Immediate Operand Next Instruction	Decode Opcode Idle Read Operand Execute Instruction	
LDI rr	4	1 2 3 4	Opcode Address(*) Opcode Address +1 Opcode Address +2 Opcode AdDress +3	Opcode (*) Register Address Immediate Operand Next Opcode	Decode Opcode Read Register Address Read Immediate Operand Write Operand To Reg.	ROM Data Space not Addressed
LDI rr	4	1 2 3 4	Opcode Address(*) Opcode Address +1 (*) Opcode Address +2 (#) Data Space Rom Add	Opcode (*) Register Address Immediate Operand Rom Operand (#)	Decode Opcode Read Register Address Read Immediate Operand Write Operand To Reg	ROM Data Space Addressed
Short Direct Addressing Mode						
DEC, INC, LD	4	1 2 3 4	Opcode Address(*) Opcode Address +1 Opcode Address +1 Opcode Address +1	Opcode (*) Next Opcode Next Opcode Next Opcode	Decode Opcode Define Data Space Add Read Operand Execute Instruction	

Notes: * Valid only at the beginning of the cycle

#. Valid only unti t 18 of the cycle

ST62 & ST63 INSTRUCTION SET (Continued)

Table 10. Instruction Set Cycle-by-Cycle Summary (Continued)

Instruction	Cycles	Cycles(#)	Address Bus	Data Bus	CPU Activity	Notes
Other Instructions						
CALL	4	1 2 3 4	Opcode Address(*) Opcode Address +1 Opcode Address +1 Opcode Address +2(*)	Opcode (*) Subroutine Address Subroutine Address Next Instruction	Decode Opcode Increment Stack Pointer Push Return Address Calculate Subroutine Add.	
COM	4	1 2 3 4	Opcode Address(*) Opcode Address +1 Opcode Address +1 Opcode Address +1	Opcode (*) Next Opcode Next Opcode Next Opcode	Decode Opcode Calculate Acc. Address Read Accumulator Complement Accumulator	
INTERRUPT	1	1	Next opcode address	Next Opcode (*)	Calculate Interrupt Add Push Return Address Switch Flag Set	Note 1
JP	4	1 2 3 4	Opcode Address(*) opcode Address +1 opcode Address +1 opcode Address +2	Opcode (*) Jump Address Following Instr Following Instr (*)	Decode Opcode Idle Read Jump Address Calculate Jump Address	
JRC, JRNC, JRZ, JRNZ	2	1 2	Opcode Address(*) Opcode Address +1	Opcode (*) Following Instr.	Decode Opcode Calculate Offset	
JRR, JRS	5	1 2 3 4 5	Opcode Address(*) Opcode Address +1(*) Opcode Address +2(*) Opcode Address +2(*) Opcode Address +3(*)	Opcode (*) Operand Address (*) Branch Value Branch Value (*) Following Instr.	Decode Opcode Read Operand Test Operand Fetch Branch Value Calculate New Address	ROM Data Space not Addressed
JRR, JRS	5	1 2 3 4 5	Opcode Address(*) Opcode Address +1(*) Data Space Rom Add.(#) Opcode Address +2(*) Data Space Rom Add (#)	Opcode (*) Operand Address (*) Rom Data (#) Branch Value (*) Rom Data (#)	Decode Opcode Read Operand Test Operand Fetch Branch Value Calculate New Address	ROM Data Space Addressed
RET	2	1 2	Opcode Address(*) Return Address	Opcode (*) Next Opcode	Decode Opcode Pop Return Address	
RETI	2	1 2	Opcode Address(*) Return Address	Opcode (*) Next Opcode	Decode Opcode Pop Return Address Switch Flag Set	
RLC	4	1 2 3 4	Opcode Address(*) Opcode Address +1 Opcode Address +1 Opcode Address +1	Opcode (*) Next Opcode Next Opcode Next Opcode	Decode Opcode Calculate Acc. Address Read Accumulator Shifted	
STOP, WAIT	2	1 2	Opcode Address(*) Opcode Address +1	Opcode (*) Next Opcode	Decode Opcode Stop/Wait the Oscillator	

Notes· *. Valid only at the beginning of the cycle
Valid only until t18 of the cycle

1. Add oscillator build up time plus 16 oscillator clocks if a stop instruction has been executed before the interrupt occured

ADD

Addition

Mnemonic: ADD

Function: Addition

Description: The contents of the source byte is added to the accumulator leaving the result in the accumulator. The source register remains unaltered.

Operation: dst ← dst + src

The destination must be the accumulator.

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
ADD dst,src				Z	C
ADD A,A	5F FF	2	4	Δ	Δ
ADD A,X	5F 80	2	4	Δ	D
ADD A,Y	5F 81	2	4	Δ	D
ADD A,V	5F 82	2	4	Δ	D
ADD A,W	5F 83	2	4	Δ	D
ADD A,(X)	47	1	4	Δ	D
ADD A,(Y)	4F	1	4	Δ	D
ADD A,rr	5F rr	2	4	Δ	D

Notes:
rr. 1 Byte dataspace address.
Δ: Z is set if the result is zero. Cleared otherwise.
C is cleared before the operation and than set if there is an overflow from the 8-bit result

Example: If data space register 22H contains the value 33H and the accumulator holds the value 20H then the instruction,

ADD A,22H

will cause the accumulator to hold 53H (i.e. 33+20).

Addressing Modes: Source: Direct, Indirect

Destination: Accumulator

ADDI

Addition Immediate

Mnemonic: ADDI

Function: Addition Immediate

Description: The immediately addressed data (source) is added to the accumulator leaving the result in the accumulator.

Operation: dst ← dst + src

The destination must be the accumulator.

Instruction Format	Opcode (Hex)	Bytes	Cycles	Flags	
ADDI dst,src				Z	C
ADDI A,nn	57 nn	2	4	Δ	Δ

Notes:
nn. 1 Byte immediate data
Δ· Z is set if result is zero Cleared otherwise
C is cleared before the operation and than set if there is an overflow from the 8-bit result

Example: If the accumulator holds the value 20H then the instruction,

ADDI A,22H

will cause the accumulator to hold 42H (i.e. 22+20).

Addressing Modes: Source: Immediate
Destination: Accumulator

AND

Logical AND

Mnemonic: AND

Function: Logical AND

Description: This instruction logically ANDs the source register and the accumulator. The result is left in the destination register and the source is unaltered.

Operation: dst ←src AND dst

The destination must be the accumulator.

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
AND dst,src				Z	C
AND A,A	BF FF	2	4	Δ	*
AND A,X	BF 80	2	4	Δ	*
AND A,Y	BF 81	2	4	Δ	*
AND A,V	BF 82	2	4	Δ	*
AND A,W	BF 83	2	4	Δ	*
AND A,(X)	A7	1	4	Δ	*
AND A,(Y)	AF	1	4	Δ	*
AND A,rr	BF rr	2	4	Δ	*

Notes:
rr. 1 Byte dataspace address
*. C is unaffected
Δ. Z is set if the result is zero. Cleared otherwise.

Example: If data space register 54H contains the binary value 11110000 and the accumulator contains the binary value 11001100 then the instruction,

AND A,54H

will cause the accumulator to be altered to 11000000.

Addressing Modes: Source: Direct, Indirect:
Destination: Accumulator

ANDI

Logical AND Immediate

Mnemonic: ANDI

Function: Logical AND Immediate

Description: This instruction logically ANDs the immediate data byte and the accumulator. The result is left in the accumulator.

Operation: dst ← src AND dst

The source is immediate data and the destination must be the accumulator.

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
ANDI dst,src				Z	C
ANDI A,nn	B7 nn	2	4	Δ	*

Notes:
nn 1 Byte ımmedıate data
*. C ıs unaffected
Δ Z ıs set ıf the result ıs zero Cleared otherwıse

Example: If the accumulator contains the binary value 00001111 then the instruction,

ANDI A,33H

will cause the accumulator to hold the value 00000011.

Addressing Modes: Source: Immediate

Destination: Accumulator

CALL

Call Subroutine

Mnemonic: CALL

Function: Call Subroutine

Description: The CALL instruction is used to call a subroutine. It "pushes" the current contents of the program counter (PC) onto the top of the stack. The specified destination address is then loaded into the PC and points to the first instruction of a procedure. At the end of the procedure a RETurn instruction can be used to return to the original program flow. RET pops the top of the stack back into the PC.
Because the ST6 stack is 4 levels deep (ST60) and 6 levels deep (ST62,ST63), a maximum of four/six calls or interrupts may be nested. If more calls are nested, the PC values stacked latest will be lost. In this case returns will return to the PC values stacked first.

Operation: PC ← dst; Top of stack ← PC

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
CALL dst				Z	C
CALL abc	c0001 ab	2	4	*	*

Notes:
abc the three half bytes of a twelve bit address, the start location of the subroutine
* C,Z not affected

Example: If the current PC is 345H then the instruction,

CALL 8DCH

The current PC 345H is pushed onto the top of the stack and the PC will be loaded with the value 8DCH. The next instruction to be executed will be the instruction at 8DCH, the first instruction of the called subroutine.

Addressing Modes: Extended

CLR

Clear

Mnemonic: CLR

Function: Clear

Description: The destination register is cleared to 00H.

Operation: dst ← 0

Inst. Format	OPCDE (Hex)	Bytes	Cycles	Flags	
CLR dst				Z	C
CLR A	DF FF	2	4	Δ	Δ
CLR X	0D 80 00	3	4	*	*
CLR Y	0D 81 00	3	4	*	*
CLR V	0D 82 00	3	4	*	*
CLR W	0D 83 00	3	4	*	*
CLR rr	0D rr 00	3	4	*	*

Notes:
rr 1 Byte dataspace address
Δ C,Z set
*. C,Z unaffected

Example: If data space register 22H contains the value 33H,
CLR 22H
will cause register 22H to hold 00H.

Addressing Modes: Direct

COM

Complement

Mnemonic: COM

Function: Complement

Description: This instruction complements each bit of the accumulator; all bits which are set to 1 are cleared to 0 and vice-versa.

Operation: dst ← NOT dst

The destination must be the accumulator.

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
COM dst				Z	C
COM A	2D	1	4	Δ	Δ

Note :
Δ: Z is set if the result is zero. Cleared otherwise.
C will contain the value of the MSB before the operation.

Example: If the accumulator contains the binary value 10111001 then the instruction

COM A

will cause the accumulator to be changed to 01000110 and the carry flag to be set (since the original MSB was 1).

Addressing Modes: Inherent

CP

Compare

Mnemonic: CP

Function: Compare

Description: This instruction compares the source byte (subtracted from) with the destination byte, which must be the accumulator. The carry and zero flags record the result of this comparison.

Operation: dst - src

The destination must be the accumulator, but it will not be changed.

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
CP dst,src				Z	C
CP A,A	3F FF	2	4	Δ	Δ
CP A,X	3F 80	2	4	Δ	Δ
CP A,Y	3F 81	2	4	Δ	Δ
CP A,V	3F 82	2	4	Δ	Δ
CP A,W	3F 83	2	4	Δ	Δ
CP A,(X)	27	1	4	Δ	Δ
CP A,(Y)	2F	1	4	Δ	Δ
CP A,rr	3F rr	2	4	Δ	Δ

Note: rr 1 Byte dataspace address

ST60 Δ: Z is set if the result is zero. Cleared otherwise.

C is set if Acc ≥ src, cleared if Acc < src.

ST62/63 Δ: Z is set if the result is zero. Cleared otehrwise.

C is set if Acc < src, cleared if Acc ≥ src.

Example: If the accumulator contains the value 11111000 and the register 34H contains the value 00011100 then the instruction,

CP A,34H

will clear the Zero flag Z and set the Carry flag C, indicating that Acc ≥ src (on ST60)

Addressing Modes: Source: Direct, Indirect

Destination: Accumulator

CPI

Compare Immediate

Mnemonic: CPI

Function: Compare Immediate

Description: This instruction compares the immediately addressed source byte (subtracted from) with the destination byte, which must be the accumulator. The carry and zero flags record the result of this comparison.

Operation: dst-src

The source must be the immediately addressed data and the destination must be the accumulator, that will not be changed.

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
CPI dst,src				Z	C
CPI A,nn	37 nn	2	4	Δ	Δ

Note: nn 1 Byte immediate data.

ST60 Δ: Z is set if the result is zero. Cleared otherwise.
C is set if Acc ≥ src, cleared if Acc < src.

ST62/63 Δ: Z is set if the result is zero. Cleared otherwise.
C is set if Acc < src, cleared if Acc ≥ src.

Example: If the accumulator contains the value 11111000 then the instruction,
CPI A,00011100B
will clear the Zero flag Z and set the Carry flag C indicating that Acc ≥ src (on ST60).

Addressing Modes: Source: Immediate
Destination: Accumulator

DEC

Decrement

Mnemonica: DEC

Function: Decrement

Description: The destination register's contents are decremented by one.

Operation: dst ← dst-1

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
DEC dst				Z	C
DEC A	FF FF	2	4	Δ	*
DEC X	1D	1	4	Δ	*
DEC Y	5D	1	4	Δ	*
DEC V	9D	1	4	Δ	*
DEC W	DD	1	4	Δ	*
DEC (X)	E7	1	4	Δ	*
DEC (Y)	EF	1	4	Δ	*
DEC rr	FF rr	2	4	Δ	*

Notes:
rr. 1 Byte dataspace address
*. C is unaffected
Δ Z ıs set if the result is zero Cleared otherwise

Example: If the X register contains the value 45H and the data space register 45H contains the value 16H then the instruction,

DEC (X)

will cause data space register 45H to contain the value 15H.

Addressing Modes: Short direct, Direct, Indirect.

INC

Increment

Mnemonic: INC

Function: Increment

Description: The destination register's contents are incremented by one.

Operation: dst ← dst+1

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
INC dst				Z	C
INC A	7F FF	2	4	Δ	*
INC X	15	1	4	Δ	*
INC Y	55	1	4	Δ	*
INC V	95	1	4	Δ	*
INC W	D5	1	4	Δ	*
INC (X)	67	1	4	Δ	*
INC (Y)	6F	1	4	Δ	*
INC rr	7F rr	2	4	Δ	*

Notes:
rr. 1 Byte dataspace address
* C is unaffected
Δ. Z is set if the result is zero. Cleared otherwise

Example: If the X register contains the value 45H and the data space register 45H contains the value 16H then the instruction

INC (X)

will cause data space register 45H to contain the value 17H.

Addressing Modes: Short direct, Direct, Indirect.

JP

Jump

Mnemonic: JP

Function: Jump (Unconditional)

Description: The JP instruction replaces the PC value with a twelve bit value thus causing a simple jump to another location in the program memory. The previous PC value is lost, not stacked.

Operation: PC ← dst

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
JP dst				Z	C
JP abc	c1001 ab	2	4	*	*

Notes:
abc. the three half bytes of a twelve bit address
*. C,Z not affected

Example: The instruction,

JP 5CDH

will cause the PC to be loaded with 5CDH and the program will continue from that location.

Addressing Modes: Extended

JRC

Jump Relative on Carry Flag

Mnemonic: JRC

Function: Jump Relative on Carry Flag

Description: This instruction causes the carry (C) flag to be tested and if this flag is set then a jump is performed within the program memory. This jump is in the range -15 to +16 and is relative to the PC value. The displacemente is of five bits. If C=0 than the next instruction is executed.

Operation: If C=1, PC ← PC + e

where e= 5 bit displacement

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
				Z	C
JRC e	e110	1	2	*	*

Notes:
e 5 bit displacement in the range –15 to + 16
* C,Z not affected

Example: If the carry flag is set then the instruction,

JRC + 8

will cause a branch forward to PC+8. The user can use labels as indentifiers and the assembler will automatically allow the jump if it is in the range -15 to +16.

Addressing Modes: Program Counter Relative

JRNC

Jump Relative on Non Carry Flag

Mnemonic: JRNC

Function: Jump Relative on Non Carry Flag

Description: This instruction causes the carry (C) flag to be tested and if this flag is cleared to zero then a jump is performed within the program memory. This jump is in the range -15 to +16 and is relative to the PC value. The dispacement is of five bits. If C=1 then the next instruction is executed.

Operation: If C=0, PC ← PC + e

where e= 5 bit displacement

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
				Z	C
JRNC e	e010	1	2	*	*

Notes:
e. 5 bit displacement in the range -15 to +16
*: C,Z not affected

Example: If the carry flag is cleared then the instruction,

JRNC -5

will cause a branch backward to PC-5. The user can use labels as identifiers and the assembler will automatically allow the jump if it is in the range -15 to +16.

Addressing Modes: Program Counter Relative

JRNZ

Jump Relative on Non Zero Flag

Mnemonic: JRNZ

Function: Jump Relative on Non Zero Flag

Description: This instruction causes the zero (Z) flag to be tested and if this flag is cleared to zero then a jump is performed within the program memory. This jump is in the range -15 to +16 and is relative to the PC value. The displacement is of five bits. If Z=1 then the next instruction is executed.

Operation: If Z=0, PC ⟵ PC + e

where e= 5 bit displacement

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
				Z	C
JRNZ e	e000	1	2	*	*

Notes:
e. 5 bit displacement in the range -15 to +16
*. C,Z not affected

Example: If the zero flag is cleared then the instruction,

JRNZ -5

will cause a branch backward to PC-5. The user can use labels as identifiers and the assembler will automatically allow the jump if it is in the range -15 to +16.

Addressing Modes: Program Counter Relative

JRR

Jump Relative if Reset

Mnemonic: JRR

Function: Jump Relative if RESET

Description: This instruction causes a specified bit in a given dataspace register to be tested. If this bit is reset (=0) then the PC value will be changed and a relative jump will be performed within the program. The relative jump range is -126 to +129. If the tested bit is not reset then the next instruction is executed.

Operation: If bit=0, PC ⟵ PC + ee

where ee= 8 bit displacement

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
				Z	C
JRR b,rr,ee	b00011 rr ee	3	5	*	Δ

Notes:
b 3 bit-address
rr 1 Byte dataspace address
ee 8 bit displacement in the range -126 to +129

*. Z is not affected
Δ The tested bit is shifted into carry

Example: If bit 4 of dataspace register 70H is reset and the PC=110 then the instruction,

JRR 4, 70H, -20

will cause the PC to be changed to 90 (110-20) and the instruction starting at that address in the program memory to be the next instruction executed.

The user is advised to use labels for conditional jumps. The relative jump will be calculated by the assembler. The jump must be in the range -126 to +129.

Addressing Modes: Bit Test

JRS

Jump Relative if Set

Mnemonic: JRS

Function: Jump Relative if set

Description: This instruction causes a specified bit in a given dataspace register to be tested. If this bit is set (=1) then the PC value will be changed and a relative jump will be performed within the program. The relative jump range is -126 to +129. If the tested bit is not set then the next instruction is executed.

Operation: If bit=1, PC ← PC + ee

where ee= 8 bit displacement

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
				Z	C
JRS b,rr,ee	b10011 rr ee	3	5	*	Δ

Notes:
b 3 bit-address
rr 1 Byte dataspace address
ee. 8 bit displacement in the range -126 to +129

* Z is not affected
Δ The tested bit is shifted into carry.

Example: If bit 7 of dataspace register AFH is set and the PC=123 then the instruction,

JRS 7,AFH,+25

will cause the PC to be changed to 148 (123+25) and the instruction starting at that address in the program memory to be the next instruction executed.

The user is advised to use labels for conditional jumps. The relative jump will be calculated by the assembler. The jump must be in the range -126 to +129.

Addressing Modes: Bit Test

JRZ

Jump Relative on Zero Flag

Mnemonic: JRZ

Function: Jump Relative on Zero Flag

Description: This instruction causes the zero (Z) flag to be tested and if this flag is set to one then a jump is performed within the program memory. This jump is in the range -15 to +16 and is relative to the PC value. The displacement is of five bits.
If Z=0 then next instruction is executed.

Operation: If Z=1, PC ← PC + e
where e= 5 bit displacement

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
				Z	C
JRZ e	e100	1	2	*	*

Notes:
e. 5 bit displacement in the range -15 to +16.
*. C,Z not affected

Example: If the zero flag is set then the instruction,

JRZ +8

will cause a branch forward to PC+8. The user can use labels as identifiers and the assembler will automatically allow the jump if it is in the range -15 to +16.

Addressing Modes: Program Counter Relative

LD

Load

Mnemonic: LD

Function: Load

Description: The contents of the source register are loaded into the destination register. The source register remains unaltered and the previous contents of the destination register are lost.

Operation: dst ← src

Either the source or the destination must be the accumulator.

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
LD dst,src				Z	C
LD A,X	35	1	4	Δ	*
LD A,Y	75	1	4	Δ	*
LD A,V	B5	1	4	Δ	*
LD A,W	.F5	1	4	Δ	*
LD X,A	3D	1	4	Δ	*
LD Y,A	7D	1	4	Δ	*
LD V,A	BD	1	4	Δ	*
LD W,A	FD	1	4	Δ	*
LD A,(X)	07	1	4	Δ	*
LD (X), A	87	1	4	Δ	*
LD A,(Y)	0F	1	4	Δ	*
LD (Y),A	8F	1	4	Δ	*
LD A,rr	1F rr	2	4	Δ	*
LD rr,A	9F rr	2	4	Δ	*

Notes:
rr. 1 Byte dataspace address
*. C not affected
Δ Z is set if the result is zero. Cleared otherwise

Example: If data space register 34H contains the value 45H then the instruction;

LD A,34H

will cause the accumulator to be loaded with the value 45H. Register 34H will keep the value 45H.

Addressing Modes: Source: Direct, Short Direct, Indirect

Destination: Direct, Short Direct, Indirect

LDI

Load Immediate

Mnemonic: LDI

Function: Load Immediate

Description: The immediately addressed data (source) is loaded into the destination data space register.

Operation: dst ← src

The source is always an immediate data while the destination can be the accumulator, one of the X,Y,V,W registers or one of the available data space registers.

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
LDI dst,src				Z	C
LDI A,nn	17 nn	2	4	Δ	*
LDI X,nn	0D 80 nn	3	4	*	*
LDI Y,nn	0D 81 nn	3	4	*	*
LDI V,nn	0D 82 nn	3	4	*	*
LDI W,nn	0D 83 nn	3	4	*	*
LDI rr,nn	0D rr nn	3	4	*	*

Notes:
rr. 1 Byte dataspace address
nn 1 Byte immediate value

*. Z, C not affected
Δ Z is set if the result is zero Cleared otherwise.

Example: The instruction

LDI 34H,45H

will cause the value 45H to be loaded into data register at location 34H.

Addressing Modes: Source: Immediate
Destination: Direct

NOP

No Operation

Mnemonic: NOP

Function: No Operation

Description: No action is performed by this instruction. It is typically used for timing delay.

Operation: No Operation

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
				Z	C
NOP	04	1	2	*	*

Note: * C,Z not affected

Addressing Modes: Program Counter Relative

RES

Reset Bit

Mnemonic: RES

Function: Reset Bit

Description: The RESET instruction is used to reset a specified bit in a given register in the data space.

Operation: dst (n) $\leftarrow$ 0, $0 \leq n \leq 7$

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
RES bit,dst				Z	C
RES b,A	b01011 FF	2	4	*	*
RES b,rr	b01011 rr	2	4	*	*

Notes:
b. 3 bit-address
rr. 1 Byte dataspace address
* C,Z not affected

Example: If register 23H of the dataspace contains 11111111 then the instruction,

RES 4,23H

will cause register 23H to hold 11101111.

Addressing Modes: Bit Direct

RET

Return from Subroutine

Mnemonic: RET

Function: Return From Subroutine

Description: This instruction is normally used at the end of a subroutine to return to the previously executed procedure. The previously stacked program counter (stacked during CALL) is popped back from the stack. The next statement executed is that addressed by the new contents of the PC. If the stack had already reached its highest level (no more PC stacked) before the RET is executed, program execution will be continued at the next instruction after the RET.

Operation: PC ← Stacked PC

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
				Z	C
RET	CD	1	2	*	*

Note: *. C,Z not affected

Example: If the current PC value is 456H and the PC value at the top of the stack is 3DFH then the instruction,

RET

will cause the PC value 456H to be lost and the current PC value to be 3DFH.

Addressing Modes: Inherent

RETI

Return from Interrupt

Mnemonic: RETI

Function: Return from Interrupt

Description: This instruction marks the end of the interrupt service routine and returns the ST60/62/63 to the state it was in before the interrupt. It "pops" the top (last in) PC value from the stack into the current PC. This instruction also causes the ST60/62/63 to switch from the interrupt flags to the normal flags. The RETI instruction also applies to the end of NMI routine for ST62/63 devices; in this case the instruction causes the switch from NMI flags to normal flags (if NMI was acknowledged inside a normal routine) or to standard interrupt flags (if NMI was acknowledged inside a standard interrupt service routine).

In addition the RETI instruction also clears the interrupt mask (also NMI mask for ST62/63) which was set when the interrupt occurred. If the stack had already reached its highest level (no more PC stacked) before the RETI is executed, program execution will be continued with the next instruction after the RETI. Because the ST60 is in interrupt mode after reset (NMI mode for ST62/63), RETI has to be executed to switch to normal flags and enable interrupts at the end of the starting routine. If no call was executed during the starting routine, program execution will continue with the instruction after the RETI (supposed no interrupt is active).

Operation: Actual Flags ← Normal Flags (1)

PC ← Stacked PC

IM ← 0

(1) Standard Interrupt flags if NMI was acknowledged inside a standard interrupt service (ST62/63 only).

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
				Z	C
RETI	4D	1	2	Δ	Δ

Note: Δ C,Z normal flag will be used from now on

Example: If the current PC value is 456H and the PC value at the top of the stack is 3DFH then the instruction

RETI

will cause the value 456H to be lost and the current PC value to be 3DFH.
The ST6 will switch from interrupt flags to normal flags and the interrupt mask is cleared.

Addressing Modes: Inherent

RLC

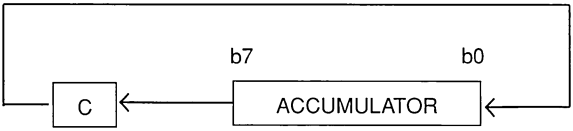
Rotate Left Through Carry

Mnemonic: RLC

Function: Rotate Left through Carry

Description: This instruction moves each bit in the accumulator one place to the left (i.e. towards the MSBit. The MSBit (bit 7) is moved into the carry flag and the carry flag is moved into the LSBit (bit0) of the accumulator.

Operation:



dst(0) ← C

C ←dst(7)

dst(n+1) ← dst(n), 0 ≤ n ≤ 6

This instruction can only be performed on the accumulator.

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
				Z	C
RLC A	AD	1	4	Δ	Δ

Note : Δ: Z is set if the result is zero Cleared otherwise
C will contain the value of the MSB before the operation

Example: If the accumulator contains the binary value 10001001 and the carry flag is set to 0 then the instruction,

RLC A

will cause the accumulator to have the binary value 00010010 and the carry flag to be set to 1.

Addressing Modes: Inherent

SET

Set Bit

Mnemonic: SET

Function: Set Bit

Description: The SET instruction is used to set a specified bit in a given register in the data space.

Operation: dst (n) ← 1, $0 \leq n \leq 7$

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
SET bit,dst				Z	C
SET b,A	b11011 FF	2	4	*	*
SET b,rr	b11011 rr	2	4	*	*

Notes:
b 3 bit-address
rr. 1 Byte dataspace address
*. C,Z not affected

Example: If register 23H of the dataspace contains 00000000 then the instruction,

SET 4,23H

will cause register 23H to hold 00010000.

Addressing Modes: Bit Direct

SLA

Shift Left Accumulator

Mnemonic: SLA

Function: Shift Left Accumulator

Description: This instruction implements an addition of the accumulator to itself (i.e a doubling of the accumulator) causing an arithmetic left shift of the value in the register.

Operation: ADD A,FFH

This instruction can only be performed on the accumulator.

Inst. Format	OPCPDE (Hex)	Bytes	Cycles	Flags	
				Z	C
SLA A	5F FF	2	4	Δ	Δ

Note: Δ Z ıs set ıf the result ıs zero Cleared otherwıse
C will contaın the value of the MSB before the operatıon.

Example: If the accumulator contains the binary value 11001101 then the instruction,

SLA A

will cause the accumulator to have the binary value 10011010 and the carry flag to be set to 1.

Addressing Modes: Inherent

STOP

Stop Operation

Mnemonic: STOP

Function: Stop operation

Description: This instruction is used for putting the ST60/62/63 into a stand-by mode in which the power consumption is reduced to a minimum. All the on-chip peripherals and oscillator are stopped (for some peripherals,A/D for example, it is necessary to individually turn-off the macrocell before entering the STOP instruction). To restart the processor an external interrupt or a reset is needed.

Operation: Stop Processor

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
				Z	C
STOP	6D	1	2	*	*

Note : *: C,Z not affected

Addressing Mode: Inherent

SUB

Subtraction

Mnemonic: SUB

Function: Subtraction

Description: This instruction subtracts the source value from the destination value.

Operation: dst ←dst-src

The destination must be the accumulator.

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
SUB·dst,src				Z	C
SUB A,A	DF FF	2	4	Δ	Δ
SUB A,X	DF 80	2	4	Δ	Δ
SUB A,Y	DF 81	2	4	Δ	Δ
SUB A,V	DF 82	2	4	Δ	Δ
SUB A,W	DF 83	2	4	Δ	Δ
SUB A,(X)	C7	1	4	Δ	Δ
SUB A,(Y)	CF	1	4	Δ	Δ
SUB A,rr	DF rr	2	4	Δ	Δ

Note: rr.1 Byte dataspace address

ST60 Δ: Z is set if the result is zero. Cleared otherwise.

C is set if Acc ≥ src, cleared if Acc < src.

ST62/63 Δ: Z is set if the result is zero. Cleared otherwise.

C is set if Acc < src, cleared if Acc ≥ src.

Example: If the Y register contains the value 23H, dataspace register 23H contains the value 53H and the accumulator contains the value 78H then the instruction,

SUB A,(Y)

will cause the accumulator to hold the value 25H (i.e. 78-53). The zero flag is cleared and the carry flag is set (on ST60), indicating that result is > 0.

Addressing Modes: Source: Indirect,Direct

Destination: Accumulator

SUBI

Subtraction Immediate

Mnemonic: SUBI

Function: Subtraction Immediate

Description: This instruction causes the immediately addressed source data to be subtracted from the accumulator.

Operation: dst ⟵ dst - src

The destination must be the accumulator.

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
SUBI dst,src				Z	C
SUBI A,nn	D7 nn	2	4	Δ	Δ

Note: nn 1 Byte of immediate data

ST60 Δ: Z is set if the result is zero. Cleared otherwise.
C is set if Acc ≥ src, cleared if Acc < src.

ST62/63 Δ: Z is set if the result is zero. Cleared otherwise.
C is set if Acc < src, cleared if Acc ≥ src.

Example: If the accumulator contains the value 56H then the instruction,

SUBI A,25

will cause the accumulator to contain the value 31H. The zero flag is cleared and the carry flag is set (on ST60), indicating that the result is > 0.

Addressing Modes: Source: Immediate
Destination: Accumulator

WAIT

Wait Processor

Mnemonic: WAIT

Function: Wait Processor

Description: This instruction is used for putting the ST60/62/63 into a stand-by mode in which the power consumption is reduced to a minimum. Instruction execution is stopped, but the oscillator and some on-chip peripherals continue to work. To restart the processor an interrupt from an active on-chip peripheral (eg. timer), an external interrupt or reset is needed. For on-chip peripherals active during wait, see ST60/62/63 data sheets.

Operation: Put ST6 in stand-by mode

Inst. Format	OPCODE (Hex)	Bytes	Cycles	Flags	
				Z	C
WAIT	ED	1	2	*	*

Note : *. C,Z not affected

Addressing Modes: Inherent

APPLICATION NOTE

SGS-THOMSON MICROELECTRONICS APPLICATION NOTE

POWER CONTROL WITH ST6210 MCU AND TRIAC

Philippe RABIER - Laurent PERIER

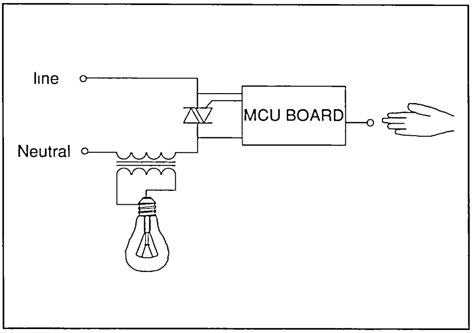
INTRODUCTION

Microcontroller (MCU) systems are progressively replacing analog controllers even in low cost applications. They are more flexible, provide a faster time to market and need few components.

With an analog IC, the designer is limited to a fixed function frozen inside the device. With a DIAC control, features such as sensor feedback or enhanced motor drive can not be implemented. With the MCU proposed in this note (ST6210), the designer can implement his own ideas and test them directly using EPROM or One Time Programmable (OTP) versions.

The LOGIC LEVEL triac BTA08-600SW is a good complement to this MCU for low cost off-line power applications. This triac requires a low gate current, and can be directly triggered by the MCU, while still maintaining a high switching capability.

This application note describes the main aspects of a highly flexible, low cost power application designed around an ST6210 MCU and a LOGIC LEVEL triac.



BOARD OPERATION

Basic function

Light dimmers with DIAC or analog controllers are currently used today. These circuits have the disavantage that they can not easily drive inductive loads like halogen lamps on the secondary of a 220V/12V transformer. They are also limited in the choice of user interfaces.

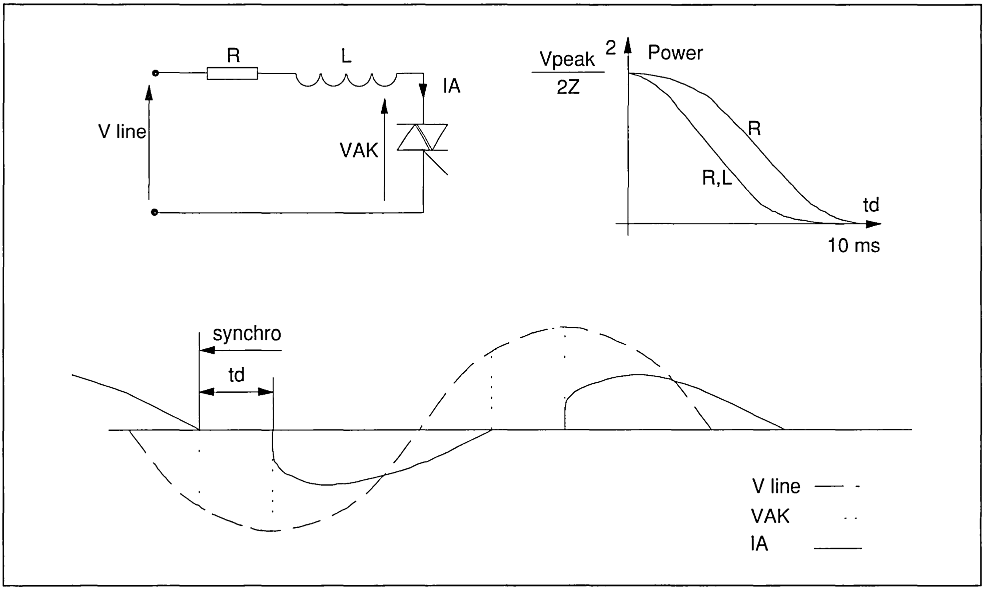
A light dimmer circuit, supplied directly from the 110V/240V mains, has been realized using a MCU ST6210 and a LOGIC LEVEL triac. This circuit drives both resistive and inductive loads (e.g. halogen or incandescent lamps, transformers). The control method is such that the same board can drive a universal motor. The user interface is either a touch sensor, a push button or a potentiometer. The board contains a minimum of components therefore saving cost and space. The auxiliary supply is derived from the voltage across the triac.

Power control

The output power is controlled by the phase delay of the triac drive. In classical designs, the delay is refered to the zero crossing of the line voltage. The detection of the zero voltage normally requires an additional connection to the mains neutral. In order to avoid this connection and connect the circuit directly in series with the load, the trigger delay is referred to the previous zero crossing of the current (fig.1).

BOARD OPERATION (Continued)

Figure 1. The Power Control Is Based On The Monitoring Of The Zero Crossing Of The Current



Mains synchronisation

When the current in the triac is zero, the mains voltage is re-applied across the triac. Synchronisation is achieved by measuring this voltage. This voltage is monitored in each halfwave, which allows the detection of spurious open load conditions. The triac is retrigged with multipulse operation if it is not latched after the first gate pulse.

Changing operation from 50Hz to 60Hz can be achieved by making simple modifications to the microcontroller EPROM/ROM table defining the triac conduction angle versus the power level.

Operation with a transformer

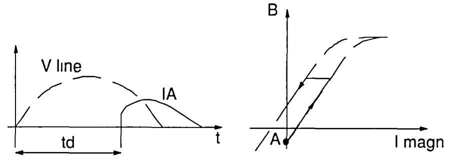
Low power halogen spots use low voltage lamps (12V typ.) usually supplied through a low voltage transformer. The light dimming of these lamps is simple with this circuit.

A phase lag between current and voltage as high as 90 does not disturb the circuit because the control method is based only on monitoring the zero current crossing.

The risk of saturation of the transformer core is avoided because the controller includes the following features:

At the start, the delay time between the first gate pulse and the synchronisation instant is greater than 5ms. This limits the induction in the transformer and hence reduces the risk of saturation.

Figure 2. First Gate Pulse Delay

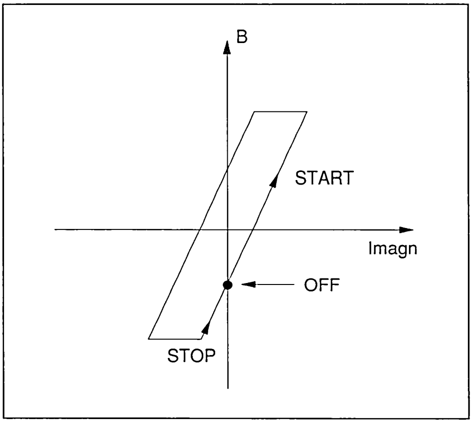


Saturation of the transformer at start is avoided.

BOARD OPERATION (Continued)

The circuit starts on a positive halfwave and stops on a negative halfwave (fig.3). So it starts with positive induction and stops after negative induction has been applied. This helps to minimize the size of the magnetic material.

Figure 3. Hysteresis Cycle in off/start/stop Phases



The timer is very precisely tuned in order to obtain precisely 10ms delay between two gate pulses. As a result, the triac is driven symetrically in both phases so continuous voltage in the transformer is avoided.

The voltage across the triac is monitored in order to detect a spurious open load condition on the secondary of the transformer.

The inrush current at the turn-on of a lamp (halogen or incandescent) is also reduced due to the soft start feature of the circuit (fig.8).

Triac drive

The triac is multi-pulse driven. Therefore, inductive loads can be driven without the use of long pulse drives. As a result, the consumption on the +5V supply can be minimized and the supply circuit made very small.

The pulse driving the triac is 50μs long. The LOGIC LEVEL triac is driven in quadrants QII and QIII with a gate current of 20mA provided by two I/O bits of the ST6210 in parallel. The LOGIC LEVEL triac has a maximum specified gate triggering current of 10 mA at 25°C.

Before supplying the first drive pulse, the triac voltage is tested. If no voltage is detected, a spurious open load or a supply disconnection is assumed to have occured and the circuit is stopped.

After the first driving pulse, the triac voltage is monitored again. If the triac is not ON, another pulse is sent. The same process can be repeated up to four times. Thereafter, if the triac is not ON, the circuit is switched off.

User Interfaces

There are three different user interfaces: a touch control, a push button or a potentiometer. Four modes can be selected on the board in order to define how the transmitted power is related to the user interface.

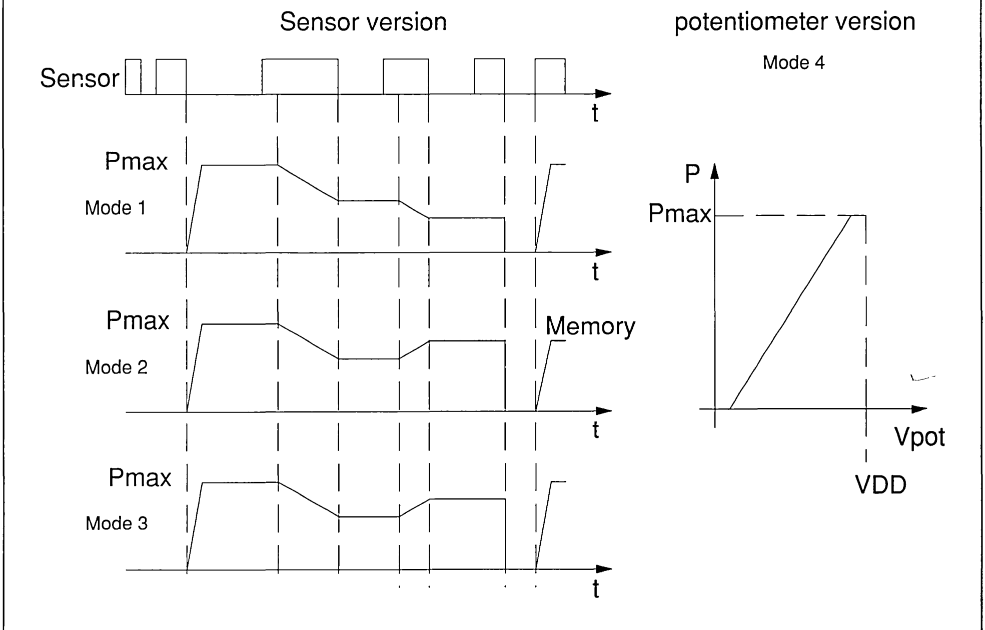
Three modes operate with the touch sensor or the push button. Dimming is obtained when the sensor or the button is touched for more than 400 ms. If the touch duration is between 60 ms and 400 ms, the circuit is switched on or off. A contact of less than 60 ms has no effect. Modes 1,2,3 differ in the way the ouput power is influenced by the contact on the sensor or on the button.

Mode 4 directly relates the transmitted power to the position of the potentiometer (fig.4).

All modes include a soft start function.

BOARD OPERATION (Continued)

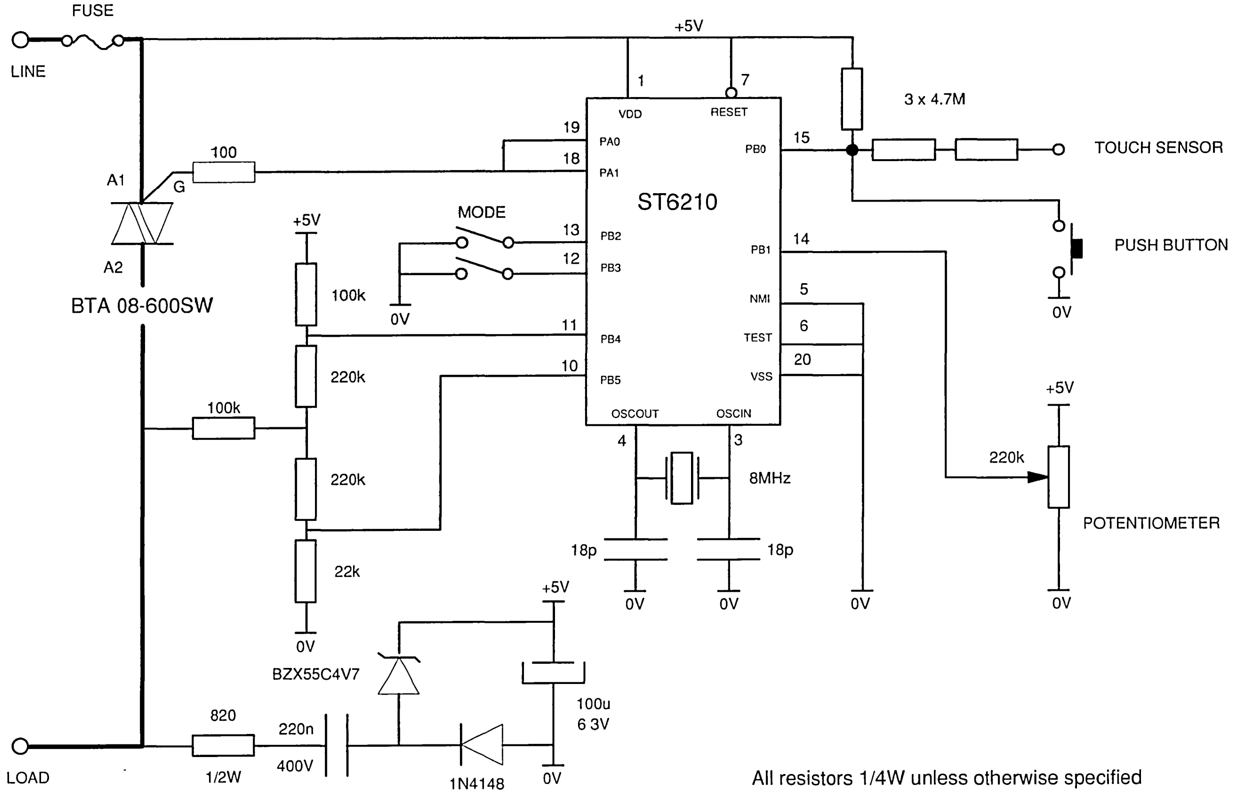
Figure 4. User Interface



Sensor Contact Duration	< 60ms	60ms to 400ms	> 400ms
Mode 1	No effect	Switched ON to full power or switched OFF	Same sense of variation as previous action
Mode 2	No effect	Switched ON to previous level or switched OFF	Opposite sense of variation to previous action
Mode 3	No effect	Switched ON to full power or switched OFF	Opposite sense of variation to previous action

BOARD OPERATION (Continued)

Figure 5. Circuit Diagram



All resistors 1/4W unless otherwise specified

HARDWARE

The circuit uses an 8 bit MCU ST6210 and a LOGIC LEVEL triac directly driven by the MCU (fig.5). It operates with 3 user interfaces, 4 modes of operation and 4 kinds of loads. When the board is dimming a resistive load, an RFI filter must be used in order to meet RFI standards (eg. VDE 875).

The ST6210 includes 2K ROM, 64 bytes RAM, an 8bit A/D converter that can be connected to eight different inputs, 4 I/O ports with 10mA sink current capability and a timer. Hysteresis protection is included in series with each I/O pin. The ST6210 is packaged in PDIP or SMD packages. The ports, timer and interrupts configurations can be chosen by software, providing great flexibility. With EPROM and OTP versions, the equipment development and preproduction can be carried out directly from the design lab thus providing a fast time to market.

The LOGIC LEVEL triac (BTA08-600SW) has been especially designed to operate with MCUs. It is a sensitive triac (I_{GT}=10 mA, I_L=50 mA) trigged in quadrants QII and QIII. In this application it is driven by two I/O bits of the ST6210 in parallel. This triac has high switching capabilities ([dl/dt]c=3.5 A/ms), ([dV/dt]c=50 V/µs), so in this circuit, it can operate without a snubber.

Total consumption of the board is less than 3mA with an 8MHz oscillator. The board receives its supply only when the triac is off. So a minimum off time of the triac (2ms) is necessary to ensure its supply.

The 5V supply capacitance is mounted as near as possible to the MCU with very short interconnecting tracks in order to maximize the RFI/EMI immunity.

The touch sensor is a voltage divider between line and neutral potentials. It operates when the supply of the circuit is connected at the line potential and not at the neutral. The user is protected from electrical shock by a very high impedance (10MΩ) connecting the sensor to the circuit.

SOFTWARE

All the features are included in a 700 byte program. More than 1kbyte of ROM is available for additional features. The architecture of the software is modular in order to provide maximum flexibility.

The table relating the delay time to the power requirement contains 64 different levels. The conduction time of the triac can vary from 2ms to 8ms. The user can easily adjust the minimum and maximum power levels because the corresponding delay times change with smaller increments at the top and bottom of the table.

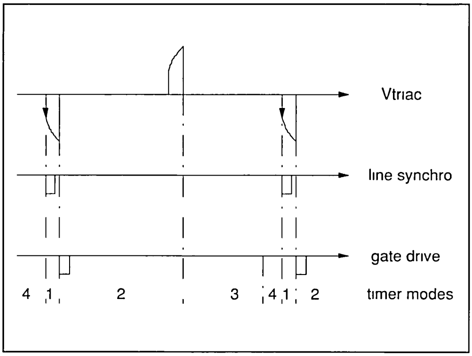
The table can be easily modified in the ROM/EPROM space to meet different conditions e.g. 60Hz operation or varying loads.

Software versions cover the four user interface modes of operation without hardware change.

All inputs are digitally filtered, so that an input is valid only if it remains constant for 10µs or more. This reduces the number of passive components required.

The mains supply carries disturbances (e.g. glitches, telecommand signals) which can disturb the triac drive and generate lamp flickering. For this

Figure 6. Internal Timer Operation

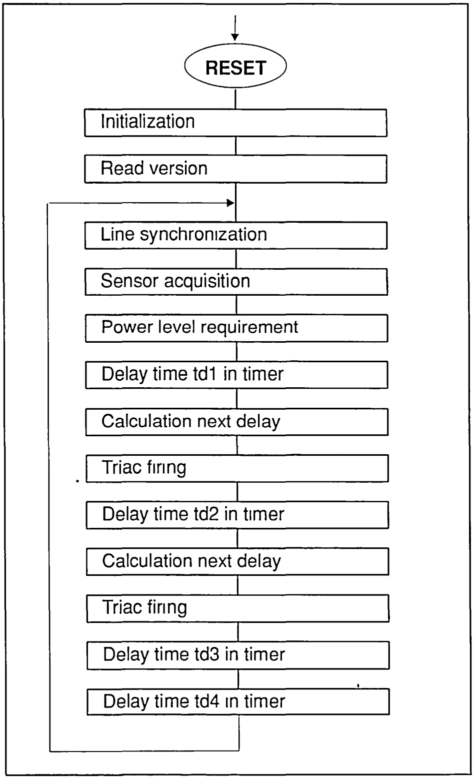


SOFTWARE (Continued)

reason, the triac voltage is not used directly as the synchronisation parameter. The timing is carried out internally by the MCU timer. The period of operation can be slowly modified to follow the variations of the mains frequency but not the spurious disturbances. The mains synchronization signal is received every cycle. The corresponding mains period is measured and compared to the internal timer period. If a difference remains for a long time, the timer period is modified to follow the mains. This block acts like a low band filter which saves external filtering components.
Each 50Hz period, the timer operation is separated in four steps (fig.6). The triac voltage synchronisation can only be validated during phase 4.

The software has been written with modular blocks (fig.7). It can be enlarged to other applications such as motor speed regulation, telecommand input or IR remote control with additional blocks.

Figure 7. Major Steps of the Software



PRACTICAL RESULTS

Figure 8 shows the soft start operation with a halogen lamp operating from the secondary of a low voltage transformer and with a tungsten filament lamp.

Figure 8. Soft Start With Lamps

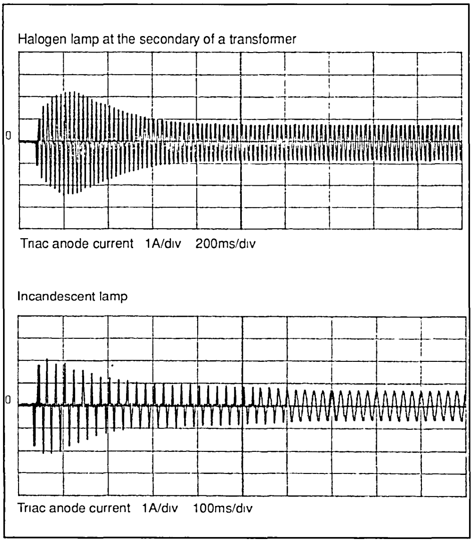
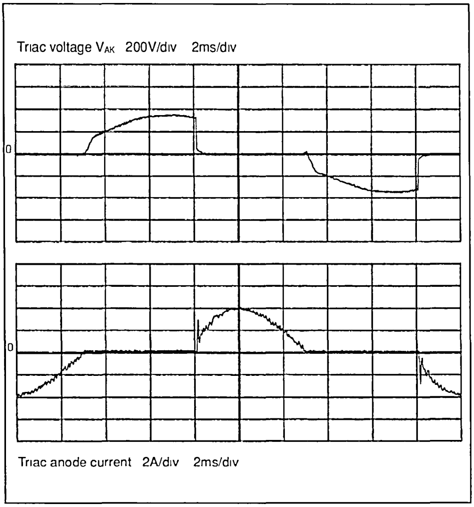


Figure 9. Universal Motor Drive



PRACTICAL RESULTS (Continued)

Due to the soft start, the peak in-rush current is about 3 times the nominal current compared with 10 to 15 times without soft start. This extends the lamp life time and prevents the input fuse from blowing.

The figure 9 shows the current and voltage in a triac driving a universal motor.

SUMMARY

Microcontrollers (MCU) are in common use in most areas of electronics. They now penetrate the very cost sensitive arena of home appliance applications.

The application described in this paper shows that enhanced appliance circuits can be designed with fast prototyping time using a ST6210 MCU and a BTA08-600SW LOGIC LEVEL triac. These circuits are low cost and provide more features with less components than classical solutions.

The circuit presented is an enhanced light dimmer operating from the 120V/240V mains. It drives incandescent and halogen lamps supplied either directly from the mains or through a low voltage transformer. The same circuit can also drive a universal motor. It includes soft start and protection features. Different user interfaces can be chosen: touch sensor, push button or potentiometer.

All this is achieved with only few components: a ST6210 MCU in PDIP/PSO package with a BTA08-600SW LOGIC LEVEL triac in TO220 package and some passive components.

Additional features like presence detection, IR remote control, homebus interface, motor speed control or 60Hz operation can be implemented from the existing solution.

Bibliography

Thyristors and triacs application manual 1989
Microcontroller based universal motor speed control . M.Querol / SGS-Thomson Microelectronics

Application Note:.

Universal Motor Speed Control / P.Rault + Y.Bahout / SGS-THOMSON Microelectronics

ANNEX : Choice of a triac driven by a MCU.

When the software includes a soft start, the inrush current in the load and therefore the current rating in the triac can be reduced.

When using a LOGIC LEVEL or a SNUBBERLESS triac the current rating of the triac can be reduced, keeping fast commutation characteristics. For instance, a LOGIC LEVEL triac BTA08-600SW can drive a 600W lamp and a SNUBBERLESS triac BTA10-600BW a 1200W universal motor.

LOGIC LEVEL triacs are optimized on the drive view point. Therefore they can be driven directly by the ST621X I/O.

SNUBBERLESS triacs are optimized on the power view point, so they can drive loads which generate very strong dynamic contraints.

These triacs are specified in a way that their behaviour can be pre-determined. The tables below present with two examples the relation between the major application constraints and the key parameters of the triac.

LIGHT DIMMER

Constraint	Key parameters on a LOGIC LEVEL triac BTA08-600 SW
Simple Drive	I_{GT} = 10mA - V_{GT} = 1.5V
No flicker	I_H = 25mA
Max power on the load	I_{RMS} = 8A
Max inrush current	$(dI/dt)_C$ = 4.5A/ms
No flashing [(1)]	I_{TSM} = 80A
Flash over (filament failure)	

Note 1 : When the lamp is cold (start or low light intensity), there must not be spurious turn-on of the triac (flashing) due to a high commutation dI/dt

UNIVERSAL MOTOR DRIVE

Constraint	Key parameters on a SNUBBERLESS triac BTA10-600 BW
Simple drive	I_{GT} = 50mA - V_{GT} = 1.5V
Max. start current	I_{TSM} = 100A I_{RMS} = 10A $(dI/dt)_C$ = 9A/ms dV/dt = 500V/µs
Max. power on the load	
Fuse sizing	$I^2t = 50A^2.s$

SALES OFFICES

EUROPE

DENMARK

2730 HERLEV
Herlev Torv, 4
Tel (45-42) 94 85 33
Telex 35411
Telefax (45-42) 948694

FINLAND

LOHJA SF-08150
Karjalankatu, 2
Tel 12 155 11
Telefax 12 155 66

FRANCE

94253 GENTILLY Cedex
7 - avenue Gallieni - BP 93
Tel (33-1) 47 40 75 75
Telex 632570 STMHQ
Telefax (33-1) 47 40 79 10

67000 STRASBOURG
20, Place des Halles
Tel (33) 88 75 50 66
Telex 870001F
Telefax (33) 88 22 29 32

GERMANY

6000 FRANKFURT
Gutleutstrasse 322
Tel (49-69) 237492
Telex 176997 689
Telefax (49-69) 231957
Teletex 6997689=STVBP

8011 GRASBRUNN
Bretonischer Ring 4
Neukeferloh Technopark
Tel (49-89) 46006-0
Telex 528211
Telefax (49-89) 4605454
Teletex 897107=STDISTR

3000 HANNOVER 1
Eckenerstrasse 5
Tel (49-511) 634191
Telex 175118418
Teletex 5118418 csfbeh
Telefax (49-511) 633552

8500 NÜRNBERG 20
Erlenstegenstrasse, 72
Tel (49-911) 59893-0
Telex 626243
Telefax (49-911) 5980701

5200 SIEGBURG
Frankfurter Str 22a
Tel (49-2241) 660 84-86
Telex 889510
Telefax (49-2241) 67584

7000 STUTTGART
Oberer Kirchhaldenweg 135
Tel (49-711) 692041
Telex 721718
Telefax (49-711) 691408

ITALY

20090 ASSAGO (MI)
V le Milanofiori - Strada 4 - Palazzo A/4/A
Tel (39-2) 89213 1 (10 linee)
Telex 330131 - 330141 SGSAGR
Telefax (39-2) 8250449

40033 CASALECCHIO DI RENO (BO)
Via R Fucini, 12
Tel (39-51) 591914
Telex 512442
Telefax (39-51) 591305

00161 ROMA
Via A Torlonia, 15
Tel (39-6) 8443341
Telex 620653 SGSATE I
Telefax (39-6) 8444474

NETHERLANDS

5652 AR EINDHOVEN
Meerenakkerweg 1
Tel (31-40) 550015
Telex 51186
Telefax (31-40) 528835

SPAIN

08021 BARCELONA
Calle Platon, 6 4th Floor, 5th Door
Tel (34-3) 4143300-4143361
Telefax (34-3) 2021461

28027 MADRID
Calle Albacete, 5
Tel (34-1) 4051615
Telex 27060 TCCEE
Telefax (34-1) 4031134

SWEDEN

S-16421 KISTA
Borgarfjordsgatan, 13 - Box 1094
Tel (46-8) 7939220
Telex 12078 THSWS
Telefax (46-8) 7504950

SWITZERLAND

1218 GRAND-SACONNEX (GENEVA)
Chemin Francois-Lehmann, 18/A
Tel (41-22) 7986462
Telex 415493 STM CH
Telefax (41-22) 7984869

UNITED KINGDOM and EIRE

MARLOW, BUCKS
Planar House, Parkway
Globe Park
Tel (44-628) 890800
Telex 847458
Telefax (44-628) 890391

AMERICAS

BRAZIL

05413 SÃO PAULO
R Henrique Schaumann 286-CJ33
Tel (55-11) 883-5455
Telex (391)11-37988 "UMBR BR"
Telefax 11-551-128-22367

CANADA

BRAMPTON, ONTARIO
341 Main St North
Tel (416) 455-0505
Telefax 416-455-2606

U.S.A.

NORTH & SOUTH AMERICAN
MARKETING HEADQUARTERS
1000 East Bell Road
Phoenix, AZ 85022
(1)-(602) 867-6100

SALES COVERAGE BY STATE

ALABAMA
Huntsville - (205) 533-5995

ARIZONA
Phoenix - (602) 867-6340

CALIFORNIA
Santa Ana - (714) 957-6018
San Jose - (408) 452-8585

COLORADO
Boulder (303) 449-9000

ILLINOIS
Schaumburg - (708) 517-1890

INDIANA
Kokomo - (317) 459-4700

MASSACHUSETTS
Lincoln - (617) 259-0300

MICHIGAN
Livonia - (313) 462-4030

NEW JERSEY
Voorhees - (609) 772-6222

NEW YORK
Poughkeepsie - (914) 454-8813

NORTH CAROLINA
Raleigh - (919) 787-6555

TEXAS
Carrollton - (214) 466-8844

FOR RF AND MICROWAVE POWER TRANSISTORS CONTACT
THE FOLLOWING REGIONAL OFFICE IN THE U.S A.

PENNSYLVANIA
Montgomeryville - (215) 362-8500

ASIA / PACIFIC

AUSTRALIA

NSW 2027 EDGECLIFF
Suite 211, Edgecliff centre
203-233, New South Head Road
Tel (61-2) 327 39 22
Telex 071 126911 TCAUS
Telefax (61-2) 327 61 76

HONG KONG

WANCHAI
22nd Floor - Hopewell centre
183 Queen's Road East
Tel (852-5) 8615788
Telex 60955 ESGIES HX
Telefax (852-5) 8656589

INDIA

NEW DELHI 110001
LiasonOffice
62, Upper Ground Floor
World Trade Centre
Barakhamba Lane
Tel 3715191
Telex 031-66816 STMI IN
Telefax 3715192

MALAYSIA

PULAU PINANG 10400
4th Floor - Suite 4-03
Bangunan FOP-123D Jalan Anson
Tel (04) 379735
Telefax (04) 379816

KOREA

SEOUL 121
8th floor Shinwon Building
823-14, Yuksam-Dong
Kang-Nam-Gu
Tel (82-2) 553-0399
Telex SGSKOR K29998
Telefax (82-2) 552-1051

SINGAPORE

SINGAPORE 2056
28 Ang Mo Kio - Industrial Park 2
Tel (65) 4821411
Telex· RS 55201 ESGIES
Telefax (65) 4820240

TAIWAN

TAIPEI
12th Floor
571, Tun Hua South Road
Tel (886-2) 755-4111
Telex 10310 ESGIE TW
Telefax (886-2) 755-4008

JAPAN

TOKYO 108
Nisseki - Takanawa Bld 4F
2-18-10 Takanawa
Minato-Ku
Tel (81-3) 3280-4121
Telefax (81-3) 3280-4131

Information furnished is believed to be accurate and reliable However, SGS-THOMSON Microelectronics assumes no resṕonsability for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may results 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 informations previously supplied.

SGS-THOMSON Microelectronics GROUP OF COMPANIES
Australia - Brazil - China - France - Hong Kong - Italy - Japan - Korea - Malaysia - Malta - Morocco - The Netherlands - Singapore - Spain - Sweden - Switzerland - Taiwan - United Kingdom - U.S.A. - West Germany

Printed by NTI - AGL Partenaires - Belcodène - France