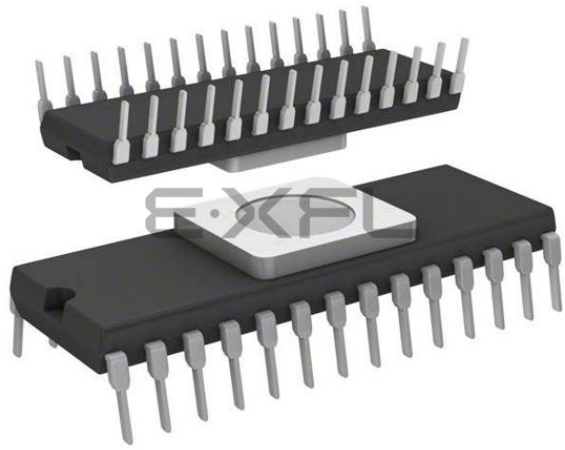


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	ST6
Core Size	8-Bit
Speed	8MHz
Connectivity	-
Peripherals	LVD, POR, WDT
Number of I/O	20
Program Memory Size	4KB (4K x 8)
Program Memory Type	EPROM, UV
EEPROM Size	-
RAM Size	64 x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 6V
Data Converters	A/D 16x8b
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Through Hole
Package / Case	28-CDIP (0.600", 15.24mm) Window
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/stmicroelectronics/st62e25cf1

2 PIN DESCRIPTION

Figure 2. 28-Pin Package Pinout

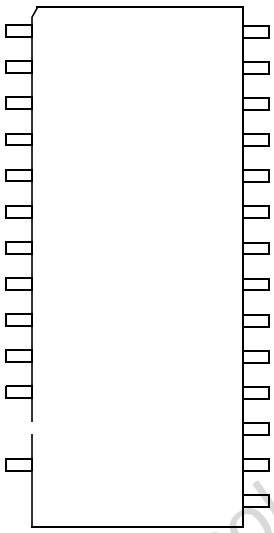


Table 1. Device Pin Description

Obsolete Product(s) - Obsolete Product(s)

MEMORY MAP (Cont'd)

3.1.6.2 Data ROM Window memory addressing

In cases where some data (look-up tables for example) are stored in program memory, reading these data requires the use of the Data ROM window mechanism. To do this:

1. The DRWR register has to be loaded with the 64-byte block number where the data are located (in program memory). This number also gives the start address of the block.

2. Then, the offset address of the byte in the Data ROM Window (corresponding to the offset in the 64-byte block in program memory) has to be loaded in a register (A, X,...).

When the above two steps are completed, the data can be read.

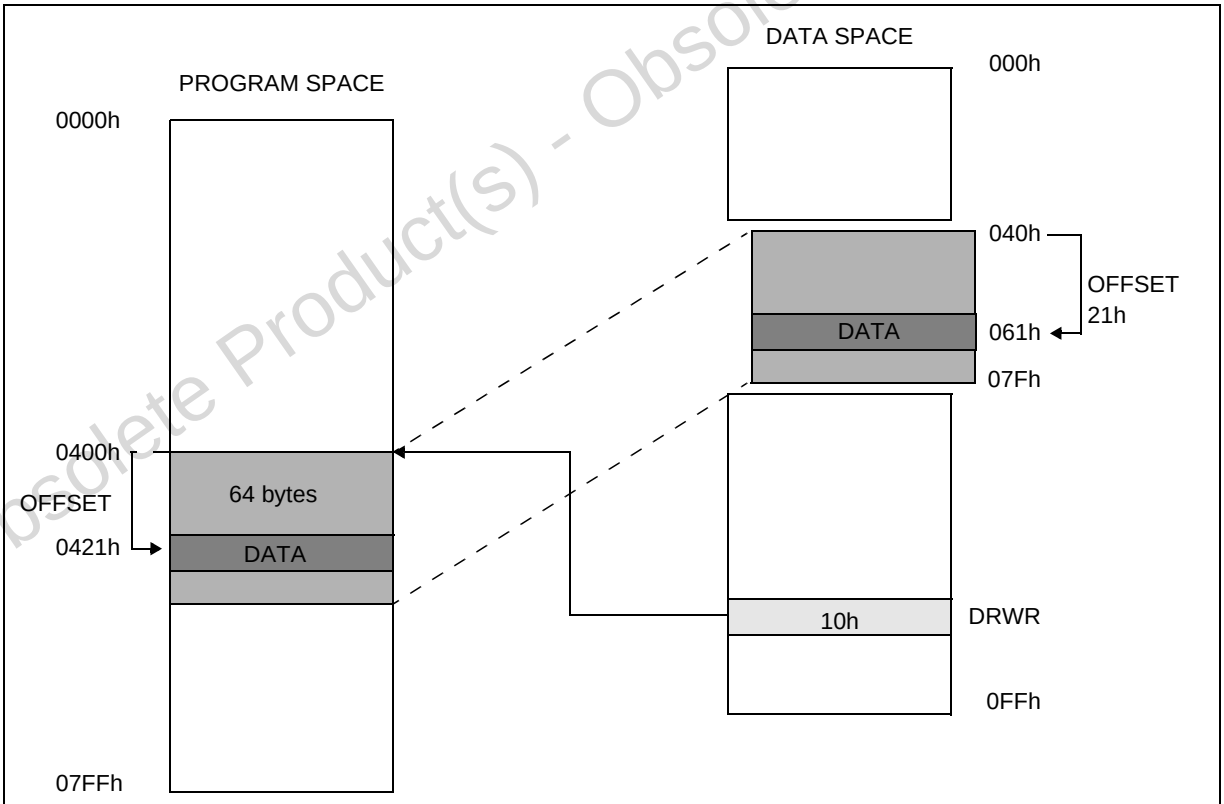
To understand how to determine the DRWR and the content of the register, please refer to the example shown in Figure 6. In any case the calculation is automatically handled by the ST6 development tools.

Please refer to the user manual of the correspoding tool.

3.1.6.3 Recommendations

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

Figure 6. Data read-only memory Window Memory Addressing



DATA address in Program memory : 421h
DRWR content : 421h / 3Fh (64) = 10H data is located in 64-bytes window number 10h
64-byte window start address : 10h x 3Fh = 400h
Register (A, X,...)content : Offset = (421h - 400h) + 40h (Data ROM Window start address in data space) = 61h

Obsolete Product(s) - Obsolete Product(s)

CPU REGISTERS (Cont'd)

The 12-bit length allows the direct addressing of 4096 bytes in Program Space.

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

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

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

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

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

C : Carry flag.

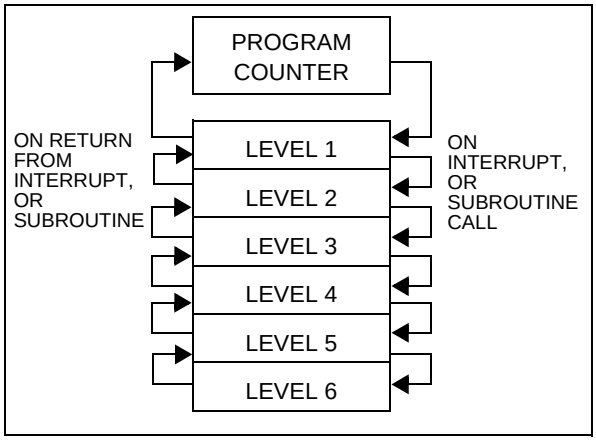
This bit is set when a carry or a borrow occurs during arithmetic operations; otherwise it is cleared. The Carry flag is also set to the value of the bit tested in a bit test instruction; it also participates in the rotate left instruction.
0: No carry has occured
1: A carry has occured

Z : Zero flag
This flag is set if the result of the last arithmetic or logical operation was equal to zero; otherwise it is cleared.
0: The result of the last operation is different from zero
1: The result of the last operation is zero

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

Stack. The ST6 CPU includes a true LIFO (Last In First Out) hardware stack which eliminates the need for a stack pointer. The stack consists of six separate 12-bit RAM locations that do not belong to the data space RAM area. When a subroutine call (or interrupt request) occurs, the contents of each level are shifted into the next level down, while the content of the PC is shifted into the first level (the original contents of the sixth stack level are lost). When a subroutine or interrupt return occurs (RET or RETI instructions), the first level register is shifted back into the PC and the value of each level is popped back into the previous level.

Figure 8. Stack manipulation



Since the accumulator, in common with all other data space registers, is not stored in this stack, management of these registers should be performed within the subroutine.

Caution: The stack will remain in its "deepest" position if more than 6 nested calls or interrupts are executed, and consequently the last return address will be lost.

It will also remain in its highest position if the stack is empty and a RET or RETI is executed. In this case the next instruction will be executed.

Obsolete Product(s) - Obsolete Product(s)

6.5 EXTERNAL INTERRUPTS (I/O Ports)

External interrupt vectors can be loaded into the PC register if the corresponding external interrupt occurred and if the GEN bit is set. These interrupts allow the processor to exit from STOP mode.

The external interrupt polarity is selected through the IOR register.

External interrupts are linked to vectors #1 and # 2.

Interrupt requests on vector #1 can be configured either as edge or level-sensitive using the LES bit in the IOR Register.

Interrupt requests from vector #2 are always edge sensitive. The edge polarity can be configured using the ESB bit in the IOR Register.

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

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

6.5.1 Notes on using External Interrupts

ESB bit Spurious Interrupt on Vector #2

If a pin associated with interrupt vector #2 is configured as interrupt with pull-up, whenever vector #2 is configured to be rising edge sensitive (by setting the ESB bit in the IOR register), an interrupt is latched although a rising edge may not have occured on the associated pin.

This is due to the vector #2 circuitry.The workaround is to discard this first interrupt request in the routine (using a flag for example).

Masking of One Interrupt by Another on Vector #2.

When two or more port pins (associated with interrupt vector #2) are configured together as input with interrupt (falling edge sensitive), as long as one pin is stuck at '0', the other pin can never generate an interrupt even if an active edge occurs at this pin. The same thing occurs when one pin is stuck at '1' and interrupt vector #2 is configured as rising edge sensitive.

To avoid this the first pin must input a signal that goes back up to '1' right after the falling edge. Otherwise, in the interrupt routine for the first pin, deactivate the “input with interrupt” mode using the port control registers (DDR, OR, DR). An active edge on another pin can then be latched.

I/O port Configuration Spurious Interrupt on Vector #2

If a pin associated with interrupt vector #2 is in ‘input with pull-up’ state, a ‘0’ level is present on the pin and the ESB bit = 0, when the I/O pin is configured as interrupt with pull-up by writing to the DDRx, ORx and DRx register bits, an interrupt is latched although a falling edge may not have occurred on the associated pin.

In the opposite case, if the pin is in interrupt with pull-up state , a 0 level is present on the pin and the ESB bit =1, when the I/O port is configured as input with pull-up by writing to the DDRx, ORx and DRx bits, an interrupt is latched although a rising edge may not have occurred on the associated pin.

Obsolete Product(s) - Obsolete Product(s)

7.2 WAIT MODE

The MCU goes into WAIT mode as soon as the WAIT instruction is executed. This has the following effects:

– Program execution is stopped, the microcontroller software can be considered as being in a “frozen” state.

– RAM contents and peripheral registers are preserved as long as the power supply voltage is higher than the RAM retention voltage.

– The oscillator is kept running to provide a clock to the peripherals; they are still active.

WAIT mode can be used when the user wants to reduce the MCU power consumption during idle periods, while not losing track of time or the ability to monitor external events. WAIT mode places the MCU in a low power consumption mode by stopping the CPU. The active oscillator (main oscillator or LFAO) is kept running in order to provide a clock signal to the peripherals.

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

Exit from Wait mode

The MCU remains in WAIT mode until one of the following events occurs:

– RESET (Watchdog, LVD or $\overline{\text{RESET}}$ pin)

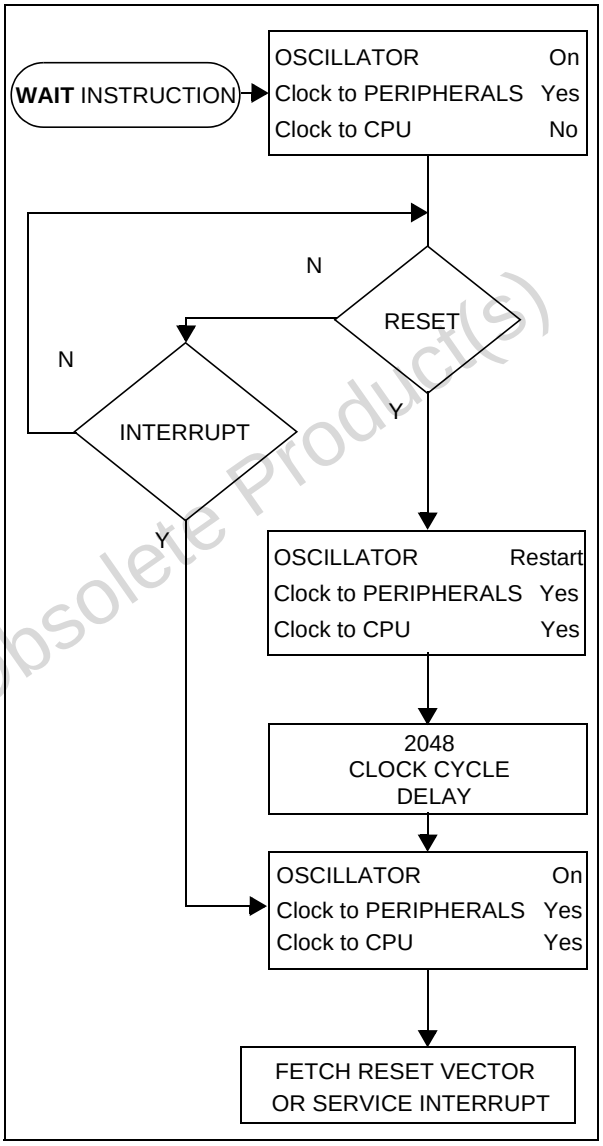
– A peripheral interrupt (timer, ADC,...),

– An external interrupt (I/O port, NMI)

The Program Counter then branches to the starting address of the interrupt or RESET service routine. Refer to Figure 20.

See also Section 7.4.1.

Figure 20. WAIT Mode Flowchart



Obsolete Product(s) - Obsolete Product(s)

7.3 STOP MODE

STOP mode is the lowest power consumption mode of the MCU (see Figure 22).

The MCU goes into STOP mode as soon as the STOP instruction is executed. This has the following effects:

– Program execution is stopped, the microcontroller can be considered as being “frozen”.

– The contents of RAM and the peripheral registers are kept safely as long as the power supply voltage is higher than the RAM retention voltage.

– The oscillator is stopped, so peripherals cannot work except the those that can be driven by an external clock.

Exit from STOP Mode

The MCU remains in STOP mode until one of the following events occurs:

– RESET (Watchdog, LVD or $\overline{\text{RESET}}$ pin)

– A peripheral interrupt (assuming this peripheral can be driven by an external clock)

– An external interrupt (I/O port, NMI)

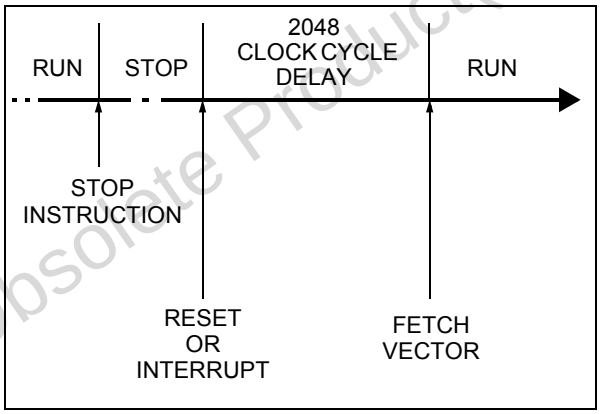
In all cases a delay of 2048 clock cycles (f_{INT}) is generated to make sure the oscillator has started properly.

The Program Counter then points to the starting address of the interrupt or RESET service routine (see Figure 21).

STOP Mode and Watchdog

When the Watchdog is active (hardware or software activation), the STOP instruction is disabled and a WAIT instruction will be executed in its place unless the EXCTNL option bit is set to 1 in the option bytes and a a high level is present on the NMI pin. In this case, the STOP instruction will be executed and the Watchdog will be frozen.

Figure 21. STOP Mode Timing Overview



Obsolete Product(s) - Obsolete Product(s)

8 I/O PORTS

8.1 INTRODUCTION

Each I/O port contains up to 8 pins. Each pin can be programmed independently as digital input (with or without pull-up and interrupt generation), digital output (open drain, push-pull) or analog input (when available).

The I/O pins can be used in either standard or alternate function mode.

Standard I/O mode is used for:

– Transfer of data through digital inputs and outputs (on specific pins):

– External interrupt generation

Alternate function mode is used for:

– Alternate signal input/output for the on-chip peripherals

The generic I/O block diagram is shown in Figure 23.

8.2 FUNCTIONAL DESCRIPTION

Each port is associated with 3 registers located in Data space:

– Data Register (DR)

– Data Direction Register (DDR)

– Option Register (OR)

Each I/O pin may be programmed using the corresponding register bits in the DDR, DR and OR registers: bit x corresponding to pin x of the port. Table 8 illustrates the various port configurations which can be selected by user software.

During MCU initialization, all I/O registers are cleared and the input mode with pull-up and no interrupt generation is selected for all the pins, thus avoiding pin conflicts.

8.2.1 Digital Input Modes

The input configuration is selected by clearing the corresponding DDR register bit.

In this case, reading the DR register returns the digital value applied to the external I/O pin.

Different input modes can be selected by software through the DR and OR registers, see Table 8.

External Interrupt Function

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

8.2.2 Analog Inputs

Some pins can be configured as analog inputs by programming the OR and DR registers accordingly, see Table 8. These analog inputs are connected to the on-chip 8-bit Analog to Digital Converter.

Caution: ONLY ONE pin should be programmed as an analog input at any time, since by selecting more than one input simultaneously their pins will be effectively shorted.

8.2.3 Output Modes

The output configuration is selected by setting the corresponding DDR register bit. In this case, writing to the DR register applies this digital value to the I/O pin through the latch. Then, reading the DR register returns the previously stored value.

Two different output modes can be selected by software through the OR register: push-pull and open-drain.

DR register value and output pin status:

DR	Push-pull	Open-drain
0	V_{SS}	V_{SS}
1	V_{DD}	Floating

Note: The open drain setting is not a true open drain. This means it has the same structure as the push-pull setting but the P-buffer is deactivated. To avoid damaging the device, please respect the V_{OUT} absolute maximum rating described in the Electrical Characteristics section.

8.2.4 Alternate Functions

When an on-chip peripheral is configured to use a pin, the alternate function (timer input/output...) is not systematically selected but has to be configured through the DDR, OR and DR registers. Refer to the chapter describing the peripheral for more details.

Obsolete Product(s) - Obsolete Product(s)

WATCHDOG TIMER (Cont'd)

9.1.3 Functional Description

The watchdog activation is selected through an option in the option bytes:

– **HARDWARE** Watchdog option

After reset, the watchdog is permanently active, the C bit in the WDGR is forced high and the user can not change it. However, this bit can be read equally as 0 or 1.

– **SOFTWARE** Watchdog option

After reset, the watchdog is deactivated. The function is activated by setting C bit in the WDGR register. Once activated, it cannot be deactivated. The counter value stored in the WDGR register (bits SR:T0), is decremented every 3072 clock cycles. The length of the timeout period can be programmed by the user in 64 steps of 3072 clock cycles.

If the watchdog is activated (by setting the C bit) and when the SR bit is cleared, the watchdog initiates a reset cycle pulling the reset pin low for typically 500ns.

The application program must write in the WDGR register at regular intervals during normal operation to prevent an MCU reset. The value to be stored in the WDGR register must be between FEh and 02h (see Table 11). To run the watchdog function the following conditions must be true:

– The C bit is set (watchdog activated)

– The SR bit is set to prevent generating an immediate reset

– The T[5:0] bits contain the number of decrements which represent the time delay before the watchdog produces a reset.

Table 11. Watchdog Timing (f_{OSC} = 8 MHz)

	WDGR Register initial value	WDG timeout period (ms)
Max.	FEh	24.576
Min.	02h	0.384

9.1.3.1 Software Reset

The SR bit can be used to generate a software reset by clearing the SR bit while the C bit is set.

9.1.4 Recommendations

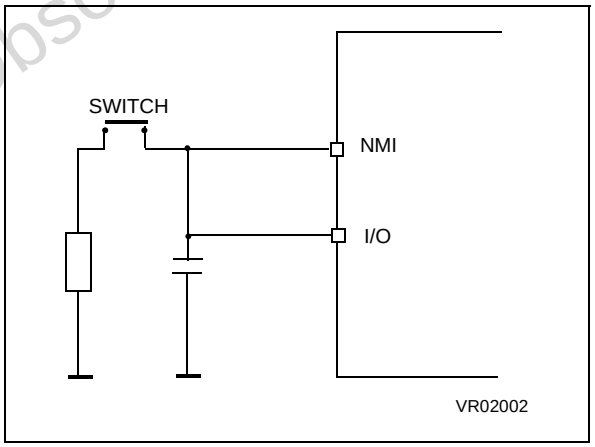
1. The Watchdog plays an important supporting role in the high noise immunity of ST62xx devices, and should be used wherever possible. Watchdog related options should be selected on the basis of a trade-off between application security and STOP mode availability (refer to the description of the WDACT and EXTCNTL bits on the Option Bytes).

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

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

The NMI pin can be connected to an I/O line (see Figure 26) to allow its state to be controlled by software. The I/O line can then be used to keep NMI low while Watchdog protection is required, or to avoid noise or key bounce. When no more processing is required, the I/O line is released and the device placed in STOP mode for lowest power consumption.

Figure 26. A typical circuit making use of the EXERNAL STOP MODE CONTROL feature



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

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

```
jrr 0, WDGR, #+3 ; If C=0,jump to next
ldi WDGR, 0FDH   ; SR=0 -> reset

next :
```

Obsolete Product(s) - Obsolete Product(s)

WATCHDOG TIMER (Cont'd)

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

For more information on the use of the watchdog, please read application note AN1015.

Note: This note applies only when the watchdog is used as a standard timer. It is recommended to read the counter twice, as it may sometimes return an invalid value if the read is performed while the counter is decremented (counter bits in transient state). To validate the return value, both values read must be equal. The counter decrements every 384 µs at 8 MHz f_{OSC}.

9.1.5 Low Power Modes

Mode	Description
WAIT	No effect on Watchdog.
STOP	Behaviour depends on the EXTCNTL option in the Option bytes: **1. Watchdog disabled:** The MCU will enter Stop mode if a STOP instruction is executed. **2. Watchdog enabled and EXTCNTL option disabled:** If a STOP instruction is encountered, it is interpreted as a WAIT. **3. Watchdog and EXTCNTL option enabled:** If a STOP instruction is encountered when the NMI pin is low, it is interpreted as a WAIT. If, however, the STOP instruction is encountered when the NMI pin is high, the Watchdog counter is frozen and the CPU enters STOP mode. When the MCU exits STOP mode (i.e. when an interrupt is generated), the Watchdog resumes its activity.

9.1.6 Interrupts

None.

Obsolete Product(s) - Obsolete Product(s)

A/D CONVERTER (Cont'd)

9.3.5 Low Power Modes

Mode	Description
WAIT	No effect on A/D Converter. ADC interrupts cause the device to exit from Wait mode.
STOP	A/D Converter disabled.

Note: The A/D converter may be disabled by clearing the PDS bit. This feature allows reduced power consumption when no conversion is needed.

9.3.6 Interrupts

Interrupt Event	Event Flag	Enable Bit	Exit from Wait	Exit from Stop
End of Conversion	EOC	EAI	Yes	No

Note: The EOC bit is cleared only when a new conversion is started (it cannot be cleared by writing 0). To avoid generating further EOC interrupt, the EAI bit has to be cleared within the ADC interrupt subroutine.

9.3.7 Register Description

A/D CONVERTER CONTROL REGISTER (ADCR)

Address: 0D1h - Read/Write (Bit 6 Read Only, Bit 5 Write Only)

Reset value: 0100 0000 (40h)

7							0
EAI	EOC	STA	PDS	ADCR 3	OSC OFF	ADCR 1	ADCR 0

Bit 7 = **EAI** *Enable A/D Interrupt.*
0: ADC interrupt disabled
1: ADC interrupt enabled

Bit 6 = **EOC** *End of conversion. Read Only*
When a conversion has been completed, this bit is set by hardware and an interrupt request is generated if the EAI bit is set. The EOC bit is automatically cleared when the STA bit is set. Data in the data conversion register are valid only when this bit is set to "1".
0: Conversion is not complete
1: Conversion can be read from the ADR register

Bit 5 = **STA***: Start of Conversion. Write Only.*
0: No effect
1: Start conversion

Note: Setting this bit automatically clears the EOC bit. If the bit is set again when a conversion is in progress, the present conversion is stopped and a new one will take place. This bit is write only, any attempt to read it will show a logical zero.

Bit 4 = **PDS** *Power Down Selection.*

0: A/D converter is switched off
1: A/D converter is switched on

Bit 3 = **ADCR3 Reserved**, must be cleared.

Bit 2 = **OSCOFF** *Main Oscillator off.*

0: Main Oscillator enabled
1: Main Oscillator disabled

Note: This bit does not apply to the ADC peripheral but to the main clock system. Refer to the Clock System section.

Bits 1:0 = **ADCR[1:0] Reserved**, must be cleared.

A/D CONVERTER DATA REGISTER (ADR)

Address: 0D0h - Read only

Reset value: xxxx xxxx (xxh)

7							0
ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0

Bits 7:0 = **ADR[7:0]***: 8 Bit A/D Conversion Result.*

Table 15. ADC Register Map and Reset Values

Address (Hex.)	Register Label	7	6	5	4	3	2	1	0
0D0h	**ADR** Reset Value	ADR7 0	ADR6 0	ADR5 0	ADR4 0	ADR3 0	ADR2 0	ADR1 0	ADR0 0
0D1h	**ADCR** Reset Value	EAI 0	EOC 1	STA 0	PDS 0	ADCR3 0	OSCOFF 0	ADCR1 0	ADCR0 0

Obsolete Product(s) Obsolete Product(s)

INSTRUCTION SET (Cont'd)

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

Table 17. Arithmetic & Logic Instructions

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

Notes:
X,Y Index Registers
V, W Short Direct Registers
' Affected
Immediate data (stored in ROM memory)
* Not Affected
rr Data space register

Obsolete Product(s) - Obsolete Product(s)

INSTRUCTION SET (Cont'd)

Conditional Branch. Branch instructions perform a branch in the program when the selected condition is met.

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

Control Instructions. Control instructions control microcontroller operations during program execution.

Jump and Call. These two instructions are used to perform long (12-bit) jumps or subroutine calls to any location in the whole program space.

Table 18. 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	*	'
JRS b, rr, ee	Bit = 1	3	5	*	'

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
' Affected. The tested bit is shifted into carry.
* Not Affected

Table 19. 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
Bit Manipulation Instructions should not be used on Port Data Registers and any registers with read only and/or write only bits (see I/O port chapter)

Table 20. Control Instructions

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

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

Table 21. 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

Obsolete Product(s) - Obsolete Product(s)

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

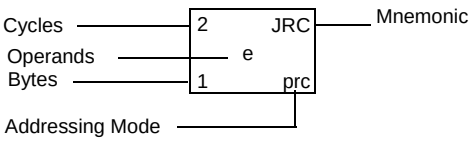
LOW / HI	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	LOW / HI
0 0000	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRR b0,rr,ee 3 bt	2 JRZ e NOP 1 pcr	#	2 JRC e 1 prc	4 LD a,(x) 1 ind	**0 0000**
1 0001	2 JRNZ e 1 pcr	4 CALL abc 2 ext	2 JRNC e 1 pcr	5 JRS b0,rr,ee 3 bt	2 JRZ e 1 pcr	4 INC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

Abbreviations for Addressing Modes:

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

Legend:

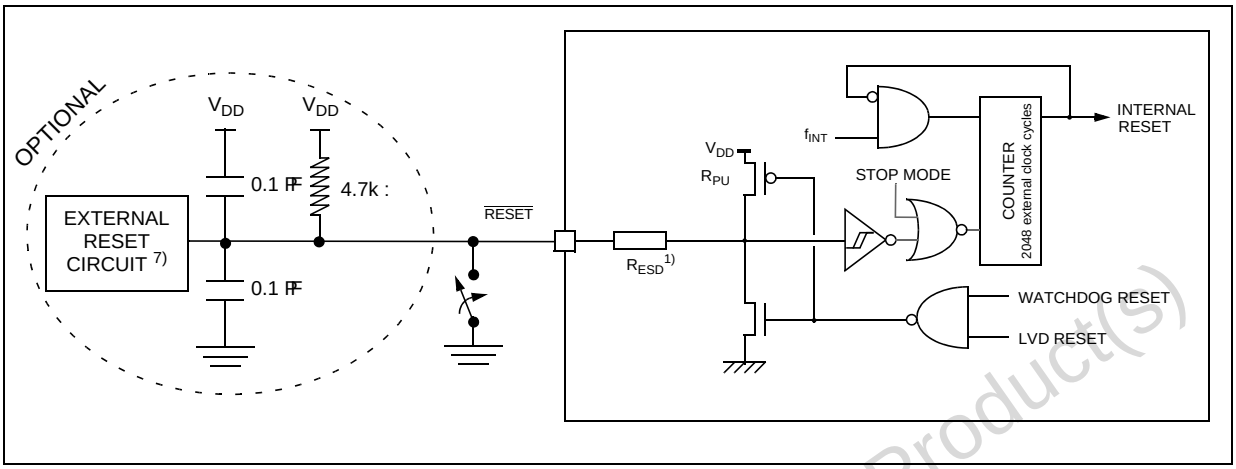
#	Indicates Illegal Instructions
e	5-bit Displacement
b	3-bit Address
rr	1-byte Data space address
nn	1-byte immediate data
abc	12-bit address
ee	8-bit displacement



Obsolete Product(s) - Obsolete Product(s)

CONTROL PIN CHARACTERISTICS (Cont'd)

Figure 70. Typical Application with $\overline{\text{RESET}}$ pin [8)]



11.9.2 NMI Pin

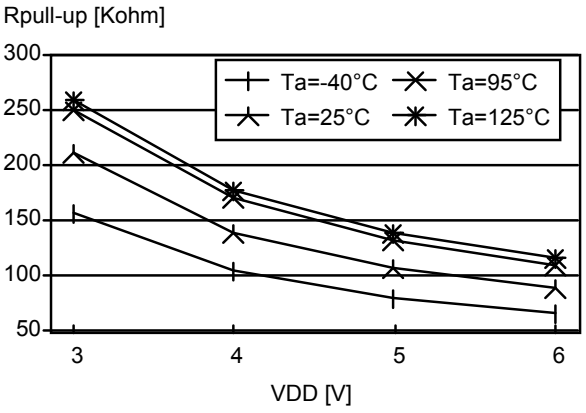
Subject to general operating conditions for V_{DD}, f_{OSC}, and T_A unless otherwise specified.

Symbol	Parameter	Conditions		Min	Typ [1)]	Max	Unit
V_{IL}	Input low level voltage [2)]					$0.3xV_{DD}$	V
V_{IH}	Input high level voltage [2)]			$0.7xV_{DD}$			
V_{hys}	Schmitt trigger voltage hysteresis [3)]			200	400		mV
$R_{pull-up}$	Weak pull-up equivalent resistor [4)]	V_{IN} V_{SS}	V_{DD}=5V	40	100	350	k :
			V_{DD}=3.3V	80	200	700	

Notes:

1. Unless otherwise specified, typical data are based on T_A=25°C and V_{DD}=5V.
2. Data based on characterization results, not tested in production.
3. Hysteresis voltage between Schmitt trigger switching levels. Based on characterization results, not tested.
4. The $R_{pull-up}$ equivalent resistor is based on a resistive transistor. This data is based on characterization results, not tested in production.

Figure 71. Typical $R_{pull-up}$ vs. V_{DD} with V_{IN}=V_{SS}



Obsolete Product(s) - Obsolete Product(s)

11.11 8-BIT ADC CHARACTERISTICS

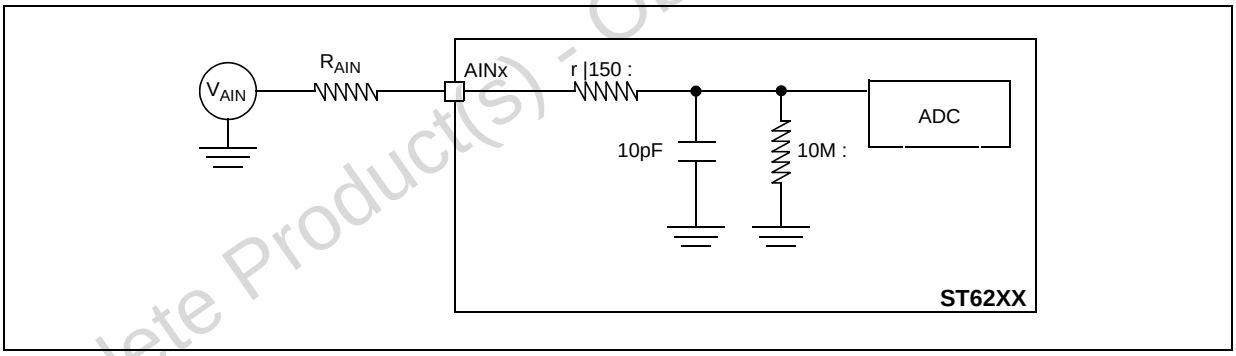
Subject to general operating conditions for V_{DD}, f_{OSC}, and T_A unless otherwise specified.

Symbol	Parameter	Conditions	Min	Typ [1]	Max	Unit
f_{OSC}	Clock frequency		1.2		f_{OSC}	MHz
V_{AIN}	Conversion range voltage [2]		V_{SS}		V_{DD}	V
R_{AIN}	External input resistor				10 [3]	k :
t_{ADC}	Total convertion time	f_{OSC}=8MHz f_{OSC}=4MHz		70 140		µs
t_{STAB}	Stabilization time [4]			2	4	t_{CPU}
		f_{OSC}=8MHz		3.25	6.5	µs
AD_I	Analog input current during conversion				1.0	µA
AC_{IN}	Analog input capacitance			2	5	pF

Notes:

1. Unless otherwise specified, typical data are based on T_A=25°C and V_{DD}=5V.
2. The ADC refers to V_{DD} and V_{SS}.
3. Any added external serial resistor will downgrade the ADC accuracy (especially for resistance greater than 10k :). Data based on characterization results, not tested in production.
4. As a stabilization time for the AD converter is required, the first conversion after the enable can be wrong.

Figure 72. Typical Application with ADC



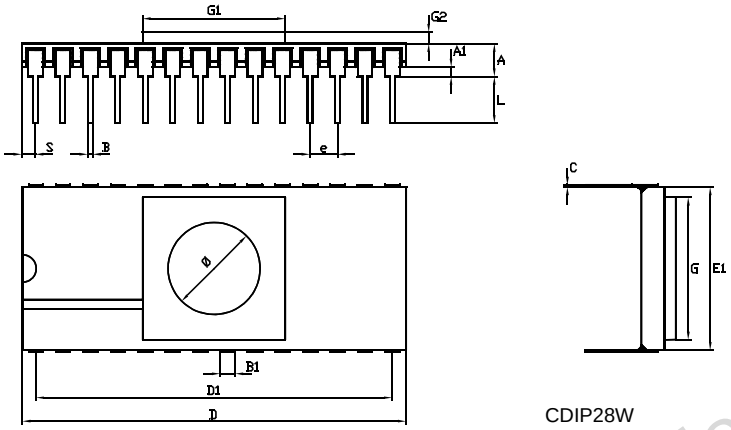
Note: ADC not present on some devices. See device summary on page 1.

Obsolete Product(s) - Obsolete Product(s)

Obsolete Product(s) - Obsolete Product(s)

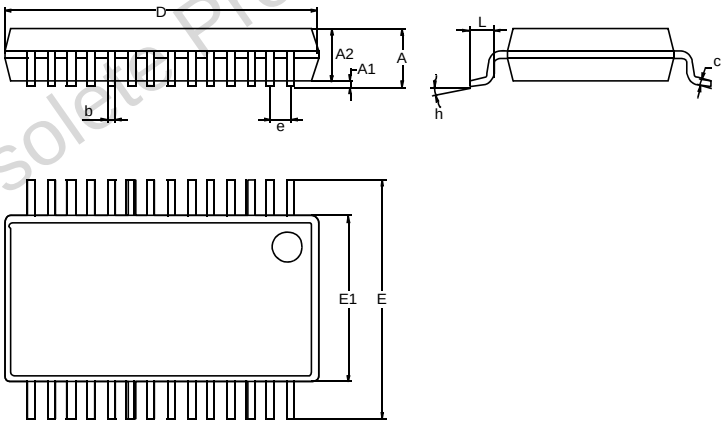
PACKAGE MECHANICAL DATA (Cont'd)

Figure 76. 28-Pin Ceramic Side-Brazed Dual In-Line Package



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A			4.17			0.164
A1	0.76			0.030		
B	0.36	0.46	0.56	0.014	0.018	0.022
B1	0.76	1.27	1.78	0.030	0.050	0.070
C	0.20	0.25	0.38	0.008	0.010	0.015
D	34.95	35.56	36.17	1.376	1.400	1.424
D1		33.02			1.300	
E1	14.61	15.11	15.62	0.575	0.595	0.615
e		2.54			0.100	
G	12.70	12.95	13.21	0.500	0.510	0.520
G1	12.70	12.95	13.21	0.500	0.510	0.520
G2		1.14			0.045	
L	2.92		5.08	0.115		0.200
S		1.27			0.050	
Ø		8.89			0.350	
	Number of Pins					
N	28					

Figure 77. 28-Pin Plastic Shrink Small Outline Package



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A			2.00			0.079
A1	0.05			0.002		
A2	1.65	1.75	1.85	0.065	0.069	0.073
b	0.22		0.38	0.009		0.015
c	0.09		0.25	0.004		0.010
D	9.90	10.20	10.50	0.390	0.402	0.413
E	7.40	7.80	8.20	0.291	0.307	0.323
E1	5.00	5.30	5.60	0.197	0.209	0.220
e		0.65			0.026	
T	0°	4°	8°	0°	4°	8°
L	0.55	0.75	0.95	0.022	0.030	0.037
	Number of Pins					
N	28					

Obsolete Product(s) - Obsolete Product(s)

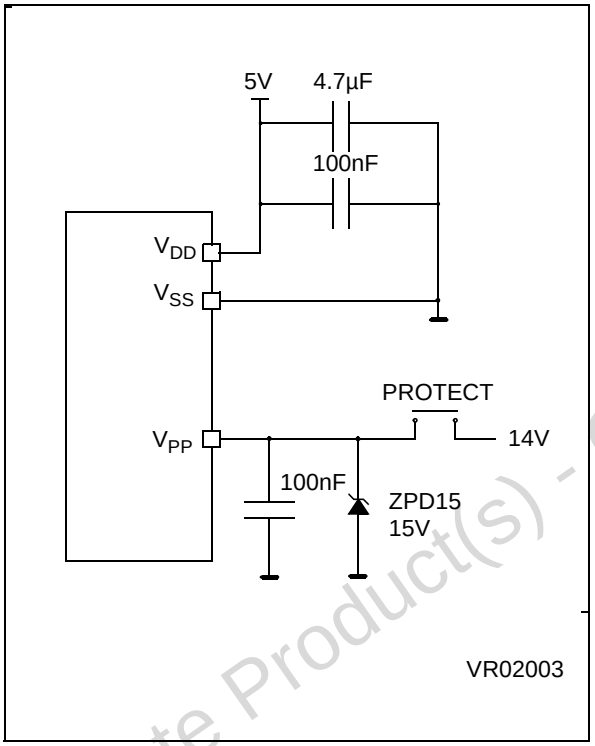
TRANSFER OF CUSTOMER CODE (Cont'd)

12.6.2 ROM Version

The ST6215C/25C are mask programmed ROM version of ST62T15C,T25C OTP devices.

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

Figure 79. Programming Circuit

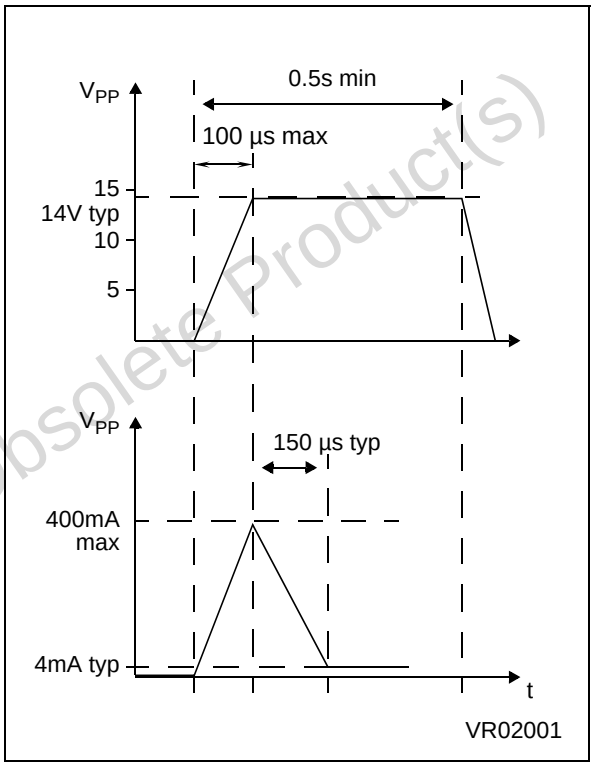


Note: ZPD15 is used for overvoltage protection

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

In case the user wants to blow this fuse, high voltage must be applied on the V_{PP} pin.

Figure 80. Programming wave form



Obsolete Product(s) - Obsolete Product(s)

IDENTIFICATION	DESCRIPTION
AN913	PWM GENERATION WITH ST62 16-BIT AUTO-RELOAD TIMER
AN914	USING ST626X SPI AS UART
AN1016	ST6 USING THE ST623XB/ST628XB UART
AN1050	ST6 INPUT CAPTURE WITH ST62 16-BIT AUTO-RELOAD TIMER
AN1127	USING THE ST62T6XC/5XC SPI IN MASTER MODE
GENERAL	
AN683	MCUS - 8/16-BIT MICROCONTROLLERS (MCUS) APPLICATION NOTES ABSTRACTS BY TOPICS
AN886	SELECTING BETWEEN ROM AND OTP FOR A MICROCONTROLLER
AN887	MAKING IT EASY WITH MICROCONTROLLERS
AN898	EMC GENERAL INFORMATION
AN899	SOLDERING RECOMMENDATIONS AND PACKAGING INFORMATION
AN900	INTRODUCTION TO SEMICONDUCTOR TECHNOLOGY
AN901	EMC GUIDE-LINES FOR MICROCONTROLLER - BASED APPLICATIONS
AN902	QUALITY AND RELIABILITY INFORMATION
AN912	A SIMPLE GUIDE TO DEVELOPMENT TOOLS
AN1181	ELECTROSTATIC DISHARGE SENSITIVITY MEASUREMENT

Obsolete Product(s) - Obsolete Product(s)