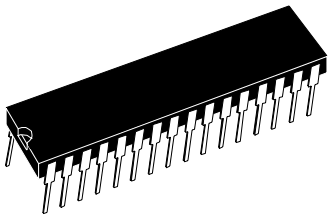
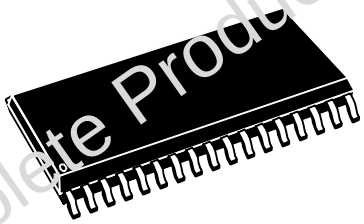
ST92141

8/16-BIT MCU FOR 3-PHASE AC MOTOR CONTROL

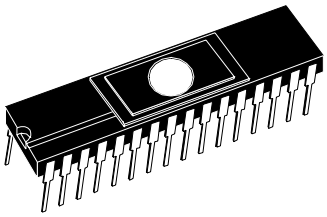
- Register File based 8/16 bit Core Architecture with RUN, WFI, SLOW, HALT and STOP modes
- 0-25 MHz Operation (internal clock) @ 5V±10% voltage range
- -40°C to +85°C Operating Temperature Range
- Fully Programmable PLL Clock Generator, with Frequency Multiplication and low frequency, low cost external crystal (3-5 MHz)
- Minimum Instruction Cycle time: 160 ns - (@ 25 MHz internal clock frequency)
- Internal Memory:
  - EPROM/OTP/FASTROM 16K bytes
  - RAM 512 bytes
- 224 general purpose registers available as RAM, accumulators or index pointers (register file)
- 32-pin Dual Inline and 34-pin Small Outline Packages
- 15 programmable I/O pins with Schmitt Trigger input, including 4 high sink outputs (20mA @ V_{OL}=3V)
- 4 Wake-up Interrupts (one usable as Non-Maskable Interrupt) for emergency event management
- 3-phase Induction Motor Controller (IMC) Peripheral with 3 pairs of PWM outputs and asynchronous emergency stop
- Serial Peripheral Interface (SPI) with Master/ Slave Mode capability
- 16-bit Timer with 8-bit Prescaler usable as a Watchdog Timer
- 16-bit Standard Timer with 8-bit Prescaler
- 16-bit Extended Function Timer with Prescaler, 2 Input Captures and 2 Output Compares
- 8-bit Analog to Digital Converter allowing up to 6 input channels with autoscan and watchdog capability
- Low Voltage Detector Reset
- Rich Instruction Set with 14 Addressing Modes
- Division-by-Zero trap generation
- Versatile Development Tools, including Assembler, Linker, C-compiler, Archiver, Source Level Debugger and Hardware Emulators with Real-Time Operating System available from Third Parties



PSDIP32



SO34 Shrink



CSDIP32W

DEVICE SUMMARY

DEVICE	Program Memory (Bytes)	RAM (Bytes)	PACKAGE
ST92P141	16K FASTROM	512	PSDIP32/ SO34
ST92E141	16K EPROM	512	CSDIP32W
ST92T141	16K OTP	512	PSDIP32/ SO34

Rev. 1.7

Table of Contents

Table of Contents

Table of Contents

Table of Contents

1 GENERAL DESCRIPTION

1.1 INTRODUCTION

The ST92141 microcontroller is developed and manufactured by STMicroelectronics using a proprietary n-well HCMOS process. Its performance derives from the use of a flexible 256-register programming model for ultra-fast context switching and real-time event response. The intelligent on-chip peripherals offload the ST9 core from I/O and data management processing tasks allowing critical application tasks to get the maximum use of core resources. The new-generation ST9 MCU devices now also support low power consumption and low voltage operation for power-efficient and low-cost embedded systems.

1.1.1 ST9+ Core

The advanced Core consists of the Central Processing Unit (CPU), the Register File, the Interrupt controller, and the Memory Management Unit. The MMU allows addressing of up to 4 Megabytes of program and data mapped into a single linear space.

Four independent buses are controlled by the Core: a 16-bit memory bus, an 8-bit register data bus, an 8-bit register address bus and a 6-bit interrupt bus which connects the interrupt controllers in the on-chip peripherals with the core.

Note: The DMA features of the ST9+ core are not used by the on-chip peripherals of the ST92141.

This multiple bus architecture makes the ST9 family devices highly efficient for accessing on and off-chip memory and fast exchange of data with the on-chip peripherals.

The general-purpose registers can be used as accumulators, index registers, or address pointers. Adjacent register pairs make up 16-bit registers for addressing or 16-bit processing. Although the ST9 has an 8-bit ALU, the chip handles 16-bit operations, including arithmetic, loads/stores, and memory/register and memory/memory exchanges.

1.1.2 Power Saving Modes

To optimize performance versus power consumption, a range of operating modes can be dynamically selected by software according to the requirements of the application.

Run Mode. This is the full speed execution mode with CPU and peripherals running at the maximum clock speed delivered either by the Phase Locked Loop controlled by the RCCU (Reset and Clock Control Unit), directly by the oscillator or by an external source (dedicated Pin or Alternate Function).

Slow Mode. Power consumption can be significantly reduced by running the CPU and the peripherals at reduced clock speed using the CPU Prescaler and RCCU Clock Divider.

Wait For Interrupt Mode. The Wait For Interrupt (WFI) instruction suspends program execution until an interrupt request is acknowledged. During WFI, the CPU clock is halted while the peripheral with interrupt capability and interrupt controller are kept running at a frequency that can be programmed by software in the RCCU registers. In this mode, the power consumption of the device can be reduced by more than 95% (Low Power WFI).

Halt Mode. When executing the HALT instruction, and if the Watchdog is not enabled, the CPU and its peripherals stop operating. If however the Watchdog is enabled, the HALT instruction has no effect. The main difference between Halt mode and Stop mode is that a reset is necessary to exit from Halt mode which causes the system to be reinitialized.

Stop Mode. When Stop mode is requested by executing the STOP sequence (see Wake-up Management Unit section), the CPU and the peripherals stop operating. Operations resume after a wake-up line is activated. The difference between Stop mode and Halt mode is in the way the CPU exits each state: when the STOP sequence is executed, the status of the registers is recorded, and when the system exits from Stop mode the CPU continues execution with the same status, without a system reset.

The Watchdog counter, if enabled, is stopped. After exiting Stop mode it restarts counting from where it left off.

When the MCU exits from STOP mode, the oscillator, which was also sleeping, requires a start-up time to restart working properly. An internal counter is present to guarantee that, after exiting Stop Mode, all operations take place with the clock stabilised.

1.1.3 System Clock

A programmable PLL Clock Generator allows standard 3 to 5 MHz crystals to be used to obtain a large range of internal frequencies up to 25MHz.

1.1.4 Low Voltage Reset

The on-chip Low Voltage Detector (LVD) generates a static reset when the supply voltage is below a reference value. The LVD works both during power-on as well as when the power supply drops (brown-out). The reference value for the voltage drop is lower than the reference value for power-on in order to avoid a parasitic reset when the MCU starts running and sinks current on the supply (hysteresis).

1.1.5 I/O Ports

The I/O lines are grouped into two I/O Ports and can be configured on a bit basis to provide timing, status signals, an address/data bus for timer inputs and outputs, analog inputs, external wake-up lines and serial or parallel I/O.

1.1.6 3-phase Induction Motor Controller

The IMC controller is designed for variable speed motor control applications. Three pairs of PWM outputs are available for controlling a three-phase motor drive. Rotor speed feedback is provided by capturing a tachogenerator input signal. Emergency stop is provided by putting the PWM outputs in high impedance mode upon asynchronous faulty event on NMI pin.

1.1.7 Watchdog Timer (WDT)

The Watchdog timer can be used to monitor system integrity. When enabled, it generates a reset after a timeout period unless the counter is refreshed by the application software. For additional security, watchdog function can be enabled by hardware using a specific pin.

1.1.8 Standard Timer

The standard timer includes a programmable 16-bit down-counter and an associated 8-bit prescaler with Single and Continuous counting modes.

1.1.9 Extended Function Timer

The Extended Function Timer can be used for a wide range of standard timing tasks. It has a 16-bit free running counter with programmable prescaler. Each timer can have up to 2 input capture and 2 output compare pins with associated registers. This allows applications to measure pulse intervals or generate pulse waveforms. Timer overflow and other events are flagged in a status register with optional interrupt generation.

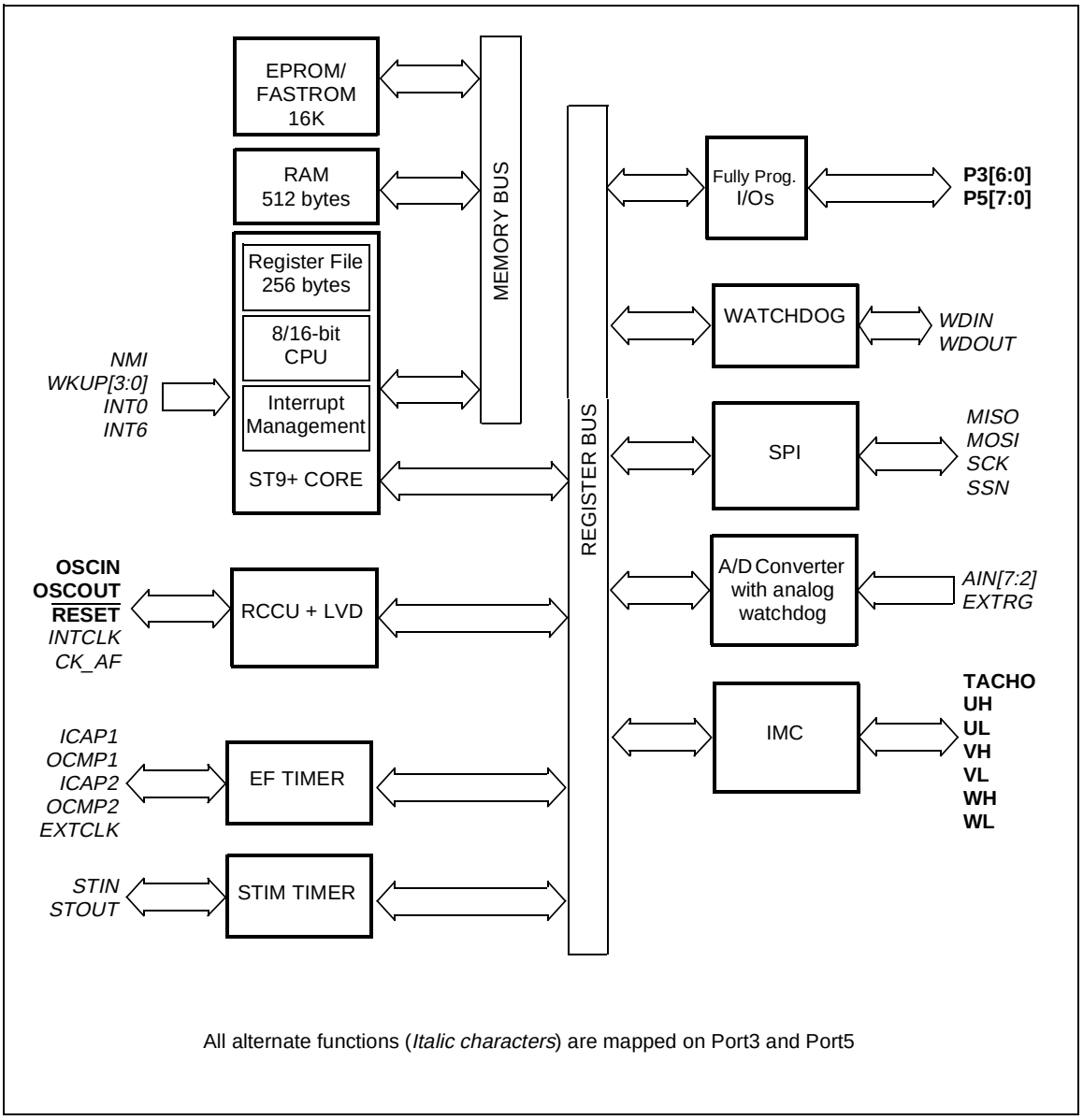
1.1.10 Serial Peripheral Interface (SPI)

The SPI bus is used to communicate with external devices via the SPI, or I²C bus communication standards.

1.1.11 Analog/Digital Converter (ADC)

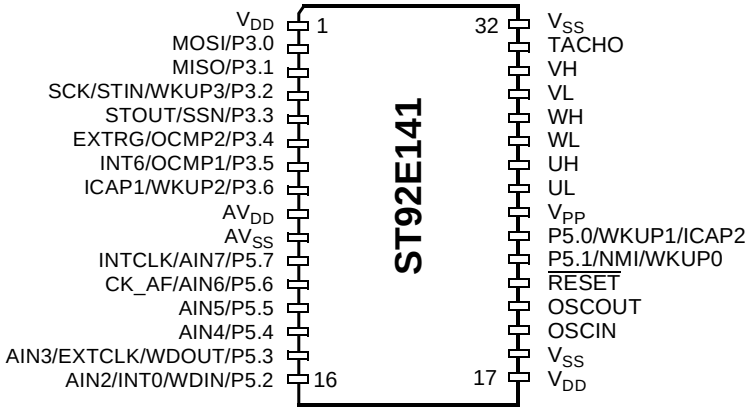
The ADC provides up to 6 analog inputs with on-chip sample and hold. The analog watchdog generates an interrupt when the input voltage moves out of a preset threshold.

Figure 1. ST92141 Block Diagram

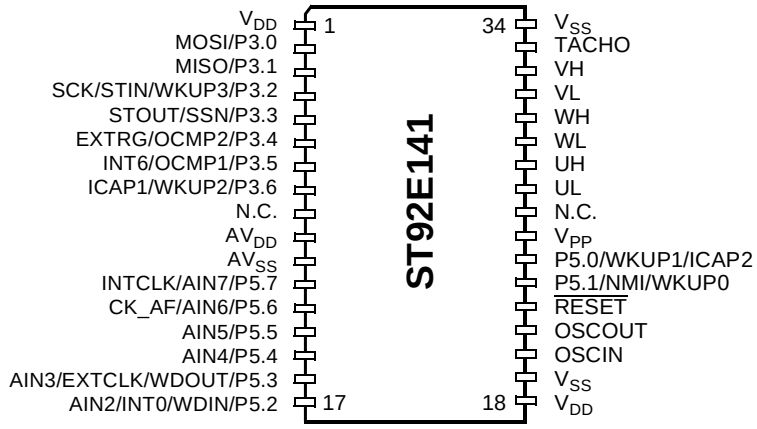


All alternate functions (*Italic characters*) are mapped on Port3 and Port5

1.2 PIN DESCRIPTION



PSDIP32/CSDIP32W Package



SO34 Package

Table 1. Power Supply Pins

Name	Function	SDIP32	SO34
V_{PP}	Programming voltage for EPROM/OTP devices. Must be connected to V_{SS} in user mode.	24	25
V_{DD}	Main power supply voltage (5V ± 10% (2 pins internally connected)	17	18
		1	1
V_{SS}	Digital Circuit Ground (2 pins internally connected)	18	19
		32	34
AV_{DD}	Analog V_{DD} of the Analog to Digital Converter	9	10
AV_{SS}	Analog V_{SS} of the Analog to Digital Converter	10	11

Table 2. Primary Function pins

Name	Function	SDIP32	SO34
TACHO	Signal input from a tachogenerator to the IMC controller for measuring the rotor speed	31	33
UH	U-phase PWM output signal	26	28
VH	V-phase PWM output signal	30	32
WH	W-phase PWM output signal	28	30
UL	The complemented UH, VH, WH output signals with added dead time to avoid crossover conduction from the power driver	25	27
VL		29	31
WL		27	29
$\overline{\text{RESET}}$	Reset (input, active low). The ST9+ is initialised by the Reset signal. With the deactivation of $\overline{\text{RESET}}$, program execution begins from the memory location pointed to by the vector contained in memory locations 00h and 01h	21	22
OSCIN	OSCIN is the input of the oscillator inverter and internal clock generator. OSCIN and OSCOUT connect a parallel-resonant crystal (3 to 5 MHz), or an external source to the on-chip clock oscillator and buffer	19	20
OSCOUT	OSCOUT is the output of the oscillator inverter	20	21

1.2.1 I/O Port Configuration

All ports can be individually configured as input, bi-directional, output, or alternate function. Refer to the Port Bit Configuration Table in the I/O Port Chapter.

All I/Os are implemented with a High Hysteresis or Standard Hysteresis Schmitt trigger function (See Electrical Characteristics).

Weak Pull-Up = This column indicates if a weak pull-up is present or not (refer to Table 3).

– If WPU = yes, then the WPU can be enabled/disable by software

– If WPU = no, then enabling the WPU by software has no effect

All port output configurations can be software selected on a bit basis to provide push-pull or open drain driving capabilities. For all ports, when configured as open-drain, the voltage on the pin must never exceed the V_{DD} power line value (refer to Electrical characteristics section).

1.2.2 I/O Port Reset State

I/Os are reset asynchronously as soon as the $\overline{\text{RESET}}$ pin is asserted low.

All I/Os are forced by the Reset in "floating input" configuration mode.

WARNING

When a common pin is declared to be connected to an alternate function input and to an alternate function output, the user must be aware of the fact that the alternate function output signal always inputs to the alternate function module declared as input.

When any given pin is declared to be connected to a digital alternate function input, the user must be aware of the fact that the alternate function input is always connected to the pin. When a given pin is declared to be connected to an analog alternate function input (ADC input for example) and if this pin is programmed in the "AF-OD" mode, the digital input path is disconnected from the pin to prevent any DC consumption.

Table 3. I/O Port Characteristics

	Input	Output	Weak Pull-Up	Reset State
Port 3[4:0]	Schmitt trigger (High Hysteresis)	Push-Pull/OD	Yes	Floating input
Port 3[6:5]	Schmitt trigger (High Hysteresis)	Push-Pull/OD (HC)	Yes	Floating input
Port 5.0	Schmitt trigger (High Hysteresis)	Push-Pull/OD (HC)	Yes	Floating input
Port 5.1	Schmitt trigger (High Hysteresis)	Push-Pull/OD	Yes	Floating input
Port 5.2	Schmitt trigger (Standard Hysteresis)	Push-Pull/OD (HC)	Yes	Floating input
Port 5[7:3]	Schmitt trigger (Standard Hysteresis)	Push-Pull/OD	Yes	Floating input

Legend: OD = Open Drain; HC= High current

Table 4. ST92141 Alternate functions

Port Name	General Purpose I/O	Pin No.		Alternate Functions		
		SDIP32	PSO34			
P3.0	All ports useable for general purpose I/O (input, output or bidirectional)	2	2	MOSI	I/O	SPI Master Output/Slave Input Data
P3.1		3	3	MISO	I/O	SPI Master Input/Slave Output Data
P3.2		4	4	WKUP3	I	Wake-up line 3
				STIN	I	Standard Timer Input
				SCK	I/O	SPI Serial Clock Input/Output
P3.3		5	5	SSN	I	SPI Slave Select
				STOUT	O	Standard Timer Output
P3.4		6	6	EXTRG	I	A/D External trigger
				OCPM2	O	Ext. Timer Output Compare 2
P3.5		7	7	INT6	I	External Interrupt 6
				OCMP1	O	Ext. Timer - Output Compare 1
P3.6		8	8	ICAP1	I	Ext. Timer - Input Capture 1
				WKUP2	I	Wake-up line 2
P5.0		23	24	ICAP2	I	Ext. Timer - Input Capture 2
				WKUP1	I	Wake-up line 1
P5.1		22	23	NMI	I	Not maskable Int.
				WKUP0	I	Wake-up line 0
P5.2		16	17	AIN2	I	Analog Data Input 2
				INT0	I	External Interrupt 0
				WDIN	I	Watchdog input
P5.3		15	16	AIN3	I	Analog Data Input 3
				EXTCLK	I	Ext. Timer - Input Clock
				WDOUT	O	Watchdog Output
P5.4		14	15	AIN4	I	Analog Data Input 4
P5.5		13	14	AIN5	I	Analog Data Input 5
P5.6		12	13	AIN6	I	Analog Data Input 6
				CK_AF	I	Clock Alternative Source
P5.7		11	12	AIN7	I	Analog Data Input 7
				INTCLK	O	Internal Main Clock

How to configure the I/O ports

To configure the I/O ports, use the information in Table 3 and Table 4 and the Port Bit Configuration Table in the I/O Ports Chapter on page 81.

I/O note = The hardware characteristics fixed for each port line in Table 3.

All I/O inputs have Schmitt trigger fixed by hardware so selecting CMOS or TTL input by software has no effect, the input will always be Schmitt Trigger. In particular, the Schmitt Triggers present on the P5[7:2] pins have a standard hysteresis whereas the remaining pins have Schmitt Triggers with High Hysteresis (refer to Electrical Specifications).

Alternate Functions (AF) = More than one AF cannot be assigned to an external pin at the same time:

An alternate function can be selected as follows.

AF Inputs:

– AF is selected implicitly by enabling the corresponding peripheral. Exceptions to this are ADC analog inputs which must be explicitly selected as AF by software.

AF Outputs or Bidirectional Lines:

– In the case of Outputs or I/Os, AF is selected explicitly by software.

Example 1: Standard Timer input

AF: STIN, Port: P3.2, I/O Note: Schmitt trigger.

Write the port configuration bits:

P3C2.2=1

P3C1.2=0

P3C0.2=1

or P3C2.2=0

P3C1.2=0

P3C0.2=1

Enable the Standard Timer input by software as described in the STIM chapter.

Example 2: Standard Timer output

AF: STOUT, Port: P3.3

Write the port configuration bits (for AF output push-pull):

P3C2.3=0

P3C1.3=1

P3C0.3=1

Example 3: ADC analog input

AF: AIN2, Port: P5.2, I/O Note: does not apply to analog inputs

Write the port configuration bits:

P5C2.2=1

P5C1.2=1

P5C0.2=1

1.3 MEMORY MAP

1.3.1 Memory Configuration

The Program memory space of the ST92141, 16K bytes of directly addressable on-chip memory, is fully available to the user.

The first 256 memory locations from address 0 to FFh hold the Reset Vector, the Top-Level (Pseudo Non-Maskable) interrupt, the Divide by Zero Trap Routine vector and, optionally, the interrupt vector table for use with the on-chip peripherals and the external interrupt sources. Apart from this case no other part of the Program memory has a predetermined function except segment 21h which is reserved for use by STMicroelectronics.

1.3.2 EPROM Programming

The 16K bytes of EPROM memory of the ST92E141 may be programmed by using the EPROM Programming Boards (EPB) or gang programmers available from STMicroelectronics.

EPROM Erasing

The EPROM of the windowed package of the ST92E141 may be erased by exposure to Ultra-Violet light.

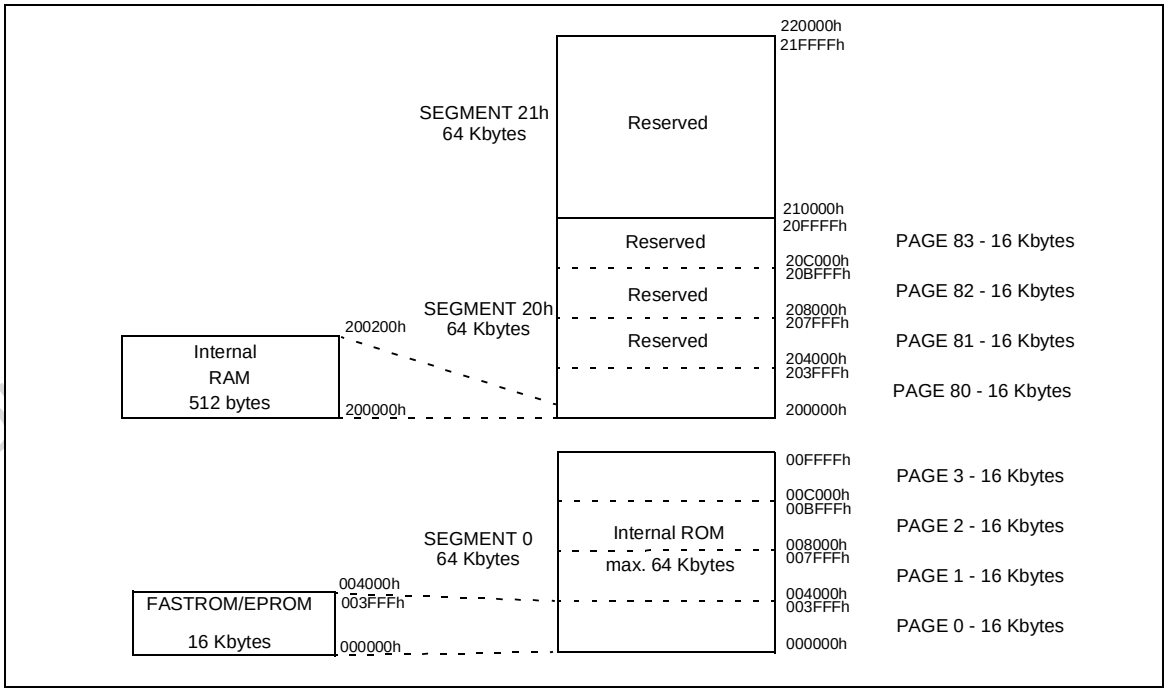
The erasure characteristic of the ST92E141 is such that erasure begins when the memory is exposed to light with a wave lengths shorter than approximately 4000Å. It should be noted that sunlight and some types of fluorescent lamps have wavelengths in the range 3000-4000Å. It is thus recommended that the window of the ST92E141 packages be covered by an opaque label to prevent unintentional erasure problems when testing the application in such an environment.

The recommended erasure procedure of the EPROM is the exposure to short wave ultraviolet light which have a wave-length 2537Å. The integrated dose (i.e. U.V. intensity x exposure time) for erasure should be a minimum of 15W-sec/cm2. The erasure time with this dosage is approximately 30 minutes using an ultraviolet lamp with 12000mW/cm2 power rating. The ST92E141 should be placed within 2.5cm (1 inch) of the lamp tubes during erasure.

Table 5. First 6 Bytes of Program Space

0	Address high of Power on Reset routine
1	Address low of Power on Reset routine
2	Address high of Divide by zero trap Subroutine
3	Address low of Divide by zero trap Subroutine
4	Address high of Top Level Interrupt routine
5	Address low of Top Level Interrupt routine

Figure 2. Memory Map



1.4 REGISTER MAP

The following pages contain a list of ST92141 registers, grouped by peripheral or function.

Be very careful to correctly program both:

– The set of registers dedicated to a particular function or peripheral.

– Registers common to other functions.

– In particular, double-check that any registers with “undefined” reset values have been correctly initialised.

WARNING: Note that in the **EIVR** and each **IVR** register, all bits are significant. Take care when defining base vector addresses that entries in the Interrupt Vector table do not overlap.

Table 6. Common Registers

Function or Peripheral	Common Registers
ADC	CICR + NICR + I/O PORT REGISTERS
WDT	CICR + NICR + EXTERNAL INTERRUPT REGISTERS + I/O PORT REGISTERS
I/O PORTS	I/O PORT REGISTERS + MODER
EXTERNAL INTERRUPT	INTERRUPT REGISTERS + I/O PORT REGISTERS
RCCU	INTERRUPT REGISTERS + MODER

Table 7. Group F Pages

Resources available on the ST92141 devices:

<table>
<tr><th>Register</th><th colspan="12">Page</th></tr>
<tr><td></td><td>0</td><td>2</td><td>3</td><td>7</td><td>11</td><td>21</td><td>28</td><td>48</td><td>51</td><td>55</td><td>57</td><td>63</td></tr>
<tr><td>R255</td><td rowspan="4">Res.</td><td>Res.</td><td rowspan="5">Res.</td><td rowspan="10">Res.</td><td rowspan="10">Res.</td><td rowspan="5">Res.</td><td rowspan="14">EFT0</td><td rowspan="5">IMC</td><td rowspan="14">IMC</td><td rowspan="5">Res.</td><td rowspan="4">WU</td><td rowspan="14">A/D0</td></tr>
<tr><td>R254</td><td rowspan="3">PORT 3</td></tr>
<tr><td>R253</td></tr>
<tr><td>R252</td><td>WCR</td></tr>
<tr><td>R251</td><td rowspan="4">WDT</td><td rowspan="14">Res.</td></tr>
<tr><td>R250</td></tr>
<tr><td>R249</td><td rowspan="2">MMU</td></tr>
<tr><td>R248</td></tr>
<tr><td>R247</td><td rowspan="6">EXT INT</td><td>Res.</td><td rowspan="14">Res.</td></tr>
<tr><td>R246</td><td rowspan="3">PORT 5</td><td rowspan="2">EM</td><td>RCCU</td></tr>
<tr><td>R245</td><td rowspan="4">Res.</td></tr>
<tr><td>R244</td><td rowspan="9">MMU</td><td rowspan="9">Res.</td></tr>
<tr><td>R243</td><td rowspan="8">Res.</td><td rowspan="8">SPI0</td><td rowspan="8">STIM0</td></tr>
<tr><td>R242</td><td>RCCU</td></tr>
<tr><td>R241</td><td rowspan="2">Res.</td><td>Res.</td></tr>
<tr><td>R240</td><td>RCCU</td></tr>
</table>

Table 8. Detailed Register Map

Page (Decimal)	Block	Reg. No.	Register Name	Description	Reset Value Hex.	Doc. Page
N/A	Core	R230	CICR	Central Interrupt Control Register	87	52
		R231	FLAGR	Flag Register	00	24
		R232	RP0	Pointer 0 Register	xx	26
		R233	RP1	Pointer 1 Register	xx	26
		R234	PPR	Page Pointer Register	xx	28
		R235	MODER	Mode Register	E0	28
		R236	USPHR	User Stack Pointer High Register	xx	30
		R237	USPLR	User Stack Pointer Low Register	xx	30
		R238	SSPHR	System Stack Pointer High Reg.	xx	30
		R239	SSPLR	System Stack Pointer Low Reg.	xx	30
	I/O Port 5:4,2:0	R224	P0DR	Port 0 Data Register	FF	79
		R225	P1DR	Port 1 Data Register	FF	
		R226	P2DR	Port 2 Data Register	FF	
		R228	P4DR	Port 4 Data Register	FF	
		R229	P5DR	Port 5 Data Register	FF	
0	INT	R242	EITR	External Interrupt Trigger Register	00	52
		R243	EIPR	External Interrupt Pending Reg.	00	53
		R244	EIMR	External Interrupt Mask-bit Reg.	00	53
		R245	EIPLR	External Interrupt Priority Level Reg.	FF	53
		R246	EIVR	External Interrupt Vector Register	x6	54
		R247	NICR	Nested Interrupt Control	00	54
	WDT	R248	WDTHR	Watchdog Timer High Register	FF	90
		R249	WDTLR	Watchdog Timer Low Register	FF	90
		R250	WDTPR	Watchdog Timer Prescaler Reg.	FF	90
		R251	WDTCR	Watchdog Timer Control Register	12	90
		R252	WCR	Wait Control Register	7F	91
2	I/O Port 3	R252	P3C0	Port 3 Configuration Register 0	00	79
		R253	P3C1	Port 3 Configuration Register 1	00	
		R254	P3C2	Port 3 Configuration Register 2	00	
3	I/O Port 5	R244	P5C0	Port 5 Configuration Register 0	FF	
		R245	P5C1	Port 5 Configuration Register 1	00	
		R246	P5C2	Port 5 Configuration Register 2	00	
7	SPI	R240	SPDR	SPI Data Register	00	145
		R241	SPCR	SPI Control Register	00	145
		R242	SPSR	SPI Status Register	00	146
		R243	SPPR	SPI Prescaler Register	00	146
11	STIM	R240	STH	Counter High Byte Register	FF	95
		R241	STL	Counter Low Byte Register	FF	95
		R242	STP	Standard Timer Prescaler Register	FF	95
		R243	STC	Standard Timer Control Register	14	95

Page (Decimal)	Block	Reg. No.	Register Name	Description	Reset Value Hex.	Doc. Page
21	MMU	R240	DPR0	Data Page Register 0	xx	35
		R241	DPR1	Data Page Register 1	xx	35
		R242	DPR2	Data Page Register 2	xx	35
		R243	DPR3	Data Page Register 3	xx	35
		R244	CSR	Code Segment Register	00	36
		R248	ISR	Interrupt Segment Register	xx	36
		R249	DMASR	DMA Segment Register	xx	36
	EM	R245	EMR1	EM Register 1	80	62
		R246	EMR2	EM Register 2	0F	62
28	EFT	R240	IC1HR	Input Capture 1 High Register	xx	108
		R241	IC1LR	Input Capture 1 Low Register	xx	108
		R242	IC2HR	Input Capture 2 High Register	xx	108
		R243	IC2LR	Input Capture 2 Low Register	xx	108
		R244	CHR	Counter High Register	FF	109
		R245	CLR	Counter Low Register	FC	109
		R246	ACHR	Alternate Counter High Register	FF	109
		R247	ACLR	Alternate Counter Low Register	FC	109
		R248	OC1HR	Output Compare 1 High Register	80	110
		R249	OC1LR	Output Compare 1 Low Register	00	110
		R250	OC2HR	Output Compare 2 High Register	80	110
		R251	OC2LR	Output Compare 2 Low Register	00	110
		R252	CR1	Control Register 1	00	111
		R253	CR2	Control Register 2	00	112
		R254	SR	Status Register	00	113
		R255	CR3	Control Register 3	00	113
48	IMC	R248	PCR0	Peripheral Control Register 0	80	130
		R249	PCR1	Peripheral Control Register 1	00	130
		R250	PCR2	Peripheral Control Register 2	00	131
		R251	PSR	Polarity Selection Register	00	131
		R252	OPR	Output Peripheral Register	00	132
		R253	IMR	Interrupt Mask Register	00	132
		R254	DTG	Dead Time Generator Register	00	133
		R255	IMCIVR	IMC Interrupt Vector Register	xx	133

Page (Decimal)	Block	Reg. No.	Register Name	Description	Reset Value Hex.	Doc. Page
51	IMC	R240	TCPTH	Tacho Capture Register High	xx	125
		R241	TCPTL	Tacho Capture Register Low	xx	125
		R242	TCMP	Tacho Compare Register	xx	125
		R243	ISR	Interrupt Status Register	3F	36
		R244	TPRSH	Tacho Prescaler Register High	00	127
		R245	TPRSL	Tacho Prescaler Register Low	00	127
		R246	CPRS	PWM Counter Prescaler Register	00	127
		R247	REP	Repetition Counter Register	00	127
		R248	CPWH	Compare Phase W Preload Register High	00	128
		R249	CPWL	Compare Phase W Preload Register Low	00	128
		R250	CPVH	Compare Phase V Preload Register High	00	128
		R251	CPVL	Compare Phase V Preload Register Low	00	128
		R252	CPUH	Compare Phase U Preload Register High	00	129
		R253	CPUL	Compare Phase U Preload Register Low	00	129
		R254	CP0H	Compare 0 Preload Register High	00	129
		R255	CP0L	Compare 0 Preload Register Low	00	129
55	RCCU	R240	CLKCTL	Clock Control Register	00	69
		R242	CLK_FLAG	Clock Flag Register	48, 28	70
		R246	PLLCONF	PLL Configuration Register	xx	71
57	WUIMU	R249	WUCTRL	Wake-Up Control Register	00	59
		R250	WUMRH	Wake-Up Mask Register High	00	60
		R251	WUMRL	Wake-Up Mask Register Low	00	60
		R252	WUTRH	Wake-Up Trigger Register High	00	61
		R253	WUTRL	Wake-Up Trigger Register Low	00	61
		R254	WUPRH	Wake-Up Pending Register High	00	61
		R255	WUPRL	Wake-Up Pending Register Low	00	61
63	ADC	R240	D0R	Channel 0 Data Register	xx	151
		R241	D1R	Channel 1 Data Register	xx	151
		R242	D2R	Channel 2 Data Register	xx	151
		R243	D3R	Channel 3 Data Register	xx	151
		R244	D4R	Channel 4 Data Register	xx	151
		R245	D5R	Channel 5 Data Register	xx	151
		R246	D6R	Channel 6 Data Register	xx	151
		R247	D7R	Channel 7 Data Register	xx	151
		R248	LT6R	Channel 6 Lower Threshold Reg.	xx	152
		R249	LT7R	Channel 7 Lower Threshold Reg.	xx	152
		R250	UT6R	Channel 6 Upper Threshold Reg.	xx	152
		R251	UT7R	Channel 7 Upper Threshold Reg.	xx	152
		R252	CRR	Compare Result Register	0F	153
		R253	CLR	Control Logic Register	00	154
		R254	ICR	Interrupt Control Register	0F	155
		R255	IVR	Interrupt Vector Register	x2	155

Note: xx denotes a byte with an undefined value, however some of the bits may have defined values. Refer to register description for details.

2 DEVICE ARCHITECTURE

2.1 CORE ARCHITECTURE

The ST9 Core or Central Processing Unit (CPU) features a highly optimised instruction set, capable of handling bit, byte (8-bit) and word (16-bit) data, as well as BCD and Boolean formats; 14 addressing modes are available.

Four independent buses are controlled by the Core: a 16-bit Memory bus, an 8-bit Register data bus, an 8-bit Register address bus and a 6-bit Interrupt/DMA bus which connects the interrupt and DMA controllers in the on-chip peripherals with the Core.

This multiple bus architecture affords a high degree of pipelining and parallel operation, thus making the ST9 family devices highly efficient, both for numerical calculation, data handling and with regard to communication with on-chip peripheral resources.

2.2 MEMORY SPACES

There are two separate memory spaces:

– The Register File, which comprises 240 8-bit registers, arranged as 15 groups (Group 0 to E), each containing sixteen 8-bit registers plus up to 64 pages of 16 registers mapped in Group F, which hold data and control bits for the on-chip peripherals and I/Os.

– A single linear memory space accommodating both program and data. All of the physically separate memory areas, including the internal ROM, internal RAM and external memory are mapped in this common address space. The total addressable memory space of 4 Mbytes (limited by the size of on-chip memory and the number of external address pins) is arranged as 64 segments of 64 Kbytes. Each segment is further subdivided into four pages of 16 Kbytes, as illustrated in Figure 3. A Memory Management Unit uses a set of pointer registers to address a 22-bit memory field using 16-bit address-based instructions.

2.2.1 Register File

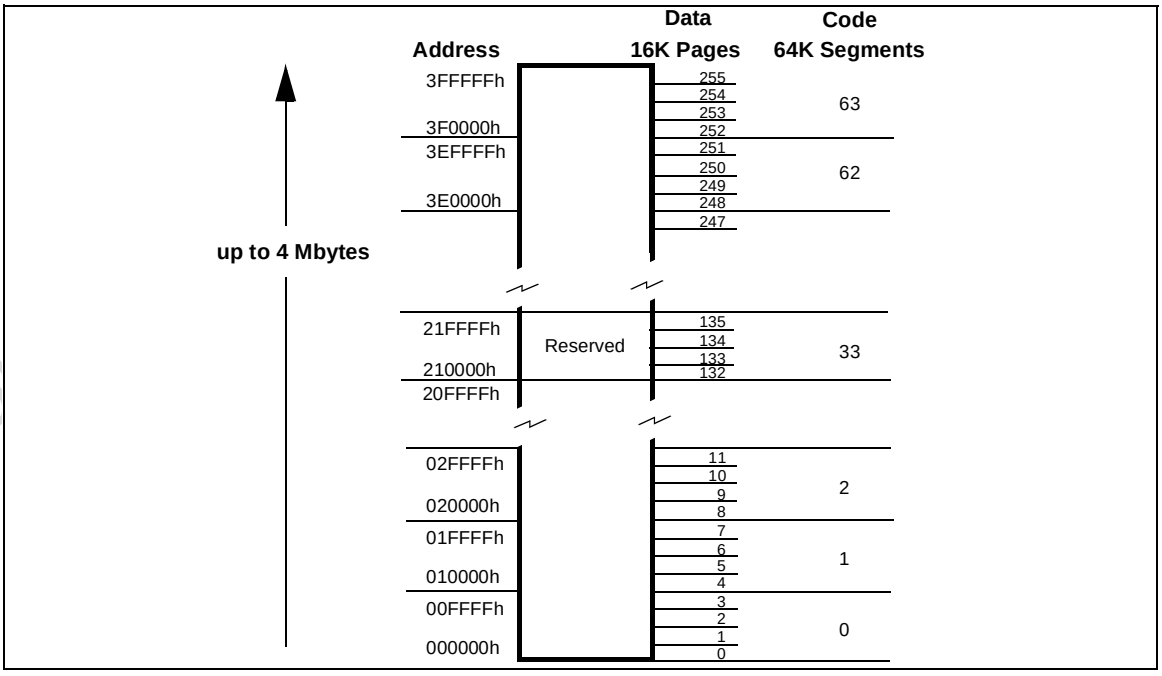
The Register File consists of (see Figure 4):

– 224 general purpose registers (Group 0 to D, registers R0 to R223)

– 6 system registers in the System Group (Group E, registers R224 to R239)

– Up to 64 pages, depending on device configuration, each containing up to 16 registers, mapped to Group F (R240 to R255), see Figure 5.

Figure 3. Single Program and Data Memory Address Space



MEMORY SPACES (Cont'd)

Figure 4. Register Groups

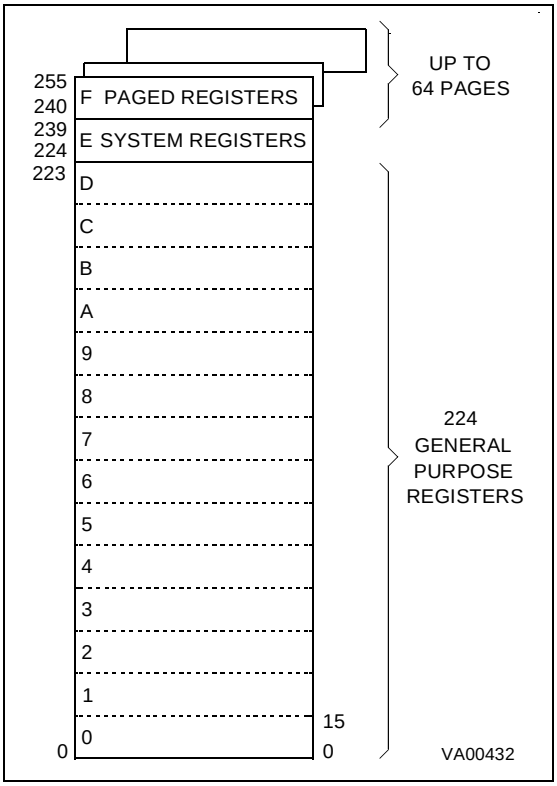


Figure 5. Page Pointer for Group F mapping

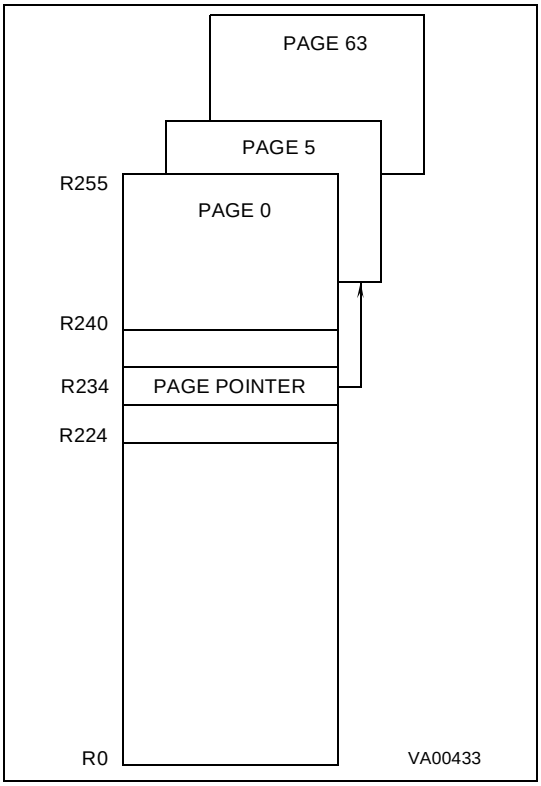
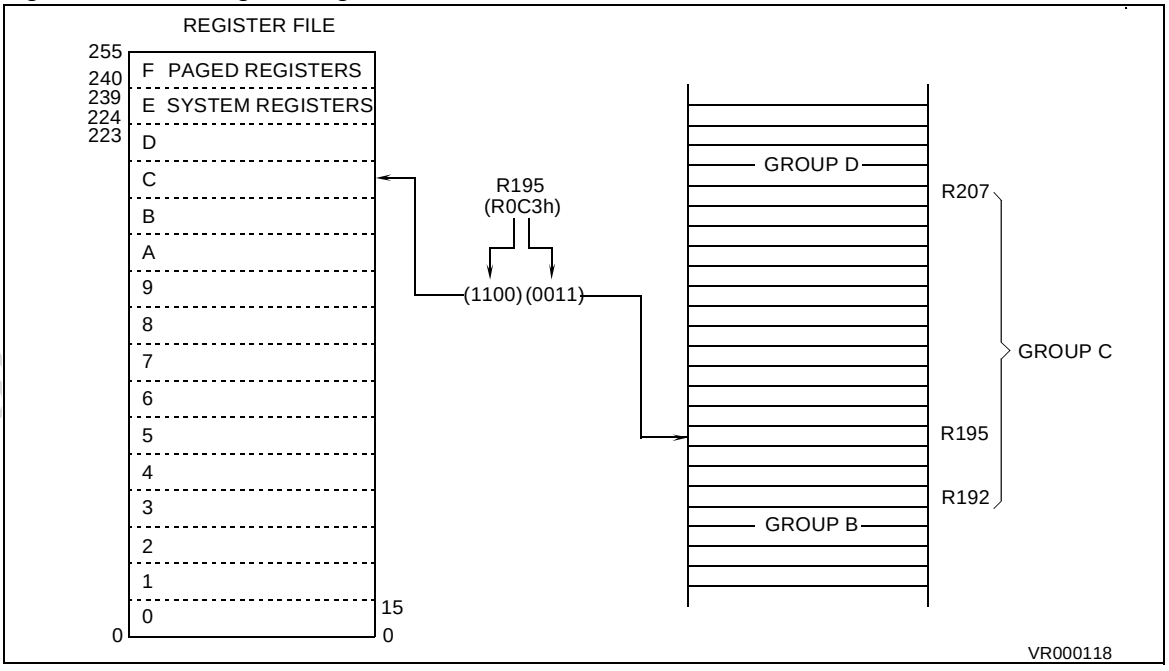


Figure 6. Addressing the Register File



MEMORY SPACES (Cont'd)

2.2.2 Register Addressing

Register File registers, including Group F paged registers (but excluding Group D), may be addressed explicitly by means of a decimal, hexadecimal or binary address; thus R231, RE7h and R11100111b represent the same register (see Figure 6). Group D registers can only be addressed in Working Register mode.

Note that an upper case "R" is used to denote this direct addressing mode.

Working Registers

Certain types of instruction require that registers be specified in the form "rx", where x is in the range 0 to 15: these are known as Working Registers.

Note that a lower case "r" is used to denote this indirect addressing mode.

Two addressing schemes are available: a single group of 16 working registers, or two separately mapped groups, each consisting of 8 working registers. These groups may be mapped starting at any 8 or 16 byte boundary in the register file by means of dedicated pointer registers. This technique is described in more detail in Section 2.3.3 Register Pointing Techniques, and illustrated in Figure 7 and in Figure 8.

System Registers

The 16 registers in Group E (R224 to R239) are System registers and may be addressed using any of the register addressing modes. These registers are described in greater detail in Section 2.3 SYSTEM REGISTERS.

Paged Registers

Up to 64 pages, each containing 16 registers, may be mapped to Group F. These are addressed using any register addressing mode, in conjunction with the Page Pointer register, R234, which is one of the System registers. This register selects the page to be mapped to Group F and, once set, does not need to be changed if two or more registers on the same page are to be addressed in succession.

Therefore if the Page Pointer, R234, is set to 5, the instructions:

```
spp #5
ld R242, r4
```

will load the contents of working register r4 into the third register of page 5 (R242).

These paged registers hold data and control information relating to the on-chip peripherals, each peripheral always being associated with the same pages and registers to ensure code compatibility between ST9 devices. The number of these registers therefore depends on the peripherals which are present in the specific ST9 family device. In other words, pages only exist if the relevant peripheral is present.

Table 9. Register File Organization

Hex. Address	Decimal Address	Function	Register File Group
F0-FF	240-255	Paged Registers	Group F
E0-EF	224-239	System Registers	Group E
D0-DF	208-223	General Purpose Registers	Group D
C0-CF	192-207		Group C
B0-BF	176-191		Group B
A0-AF	160-175		Group A
90-9F	144-159		Group 9
80-8F	128-143		Group 8
70-7F	112-127		Group 7
60-6F	96-111		Group 6
50-5F	80-95		Group 5
40-4F	64-79		Group 4
30-3F	48-63		Group 3
20-2F	32-47		Group 2
10-1F	16-31		Group 1
00-0F	00-15		Group 0

2.3 SYSTEM REGISTERS

The System registers are listed in Table 10 System Registers (Group E). They are used to perform all the important system settings. Their purpose is described in the following pages. Refer to the chapter dealing with I/O for a description of the PORT[5:0] Data registers.

Table 10. System Registers (Group E)

R239 (EFh)	SSPLR
R238 (EEh)	SSPHR
R237 (EDh)	USPLR
R236 (ECh)	USPHR
R235 (EBh)	MODE REGISTER
R234 (EAh)	PAGE POINTER REGISTER
R233 (E9h)	REGISTER POINTER 1
R232 (E8h)	REGISTER POINTER 0
R231 (E7h)	FLAG REGISTER
R230 (E6h)	CENTRAL INT. CNTL REG
R229 (E5h)	PORT5 DATA REG.
R228 (E4h)	PORT4 DATA REG.
R227 (E3h)	PORT3 DATA REG.
R226 (E2h)	PORT2 DATA REG.
R225 (E1h)	PORT1 DATA REG.
R224 (E0h)	PORT0 DATA REG.

2.3.1 Central Interrupt Control Register

Please refer to the "INTERRUPT" chapter for a detailed description of the ST9 interrupt philosophy.

CENTRAL INTERRUPT CONTROL REGISTER (CICR)
R230 - Read/Write
Register Group: E (System)
Reset Value: 1000 0111 (87h)

7							0
GCEN	TLIP	TLI	IEN	IAM	CPL2	CPL1	CPL0

Bit 7 = **GCEN**: *Global Counter Enable*.
This bit is the Global Counter Enable of the Multifunction Timers. The GCEN bit is ANDed with the CE bit in the TCR Register (only in devices featuring the MFT Multifunction Timer) in order to enable the Timers when both bits are set. This bit is set after the Reset cycle.

Note: If an MFT is not included in the ST9 device, then this bit has no effect.

Bit 6 = **TLIP**: *Top Level Interrupt Pending*.
This bit is set by hardware when a Top Level Interrupt Request is recognized. This bit can also be set by software to simulate a Top Level Interrupt Request.
0: No Top Level Interrupt pending
1: Top Level Interrupt pending

Bit 5 = **TLI**: *Top Level Interrupt bit*.
0: Top Level Interrupt is acknowledged depending on the TLNM bit in the NICR Register.
1: Top Level Interrupt is acknowledged depending on the IEN and TLNM bits in the NICR Register (described in the Interrupt chapter).

Bit 4 = **IEN**: *Interrupt Enable .*
This bit is cleared by interrupt acknowledgement, and set by interrupt return (`iret`). IEN is modified implicitly by `iret`, `ei` and `di` instructions or by an interrupt acknowledge cycle. It can also be explicitly written by the user, but only when no interrupt is pending. Therefore, the user should execute a `di` instruction (or guarantee by other means that no interrupt request can arrive) before any write operation to the CICR register.
0: Disable all interrupts except Top Level Interrupt.
1: Enable Interrupts

Bit 3 = **IAM**: *Interrupt Arbitration Mode*.
This bit is set and cleared by software to select the arbitration mode.
0: Concurrent Mode
1: Nested Mode.

Bits 2:0 = **CPL[2:0]**: *Current Priority Level*.
These three bits record the priority level of the routine currently running (i.e. the Current Priority Level, CPL). The highest priority level is represented by 000, and the lowest by 111. The CPL bits can be set by hardware or software and provide the reference according to which subsequent interrupts are either left pending or are allowed to interrupt the current interrupt service routine. When the current interrupt is replaced by one of a higher priority, the current priority value is automatically stored until required in the NICR register.

SYSTEM REGISTERS (Cont'd)

2.3.2 Flag Register

The Flag Register contains 8 flags which indicate the CPU status. During an interrupt, the flag register is automatically stored in the system stack area and recalled at the end of the interrupt service routine, thus returning the CPU to its original status.

This occurs for all interrupts and, when operating in nested mode, up to seven versions of the flag register may be stored.

FLAG REGISTER (FLAGR)
R231- Read/Write
Register Group: E (System)
Reset value: 0000 0000 (00h**)**

7							0
C	Z	S	V	DA	H	-	DP

Bit 7 = **C**: *Carry Flag*.
The carry flag is affected by:
Addition (`add, addw, adc, adcw`),
Subtraction (`sub, subw, sbc, sbcw`),
Compare (`cp, cpw`),
Shift Right Arithmetic (`sra, sraw`),
Shift Left Arithmetic (`sla, slaw`),
Swap Nibbles (`swap`),
Rotate (`rrc, rrcw, rlc, rlcw, ror, rol`),
Decimal Adjust (`da`),
Multiply and Divide (`mul, div, divws`).

When set, it generally indicates a carry out of the most significant bit position of the register being used as an accumulator (bit 7 for byte operations and bit 15 for word operations).

The carry flag can be set by the Set Carry Flag (`scf`) instruction, cleared by the Reset Carry Flag (`rcf`) instruction, and complemented by the Complement Carry Flag (`ccf`) instruction.

Bit 6 = **Z:** *Zero Flag*. The Zero flag is affected by:
Addition (`add, addw, adc, adcw`),
Subtraction (`sub, subw, sbc, sbcw`),
Compare (`cp, cpw`),
Shift Right Arithmetic (`sra, sraw`),
Shift Left Arithmetic (`sla, slaw`),
Swap Nibbles (`swap`),
Rotate (rrc, `rrcw, rlc, rlcw, ror, rol`),
Decimal Adjust (`da`),
Multiply and Divide (`mul, div, divws`),
Logical (`and, andw, or, orw, xor, xorw, cpl`),
Increment and Decrement (`inc, incw, dec, decw`),
Test (`tm, tmw, tcm, tcmw, btset`).

In most cases, the Zero flag is set when the contents of the register being used as an accumulator become zero, following one of the above operations.

Bit 5 = **S**: *Sign Flag*.
The Sign flag is affected by the same instructions as the Zero flag.

The Sign flag is set when bit 7 (for a byte operation) or bit 15 (for a word operation) of the register used as an accumulator is one.

Bit 4 = **V**: *Overflow Flag*.
The Overflow flag is affected by the same instructions as the Zero and Sign flags.

When set, the Overflow flag indicates that a two's-complement number, in a result register, is in error, since it has exceeded the largest (or is less than the smallest), number that can be represented in two's-complement notation.

Bit 3 = **DA**: *Decimal Adjust Flag*.
The DA flag is used for BCD arithmetic. Since the algorithm for correcting BCD operations is different for addition and subtraction, this flag is used to specify which type of instruction was executed last, so that the subsequent Decimal Adjust (`da`) operation can perform its function correctly. The DA flag cannot normally be used as a test condition by the programmer.

Bit 2 = **H**: *Half Carry Flag.*
The H flag indicates a carry out of (or a borrow into) bit 3, as the result of adding or subtracting two 8-bit bytes, each representing two BCD digits. The H flag is used by the Decimal Adjust (`da`) instruction to convert the binary result of a previous addition or subtraction into the correct BCD result. Like the DA flag, this flag is not normally accessed by the user.

Bit 1 = Reserved bit (must be 0).

Bit 0 = **DP**: *Data/Program Memory Flag*.
This bit indicates the memory area addressed. Its value is affected by the Set Data Memory (`sdm`) and Set Program Memory (`spm`) instructions. Refer to the Memory Management Unit for further details.

SYSTEM REGISTERS (Cont'd)

If the bit is set, data is accessed using the Data Pointers (DPRs registers), otherwise it is pointed to by the Code Pointer (CSR register); therefore, the user initialization routine must include a Sdm instruction. Note that code is always pointed to by the Code Pointer (CSR).

Note: In the current ST9 devices, the DP flag is only for compatibility with software developed for the first generation of ST9 devices. With the single memory addressing space, its use is now redundant. It must be kept to 1 with a Sdm instruction at the beginning of the program to ensure a normal use of the different memory pointers.

2.3.3 Register Pointing Techniques

Two registers within the System register group, are used as pointers to the working registers. Register Pointer 0 (R232) may be used on its own as a single pointer to a 16-register working space, or in conjunction with Register Pointer 1 (R233), to point to two separate 8-register spaces.

For the purpose of register pointing, the 16 register groups of the register file are subdivided into 32 8-register blocks. The values specified with the Set Register Pointer instructions refer to the blocks to be pointed to in twin 8-register mode, or to the lower 8-register block location in single 16-register mode.

The Set Register Pointer instructions srp, srp0 and srp1 automatically inform the CPU whether the Register File is to operate in single 16-register mode or in twin 8-register mode. The srp instruction selects the single 16-register group mode and specifies the location of the lower 8-register block, while the srp0 and srp1 instructions automatically select the twin 8-register group mode and specify the locations of each 8-register block.

There is no limitation on the order or position of these register groups, other than that they must start on an 8-register boundary in twin 8-register mode, or on a 16-register boundary in single 16-register mode.

The block number should always be an even number in single 16-register mode. The 16-register group will always start at the block whose number is the nearest even number equal to or lower than the block number specified in the srp instruction. Avoid using odd block numbers, since this can be confusing if twin mode is subsequently selected.

Thus:

srp #3 will be interpreted as srp #2 and will allow using R16 ..R31 as r0 .. r15.

In single 16-register mode, the working registers are referred to as r0 to r15. In twin 8-register mode, registers r0 to r7 are in the block pointed to by RP0 (by means of the srp0 instruction), while registers r8 to r15 are in the block pointed to by RP1 (by means of the srp1 instruction).

Caution: *Group D registers can only be accessed as working registers using the Register Pointers, or by means of the Stack Pointers. They cannot be addressed explicitly in the form "Rxxx".*

SYSTEM REGISTERS (Cont'd)

POINTER 0 REGISTER (RP0)
R232 - Read/Write
Register Group: E (System)
Reset Value: xxxx xx00 (xxh)

7							0
RG4	RG3	RG2	RG1	RG0	RPS	0	0

Bits 7:3 = **RG[4:0]**: *Register Group number.*
These bits contain the number (in the range 0 to 31) of the register block specified in the `srp0` or `srp` instructions. In single 16-register mode the number indicates the lower of the two 8-register blocks to which the 16 working registers are to be mapped, while in twin 8-register mode it indicates the 8-register block to which `r0` to `r7` are to be mapped.

Bit 2 = **RPS**: *Register Pointer Selector.*
This bit is set by the instructions `srp0` and `srp1` to indicate that the twin register pointing mode is selected. The bit is reset by the `srp` instruction to indicate that the single register pointing mode is selected.
0: Single register pointing mode
1: Twin register pointing mode

Bits 1:0: Reserved. Forced by hardware to zero.

POINTER 1 REGISTER (RP1)
R233 - Read/Write
Register Group: E (System)
Reset Value: xxxx xx00 (xxh)

7							0
RG4	RG3	RG2	RG1	RG0	RPS	0	0

This register is only used in the twin register pointing mode. When using the single register pointing mode, or when using only one of the twin register groups, the RP1 register must be considered as RESERVED and may NOT be used as a general purpose register.

Bits 7:3 = **RG[4:0]:** *Register Group number.*
These bits contain the number (in the range 0 to 31) of the 8-register block specified in the `srp1` instruction, to which `r8` to `r15` are to be mapped.

Bit 2 = **RPS**: *Register Pointer Selector.*
This bit is set by the `srp0` and `srp1` instructions to indicate that the twin register pointing mode is selected. The bit is reset by the `srp` instruction to indicate that the single register pointing mode is selected.
0: Single register pointing mode
1: Twin register pointing mode

Bits 1:0: Reserved. Forced by hardware to zero.

SYSTEM REGISTERS (Cont'd)

Figure 7. Pointing to a single group of 16 registers

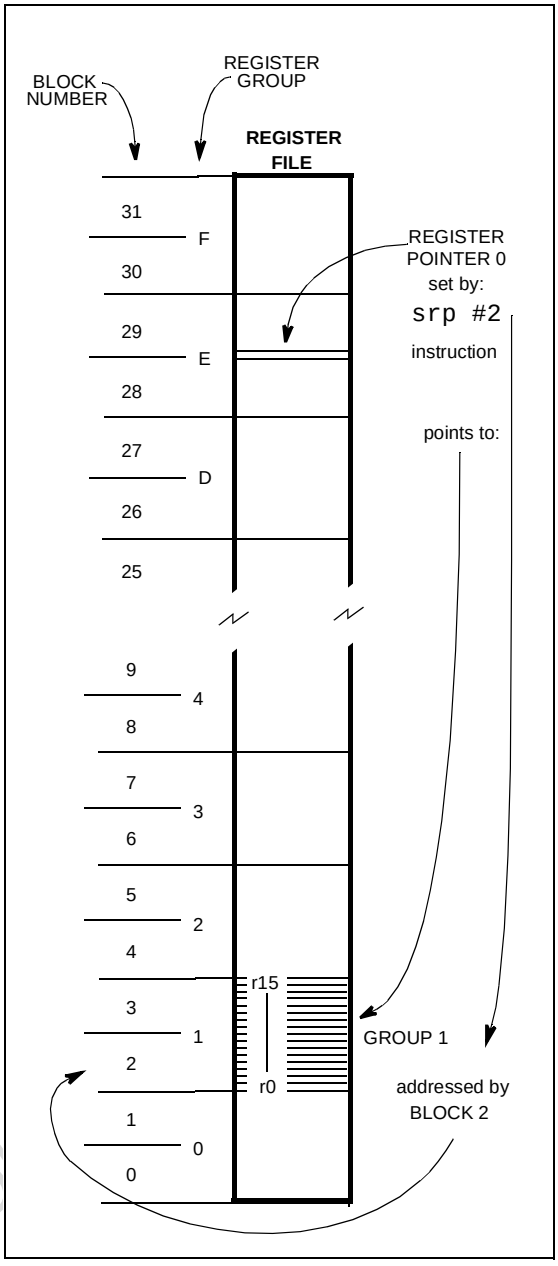
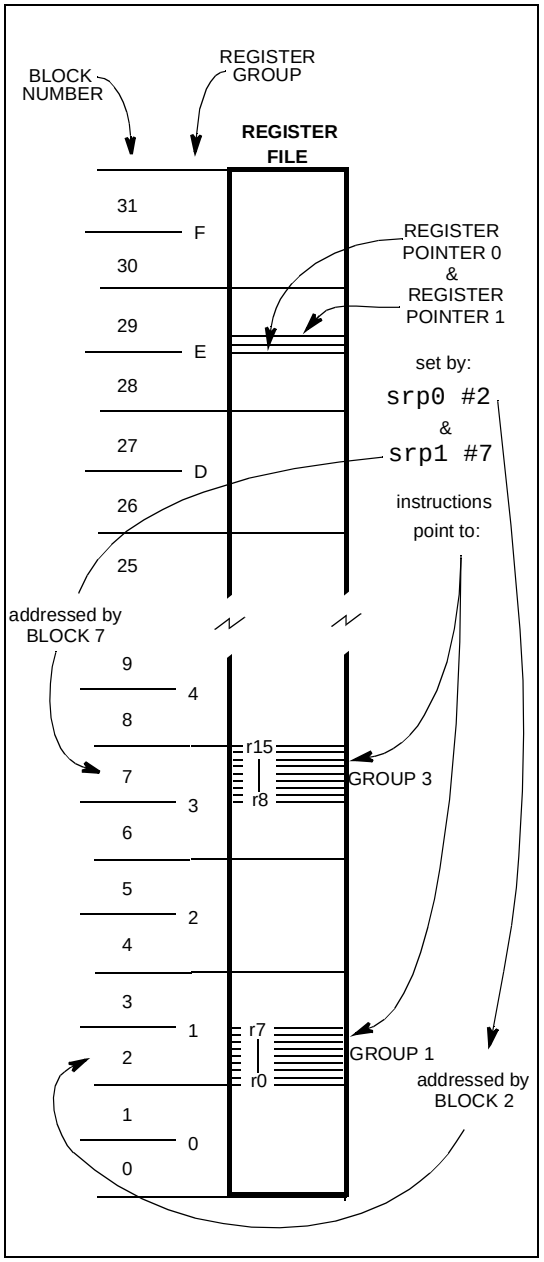


Figure 8. Pointing to two groups of 8 registers



SYSTEM REGISTERS (Cont'd)

2.3.4 Paged Registers

Up to 64 pages, each containing 16 registers, may be mapped to Group F. These paged registers hold data and control information relating to the on-chip peripherals, each peripheral always being associated with the same pages and registers to ensure code compatibility between ST9 devices. The number of these registers depends on the peripherals present in the specific ST9 device. In other words, pages only exist if the relevant peripheral is present.

The paged registers are addressed using the normal register addressing modes, in conjunction with the Page Pointer register, R234, which is one of the System registers. This register selects the page to be mapped to Group F and, once set, does not need to be changed if two or more registers on the same page are to be addressed in succession.

Thus the instructions:

```
spp #5
ld R242, r4
```

will load the contents of working register r4 into the third register of page 5 (R242).

Warning: During an interrupt, the PPR register is not saved automatically in the stack. If needed, it should be saved/restored by the user within the interrupt routine.

PAGE POINTER REGISTER (PPR)
R234 - Read/Write
Register Group: E (System)
Reset value: xxxx xx00 (xxh**)**

7							0
PP5	PP4	PP3	PP2	PP1	PP0	0	0

Bits 7:2 = **PP[5:0]**: *Page Pointer*.
These bits contain the number (in the range 0 to 63) of the page specified in the spp instruction. Once the page pointer has been set, there is no need to refresh it unless a different page is required.

Bits 1:0: Reserved. Forced by hardware to 0.

2.3.5 Mode Register

The Mode Register allows control of the following operating parameters:

– Selection of internal or external System and User Stack areas,
– Management of the clock frequency,
– Enabling of Bus request and Wait signals when interfacing to external memory.

MODE REGISTER (MODER)
R235 - Read/Write
Register Group: E (System)
Reset value: 1110 0000 (E0h)

7							0
SSP	USP	DIV2	PRS2	PRS1	PRS0	BRQEN	HIMP

Bit 7 = **SSP**: *System Stack Pointer*.
This bit selects an internal or external System Stack area.
0: External system stack area, in memory space.
1: Internal system stack area, in the Register File (reset state).

Bit 6 = **USP**: *User Stack Pointer*.
This bit selects an internal or external User Stack area.
0: External user stack area, in memory space.
1: Internal user stack area, in the Register File (reset state).

Bit 5 = **DIV2**: *Crystal Oscillator Clock Divided by 2*.
This bit controls the divide-by-2 circuit operating on the crystal oscillator clock (CLOCK1).
0: Clock divided by 1
1: Clock divided by 2

Bits 4:2 = **PRS[2:0]**: *CPUCLK Prescaler*.
These bits load the prescaler division factor for the internal clock (INTCLK). The prescaler factor selects the internal clock frequency, which can be divided by a factor from 1 to 8. Refer to the Reset and Clock Control chapter for further information.

Bit 1 = **BRQEN**: *Bus Request Enable*.
0: External Memory Bus Request disabled
1: External Memory Bus Request enabled on $\overline{\text{BREQ}}$ pin (where available).

Note: Disregard this bit if $\overline{\text{BREQ}}$ pin is not available.

Bit 0 = **HIMP**: *High Impedance Enable*.
When any of Ports 0, 1, 2 or 6 depending on device configuration, are programmed as Address and Data lines to interface external Memory, these lines and the Memory interface control lines (AS, DS, R/W) can be forced into the High Impedance

SYSTEM REGISTERS (Cont'd)

state by setting the HIMP bit. When this bit is reset, it has no effect.

Setting the HIMP bit is recommended for noise reduction when only internal Memory is used.

If Port 1 and/or 2 are declared as an address AND as an I/O port (for example: P10... P14 = Address, and P15... P17 = I/O), the HIMP bit has no effect on the I/O lines.

2.3.6 Stack Pointers

Two separate, double-register stack pointers are available: the System Stack Pointer and the User Stack Pointer, both of which can address registers or memory.

The stack pointers point to the "bottom" of the stacks which are filled using the push commands and emptied using the pop commands. The stack pointer is automatically pre-decremented when data is "pushed" in and post-incremented when data is "popped" out.

The push and pop commands used to manage the System Stack may be addressed to the User Stack by adding the suffix "u". To use a stack instruction for a word, the suffix "w" is added. These suffixes may be combined.

When bytes (or words) are "popped" out from a stack, the contents of the stack locations are unchanged until fresh data is loaded. Thus, when data is "popped" from a stack area, the stack contents remain unchanged.

Note: Instructions such as: `pushuw RR236` or `pushw RR238`, as well as the corresponding `pop` instructions (where R236 & R237, and R238 & R239 are themselves the user and system stack pointers respectively), must not be used, since the pointer values are themselves automatically changed by the `push` or `pop` instruction, thus corrupting their value.

System Stack

The System Stack is used for the temporary storage of system and/or control data, such as the Flag register and the Program counter.

The following automatically push data onto the System Stack:

– **Interrupts**

When entering an interrupt, the PC and the Flag Register are pushed onto the System Stack. If the ENCSR bit in the EMR2 register is set, then the Code Segment Register is also pushed onto the System Stack.

– **Subroutine Calls**

When a `call` instruction is executed, only the PC is pushed onto stack, whereas when a `calls` instruction (call segment) is executed, both the PC and the Code Segment Register are pushed onto the System Stack.

– **Link Instruction**

The `link` or `linku` instructions create a C language stack frame of user-defined length in the System or User Stack.

All of the above conditions are associated with their counterparts, such as return instructions, which pop the stored data items off the stack.

User Stack

The User Stack provides a totally user-controlled stacking area.

The User Stack Pointer consists of two registers, R236 and R237, which are both used for addressing a stack in memory. When stacking in the Register File, the User Stack Pointer High Register, R236, becomes redundant but must be considered as reserved.

Stack Pointers

Both System and User stacks are pointed to by double-byte stack pointers. Stacks may be set up in RAM or in the Register File. Only the lower byte will be required if the stack is in the Register File. The upper byte must then be considered as reserved and must not be used as a general purpose register.

The stack pointer registers are located in the System Group of the Register File, this is illustrated in Table 10 System Registers (Group E).

Stack Location

Care is necessary when managing stacks as there is no limit to stack sizes apart from the bottom of any address space in which the stack is placed. Consequently programmers are advised to use a stack pointer value as high as possible, particularly when using the Register File as a stacking area.

Group D is a good location for a stack in the Register File, since it is the highest available area. The stacks may be located anywhere in the first 14 groups of the Register File (internal stacks) or in RAM (external stacks).

Note. Stacks must not be located in the Paged Register Group or in the System Register Group.

SYSTEM REGISTERS (Cont'd)

USER STACK POINTER HIGH REGISTER (USPHR)
R236 - Read/Write
Register Group: E (System)
Reset value: undefined

7							0
USP15	USP14	USP13	USP12	USP11	USP10	USP9	USP8

USER STACK POINTER LOW REGISTER (USPLR)
R237 - Read/Write
Register Group: E (System)
Reset value: undefined

7							0
USP7	USP6	USP5	USP4	USP3	USP2	USP1	USP0

SYSTEM STACK POINTER HIGH REGISTER (SSPHR)
R238 - Read/Write
Register Group: E (System)
Reset value: undefined

7							0
SSP15	SSP14	SSP13	SSP12	SSP11	SSP10	SSP9	SSP8

SYSTEM STACK POINTER LOW REGISTER (SSPLR)
R239 - Read/Write
Register Group: E (System)
Reset value: undefined

7							0
SSP7	SSP6	SSP5	SSP4	SSP3	SSP2	SSP1	SSP0

Figure 9. Internal Stack Mode

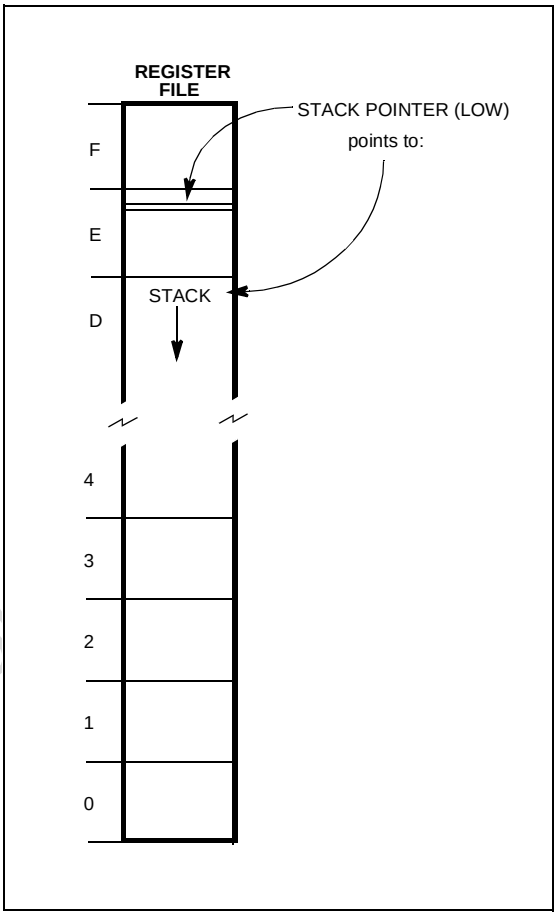
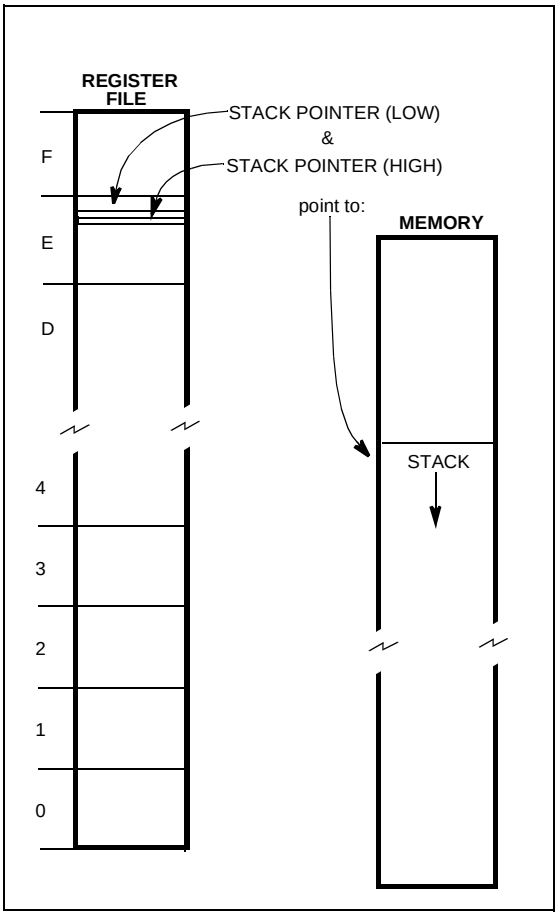


Figure 10. External Stack Mode



2.4 MEMORY ORGANIZATION

Code and data are accessed within the same linear address space. All of the physically separate memory areas, including the internal ROM, internal RAM and external memory are mapped in a common address space.

The ST9 provides a total addressable memory space of 4 Mbytes. This address space is arranged as 64 segments of 64 Kbytes; each segment is again subdivided into four 16 Kbyte pages.

The mapping of the various memory areas (internal RAM or ROM, external memory) differs from device to device. Each 64-Kbyte physical memory segment is mapped either internally or externally; if the memory is internal and smaller than 64 Kbytes, the remaining locations in the 64-Kbyte segment are not used (reserved).

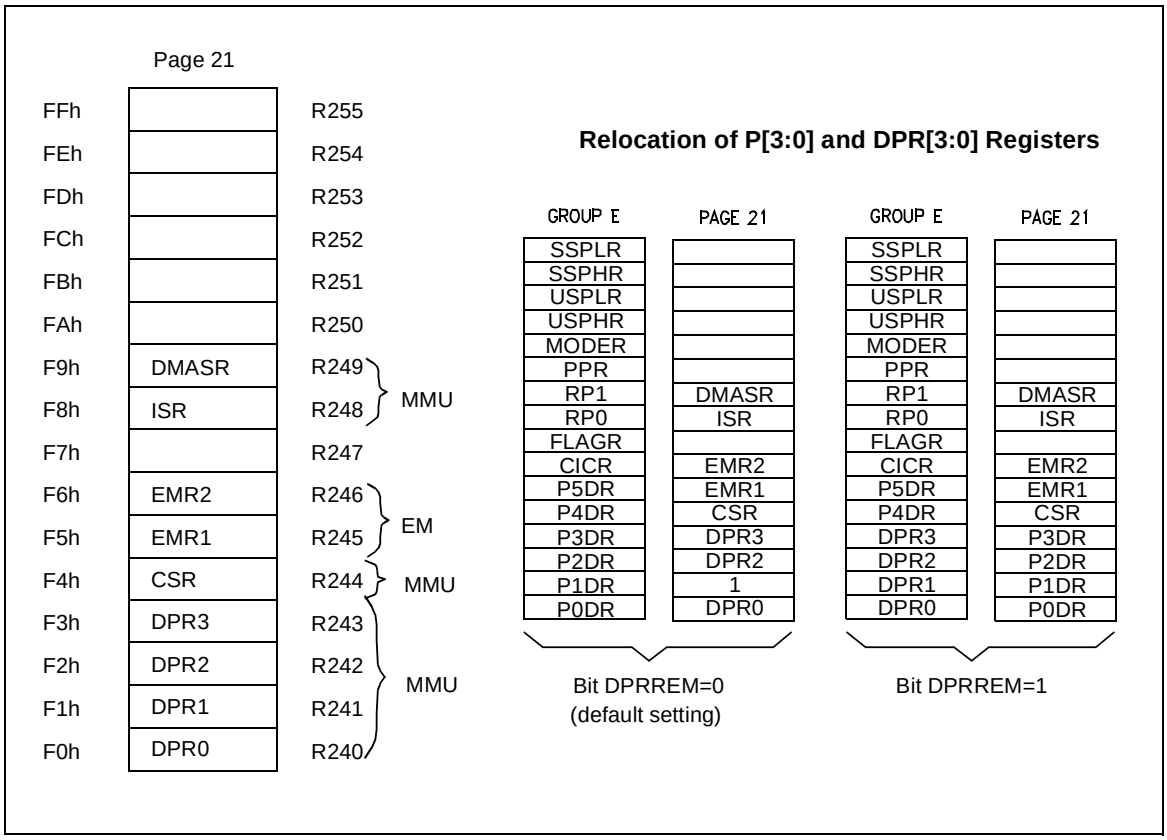
Refer to the Register and Memory Map Chapter for more details on the memory map.

2.5 MEMORY MANAGEMENT UNIT

The CPU Core includes a Memory Management Unit (MMU) which must be programmed to perform memory accesses (even if external memory is not used).

The MMU is controlled by 7 registers and 2 bits (ENCSR and DPRREM) present in EMR2, which may be written and read by the user program. These registers are mapped within group F, Page 21 of the Register File. The 7 registers may be sub-divided into 2 main groups: a first group of four 8-bit registers (DPR[3:0]), and a second group of three 6-bit registers (CSR, ISR, and DMASR). The first group is used to extend the address during Data Memory access (DPR[3:0]). The second is used to manage Program and Data Memory accesses during Code execution (CSR), Interrupts Service Routines (ISR or CSR), and DMA transfers (DMASR or ISR).

Figure 11. Page 21 Registers



2.6 ADDRESS SPACE EXTENSION

To manage 4 Mbytes of addressing space, it is necessary to have 22 address bits. The MMU adds 6 bits to the usual 16-bit address, thus translating a 16-bit virtual address into a 22-bit physical address. There are 2 different ways to do this depending on the memory involved and on the operation being performed.

2.6.1 Addressing 16-Kbyte Pages

This extension mode is implicitly used to address Data memory space if no DMA is being performed.

The Data memory space is divided into 4 pages of 16 Kbytes. Each one of the four 8-bit registers (DPR[3:0], Data Page Registers) selects a different 16-Kbyte page. The DPR registers allow access to the entire memory space which contains 256 pages of 16 Kbytes.

Data paging is performed by extending the 14 LSB of the 16-bit address with the contents of a DPR register. The two MSBs of the 16-bit address are interpreted as the identification number of the DPR register to be used. Therefore, the DPR registers are involved in the following virtual address ranges:

DPR0: from 0000h to 3FFFh;

DPR1: from 4000h to 7FFFh;

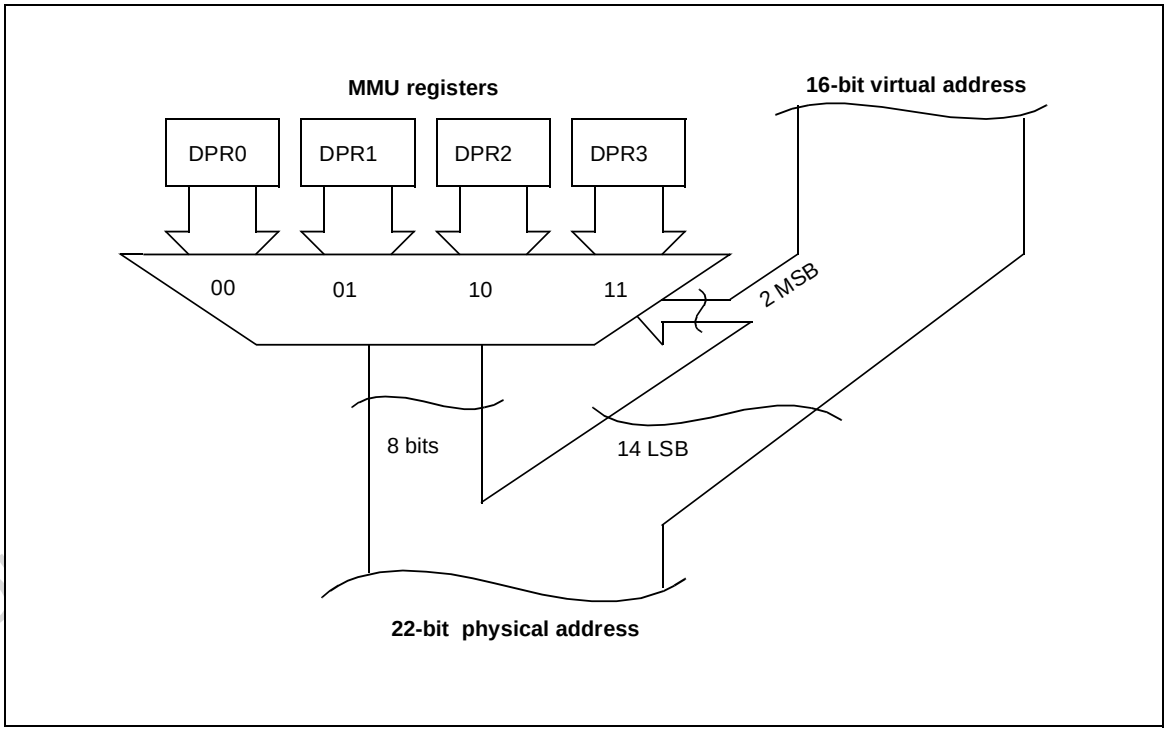
DPR2: from 8000h to BFFFh;

DPR3: from C000h to FFFFh.

The contents of the selected DPR register specify one of the 256 possible data memory pages. This 8-bit data page number, in addition to the remaining 14-bit page offset address forms the physical 22-bit address (see Figure 12).

A DPR register cannot be modified via an addressing mode that uses the same DPR register. For instance, the instruction “POPW DPR0” is legal only if the stack is kept either in the register file or in a memory location above 8000h, where DPR2 and DPR3 are used. Otherwise, since DPR0 and DPR1 are modified by the instruction, unpredictable behaviour could result.

Figure 12. Addressing via DPR[3:0]



ADDRESS SPACE EXTENSION (Cont'd)

2.6.2 Addressing 64-Kbyte Segments

This extension mode is used to address Data memory space during a DMA and Program memory space during any code execution (normal code and interrupt routines).

Three registers are used: CSR, ISR, and DMASR. The 6-bit contents of one of the registers CSR, ISR, or DMASR define one out of 64 Memory segments of 64 Kbytes within the 4 Mbytes address space. The register contents represent the 6 MSBs of the memory address, whereas the 16 LSBs of the address (intra-segment address) are given by the virtual 16-bit address (see Figure 13).

2.7 MMU REGISTERS

The MMU uses 7 registers mapped into Group F, Page 21 of the Register File and 2 bits of the EMR2 register.

Most of these registers do not have a default value after reset.

2.7.1 DPR[3:0]: Data Page Registers

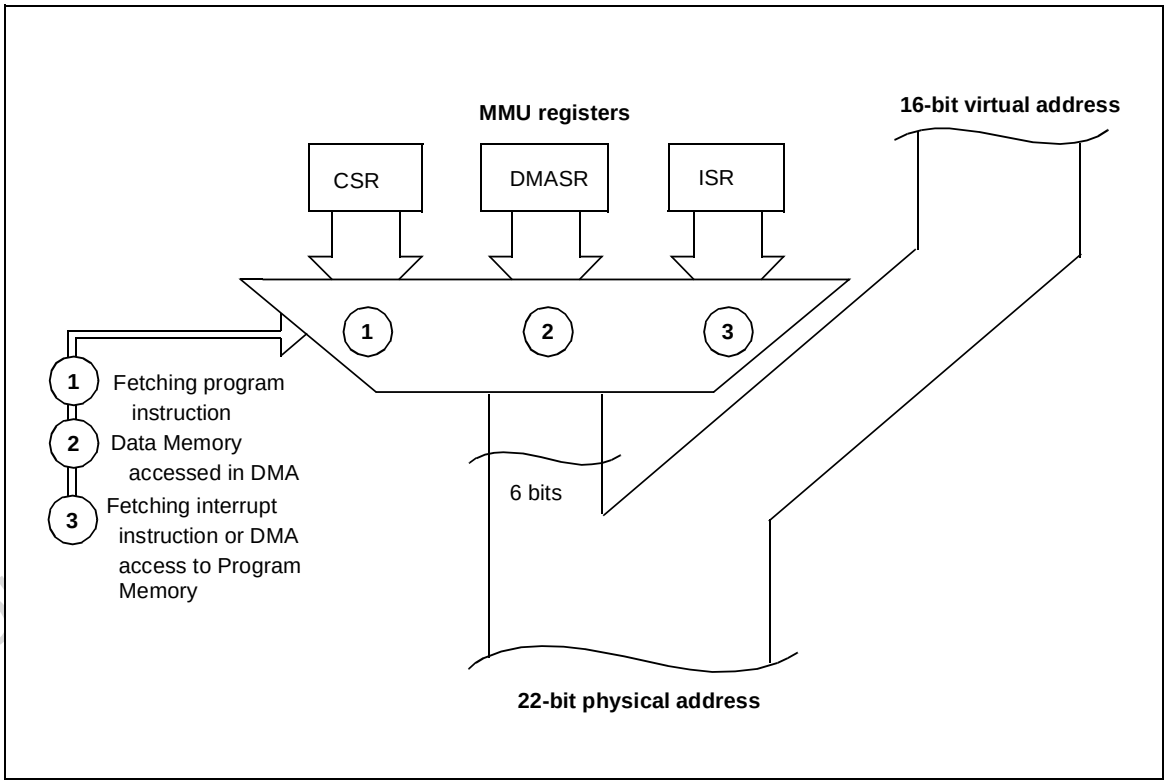
The DPR[3:0] registers allow access to the entire 4 Mbyte memory space composed of 256 pages of 16 Kbytes.

2.7.1.1 Data Page Register Relocation

If these registers are to be used frequently, they may be relocated in register group E, by programming bit 5 of the EMR2-R246 register in page 21. If this bit is set, the DPR[3:0] registers are located at R224-227 in place of the Port 0-3 Data Registers, which are re-mapped to the default DPR's locations: R240-243 page 21.

Data Page Register relocation is illustrated in Figure 11.

Figure 13. Addressing via CSR, ISR, and DMASR



MMU REGISTERS (Cont'd)

DATA PAGE REGISTER 0 (DPR0)
R240 - Read/Write
Register Page: 21
Reset value: undefined

This register is relocated to R224 if EMR2.5 is set.

7							0
DPR0_7	DPR0_6	DPR0_5	DPR0_4	DPR0_3	DPR0_2	DPR0_1	DPR0_0

Bits 7:0 = **DPR0_[7:0]**: These bits define the 16-Kbyte Data Memory page number. They are used as the most significant address bits (A21-14) to extend the address during a Data Memory access. The DPR0 register is used when addressing the virtual address range 0000h-3FFFh.

DATA PAGE REGISTER 1 (DPR1)
R241 - Read/Write
Register Page: 21
Reset value: undefined

This register is relocated to R225 if EMR2.5 is set.

7							0
DPR1_7	DPR1_6	DPR1_5	DPR1_4	DPR1_3	DPR1_2	DPR1_1	DPR1_0

Bits 7:0 = **DPR1_[7:0]**: These bits define the 16-Kbyte Data Memory page number. They are used as the most significant address bits (A21-14) to extend the address during a Data Memory access. The DPR1 register is used when addressing the virtual address range 4000h-7FFFh.

DATA PAGE REGISTER 2 (DPR2)
R242 - Read/Write
Register Page: 21
Reset value: undefined

This register is relocated to R226 if EMR2.5 is set.

7							0
DPR2_7	DPR2_6	DPR2_5	DPR2_4	DPR2_3	DPR2_2	DPR2_1	DPR2_0

Bits 7:0 = **DPR2_[7:0]**: These bits define the 16-Kbyte Data memory page. They are used as the most significant address bits (A21-14) to extend the address during a Data memory access. The DPR2 register is involved when the virtual address is in the range 8000h-BFFFh.

DATA PAGE REGISTER 3 (DPR3)
R243 - Read/Write
Register Page: 21
Reset value: undefined

This register is relocated to R227 if EMR2.5 is set.

7							0
DPR3_7	DPR3_6	DPR3_5	DPR3_4	DPR3_3	DPR3_2	DPR3_1	DPR3_0

Bits 7:0 = **DPR3_[7:0]**: These bits define the 16-Kbyte Data memory page. They are used as the most significant address bits (A21-14) to extend the address during a Data memory access. The DPR3 register is involved when the virtual address is in the range C000h-FFFFh.

MMU REGISTERS (Cont'd)

2.7.2 CSR: Code Segment Register

This register selects the 64-Kbyte code segment being used at run-time to access instructions. It can also be used to access data if the `spm` instruction has been executed (or `ldpp, ldpd, lddp`). Only the 6 LSBs of the CSR register are implemented, and bits 6 and 7 are reserved. The CSR register allows access to the entire memory space, divided into 64 segments of 64 Kbytes.

To generate the 22-bit Program memory address, the contents of the CSR register is directly used as the 6 MSBs, and the 16-bit virtual address as the 16 LSBs.

Note: The CSR register should only be read and not written for data operations (there are some exceptions which are documented in the following paragraph). It is, however, modified either directly by means of the `jps` and `calls` instructions, or indirectly via the stack, by means of the `rets` instruction.

CODE SEGMENT REGISTER (CSR)
R244 **-** Read/Write
Register Page: 21
Reset value: 0000 0000 (00h)

7							0
0	0	CSR_5	CSR_4	CSR_3	CSR_2	CSR_1	CSR_0

Bits 7:6 = Reserved, keep in reset state.

Bits 5:0 = **CSR_[5:0]**: These bits define the 64-Kbyte memory segment (among 64) which contains the code being executed. These bits are used as the most significant address bits (A21-16).

2.7.3 ISR: Interrupt Segment Register

INTERRUPT SEGMENT REGISTER (ISR)
R248 - Read/Write
Register Page: 21
Reset value: undefined

7							0
0	0	ISR_5	ISR_4	ISR_3	ISR_2	ISR_1	ISR_0

ISR and ENCSR bit (EMR2 register) are also described in the chapter relating to Interrupts, please refer to this description for further details.

Bits 7:6 = Reserved, keep in reset state.

Bits 5:0 = **ISR_[5:0]**: These bits define the 64-Kbyte memory segment (among 64) which contains the interrupt vector table and the code for interrupt service routines and DMA transfers (when the PS bit of the DAPR register is reset). These bits are used as the most significant address bits (A21-16). The ISR is used to extend the address space in two cases:

– Whenever an interrupt occurs: ISR points to the 64-Kbyte memory segment containing the interrupt vector table and the interrupt service routine code. See also the Interrupts chapter.

– During DMA transactions between the peripheral and memory when the PS bit of the DAPR register is reset : ISR points to the 64 K-byte Memory segment that will be involved in the DMA transaction.

2.7.4 DMASR: DMA Segment Register

DMA SEGMENT REGISTER (DMASR)
R249 - Read/Write
Register Page: 21
Reset value: undefined

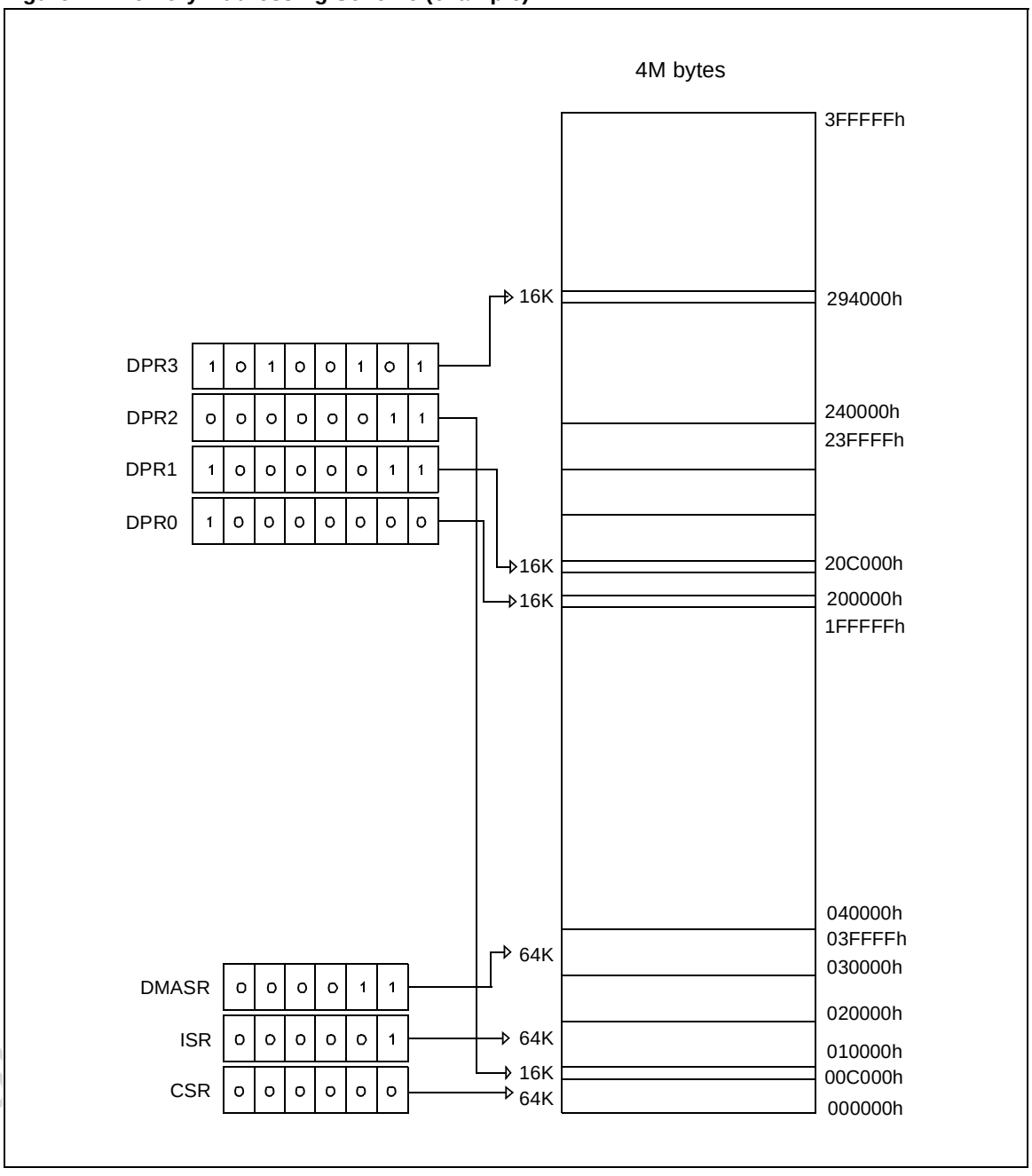
7							0
0	0	DMA SR_5	DMA SR_4	DMA SR_3	DMA SR_2	DMA SR_1	DMA SR_0

Bits 7:6 = Reserved, keep in reset state.

Bits 5:0 = **DMASR_[5:0]**: These bits define the 64-Kbyte Memory segment (among 64) used when a DMA transaction is performed between the peripheral's data register and Memory, with the PS bit of the DAPR register set. These bits are used as the most significant address bits (A21-16). If the PS bit is reset, the ISR register is used to extend the address.

MMU REGISTERS (Cont'd)

Figure 14. Memory Addressing Scheme (example)



2.8 MMU USAGE

2.8.1 Normal Program Execution

Program memory is organized as a set of 64-Kbyte segments. The program can span as many segments as needed, but a procedure cannot stretch across segment boundaries. `jps`, `calls` and `rets` instructions, which automatically modify the CSR, must be used to jump across segment boundaries. Writing to the CSR is forbidden during normal program execution because it is not synchronized with the opcode fetch. This could result in fetching the first byte of an instruction from one memory segment and the second byte from another. Writing to the CSR is allowed when it is not being used, i.e during an interrupt service routine if ENCSR is reset.

Note that a routine must always be called in the same way, i.e. either always with `call` or always with `calls`, depending on whether the routine ends with `ret` or `rets`. This means that if the routine is written without prior knowledge of the location of other routines which call it, and all the program code does not fit into a single 64-Kbyte segment, then `calls`/`rets` should be used.

In typical microcontroller applications, less than 64 Kbytes of RAM are used, so the four Data space pages are normally sufficient, and no change of DPR[3:0] is needed during Program execution. It may be useful however to map part of the ROM into the data space if it contains strings, tables, bit maps, etc.

If there is to be frequent use of paging, the user can set bit 5 (DPRREM) in register R246 (EMR2) of Page 21. This swaps the location of registers DPR[3:0] with that of the data registers of Ports 0-3. In this way, DPR registers can be accessed without the need to save/set/restore the Page Pointer Register. Port registers are therefore moved to page 21. Applications that require a lot of paging typically use more than 64 Kbytes of external memory, and as ports 0, 1 and 2 are required to address it, their data registers are unused.

2.8.2 Interrupts

The ISR register has been created so that the interrupt routines may be found by means of the same vector table even after a segment jump/call.

When an interrupt occurs, the CPU behaves in one of 2 ways, depending on the value of the ENCSR bit in the EMR2 register (R246 on Page 21).

If this bit is reset (default condition), the CPU works in original ST9 compatibility mode. For the duration of the interrupt service routine, the ISR is used instead of the CSR, and the interrupt stack frame is kept exactly as in the original ST9 (only the PC and flags are pushed). This avoids the need to save the CSR on the stack in the case of an interrupt, ensuring a fast interrupt response time. The drawback is that it is not possible for an interrupt service routine to perform segment `calls`/`jps`: these instructions would update the CSR, which, in this case, is not used (ISR is used instead). The code size of all interrupt service routines is thus limited to 64 Kbytes.

If, instead, bit 6 of the EMR2 register is set, the ISR is used only to point to the interrupt vector table and to initialize the CSR at the beginning of the interrupt service routine: the old CSR is pushed onto the stack together with the PC and the flags, and then the CSR is loaded with the ISR. In this case, an `iret` will also restore the CSR from the stack. This approach lets interrupt service routines access the whole 4-Mbyte address space. The drawback is that the interrupt response time is slightly increased, because of the need to also save the CSR on the stack. Compatibility with the original ST9 is also lost in this case, because the interrupt stack frame is different; this difference, however, would not be noticeable for a vast majority of programs.

Data memory mapping is independent of the value of bit 6 of the EMR2 register, and remains the same as for normal code execution: the stack is the same as that used by the main program, as in the ST9. If the interrupt service routine needs to access additional Data memory, it must save one (or more) of the DPRs, load it with the needed memory page and restore it before completion.

2.8.3 DMA

Depending on the PS bit in the DAPR register (see DMA chapter) DMA uses either the ISR or the DMASR for memory accesses: this guarantees that a DMA will always find its memory segment(s), no matter what segment changes the application has performed. Unlike interrupts, DMA transactions cannot save/restore paging registers, so a dedicated segment register (DMASR) has been created. Having only one register of this kind means that all DMA accesses should be programmed in one of the two following segments: the one pointed to by the ISR (when the PS bit of the DAPR register is reset), and the one referenced by the DMASR (when the PS bit is set).

3 INTERRUPTS

3.1 INTRODUCTION

The ST9 responds to peripheral and external events through its interrupt channels. Current program execution can be suspended to allow the ST9 to execute a specific response routine when such an event occurs, providing that interrupts have been enabled, and according to a priority mechanism. If an event generates a valid interrupt request, the current program status is saved and control passes to the appropriate Interrupt Service Routine (refer to Figure 15).

The ST9 CPU can receive requests from the following sources:

– On-chip peripherals

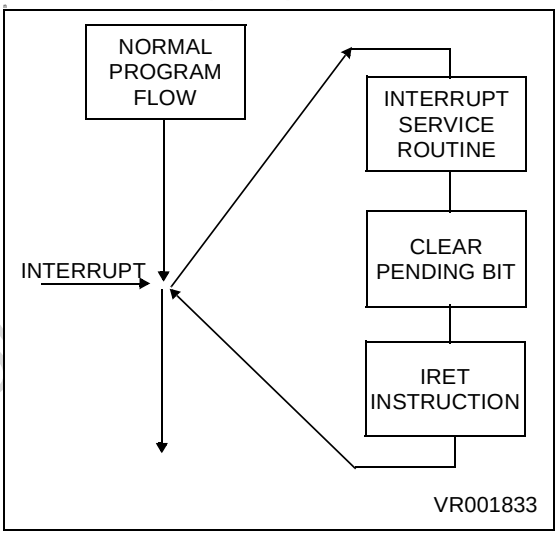
– External pins

– Top-Level Pseudo-non-maskable interrupt

According to the on-chip peripheral features, an event occurrence can generate an Interrupt request which depends on the selected mode.

Up to eight external interrupt channels, with programmable input trigger edge, are available. In addition, a dedicated interrupt channel, set to the Top-level priority, can be devoted either to the external NMI pin (where available) to provide a Non-Maskable Interrupt, or to the Timer/Watchdog. Interrupt service routines are addressed through a vector table mapped in Memory.

Figure 15. Interrupt Response



3.2 INTERRUPT VECTORING

The ST9 implements an interrupt vectoring structure which allows the on-chip peripheral to identify the location of the first instruction of the Interrupt Service Routine automatically.

When an interrupt request is acknowledged, the peripheral interrupt module provides, through its Interrupt Vector Register (IVR), a vector to point into the vector table of locations containing the start addresses of the Interrupt Service Routines (defined by the programmer).

Each peripheral has a specific IVR mapped within its Register File pages.

The Interrupt Vector table, containing the addresses of the Interrupt Service Routines, is located in the first 256 locations of Memory pointed to by the ISR register, thus allowing 8-bit vector addressing. For a description of the ISR register refer to the chapter describing the MMU.

The user Power on Reset vector is stored in the first two physical bytes in memory, 000000h and 000001h.

The Top Level Interrupt vector is located at addresses 0004h and 0005h in the segment pointed to by the Interrupt Segment Register (ISR).

With one Interrupt Vector register, it is possible to address several interrupt service routines; in fact, peripherals can share the same interrupt vector register among several interrupt channels. The most significant bits of the vector are user programmable to define the base vector address within the vector table, the least significant bits are controlled by the interrupt module, in hardware, to select the appropriate vector.

Note: The first 256 locations of the memory segment pointed to by ISR can contain program code.

3.2.1 Divide by Zero trap

The Divide by Zero trap vector is located at addresses 0002h and 0003h of each code segment; it should be noted that for each code segment a Divide by Zero service routine is required.

Warning. *Although the Divide by Zero Trap operates as an interrupt, the FLAG Register is not pushed onto the system Stack automatically. As a result it must be regarded as a subroutine, and the service routine must end with the RET instruction (not IRET).*

3.2.2 Segment Paging During Interrupt Routines

The ENCSR bit in the EMR2 register can be used to select between original ST9 backward compatibility mode and ST9+ interrupt management mode.

ST9 backward compatibility mode (ENCSR = 0)

If ENCSR is reset, the CPU works in original ST9 compatibility mode. For the duration of the interrupt service routine, ISR is used instead of CSR, and the interrupt stack frame is identical to that of the original ST9: only the PC and Flags are pushed.

This avoids saving the CSR on the stack in the event of an interrupt, thus ensuring a faster interrupt response time.

It is not possible for an interrupt service routine to perform inter-segment calls or jumps: these instructions would update the CSR, which, in this case, is not used (ISR is used instead). The code segment size for all interrupt service routines is thus limited to 64K bytes.

ST9+ mode (ENCSR = 1)

If ENCSR is set, ISR is only used to point to the interrupt vector table and to initialize the CSR at the beginning of the interrupt service routine: the old CSR is pushed onto the stack together with the PC and flags, and CSR is then loaded with the contents of ISR.

In this case, `iret` will also restore CSR from the stack. This approach allows interrupt service routines to access the entire 4 Mbytes of address space. The drawback is that the interrupt response time is slightly increased, because of the need to also save CSR on the stack.

Full compatibility with the original ST9 is lost in this case, because the interrupt stack frame is different.

ENCSR Bit	0	1
Mode	ST9 Compatible	ST9+
Pushed/Popped Registers	PC, FLAGR	PC, FLAGR, CSR
Max. Code Size for interrupt service routine	64KB Within 1 segment	No limit Across segments

3.3 INTERRUPT PRIORITY LEVELS

The ST9 supports a fully programmable interrupt priority structure. Nine priority levels are available to define the channel priority relationships:

– The on-chip peripheral channels and the eight external interrupt sources can be programmed within eight priority levels. Each channel has a 3-bit field, PRL (Priority Level), that defines its priority level in the range from 0 (highest priority) to 7 (lowest priority).

– The 9th level (Top Level Priority) is reserved for the Timer/Watchdog or the External Pseudo Non-Maskable Interrupt. An Interrupt service routine at this level cannot be interrupted in any arbitration mode. Its mask can be both maskable (TLI) or non-maskable (TLNM).

3.4 PRIORITY LEVEL ARBITRATION

The 3 bits of CPL (Current Priority Level) in the Central Interrupt Control Register contain the priority of the currently running program (CPU priority). CPL is set to 7 (lowest priority) upon reset and can be modified during program execution either by software or automatically by hardware according to the selected Arbitration Mode.

During every instruction, an arbitration phase takes place, during which, for every channel capable of generating an Interrupt, each priority level is compared to all the other requests (interrupts or DMA).

If the highest priority request is an interrupt, its PRL value must be strictly lower (that is, higher priority) than the CPL value stored in the CICR register (R230) in order to be acknowledged. The Top Level Interrupt overrides every other priority.

3.4.1 Priority level 7 (Lowest)

Interrupt requests at PRL level 7 cannot be acknowledged, as this PRL value (the lowest possible priority) cannot be strictly lower than the CPL value. This can be of use in a fully polled interrupt environment.

3.4.2 Maximum depth of nesting

No more than 8 routines can be nested. If an interrupt routine at level N is being serviced, no other Interrupts located at level N can interrupt it. This guarantees a maximum number of 8 nested levels including the Top Level Interrupt request.

3.4.3 Simultaneous Interrupts

If two or more requests occur at the same time and at the same priority level, an on-chip daisy chain, specific to every ST9 version, selects the channel

PRIORITY LEVEL ARBITRATION (Cont'd)

with the highest position in the chain, as shown in Table 11.

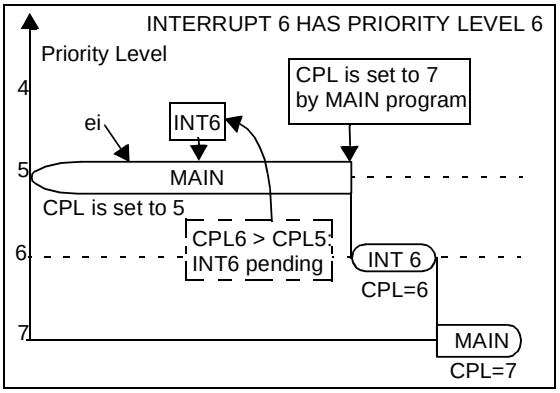
Table 11. Daisy Chain Priority

Highest Position	INTA0 / Watchdog Timer
	INTA1 / Standard Timer
	INTB0 / Extended Function Timer
	INTC1 / SPI
	INTD0 / RCCU
	INTD1 / WKUP MGT
	Induction Motor Controller
Lowest Position	AD Converter

3.4.4 Dynamic Priority Level Modification

The main program and routines can be specifically prioritized. Since the CPL is represented by 3 bits in a read/write register, it is possible to modify dynamically the current priority value during program execution. This means that a critical section can have a higher priority with respect to other interrupt requests. Furthermore it is possible to prioritize even the Main Program execution by modifying the CPL during its execution. See Figure 16

Figure 16. Example of Dynamic priority level modification in Nested Mode



3.5 ARBITRATION MODES

The ST9 provides two interrupt arbitration modes: Concurrent mode and Nested mode. Concurrent mode is the standard interrupt arbitration mode. Nested mode improves the effective interrupt response time when service routine nesting is required, depending on the request priority levels.

The IAM control bit in the CICR Register selects Concurrent Arbitration mode or Nested Arbitration Mode.

3.5.1 Concurrent Mode

This mode is selected when the IAM bit is cleared (reset condition). The arbitration phase, performed during every instruction, selects the request with the highest priority level. The CPL value is not modified in this mode.

Start of Interrupt Routine

The interrupt cycle performs the following steps:

– All maskable interrupt requests are disabled by clearing CICR.IEN.

– The PC low byte is pushed onto system stack.

– The PC high byte is pushed onto system stack.

– If ENCSR is set, CSR is pushed onto system stack.

– The Flag register is pushed onto system stack.

– The PC is loaded with the 16-bit vector stored in the Vector Table, pointed to by the IVR.

– If ENCSR is set, CSR is loaded with ISR contents; otherwise ISR is used in place of CSR until `iret` instruction.

End of Interrupt Routine

The Interrupt Service Routine must be ended with the `iret` instruction. The `iret` instruction executes the following operations:

– The Flag register is popped from system stack.

– If ENCSR is set, CSR is popped from system stack.

– The PC high byte is popped from system stack.

– The PC low byte is popped from system stack.

– All unmasked Interrupts are enabled by setting the CICR.IEN bit.

– If ENCSR is reset, CSR is used instead of ISR.

Normal program execution thus resumes at the interrupted instruction. All pending interrupts remain pending until the next `ei` instruction (even if it is executed during the interrupt service routine).

Note: In Concurrent mode, the source priority level is only useful during the arbitration phase, where it is compared with all other priority levels and with the CPL. No trace is kept of its value during the ISR. If other requests are issued during the interrupt service routine, once the global CICR.IEN is re-enabled, they will be acknowledged regardless of the interrupt service routine's priority. This may cause undesirable interrupt response sequences.

ARBITRATION MODES (Cont'd)

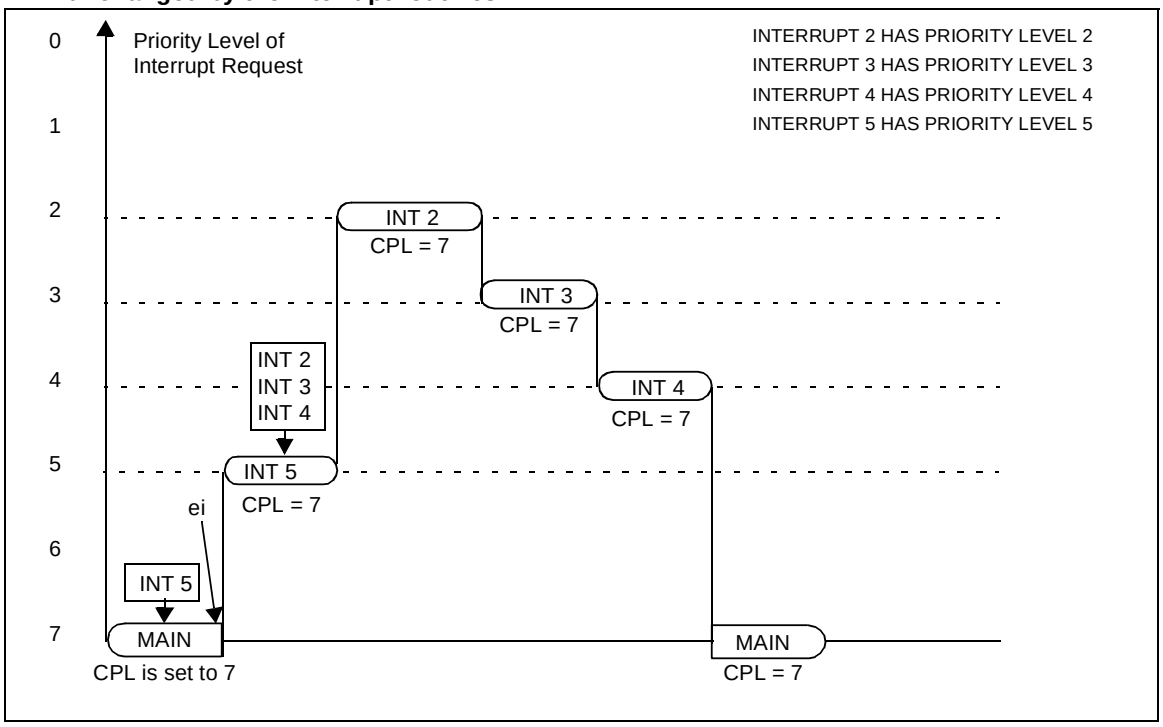
Examples

In the following two examples, three interrupt requests with different priority levels (2, 3 & 4) occur simultaneously during the interrupt 5 service routine.

Example 1

In the first example, (simplest case, Figure 17) the `ei` instruction is not used within the interrupt service routines. This means that no new interrupt can be serviced in the middle of the current one. The interrupt routines will thus be serviced one after another, in the order of their priority, until the main program eventually resumes.

Figure 17. Simple Example of a Sequence of Interrupt Requests with:
- Concurrent mode selected and
- IEN unchanged by the interrupt routines



ARBITRATION MODES (Cont'd)

Example 2

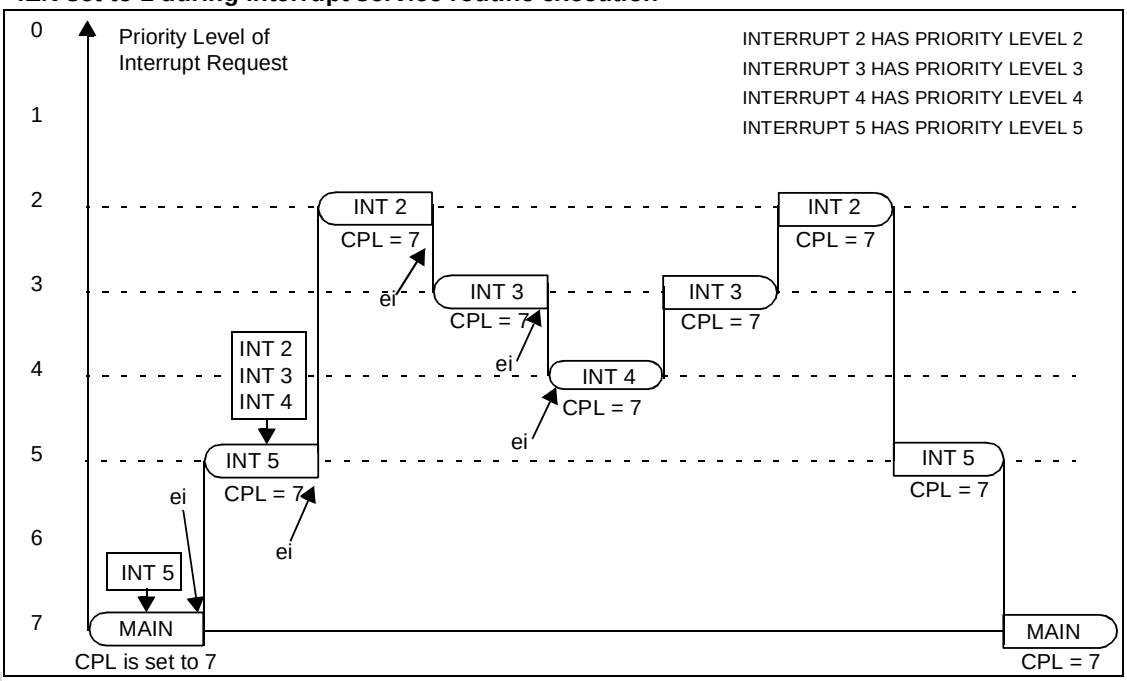
In the second example, (more complex, Figure 18), each interrupt service routine sets Interrupt Enable with the `ei` instruction at the beginning of the routine. Placed here, it minimizes response time for requests with a higher priority than the one being serviced.

The level 2 interrupt routine (with the highest priority) will be acknowledged first, then, when the `ei` instruction is executed, it will be interrupted by the level 3 interrupt routine, which itself will be interrupted by the level 4 interrupt routine. When the level 4 interrupt routine is completed, the level 3 interrupt routine resumes and finally the level 2 interrupt routine. This results in the three interrupt service routines being executed in the opposite order of their priority.

It is therefore recommended to avoid inserting the `ei` instruction in the interrupt service routine in Concurrent mode. Use the `ei` instruction only in nested mode.

WARNING: If, in Concurrent Mode, interrupts are nested (by executing `ei` in an interrupt service routine), make sure that either ENCSR is set or CSR=ISR, otherwise the `iret` of the innermost interrupt will make the CPU use CSR instead of ISR before the outermost interrupt service routine is terminated, thus making the outermost routine fail.

Figure 18. Complex Example of a Sequence of Interrupt Requests with:
- Concurrent mode selected
- IEN set to 1 during interrupt service routine execution



ARBITRATION MODES (Cont'd)

3.5.2 Nested Mode

The difference between Nested mode and Concurrent mode, lies in the modification of the Current Priority Level (CPL) during interrupt processing.

The arbitration phase is basically identical to Concurrent mode, however, once the request is acknowledged, the CPL is saved in the Nested Interrupt Control Register (NICR) by setting the NICR bit corresponding to the CPL value (i.e. if the CPL is 3, the bit 3 will be set).

The CPL is then loaded with the priority of the request just acknowledged; the next arbitration cycle is thus performed with reference to the priority of the interrupt service routine currently being executed.

Start of Interrupt Routine

The interrupt cycle performs the following steps:

– All maskable interrupt requests are disabled by clearing CICR.IEN.

– CPL is saved in the special NICR stack to hold the priority level of the suspended routine.

– Priority level of the acknowledged routine is stored in CPL, so that the next request priority will be compared with the one of the routine currently being serviced.

– The PC low byte is pushed onto system stack.

– The PC high byte is pushed onto system stack.

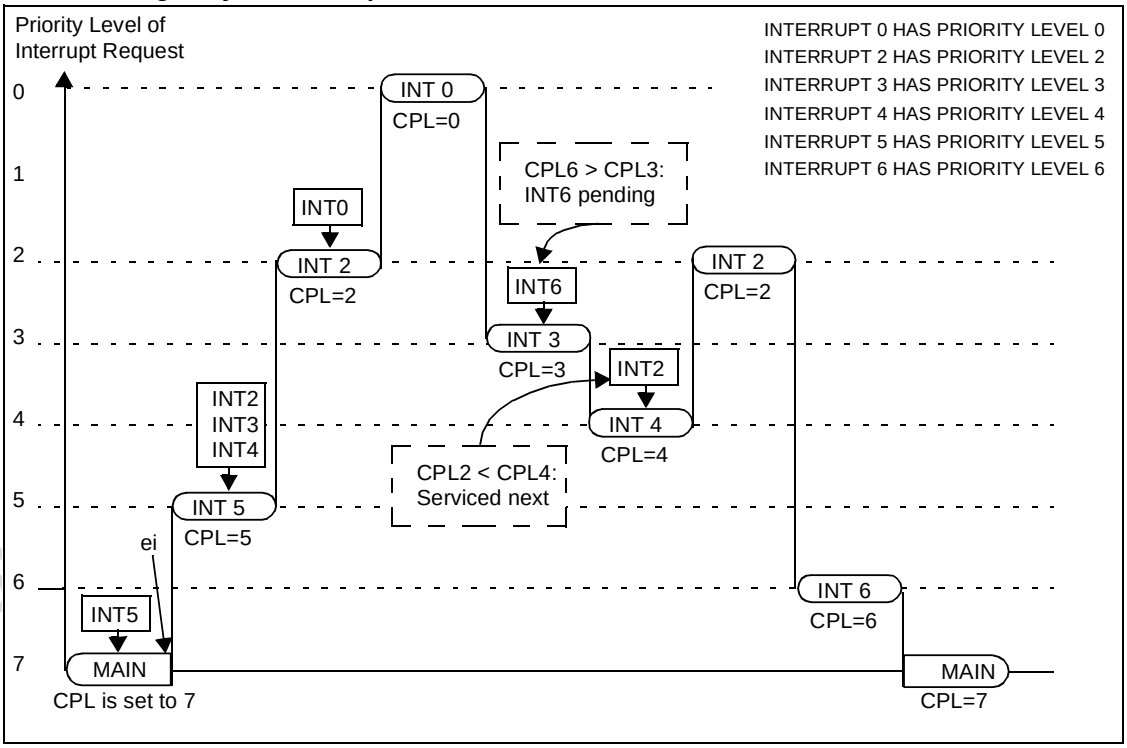
– If ENCSR is set, CSR is pushed onto system stack.

– The Flag register is pushed onto system stack.

– The PC is loaded with the 16-bit vector stored in the Vector Table, pointed to by the IVR.

– If ENCSR is set, CSR is loaded with ISR contents; otherwise ISR is used in place of CSR until `iret` instruction.

Figure 19. Simple Example of a Sequence of Interrupt Requests with:
- Nested mode
- IEN unchanged by the interrupt routines



ARBITRATION MODES (Cont'd)

End of Interrupt Routine

The `iret` Interrupt Return instruction executes the following steps:

– The Flag register is popped from system stack.

– If ENCSR is set, CSR is popped from system stack.

– The PC high byte is popped from system stack.

– The PC low byte is popped from system stack.

– All unmasked Interrupts are enabled by setting the CICR.IEN bit.

– The priority level of the interrupted routine is popped from the special register (NICR) and copied into CPL.

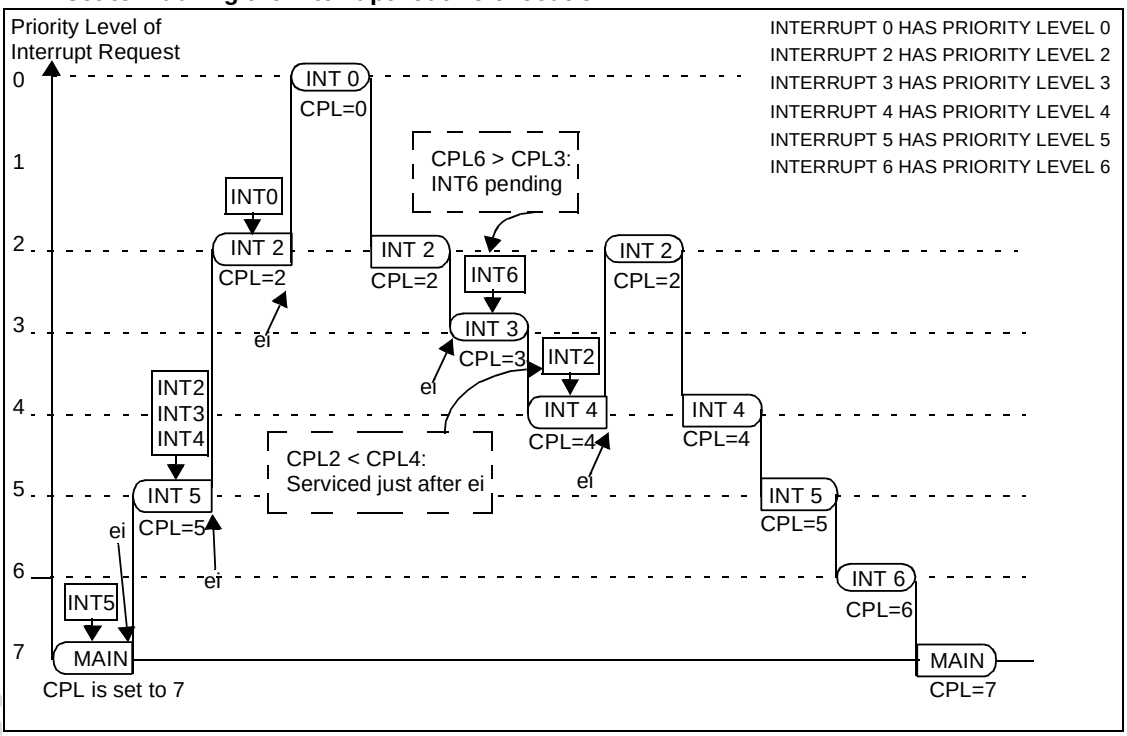
– If ENCSR is reset, CSR is used instead of ISR, unless the program returns to another nested routine.

The suspended routine thus resumes at the interrupted instruction.

Figure 19 contains a simple example, showing that if the `ei` instruction is not used in the interrupt service routines, nested and concurrent modes are equivalent.

Figure 20 contains a more complex example showing how nested mode allows nested interrupt processing (enabled inside the interrupt service routines using the `ei` instruction) according to their priority level.

Figure 20. Complex Example of a Sequence of Interrupt Requests with:
- Nested mode
- IEN set to 1 during the interrupt routine execution



3.6 EXTERNAL INTERRUPTS

The standard ST9 core contains 8 external interrupts sources grouped into four pairs.

Table 12. External Interrupt Channel Grouping

External Interrupt	Channel
none INT6	INTD1 INTD0
none none	INTC1 INTC0
none none	INTB1 INTB0
none INT0	INTA1 INTA0

Each source has a trigger control bit TEA0,..TED1 (R242,EITR.0,..,7 Page 0) to select triggering on the rising or falling edge of the external pin. If the Trigger control bit is set to “1”, the corresponding pending bit IPA0,..,IPD1 (R243,EIPR.0,..,7 Page 0) is set on the input pin rising edge, if it is cleared, the pending bit is set on the falling edge of the input pin. Each source can be individually masked through the corresponding control bit IMA0,..,IMD1 (EIMR.7,..,0). See Figure 22.

The priority level of the external interrupt sources can be programmed among the eight priority levels with the control register EIPLR (R245). The priority level of each pair is software defined using the bits PRL2, PRL1. For each pair, the even channel (A0,B0,C0,D0) of the group has the even priority level and the odd channel (A1,B1,C1,D1) has the odd (lower) priority level.

Figure 21. Priority Level Examples

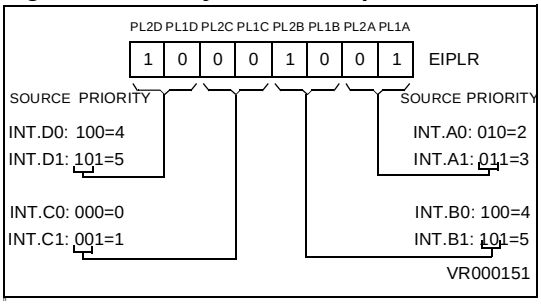


Figure 21 shows an example of priority levels.

Figure 22 gives an overview of the External interrupt control bits and vectors.

– The source of the interrupt channel A0 can be selected between the external pin INT0 (when IA0S = “1”, the reset value) or the On-chip Timer/ Watchdog peripheral (when IA0S = “0”).

– The source of the interrupt channel D0 can be selected between the external pin INT6 (when INT_SEL = “0”) or the on-chip RCCU.

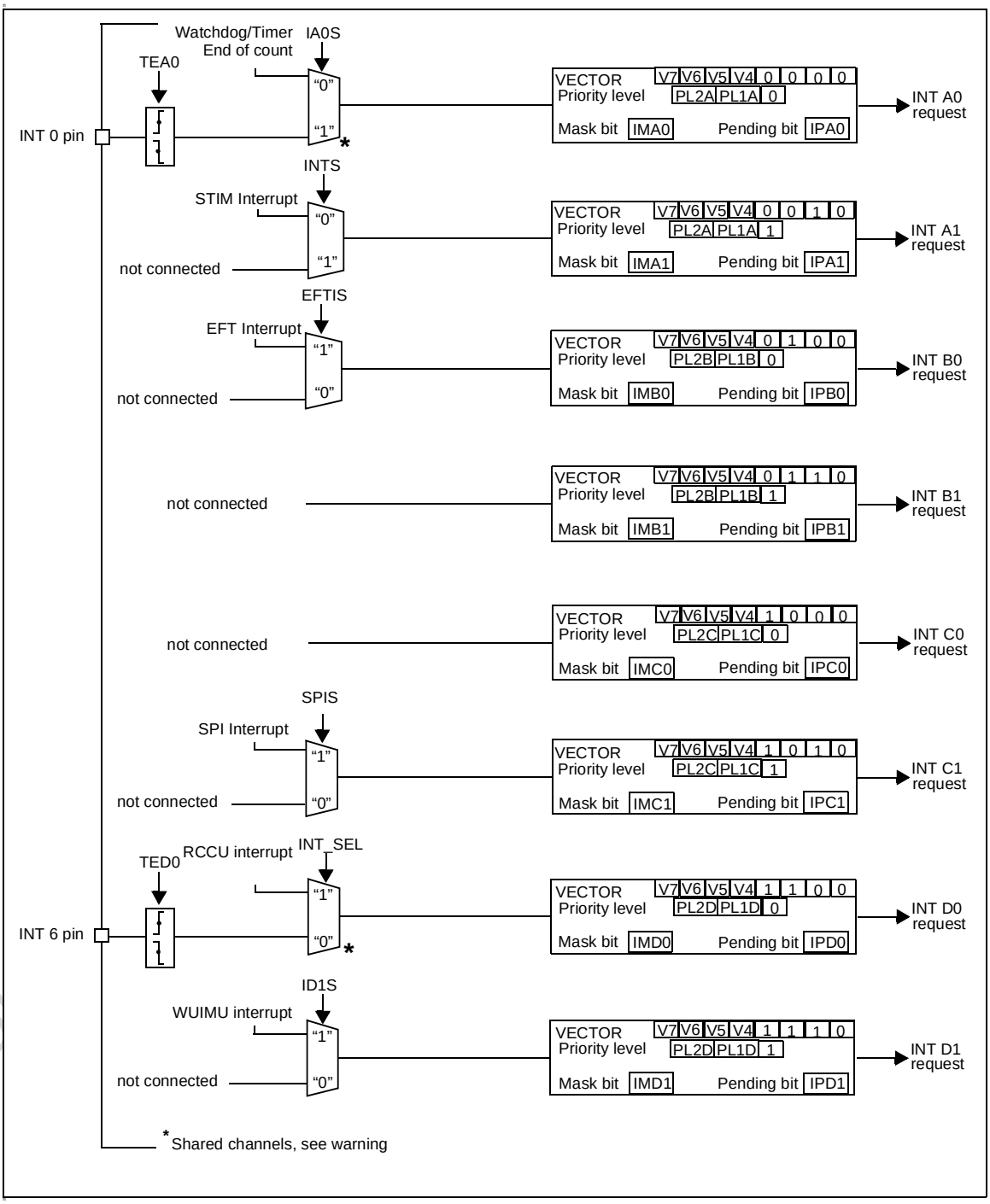
WARNING: When using channels shared by both external interrupts and peripherals, special care must be taken to configure their control registers for both peripherals and interrupts.

Table 13. Multiplexed Interrupt Sources

Channel	Internal Interrupt Source	External Interrupt Source
INTA0	Timer/Watchdog	INT0
INTD0	RCCU	INT6

EXTERNAL INTERRUPTS (Cont'd)

Figure 22. External Interrupts Control Bits and Vectors



3.7 TOP LEVEL INTERRUPT

The Top Level Interrupt channel can be assigned either to the external pin NMI or to the Timer/ Watchdog according to the status of the control bit EIVR.TLIS (R246.2, Page 0). If this bit is high (the reset condition) the source is the external pin NMI. If it is low, the source is the Timer/ Watchdog End Of Count. When the source is the NMI external pin, the control bit EIVR.TLTEV (R246.3; Page 0) selects between the rising (if set) or falling (if reset) edge generating the interrupt request. When the selected event occurs, the CICR.TLIP bit (R230.6) is set. Depending on the mask situation, a Top Level Interrupt request may be generated. Two kinds of masks are available, a Maskable mask and a Non-Maskable mask. The first mask is the CICR.TLI bit (R230.5): it can be set or cleared to enable or disable respectively the Top Level Interrupt request. If it is enabled, the global Enable Interrupt bit, CICR.IEN (R230.4) must also be enabled in order to allow a Top Level Request.

The second mask NICR.TLNM (R247.7) is a set-only mask. Once set, it enables the Top Level Interrupt request independently of the value of CICR.IEN and it cannot be cleared by the program. Only the processor RESET cycle can clear this bit. This does not prevent the user from ignoring some sources due to a change in TLIS.

The Top Level Interrupt Service Routine cannot be interrupted by any other interrupt or DMA request, in any arbitration mode, not even by a subsequent Top Level Interrupt request.

WARNING. The interrupt machine cycle of the Top Level Interrupt does not clear the CICR.IEN bit, and the corresponding `iret` does not set it. Furthermore the TLI never modifies the CPL bits and the NICR register.

3.8 ON-CHIP PERIPHERAL INTERRUPTS

The general structure of the peripheral interrupt unit is described here, however each on-chip peripheral has its own specific interrupt unit containing one or more interrupt channels, or DMA channels. Please refer to the specific peripheral chapter for the description of its interrupt features and control registers.

The on-chip peripheral interrupt channels provide the following control bits:

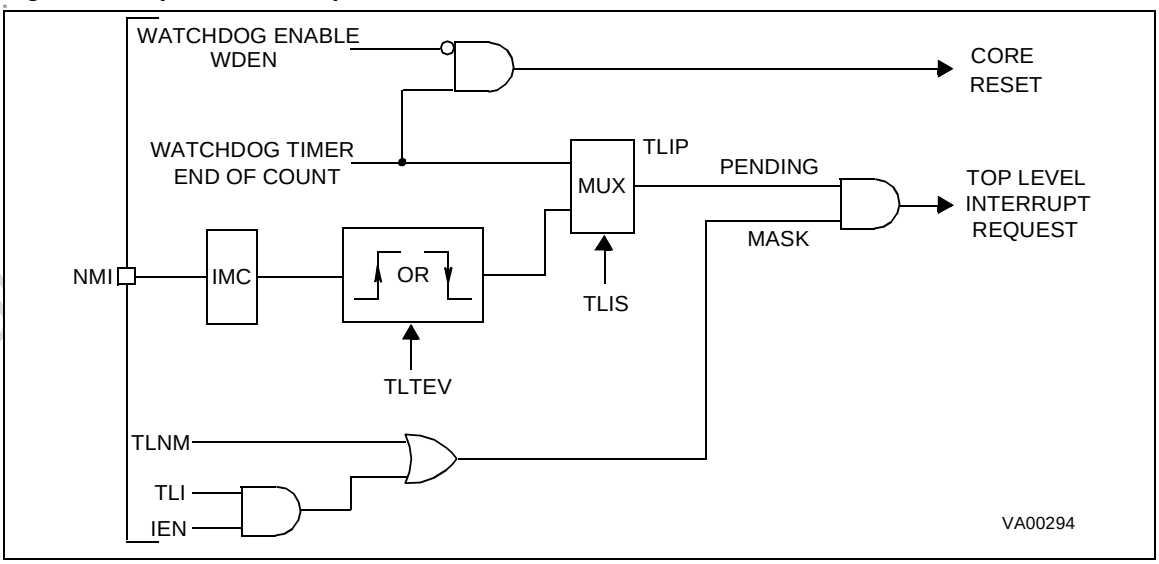
– **Interrupt Pending bit** (IP). Set by hardware when the Trigger Event occurs. Can be set/ cleared by software to generate/cancel pending interrupts and give the status for Interrupt polling.

– **Interrupt Mask bit** (IM). If IM = “0”, no interrupt request is generated. If IM =“1” an interrupt request is generated whenever IP = “1” and CICR.IEN = “1”.

– **Priority Level** (PRL, 3 bits). These bits define the current priority level, PRL=0: the highest priority, PRL=7: the lowest priority (the interrupt cannot be acknowledged)

– **Interrupt Vector Register** (IVR, up to 7 bits). The IVR points to the vector table which itself contains the interrupt routine start address.

Figure 23. Top Level Interrupt Structure



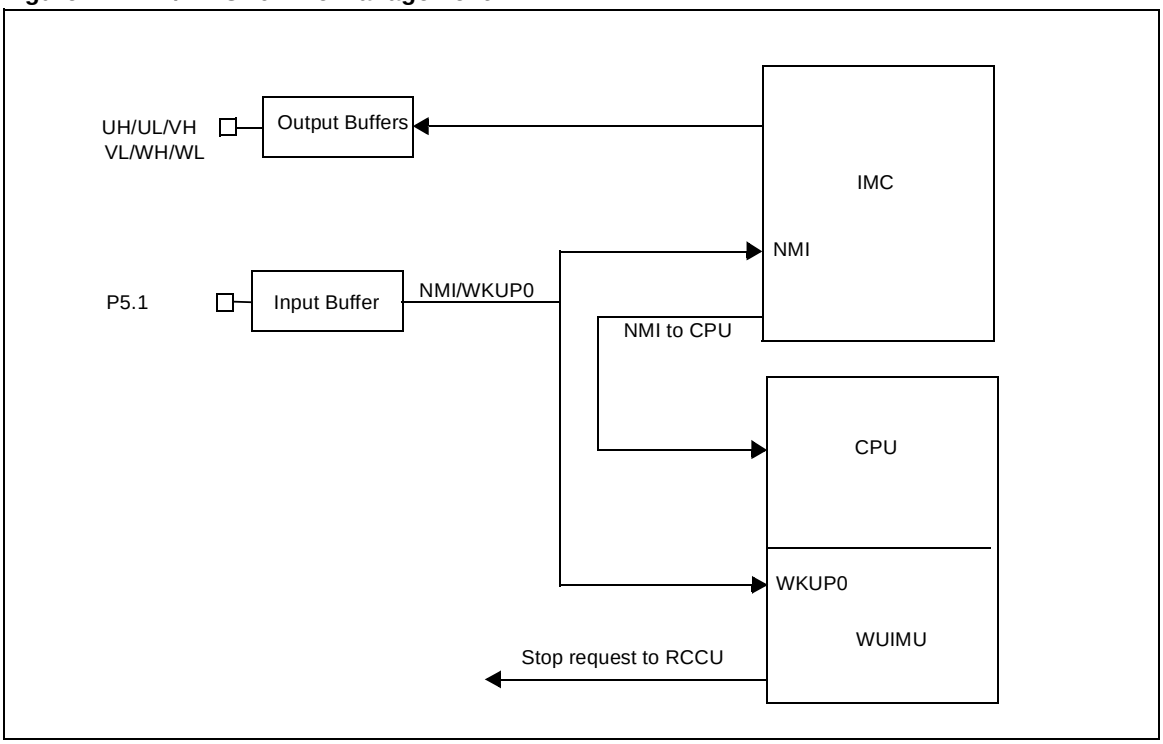
3.9 NMI/WKP0 LINE MANAGEMENT

In the ST92141, the Non Maskable Interrupt (NMI) and the Wake Up 0 line (WKUP0) functionalities are both physically mapped on the same I/O Port pin P5.1 (refer to Section 1.2).

The NMI/WKUP0 is a single alternate function Input, associated with pin P5.1. It is input to the Induction Motor Controller (IMC) and the Wake Up Management Unit (WUIMU).

The IMC Controller processes the NMI Input and generates the Non Maskable Interrupt request to the CPU (refer to Figure 24 and IMC Figure 71).

Figure 24. NMI/WKUP0 Line Management



NMI Event Handling

To enable an NMI event on the NMI/WKUP0 line, the following bits must be programmed:

– TLNM bit in the NICR register,

– TLI and IEN bits in the CICR register

– NMI bit in the IMCIVR register

– NMIL bit in the PBR register

– NMIE bit in the PCR1 register

An event on the NMI/WKUP0 line is handled by the ST92141 in the following way:

– a NMI event is acknowledged in the CPU only when the internal clock INTCLK is running (i.e. when the ST92141 is not in Stop Mode).

– a NMI event is immediately acknowledged in the IMC. The ST92141 can be either in Stop or in Run Mode (the NMI/WKUP0 line is detected asynchronously).

Wake-up Event Handling

To enable a wake-up event on the NMI/WKUP0 line, the following bits must be programmed:

– WUMx bits in the WUMRL register

– WUTx bits in the WUTRL register

An event on the NMI/WKUP0 or the WKUP[3:1] lines is handled by the ST92141 in the following way:

– a wake up event of one external line (out of the four available), is immediately acknowledged in the WUIMU. The ST92141 can be either in Stop or in Run mode (the NMI/WKUP0 and WKUP[3:1] lines are detected asynchronously).

NMI/WKP0 LINE MANAGEMENT (Cont'd)

3.9.1 NMI/Wake-Up Event Handling in Run mode

The four external lines WKUP0/NMI, WKUP1-3 can also be used when the device is in Run Mode. In addition, if the WKUP0/NMI line is used and the NMI and WKUP0 events are enabled by programming the CPU, IMC and WUIMU registers, a transition on the input pin can generate the following events:

– **IMC**: the six output phases UH/UL/VH/VL/WH/WL are released in High Impedance. The NMI bit of the IMCIVR register is automatically set to "1". A non maskable interrupt request is then sent to the CPU.

– **CPU**: the NMI pending bit of the CICR register is set and the corresponding NMI interrupt routine is immediately executed.

Note 1: The NMI pending bits of the IMCIVR register must be cleared by software in the NMI routine, whereas the NMI pending bit of the CICR register is cleared by hardware when NMI routine is acknowledged.

Note 2: The external NMI/WKUP0 event is flagged in the NMI pending bit of the IMCIVR register. The NMI routine must clear this bit. This operation must occur after disactivation of the NMI/WKUP0 line (otherwise, the next NMI/WKUP0 event will be lost, if the CPU is sensitive to a rising edge on the NMI input).

The flexibility of the ST9 also allows the use of the NMI/WKUP0 line as a wake up function only or as a Non Maskable Interrupt only.

WARNING:

1. The NMI management implemented in the ST92141 imposes the following constraints on the P5.3 (NMI/WKUP0) I.O pin:

– No glitches should occur on the pin to avoid unintentional NMI/wake up requests.

– A minimum pulse width is requested for the pin activation (refer to ST92141 Electrical Specification).

2. The WKUP0-3 management implemented in the ST92141 imposes the following constraints on the P5.0 (WKUP1), P5.2 (WKUP0), P3.2 (WKUP3) and P3.6 (WKUP2):

– No glitches should occur on WKUP0-3 pins to avoid unintentional requests.

3.9.2 NMI/Wake-Up Event Handling in STOP mode

The ST92141 enters Stop Mode by software writing a special Stop bit setting sequence in the WUCTRL register of the WUIMU. After entering Stop Mode, the device can be woken up by one of the four Wake Up external lines (refer to Section 3.12 WAKE-UP / INTERRUPT LINES MANAGEMENT UNIT (WUIMU).

In addition, if the WKUP0/NMI line is used and the NMI and WKUP0 events are enabled by programming the CPU, IMC and WUIMU registers, a transition on the input pin can generate the following events:

– **IMC**: the six output phases UH/UL/VH/VL/WH/WL are released in High Impedance. The NMI bit of the IMCIVR register is automatically set to "1". A non maskable interrupt request is then sent to the CPU.

– **WUIMU**: the NMI/WKUP0 activation wakes up the ST92141 from Stop mode, allowing the CPU to acknowledge the NMI request from IMC

– **CPU**: the NMI pending bit of the CICR register is set and the corresponding NMI interrupt routine is executed as soon as the ST92141 is exited from Stop mode.

Note: The NMI pending bits of the IMCIVR register must be cleared by software in the NMI routine, whereas the NMI pending bit of the CICR register is cleared by hardware when the NMI routine is acknowledged.

3.9.3 Unused Wake Up Management Unit lines

The WUIMU can manage up to 16 External-Interrupt/ Wake up lines. Usually, only a subset of these 16 lines is used.

In the ST92141, 4 lines out of 16 are available as external lines (WKUP0/1/2/3) but the Pending and Mask bits of the unused lines (WKUP4 to WKUP 15) are also accessible by software (refer to Section 3.12 WAKE-UP / INTERRUPT LINES MANAGEMENT UNIT (WUIMU)). Therefore, it is possible to generate a software interrupt by disabling the Mask and by setting the Pending bit of an unused channel.

3.10 INTERRUPT RESPONSE TIME

The interrupt arbitration protocol functions completely asynchronously from instruction flow and requires 5 clock cycles. One more CPUCLK cycle is required when an interrupt is acknowledged. Requests are sampled every 5 CPUCLK cycles.

If the interrupt request comes from an external pin, the trigger event must occur a minimum of one INTCLK cycle before the sampling time.

When an arbitration results in an interrupt request being generated, the interrupt logic checks if the current instruction (which could be at any stage of execution) can be safely aborted; if this is the case, instruction execution is terminated immediately and the interrupt request is serviced; if not, the CPU waits until the current instruction is terminated and then services the request. Instruction execution can normally be aborted provided no write operation has been performed.

For an interrupt deriving from an external interrupt channel, the response time between a user event and the start of the interrupt service routine can range from a minimum of 26 clock cycles to a maximum of 55 clock cycles (DIV instruction), 53 clock cycles (DIVWS and MUL instructions) or 49 for other instructions.

For a non-maskable Top Level interrupt, the response time between a user event and the start of the interrupt service routine can range from a minimum of 22 clock cycles to a maximum of 51 clock cycles (DIV instruction), 49 clock cycles (DIVWS and MUL instructions) or 45 for other instructions.

In order to guarantee edge detection, input signals must be kept low/high for a minimum of one INTCLK cycle.

An interrupt machine cycle requires a basic 18 internal clock cycles (CPUCLK), to which must be added a further 2 clock cycles if the stack is in the Register File. 2 more clock cycles must further be added if the CSR is pushed (ENCSR =1).

The interrupt machine cycle duration forms part of the two examples of interrupt response time previously quoted; it includes the time required to push values on the stack, as well as interrupt vector handling.

In Wait for Interrupt mode, a further cycle is required as wake-up delay.

3.11 INTERRUPT REGISTERS

CENTRAL INTERRUPT CONTROL REGISTER (CICR)
R230 - Read/Write
Register Group: System
Reset value: 1000 0111 (87h)

7							0
GCEN	TLIP	TLI	IEN	IAM	CPL2	CPL1	CPL0

Bit 7 = **GCEN**: *Global Counter Enable.*
This bit enables the 16-bit Multifunction Timer peripheral.
0: MFT disabled
1: MFT enabled

Bit 6 = **TLIP**: *Top Level Interrupt Pending.*
This bit is set by hardware when Top Level Interrupt (TLI) trigger event occurs. It is cleared by hardware when a TLI is acknowledged. It can also be set by software to implement a software TLI.
0: No TLI pending
1: TLI pending

Bit 5 = **TLI**: *Top Level Interrupt.*
This bit is set and cleared by software.
0: A Top Level Interrupt is generated when TLIP is set, only if TLNM=1 in the NICR register (independently of the value of the IEN bit).
1: A Top Level Interrupt request is generated when IEN=1 and the TLIP bit are set.

Bit 4 = **IEN**: *Interrupt Enable.*
This bit is cleared by the interrupt machine cycle (except for a TLI).
It is set by the `iret` instruction (except for a return from TLI).
It is set by the `EI` instruction.
It is cleared by the `DI` instruction.
0: Maskable interrupts disabled
1: Maskable Interrupts enabled

Note: The IEN bit can also be changed by software using any instruction that operates on register CICR, however in this case, take care to avoid spurious interrupts, since IEN cannot be cleared in the middle of an interrupt arbitration. Only modify the IEN bit when interrupts are disabled or when no peripheral can generate interrupts. For example, if the state of IEN is not known in advance, and its value must be restored from a previous push of CICR on the stack, use the sequence `DI; POP CICR` to make sure that no interrupts are being arbitrated when CICR is modified.

Bit 3 = **IAM**: *Interrupt Arbitration Mode.*
This bit is set and cleared by software.
0: Concurrent Mode
1: Nested Mode

Bit 2:0 = **CPL[2:0]:** *Current Priority Level.*
These bits define the Current Priority Level. CPL=0 is the highest priority. CPL=7 is the lowest priority. These bits may be modified directly by the interrupt hardware when Nested Interrupt Mode is used.

EXTERNAL INTERRUPT TRIGGER REGISTER (EITR)
R242 - Read/Write
Register Page: 0
Reset value: 0000 0000 (00h)

7							0
TED1	TED0	TEC1	TEC0	TEB1	TEB0	TEA1	TEA0

Bit 7 = **TED1**: *INTD1 Trigger Event*
Bit 6 = **TED0**: *INTD0 Trigger Event*
Bit 5 = **TEC1**: *INTC1 Trigger Event*
Bit 4 = **TEC0**: *INTC0 Trigger Event*
Bit 3 = **TEB1**: *INTB1 Trigger Event*
Bit 2 = **TEB0**: *INTB0 Trigger Event*
Bit 1 = **TEA1**: *INTA1 Trigger Event*
Bit 0 = **TEA0**: *INTA0 Trigger Event*
These bits are set and cleared by software.
0: Select falling edge as interrupt trigger event
1: Select rising edge as interrupt trigger event

INTERRUPT REGISTERS (Cont'd)

EXTERNAL INTERRUPT PENDING REGISTER (EIPR)
R243 - Read/Write
Register Page: 0
Reset value: 0000 0000 (00h)

7							0
IPD1	IPD0	IPC1	IPC0	IPB1	IPB0	IPA1	IPA0

Bit 7 = **IPD1**: *INTD1 Interrupt Pending bit*
Bit 6 = **IPD0**: *INTD0 Interrupt Pending bit*
Bit 5 = **IPC1**: *INTC1 Interrupt Pending bit*
Bit 4 = **IPC0**: *INTC0 Interrupt Pending bit*
Bit 3 = **IPB1**: *INTB1 Interrupt Pending bit*
Bit 2 = **IPB0**: *INTB0 Interrupt Pending bit*
Bit 1 = **IPA1**: *INTA1 Interrupt Pending bit*
Bit 0 = **IPA0**: *INTA0 Interrupt Pending bit*

These bits are set by hardware on occurrence of a trigger event (as specified in the EITR register) and are cleared by hardware on interrupt acknowledge. They can also be set by software to implement a software interrupt.
0: No interrupt pending
1: Interrupt pending

EXTERNAL INTERRUPT MASK-BIT REGISTER (EIMR)
R244 - Read/Write
Register Page: 0
Reset value: 0000 0000 (00h**)**

7							0
IMD1	IMD0	IMC1	IMC0	IMB1	IMB0	IMA1	IMA0

Bit 7 = **IMD1**: *INTD1 Interrupt Mask*
Bit 6 = **IMD0**: *INTD0 Interrupt Mask*
Bit 5 = **IMC1**: *INTC1 Interrupt Mask*
Bit 4 = **IMC0**: *INTC0 Interrupt Mask*
Bit 3 = **IMB1**: *INTB1 Interrupt Mask*
Bit 2 = **IMB0**: *INTB0 Interrupt Mask*
Bit 1 = **IMA1**: *INTA1 Interrupt Mask*
Bit 0 = **IMA0**: *INTA0 Interrupt Mask*

These bits are set and cleared by software.
0: Interrupt masked
1: Interrupt not masked (an interrupt is generated if the IPxx and IEN bits = 1)

EXTERNAL INTERRUPT PRIORITY LEVEL REGISTER (EIPLR)
R245 - Read/Write
Register Page: 0
Reset value: 1111 1111 (FFh**)**

7							0
PL2D	PL1D	PL2C	PL1C	PL2B	PL1B	PL2A	PL1A

Bit 7:6 = **PL2D, PL1D:** *INTD0, D1 Priority Level.*
Bit 5:4 = **PL2C, PL1C**: *INTC0, C1 Priority Level.*
Bit 3:2 = **PL2B, PL1B**: *INTB0, B1 Priority Level.*
Bit 1:0 = **PL2A, PL1A**: *INTA0, A1 Priority Level.*

These bits are set and cleared by software.

The priority is a three-bit value. The LSB is fixed by hardware at 0 for Channels A0, B0, C0 and D0 and at 1 for Channels A1, B1, C1 and D1.

PL2x	PL1x	Hardware bit	Priority
0	0	0 1	0 (Highest) 1
0	1	0 1	2 3
1	0	0 1	4 5
1	1	0 1	6 7 (Lowest)

INTERRUPT REGISTERS (Cont'd)

EXTERNAL INTERRUPT VECTOR REGISTER (EIVR)

R246 - Read/Write
Register Page: 0
Reset value: xxxx 0110b (x6h)

7							0
V7	V6	V5	V4	TLTEV	TLIS	IAOS	EWEN

Bit 7:4 = **V[7:4]**: *Most significant nibble of External Interrupt Vector.*
These bits are not initialized by reset. For a representation of how the full vector is generated from V[7:4] and the selected external interrupt channel, refer to Figure 22.

Bit 3 = **TLTEV**: *Top Level Trigger Event bit.*
This bit is set and cleared by software.
0: Select falling edge as NMI trigger event
1: Select rising edge as NMI trigger event

Bit 2 = **TLIS**: *Top Level Input Selection.*
This bit is set and cleared by software.
0: Watchdog End of Count is TL interrupt source
1: NMI is TL interrupt source

Bit 1 = **IA0S**: *Interrupt Channel A0 Selection.*
This bit is set and cleared by software.
0: Watchdog End of Count is INTA0 source
1: External Interrupt pin is INTA0 source

Bit 0 = **EWEN**: *External Wait Enable.*
This bit is set and cleared by software.
0: WAITN pin disabled
1: WAITN pin enabled (to stretch the external memory access cycle).

Note: For more details on Wait mode refer to the section describing the WAITN pin in the External Memory Chapter.

NESTED INTERRUPT CONTROL (NICR)

R247 - Read/Write
Register Page: 0
Reset value: 0000 0000 (00h)

7							0
TLNM	HL6	HL5	HL4	HL3	HL2	HL1	HL0

Bit 7 = **TLNM**: *Top Level Not Maskable.*
This bit is set by software and cleared only by a hardware reset.
0: Top Level Interrupt Maskable. A top level request is generated if the IEN, TLI and TLIP bits =1
1: Top Level Interrupt Not Maskable. A top level request is generated if the TLIP bit =1

Bit 6:0 = **HL[6:0]**: *Hold Level* x
These bits are set by hardware when, in Nested Mode, an interrupt service routine at level x is interrupted from a request with higher priority (other than the Top Level interrupt request). They are cleared by hardware at the `iret` execution when the routine at level x is recovered.

3.12 WAKE-UP / INTERRUPT LINES MANAGEMENT UNIT (WUIMU)

3.12.1 Introduction

The Wake-up/Interrupt Management Unit extends the number of external interrupt lines from 8 to 23 (depending on the number of external interrupt lines mapped on external pins of the device). It allows the source of the INTD1 external interrupt channel to be used for up to 16 additional external Wake-up/interrupt pins.

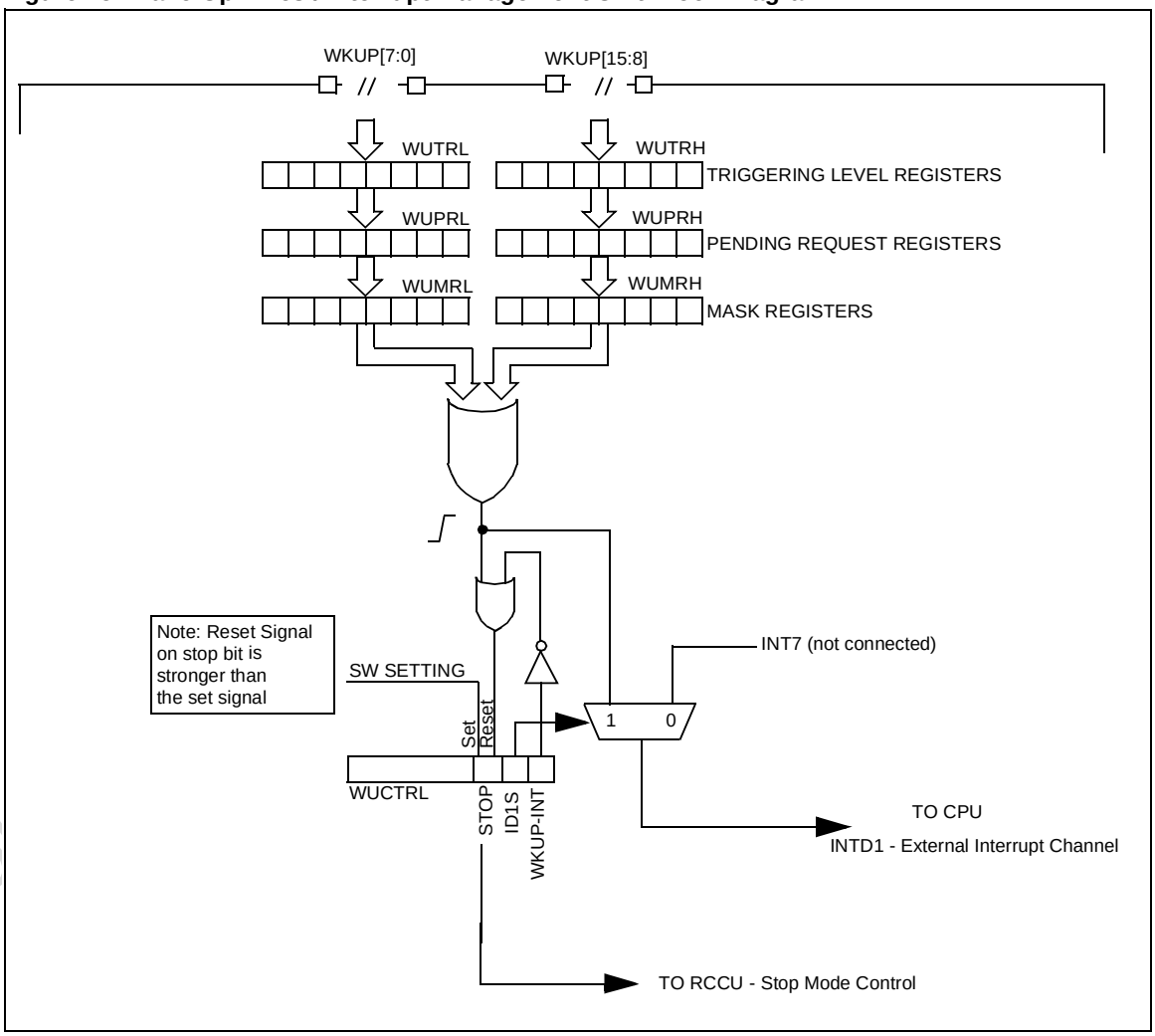
These 16 WKUP pins can be programmed as external interrupt lines or as wake-up lines, able to exit the microcontroller from low power mode (STOP mode) (see Figure 25).

3.12.2 Main Features

- Supports up to 16 additional external wake-up or interrupt lines
- Wake-Up lines can be used to wake-up the ST9 from STOP mode.
- Programmable selection of wake-up or interrupt
- Programmable wake-up trigger edge polarity
- All Wake-Up Lines maskable

Note: The number of available pins is device dependent. Refer to the device pinout description.

Figure 25. Wake-Up Lines / Interrupt Management Unit Block Diagram



WAKE-UP / INTERRUPT LINES MANAGEMENT UNIT (Cont'd)

3.12.3 Functional Description

3.12.3.1 Interrupt Mode

To configure the 16 wake-up lines as interrupt sources, use the following procedure:

1. Configure the mask bits of the 16 wake-up lines (WUMRL, WUMRH).
2. Configure the triggering edge registers of the wake-up lines (WUTRL, WUTRH).
3. Set bit 7 of EIMR (R244 Page 0) and EITR (R242 Page 0) registers of the CPU: so an interrupt coming from one of the 16 lines can be correctly acknowledged.
4. Reset the WKUP-INT bit in the WUCTRL register to disable Wake-up Mode.
5. Set the ID1S bit in the WUCTRL register to disable the INT7 external interrupt source and enable the 16 wake-up lines as external interrupt source lines.

To return to standard mode (INT7 external interrupt source enabled and 16 wake-up lines disabled) it is sufficient to reset the ID1S bit.

3.12.3.2 Wake-up Mode Selection

To configure the 16 lines as wake-up sources, use the following procedure:

1. Configure the mask bits of the 16 wake-up lines (WUMRL, WUMRH).
2. Configure the triggering edge registers of the wake-up lines (WUTRL, WUTRH).
3. Set, as for Interrupt Mode selection, bit 7 of EIMR and EITR registers only if an interrupt routine is to be executed after a wake-up event. Otherwise, if the wake-up event only restarts the execution of the code from where it was stopped, the INTD1 interrupt channel must be masked or the external source must be selected by resetting the ID1S bit.
4. Since the RCCU can generate an interrupt request when exiting from STOP mode, take care to mask it even if the wake-up event is only to restart code execution.
5. Set the WKUP-INT bit in the WUCTRL register to select Wake-up Mode.
6. Set the ID1S bit in the WUCTRL register to disable the INT7 external interrupt source and enable the 16 wake-up lines as external interrupt source lines. This is not mandatory if the wake-up event does not require an interrupt response.
7. Write the sequence 1,0,1 to the STOP bit of the WUCTRL register with three consecutive write operations. This is the STOP bit setting sequence.

To detect if STOP Mode was entered or not, immediately after the STOP bit setting sequence, poll the RCCU EX_STP bit (R242.7, Page 55) and the STOP bit itself.

3.12.3.3 STOP Mode Entry Conditions

Assuming the ST9 is in Run mode: during the STOP bit setting sequence the following cases may occur:

Case 1: Wrong STOP bit setting sequence

This can happen if an Interrupt/DMA request is acknowledged during the STOP bit setting sequence. In this case polling the STOP and EX_STP bits will give:

STOP = 0, EX_STP = 0

This means that the ST9 did not enter STOP mode due to a bad STOP bit setting sequence: the user must retry the sequence.

Case 2: Correct STOP bit setting sequence

In this case the ST9 enters STOP mode.

To exit STOP mode, a wake-up interrupt must be acknowledged. That implies:

STOP = 0, EX_STP = 1

This means that the ST9 entered and exited STOP mode due to an external wake-up line event.

Case 3: A wake-up event on the external wake-up lines occurs during the STOP bit setting sequence

There are two possible cases:

1. Interrupt requests to the CPU are disabled: in this case the ST9 will not enter STOP mode, no interrupt service routine will be executed and the program execution continues from the instruction following the STOP bit setting sequence. The status of STOP and EX_STP bits will be again:

   STOP = 0, EX_STP = 0

   The application can determine why the ST9 did not enter STOP mode by polling the pending bits of the external lines (at least one must be at 1).

WAKE-UP / INTERRUPT LINES MANAGEMENT UNIT (Cont'd)

2. Interrupt requests to CPU are enabled: in this case the ST9 will not enter STOP mode and the interrupt service routine will be executed. The status of STOP and EX_STP bits will be again:

   STOP = 0, EX_STP = 0

   The interrupt service routine can determine why the ST9 did not enter STOP mode by polling the pending bits of the external lines (at least one must be at 1).

If the MCU really exits from STOP Mode, the RCCU EX_STP bit is still set and must be reset by software. Otherwise, if an Interrupt/DMA request was acknowledged during the STOP bit setting sequence, the RCCU EX_STP bit is reset. This means that the MCU has filtered the STOP Mode entry request.

The WKUP-INT bit can be used by an interrupt routine to detect and to distinguish events coming from Interrupt Mode or from Wake-up Mode, allowing the code to execute different procedures.

To exit STOP mode, it is sufficient that one of the 16 wake-up lines (not masked) generates an event: the clock restarts after the delay needed for the oscillator to restart.

Note: After exiting from STOP Mode, the software can successfully reset the pending bits (edge sensitive), even though the corresponding wake-up line is still active (high or low, depending on the Trigger Event register programming); the user must poll the external pin status to detect and distinguish a short event from a long one (for example keyboard input with keystrokes of varying length).

3.12.4 Programming Considerations

The following paragraphs give some guidelines for designing an application program.

3.12.4.1 Procedure for Entering/Exiting STOP mode

1. Program the polarity of the trigger event of external wake-up lines by writing registers WUTRH and WUTRL.
2. Check that at least one mask bit (registers WUMRH, WUMRL) is equal to 1 (so at least one external wake-up line is not masked).
3. Reset at least the unmasked pending bits: this allows a rising edge to be generated on the INTD1 channel when the trigger event occurs (an interrupt on channel INTD1 is recognized when a rising edge occurs).
4. Select the interrupt source of the INTD1 channel (see description of ID1S bit in the WUCTRL register) and set the WKUP-INT bit.
5. To generate an interrupt on channel INTD1, bits EITR.1 (R242.7, Page 0) and EIMR.1 (R244.7, Page 0) must be set and bit EIPR.7 must be reset. Bits 7 and 6 of register R245, Page 0 must be written with the desired priority level for interrupt channel INTD1.
6. Reset the STOP bit in register WUCTRL and the EX_STP bit in the CLK_FLAG register (R242.7, Page 55). Refer to the RCCU chapter.
7. To enter STOP mode, write the sequence 1, 0, 1 to the STOP bit in the WUCTRL register with three consecutive write operations.
8. The code to be executed just after the STOP sequence must check the status of the STOP and RCCU EX_STP bits to determine if the ST9 entered STOP mode or not (See "Wake-up Mode Selection" on page 56. for details). If the ST9 did not enter in STOP mode it is necessary to reloop the procedure from the beginning, otherwise the procedure continues from next point.
9. Poll the wake-up pending bits to determine which wake-up line caused the exit from STOP mode.
10. Clear the wake-up pending bit that was set.

WAKE-UP / INTERRUPT LINES MANAGEMENT UNIT (Cont'd)

3.12.4.2 Simultaneous Setting of Pending Bits

It is possible that several simultaneous events set different pending bits. In order to accept subsequent events on external wake-up/interrupt lines, it is necessary to clear at least one pending bit: this operation allows a rising edge to be generated on the INTD1 line (if there is at least one more pending bit set and not masked) and so to set EIPR.7 bit again. A further interrupt on channel INTD1 will be serviced depending on the status of bit EIMR.7. Two possible situations may arise:

1. The user chooses to reset all pending bits: no further interrupt requests will be generated on channel INTD1. In this case the user has to:
   - Reset EIMR.7 bit (to avoid generating a spurious interrupt request during the next reset operation on the WUPRH register)
   - Reset WUPRH register using a read-modify-write instruction (AND, BRES, BAND)
   - Clear the EIPR.7 bit
   - Reset the WUPRL register using a read-modify-write instruction (AND, BRES, BAND)
2. The user chooses to keep at least one pending bit active: at least one additional interrupt request will be generated on the INTD1 channel. In this case the user has to reset the desired pending bits with a read-modify-write instruction (AND, BRES, BAND). This operation will generate a rising edge on the INTD1 channel and the EIPR.7 bit will be set again. An interrupt on the INTD1 channel will be serviced depending on the status of EIMR.7 bit.

WAKE-UP / INTERRUPT LINES MANAGEMENT UNIT (Cont'd)

3.12.5 Register Description

WAKE-UP CONTROL REGISTER (**WUCTRL)**
R249 - Read/Write
Register Page: 57
Reset Value: 0000 0000 (00h)

7							0
-	-	-	-	-	STOP	ID1S	WKUP-INT

Bit 2 = **STOP:** *Stop bit.*
To enter STOP Mode, write the sequence 1,0,1 to this bit with **three consecutive write operations**. When a correct sequence is recognized, the STOP bit is set and the RCCU puts the MCU in STOP Mode. The software sequence succeeds only if the following conditions are true:

– The WKUP-INT bit is 1,

– All unmasked pending bits are reset,

– At least one mask bit is equal to 1 (at least one external wake-up line is not masked).

Otherwise the MCU cannot enter STOP mode, the program code continues executing and the STOP bit remains cleared.

The bit is reset by hardware if, while the MCU is in STOP mode, a wake-up interrupt comes from any of the unmasked wake-up lines. The STOP bit is at 1 in the two following cases (See "Wake-up Mode Selection" on page 56. for details):

– After the first write instruction of the sequence (a 1 is written to the STOP bit)

– At the end of a successful sequence (i.e. after the third write instruction of the sequence)

WARNING: Writing the sequence 1,0,1 to the STOP bit will enter STOP mode only if no other register write instructions are executed during the sequence. If Interrupt or DMA requests (which always perform register write operations) are acknowledged during the sequence, the ST9 will not enter STOP mode: the user must re-enter the sequence to set the STOP bit.

WARNING: Whenever a STOP request is issued to the MCU, a few clock cycles are needed to enter STOP mode (see RCCU chapter for further details). Hence the execution of the instruction following the STOP bit setting sequence might start before entering STOP mode: if such instruction performs a register write operation, the ST9 will not enter in STOP mode. In order to avoid to execute register write instructions after a correct STOP bit setting sequence and before entering the STOP mode, it is mandatory to execute 3 NOP instructions after the STOP bit setting sequence.

Bit 1 = **ID1S:** *Interrupt Channel INTD1 Source.*
This bit is set and cleared by software.
0: INT7 external interrupt source selected, excluding wake-up line interrupt requests
1: The 16 external wake-up lines enabled as interrupt sources, replacing the INT7 external pin function

WARNING: To avoid spurious interrupt requests on the INTD1 channel due to changing the interrupt source, do the following before modifying the ID1S bit:

1. Mask the INTD1 interrupt channel (bit 7 of register EIMR - R244, Page 0 - reset to 0).
2. Program the ID1S bit as needed.
3. Clear the IPD1 interrupt pending bit (bit 7 of register EIPR - R243, Page 0).
4. Remove the mask on INTD1 (bit EIMR.7=1).

Bit 0 = **WKUP-INT:** *Wakeup Interrupt.*
This bit is set and cleared by software.
0: The 16 external wakeup lines can be used to generate interrupt requests
1: The 16 external wake-up lines to work as wake-up sources for exiting from STOP mode

WAKE-UP / INTERRUPT LINES MANAGEMENT UNIT (Cont'd)

WAKE-UP MASK REGISTER HIGH (**WUMRH)**
R250 - Read/Write
Register Page: 57
Reset Value: 0000 0000 (00h)

7							0
WUM15	WUM14	WUM13	WUM12	WUM11	WUM10	WUM9	WUM8

Bit 7:0 = **WUM[15:8]:** *Wake-Up Mask bits.*
If WUMx is set, an interrupt on channel INTD1 and/or a wake-up event (depending on ID1S and WKUP-INT bits) are generated if the corresponding WUPx pending bit is set. More precisely, if WUMx=1 and WUPx=1 then:

– If ID1S=1 and WKUP-INT=1 then an interrupt on channel INTD1 and a wake-up event are generated.

– If ID1S=1 and WKUP-INT=0 only an interrupt on channel INTD1 is generated.

– If ID1S=0 and WKUP-INT=1 only a wake-up event is generated.

– If ID1S=0 and WKUP-INT=0 neither interrupts on channel INTD1 nor wake-up events are generated. Interrupt requests on channel INTD1 may be generated only from external interrupt source INT7.

If WUMx is reset, no wake-up events can be generated. Interrupt requests on channel INTD1 may be generated only from external interrupt source INT7 (resetting ID1S bit to 0).

WAKE-UP MASK REGISTER LOW (**WUMRL)**
R251 **-** Read/Write
Register Page: 57
Reset Value: 0000 0000 (00h)

7							0
WUM7	WUM6	WUM5	WUM4	WUM3	WUM2	WUM1	WUM0

Bit 7:0 = **WUM[7:0]:** *Wake-Up Mask bits.*
If WUMx is set, an interrupt on channel INTD1 and/or a wake-up event (depending on ID1S and WKUP-INT bits) are generated if the corresponding WUPx pending bit is set. More precisely, if WUMx=1 and WUPx=1 then:

– If ID1S=1 and WKUP-INT=1 then an interrupt on channel INTD1 and a wake-up event are generated.

– If ID1S=1 and WKUP-INT=0 only an interrupt on channel INTD1 is generated.

– If ID1S=0 and WKUP-INT=1 only a wake-up event is generated.

– If ID1S=0 and WKUP-INT=0 neither interrupts on channel INTD1 nor wake-up events are generated. Interrupt requests on channel INTD1 may be generated only from external interrupt source INT7.

If WUMx is reset, no wake-up events can be generated. Interrupt requests on channel INTD1 may be generated only from external interrupt source INT7 (resetting ID1S bit to 0).

WAKE-UP / INTERRUPT LINES MANAGEMENT UNIT (Cont'd)

WAKE-UP TRIGGER REGISTER HIGH (WUTRH)
R252 - Read/Write
Register Page: 57
Reset Value: 0000 0000 (00h)

7							0
WUT15	WUT14	WUT13	WUT12	WUT11	WUT10	WUT9	WUT8

Bit 7:0 = **WUT[15:8]:** *Wake-Up Trigger Polarity Bits*
These bits are set and cleared by software.
0: The corresponding WUPx pending bit will be set on the falling edge of the input wake-up line.
1: The corresponding WUPx pending bit will be set on the rising edge of the input wake-up line.

WAKE-UP TRIGGER REGISTER LOW (WUTRL)
R253 - Read/Write
Register Page: 57
Reset Value: 0000 0000 (00h)

7							0
WUT7	WUT6	WUT5	WUT4	WUT3	WUT2	WUT1	WUT0

Bit 7:0 = **WUT[7:0]:** *Wake-Up Trigger Polarity Bits*
These bits are set and cleared by software.
0: The corresponding WUPx pending bit will be set on the falling edge of the input wake-up line.
1: The corresponding WUPx pending bit will be set on the rising edge of the input wake-up line.

WARNING

1. As the external wake-up lines are edge triggered, no glitches must be generated on these lines.
2. If either a rising or a falling edge on the external wake-up lines occurs while writing the WUTRH or WUTRL registers, the pending bit will not be set.

WAKE-UP PENDING REGISTER HIGH (WUPRH)
R254 - Read/Write
Register Page: 57
Reset Value: 0000 0000 (00h)

7							0
WUP15	WUP14	WUP13	WUP12	WUP11	WUP10	WUP9	WUP8

Bit 7:0 = **WUP[15:8]:** *Wake-Up Pending Bits*
These bits are set by hardware on occurrence of the trigger event on the corresponding wake-up line. They must be cleared by software. They can be set by software to implement a software interrupt.
0: No Wake-up Trigger event occurred
1: Wake-up Trigger event occurred

WAKE-UP PENDING REGISTER LOW (WUPRL)
R255 - Read/Write
Register Page: 57
Reset Value: 0000 0000 (00h)

7							0
WUP7	WUP6	WUP5	WUP4	WUP3	WUP2	WUP1	WUP0

Bit 7:0 = **WUP[7:0]:** *Wake-Up Pending Bits*
These bits are set by hardware on occurrence of the trigger event on the corresponding wake-up line. They must be cleared by software. They can be set by software to implement a software interrupt.
0: No Wake-up Trigger event occurred
1: Wake-up Trigger event occurred

Note: To avoid losing a trigger event while clearing the pending bits, **it is recommended** to use read-modify-write instructions (AND, BRES, BAND) to clear them.

4 EM CONFIGURATION REGISTERS (EM)

In ST9 devices with external memory, the EM registers (External Memory Registers) are used to configure the external memory interface. In the ST92141, only the BSZ, ENCSR and DPREM bits must be programmed. All other bits in these registers must be left at their reset values.

EM REGISTER 1 (EMR1)
R245 - Read/Write
Register Page: 21
Reset value: 1000 0000 (80h)

7							0
1	0	0	0	0	0	BSZ	0

Bit 7:2 = Reserved.

Bit 1 = **BSZ**: *Buffer size.*
0: I/O ports P3.6, P3.5, P5.0, P5.2 use output buffers with standard current capability (less noisy).
1: I/O ports P3.6, P3.5, P5.0, P5.2 use output buffers with high current capability (more noisy)

Bit 0 = Reserved.

EM REGISTER 2 (EMR2)
R246 - Read/Write
Register Page: 21
Reset value: 0000 1111 (0Fh)

7							0
0	ENCSR	DPREM	0	1	1	1	1

Bit 7 = Reserved, keep in reset state.

Bit 6 = **ENCSR**: *Enable Code Segment Register.* This bit is set and cleared by software. It affects the ST9 CPU behaviour whenever an interrupt request is issued.
0: The CPU works in original ST9 compatibility mode. For the duration of the interrupt service routine, ISR is used instead of CSR, and the interrupt stack frame is identical to that of the original ST9: only the PC and Flags are pushed. This avoids saving the CSR on the stack in the event of an interrupt, thus ensuring a faster interrupt response time. The drawback is that it is not possible for an interrupt service routine to perform inter-segment calls or jumps: these instructions would update the CSR, which, in this case, is not used (ISR is used instead). The code segment size for all interrupt service routines is thus limited to 64K bytes.
1: ISR is only used to point to the interrupt vector table and to initialize the CSR at the beginning of the interrupt service routine: the old CSR is pushed onto the stack together with the PC and flags, and CSR is then loaded with the contents of ISR. In this case, `iret` will also restore CSR from the stack. This approach allows interrupt service routines to access the entire 4 Mbytes of address space; the drawback is that the interrupt response time is slightly increased, because of the need to also save CSR on the stack. Full compatibility with the original ST9 is lost in this case, because the interrupt stack frame is different; this difference, however, should not affect the vast majority of programs.

Bit 5 = **DPRREM**: *Data Page Registers remapping*
0: The locations of the four MMU (Memory Management Unit) Data Page Registers (DPR0, DPR1, DPR2 and DPR3) are in page 21.
1: The four MMU Data Page Registers are swapped with that of the Data Registers of ports 0-3.
Refer to Figure 11

Bit 4:0 = Reserved, keep in reset state.

5 RESET AND CLOCK CONTROL UNIT (RCCU)

5.1 INTRODUCTION

The Reset and Clock Control Unit (RCCU) comprises two distinct sections:

– the Clock Control Unit, which generates and manages the internal clock signals.

– the Reset/Stop Manager, which detects and flags Hardware, Software and Watchdog generated resets.

In Stop mode and Halt mode, all oscillators are frozen in order to achieve the lowest possible power consumption.

Entering and exiting Stop mode is controlled by the WUIMU.

Halt mode is entered by executing the HALT instruction. Halt mode can only be exited by a reset event.

5.2 CLOCK CONTROL UNIT

The Clock Control Unit generates the internal clocks for the CPU core (CPUCLK) and for the on-chip peripherals (INTCLK). The Clock Control Unit may be driven by an external crystal circuit, connected to the OSCIN and OSCOUT pins, or by an external pulse generator, connected to OSCIN (see Figure 34 and Figure 36). If present, another clock source named CK_AF can be provided to the system. Depending on the device, it can be a periodic signal applied to the CK_AF pin or a signal generated internally by the MCU (RC oscillator).

5.2.1 Clock Control Unit Overview

As shown in Figure 26, a programmable divider can divide the CLOCK1 input clock signal by two. The resulting signal, CLOCK2, is the reference input clock to the programmable Phase Locked Loop frequency multiplier, which is capable of multiplying the clock frequency by a factor of 6, 8, 10 or 14; the multiplied clock is then divided by a programmable divider, by a factor of 1 to 7. By this means, the ST9 can operate with cheaper, medium frequency (3-5 MHz) crystals, while still providing a high frequency internal clock for maximum system performance; the range of available multiplication and division factors allow a great number of operating clock frequencies to be derived from a single crystal frequency. The undivided PLL clock is also available for special purposes (high-speed peripheral).

For low power operation, especially in Wait for Interrupt mode, the Clock Multiplier unit may be turned off, whereupon the output clock signal may be programmed as CLOCK2 divided by 16. Furthermore, during the execution of a WFI in Low Power mode, the CK_AF clock is automatically divided by 16 for further consumption reduction. (for the selection of this signal refer to the description the CK_AF clock source in the following sections of this chapter).

The internal system clock, INTCLK, is routed to all on-chip peripherals, as well as to the programmable Clock Prescaler Unit which generates the clock for the CPU core (CPUCLK).

The Clock Prescaler is programmable and can slow the CPU clock by a factor of up to 8, allowing the programmer to reduce CPU processing speed, and thus power consumption, while maintaining a high speed clock to the peripherals. This is particularly useful when little actual processing is being done by the CPU and the peripherals are doing most of the work.

Figure 26. Clock Control Unit Simplified Block Diagram

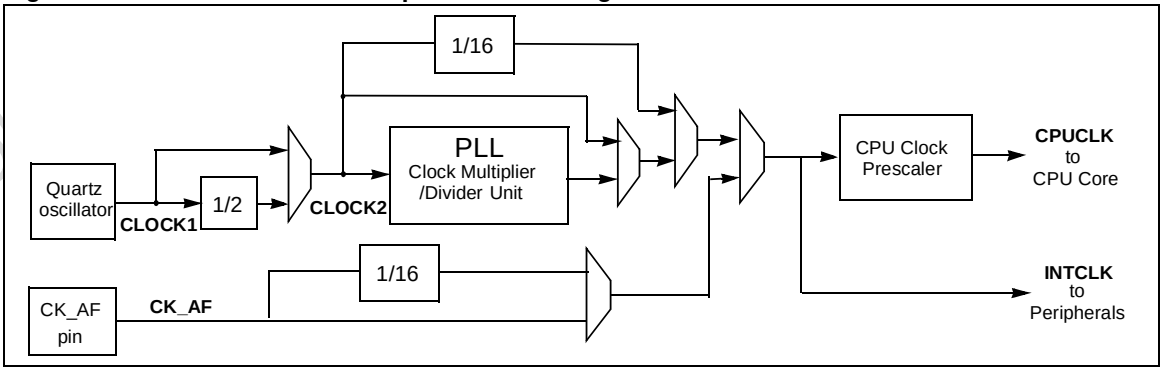
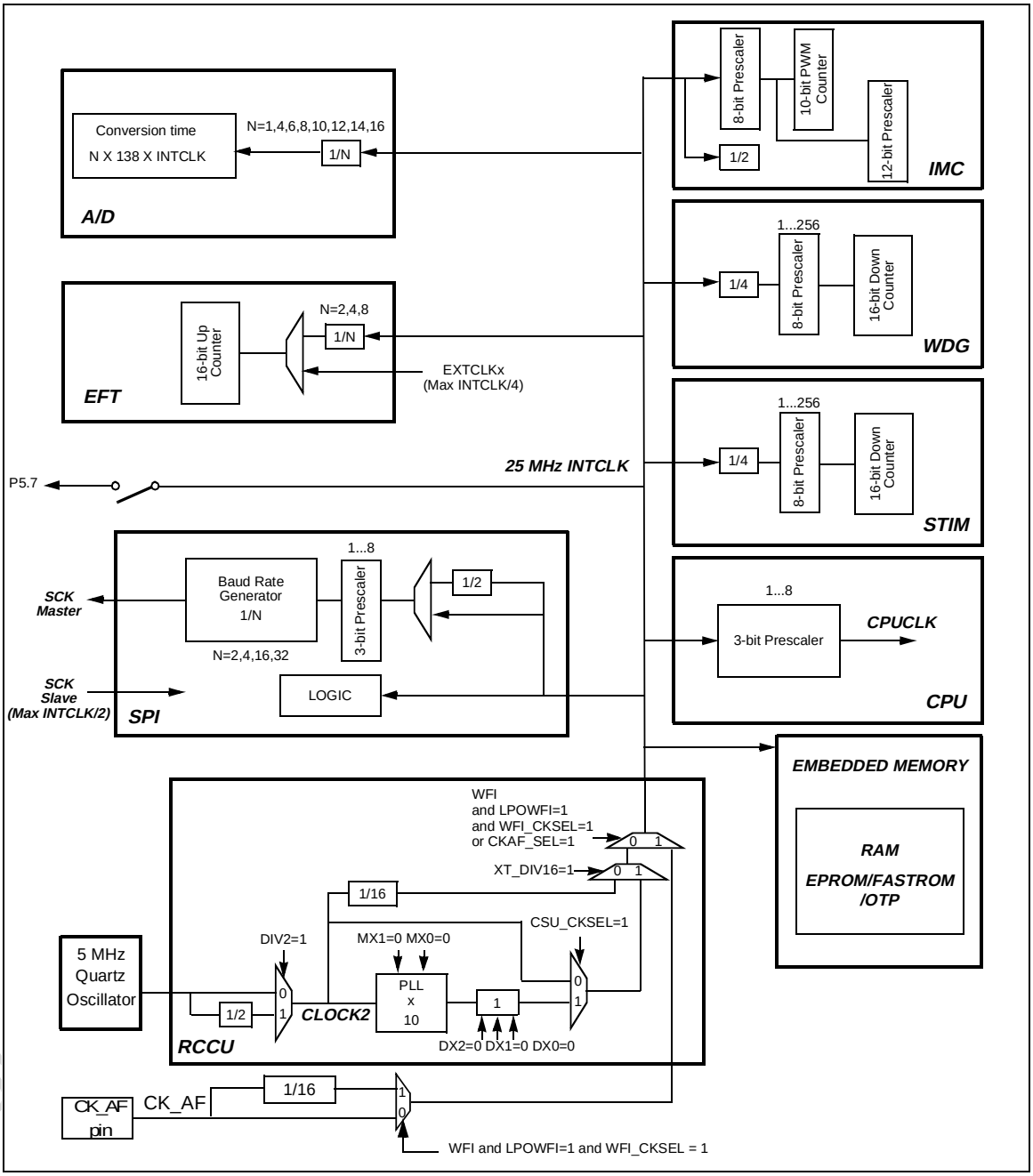


Figure 27. ST92141 Clock Distribution Diagram (settings given for 5MHz crystal & 25MHz Internal clock)



5.3 CLOCK MANAGEMENT

The various programmable features and operating modes of the CCU are handled by four registers:

– **MODER** (Mode Register)
This is a System Register (R235, Group E).

The input clock divide-by-two and the CPU clock prescaler factors are handled by this register.

– **CLKCTL** (Clock Control Register)
This is a Paged Register (R240, Page 55).

The low power modes and the interpretation of the HALT instruction are handled by this register.

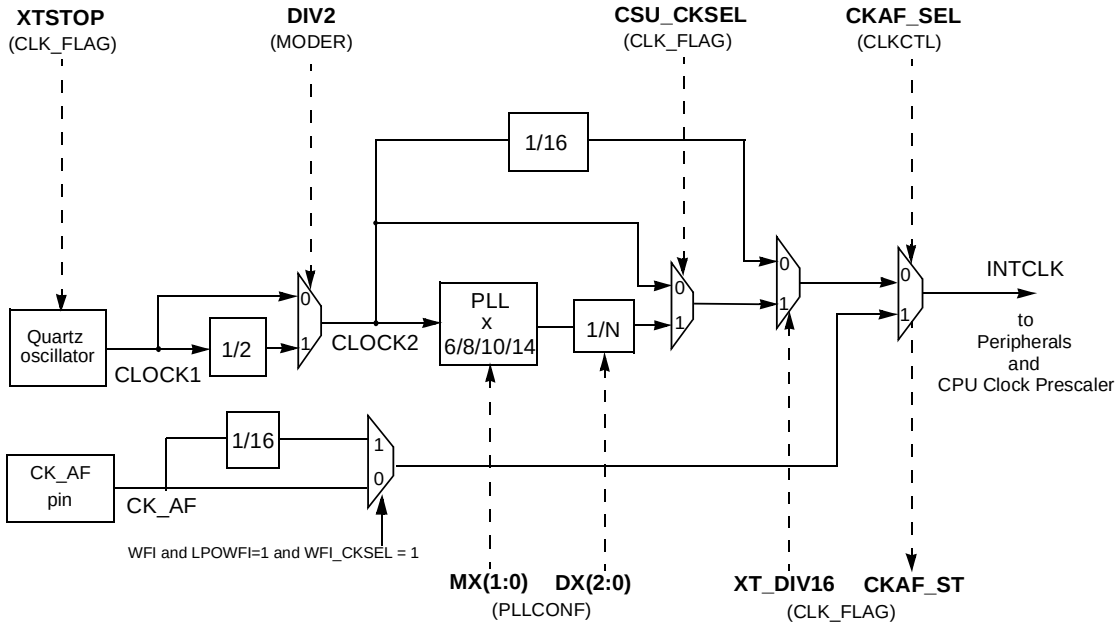
– **CLK_FLAG** (Clock Flag Register)
This is a Paged Register (R242, Page 55).

This register contains various status flags, as well as control bits for clock selection.

– **PLLCONF** (PLL Configuration Register)
This is a Paged Register (R246, Page 55).

The PLL multiplication and division factors are programmed in this register.

Figure 28. Clock Control Unit Programming



Wait for Interrupt and Low Power Modes:

LPOWFI (CLKCTL) selects Low Power operation automatically on entering WFI mode.
WFI_CKSEL (CLKCTL) selects the CK_AF clock automatically, if present, on entering WFI mode.
XTSTOP (CLK_FLAG) automatically stops the Xtal oscillator when the CK_AF clock is present and selected.

CLOCK MANAGEMENT (Cont'd)

5.3.1 PLL Clock Multiplier Programming

The CLOCK1 signal generated by the oscillator drives a programmable divide-by-two circuit. If the DIV2 control bit in MODER is set (Reset Condition), CLOCK2, is equal to CLOCK1 divided by two; if DIV2 is reset, CLOCK2 is identical to CLOCK1. Since the input clock to the Clock Multiplier circuit requires a 50% duty cycle for correct PLL operation, the divide by two circuit should be enabled when a crystal oscillator is used, or when the external clock generator does not provide a 50% duty cycle. In practice, the divide-by-two is virtually always used in order to ensure a 50% duty cycle signal to the PLL multiplier circuit. A CLOCK1 signal with a semiperiod (high or low) shorter than 40ns is forbidden if the divider by two is disabled.

When the PLL is active, it multiplies CLOCK2 by 6, 8, 10 or 14, depending on the status of the MX0 -1 bits in PLLCONF. The multiplied clock is then divided by a factor in the range 1 to 7, determined by the status of the DX0-2 bits; when these bits are programmed to 111, the PLL is switched off.

Following a RESET phase, programming bits DX0-2 to a value different from 111 will turn the PLL on. After allowing a stabilisation period for the PLL, setting the CSU_CKSEL bit in the CLK_FLAG Register selects the multiplier clock This peripheral contains a lock-in logic that verifies if the PLL is locked to the CLOCK2 frequency. The bit LOCK in CLK_FLAG register becomes 1 when this event occurs.

The maximum frequency allowed for INTCLK is 25MHz for 5V operation, and 12MHz for 3V operation. Care is required, when programming the PLL multiplier and divider factors, not to exceed the maximum permissible operating frequency for INTCLK, according to supply voltage.

The ST9 being a static machine, there is no lower limit for INTCLK. However, below 1MHz, A/D converter precision (if present) decreases.

5.3.2 CPU Clock Prescaling

The system clock, INTCLK, which may be the output of the PLL clock multiplier, CLOCK2, CLOCK2/16 or CK_AF, drives a programmable prescaler which generates the basic time base, CPUCLK, for the instruction executer of the ST9 CPU core. This allows the user to slow down program execution during non processor intensive routines, thus reducing power dissipation.

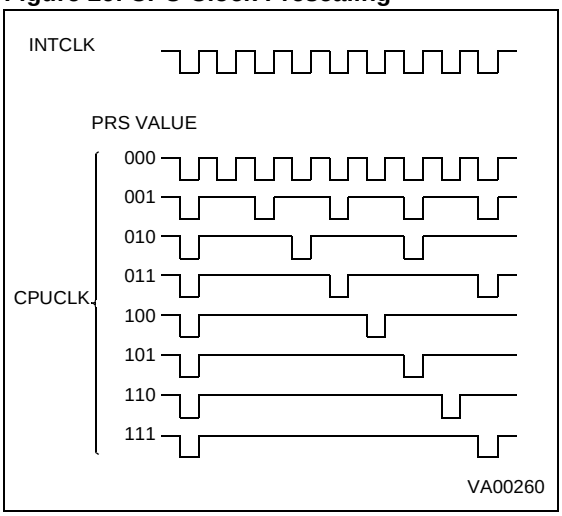
The internal peripherals are not affected by the CPUCLK prescaler and continue to operate at the full INTCLK frequency. This is particularly useful when little processing is being done and the peripherals are doing most of the work.

The prescaler divides the input clock by the value programmed in the control bits PRS2,1,0 in the MODER register. If the prescaler value is zero, no prescaling takes place, thus CPUCLK has the same period and phase as INTCLK. If the value is different from 0, the prescaling is equal to the value plus one, ranging thus from two (PRS2,1,0 = 1) to eight (PRS2,1,0 = 7).

The clock generated is shown in Figure 29, and it will be noted that the prescaling of the clock does not preserve the 50% duty cycle, since the high level is stretched to replace the missing cycles.

This is analogous to the introduction of wait cycles for access to external memory. When External Memory Wait or Bus Request events occur, CPUCLK is stretched at the high level for the whole period required by the function.

Figure 29. CPU Clock Prescaling



5.3.3 Peripheral Clock

The system clock, INTCLK, which may be the output of the PLL clock multiplier, CLOCK2, CLOCK2/16 or CK_AF, is also routed to all ST9 on-chip peripherals and acts as the central timebase for all timing functions.

CLOCK MANAGEMENT (Cont'd)

5.3.4 Low Power Modes

The user can select an automatic slowdown of clock frequency during Wait for Interrupt operation, thus idling in low power mode while waiting for an interrupt. In WFI operation the clock to the CPU core (CPUCLK) is stopped, thus suspending program execution, while the clock to the peripherals (INTCLK) may be programmed as described in the following paragraphs. An example of Low Power operation in WFI is illustrated in Figure 30.

If low power operation during WFI is disabled (LPOWFI bit = 0 in the CLKCTL Register), the CPU CLK is stopped but INTCLK is unchanged.

If low power operation during Wait for Interrupt is enabled (LPOWFI bit = 1 in the CLKCTL Register), as soon as the CPU executes the WFI instruction, the PLL is turned off and the system clock will be forced to CLOCK2 divided by 16, or to the external low frequency clock, CK_AF divided by 16 if this has been selected by setting WFI_CKSEL, and providing CKAF_ST is set, thus indicating that the external clock is selected and actually present on the CK_AF pin. The division by 16 is only selected by Hardware after entering Low Power WFI mode.

If the external clock source is used, the crystal oscillator may be stopped by setting the XTSTOP bit, providing that the CK_AF clock is present and selected, indicated by CKAF_ST being set. The crystal oscillator will be stopped automatically on entering WFI if the WFI_CKSEL bit has been set. It should be noted that selecting a non-existent CK_AF clock source is impossible, since such a selection requires that the auxiliary clock source be actually present and selected. In no event can a non-existent clock source be selected inadvertently.

It is up to the user program to switch back to a faster clock on the occurrence of an interrupt, taking care to respect the oscillator and PLL stabilisation delays, as appropriate.It should be noted that any of the low power modes may also be selected explicitly by the user program even when not in Wait for Interrupt mode, by setting the appropriate bits.

5.3.5 Interrupt Generation

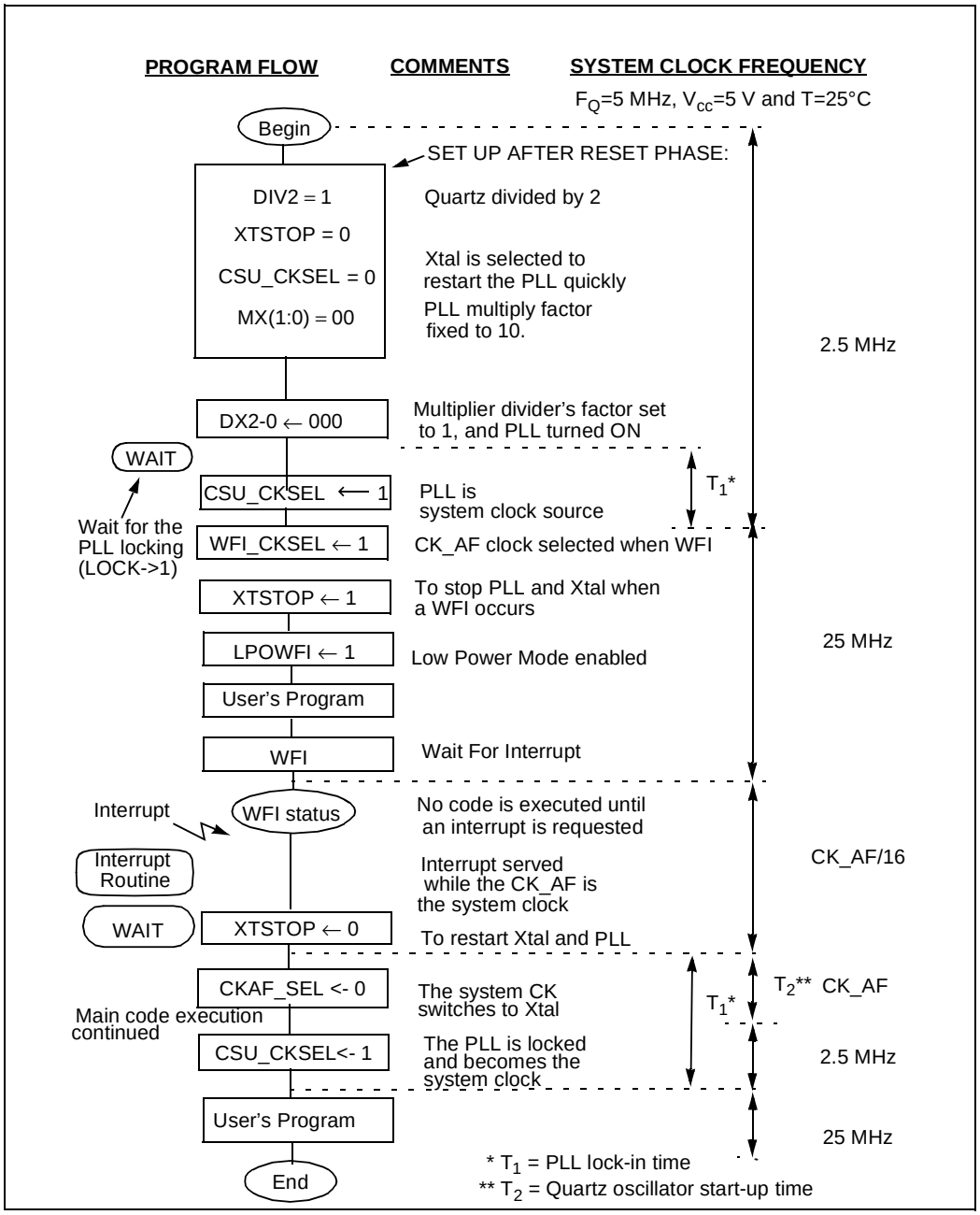
System clock selection modifies the CLKCTL and CLK_FLAG registers.

The clock control unit generates an external interrupt request when CK_AF and CLOCK2/16 are selected or deselected as system clock source, as well as when the system clock restarts after a stop request (when the STOP MODE feature is available on the specific device). This interrupt can be masked by resetting the INT_SEL bit in the CLKCTL register. In the RCCU the interrupt is generated with a high to low transition (see interrupt and DMA chapters for further information).

Table 14. Summary of Operating Modes using main Crystal Controlled Oscillator

MODE	INTCLK	CPUCLK	DIV2	PRS0-2	CSU_CKSEL	MX0-1	DX2-0	LPOWFI	XT_DIV16
PLL x BY 14	XTAL/2 x (14/D)	INTCLK/N	1	N-1	1	1 0	D-1	X	1
PLL x BY 10	XTAL/2 x (10/D)	INTCLK/N	1	N-1	1	0 0	D-1	X	1
PLL x BY 8	XTAL/2 x (8/D)	INTCLK/N	1	N-1	1	1 1	D-1	X	1
PLL x BY 6	XTAL/2 x (6/D)	INTCLK/N	1	N-1	1	0 1	D-1	X	1
SLOW 1	XTAL/2	INTCLK/N	1	N-1	X	X	111	X	1
SLOW 2	XTAL/32	INTCLK/N	1	N-1	X	X	X	X	0
WAIT FOR INTERRUPT	If LPOWFI=0, no changes occur on INTCLK, but CPUCLK is stopped anyway.								
LOW POWER WAIT FOR INTERRUPT	XTAL/32	STOP	1	X	X	X	X	1	1
RESET	XTAL/2	INTCLK	1	0	0	00	111	0	1
EXAMPLE XTAL=4.4 MHz	2.2*10/2 = 11MHz	11MHz	1	0	1	00	001	X	1

Figure 30. Example of Low Power Mode programming



5.4 CLOCK CONTROL REGISTERS

MODE REGISTER (MODER)

R235 - Read/Write
System Register
Reset Value: 1110 0000 (E0h)

7							0
-	-	DIV2	PRS2	PRS1	PRS0	-	-

***Note**: *This register contains bits which relate to other functions; these are described in the chapter dealing with Device Architecture. Only those bits relating to Clock functions are described here.*

Bit 5 = **DIV2**: *OSCIN Divided by 2*.
This bit controls the divide by 2 circuit which operates on the OSCIN Clock.
0: No division of the OSCIN Clock
1: OSCIN clock is internally divided by 2

Bit 4:2 = **PRS[2:0]**: *Clock Prescaling*.
These bits define the prescaler value used to prescale CPUCLK from INTCLK. When these three bits are reset, the CPUCLK is not prescaled, and is equal to INTCLK; in all other cases, the internal clock is prescaled by the value of these three bits plus one.

CLOCK CONTROL REGISTER (CLKCTL)

R240 - Read Write
Register Page: 55

Reset Value: 0000 0000 (00h)

7							0
INT_SEL	-	-	-	SRESEN	CKAF_SEL	WFI_CKSEL	LPOWFI

Bit 7 = **INT_SEL**: *Interrupt Selection*.
0: Select the external interrupt pin as interrupt source (Reset state)
1: Select the internal RCCU interrupt (see Section 5.3.5)

Bit 4:6 = **Reserved.**
Must be kept reset for normal operation.

Bit 3 = **SRESEN**: *Software Reset Enable.*
0: The HALT instruction turns off the quartz, the PLL and the CCU
1: A Reset is generated when HALT is executed

Bit 2 = **CKAF_SEL**: *Alternate Function Clock Select.*
0: CK_AF clock not selected
1: Select CK_AF clock

Note: To check if the selection has actually occurred, check that CKAF_ST is set. If no clock is present on the CK_AF pin, the selection will not occur.

Bit 1 = **WFI_CKSEL**: *WFI Clock Select*.
This bit selects the clock used during Low power WFI mode if LPOWFI = 1.
0: INTCLK during WFI is CLOCK2/16
1: INTCLK during WFI is CK_AF, further divided by 16, providing it is present. In effect this bit sets CKAF_SEL in WFI mode

WARNING: When the CK_AF is selected as Low Power WFI clock but the XTAL is not turned off (R242.4 = 0), after exiting from the WFI, CK_AF will be still selected as system clock. In this case, reset the R240.2 bit to switch back to the XT.

Bit 0 = **LPOWFI**: *Low Power mode during Wait For Interrupt*.
0: Low Power mode during WFI disabled. When WFI is executed, the CPUCLK is stopped and INTCLK is unchanged
1: The ST9 enters Low Power mode when the WFI instruction is executed. The clock during this state depends on WFI_CKSEL

CLOCK CONTROL REGISTERS (Cont'd)

CLOCK FLAG REGISTER (CLK_FLAG)
R242 -Read/Write
Register Page: 55
Reset Value: 0100 1000 after a Watchdog Reset
Reset Value: 0010 1000 after a Software Reset
Reset Value: 0000 1000 after a Power-On Reset

7							0
EX_ STP	WDG RES	SOFT RES	XTSTOP	XT_ DIV16	CKAF_ ST	LOC K	CSU_ CKSEL

WARNING: If this register is accessed with a logical instruction, such as AND or OR, some bits may not be set as expected.

WARNING: If you select the CK_AF as system clock and turn off the oscillator (bits R240.2 and R242.4 at 1), and then switch back to the XT clock by resetting the R240.2 bit, you must wait for the oscillator to restart correctly.

Bit 7 = **EX_STP**: *Stop Mode flag*
This bit is set by hardware and cleared by software.
0: No Stop condition occurred
1: Stop condition occurred

Bit 6 = **WDGRES**: *Watchdog reset flag.*
This bit is read only.
0: No Watchdog reset occurred
1: Watchdog reset occurred

Bit 5 = **SOFTRES**: *Software Reset Flag.*
This bit is read only.
0: No software reset occurred
1: Software reset occurred (HALT instruction)

Bit 4 = **XTSTOP**: *Oscillator Stop Enable.*
0: Xtal oscillator stop disabled
1: The Xtal oscillator will be stopped as soon as the CK_AF clock is present and selected, whether this is done explicitly by the user program, or as a result of WFI, if WFI_CKSEL has previously been set to select the CK_AF clock during WFI.

WARNING: When the program writes '1' to the XTSTOP bit, it will still be read as 0 and is only set when the CK_AF clock is running (CKAF_ST=1). Take care, as any operation such as a subsequent AND with `1' or an OR with `0' to the XTSTOP bit will reset it and the oscillator will not be stopped even if CKAF_ST is subsequently set.

Bit 3 = **XT_DIV16**: *CLOCK/16 Selection*

This bit is set and cleared by software. An interrupt is generated when the bit is toggled.
0: CLOCK2/16 is selected and the PLL is off
1: The input is CLOCK2 (or the PLL output depending on the value of CSU_CKSEL)

WARNING: After this bit is modified from 0 to 1, take care that the PLL lock-in time has elapsed before setting the CSU_CKSEL bit.

Bit 2 = **CKAF_ST**: (Read Only)

If set, indicates that the alternate function clock has been selected. If no clock signal is present on the CK_AF pin, the selection will not occur. If reset, the PLL clock, CLOCK2 or CLOCK2/16 is selected (depending on bit 0).

Bit 1= **LOCK**: *PLL locked-in*
This bit is read only.
0: The PLL is turned off or not locked and cannot be selected as system clock source.
1: The PLL is locked

Bit 0 = **CSU_CKSEL**: *CSU Clock Select*
This bit is set and cleared by software. It also cleared by hardware when:

– bits DX[2:0] (PLLCONF) are set to 111;

– the quartz is stopped (by hardware or software);

– WFI is executed while the LPOWFI bit is set;

– the XT_DIV16 bit (CLK_FLAG) is forced to '0'.

This prevents the PLL, when not yet locked, from providing an irregular clock. Furthermore, a '0' stored in this bit speeds up the PLL's locking.

0: CLOCK2 provides the system clock
1: The PLL Multiplier provides the system clock.

NOTE: Setting the CKAF_SEL bit overrides any other clock selection. Resetting the XT_DIV16 bit overrides the CSU_CKSEL selection (see Figure

CLOCK CONTROL REGISTERS (Cont'd)

PLL CONFIGURATION REGISTER (PLLCONF)
R246 - Read/Write
Register Page: 55
Reset Value: xx00 x111

7							0
-	-	MX1	MX0	-	DX2	DX1	DX0

Bit 5:4 = **MX[1:0]**: *PLL Multiplication Factor.*
Refer to Table 15 PLL Multiplication Factors for multiplier settings.

Bit 2:0 = **DX[2:0]**: *PLL output clock divider factor.*
Refer to Table 16 Divider Configuration for divider settings.

Table 15. PLL Multiplication Factors

MX1	MX0	CLOCK2 x
1	0	14
0	0	10
1	1	8
0	1	6

Table 16. Divider Configuration

DX2	DX1	DX0	CK
0	0	0	PLL CLOCK/1
0	0	1	PLL CLOCK/2
0	1	0	PLL CLOCK/3
0	1	1	PLL CLOCK/4
1	0	0	PLL CLOCK/5
1	0	1	PLL CLOCK/6
1	1	0	PLL CLOCK/7
1	1	1	CLOCK2 (PLL OFF, Reset State)

Figure 31. RCCU General Timing

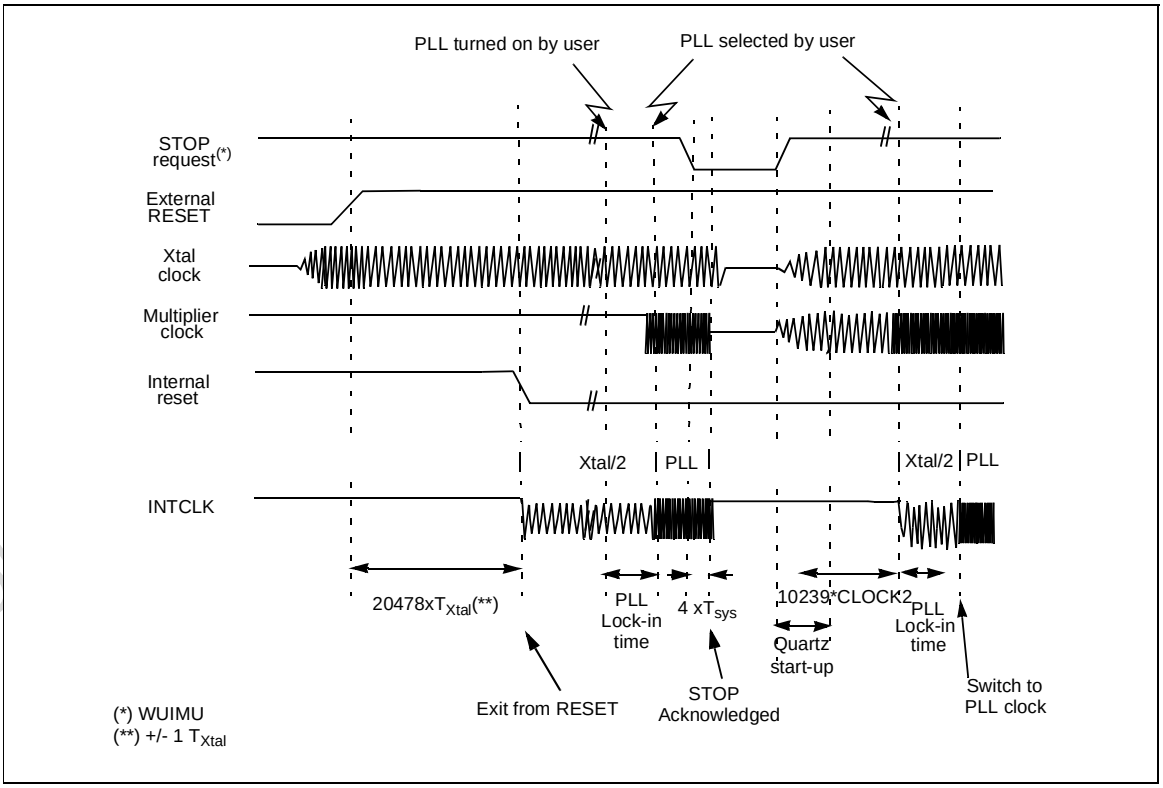


Figure 32. RCCU Timing during STOP (CK_AF System Clock)

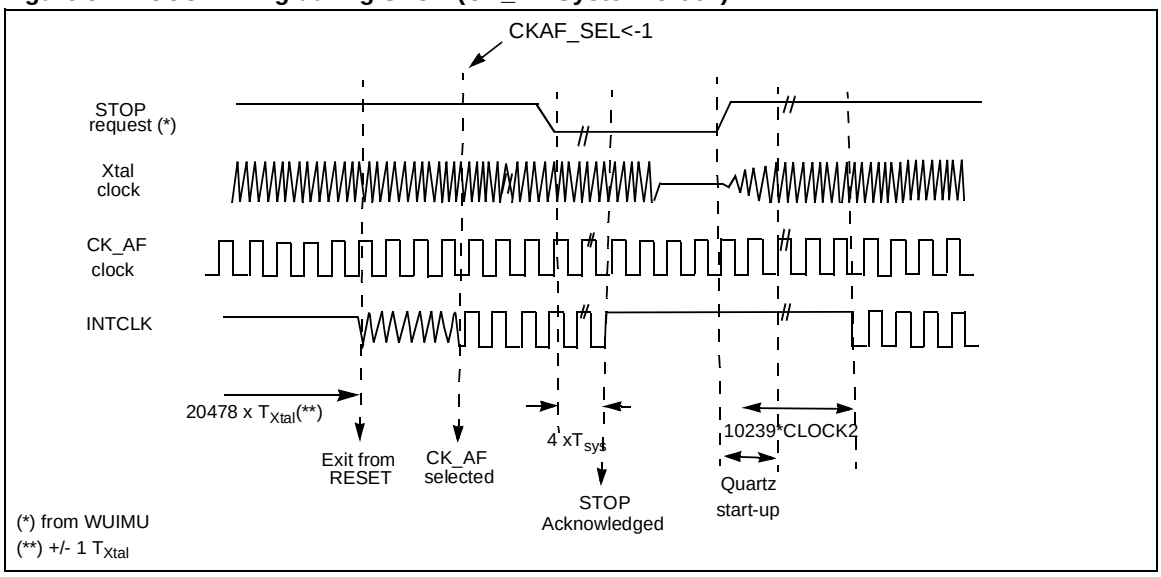
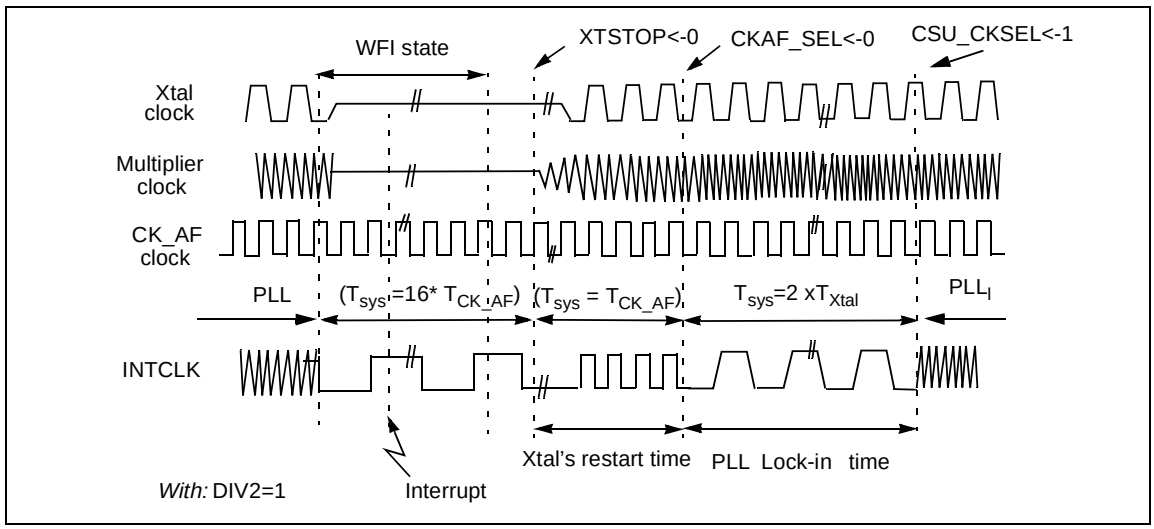


Figure 33. Low Power WFI Mode with a Stopped Quartz Oscillator



5.5 OSCILLATOR CHARACTERISTICS

The on-chip oscillator circuit uses an inverting gate circuit with tri-state output.

Notes: *Owing to the Q factor required, Ceramic Resonators may not provide a reliable oscillator source*.

OSCOUT must not be directly used to drive external circuits.

When the oscillator is stopped, OSCOUT goes high impedance.

The parallel resistor, R, is disconnected and the oscillator is disabled, forcing the internal clock, CLOCK1, to a high level, and OSCOUT to a high impedance state.

To exit the HALT condition and restart the oscillator, an external RESET pulse is required.

It should be noted that, if the Watchdog function is enabled, a HALT instruction will not disable the oscillator. This to avoid stopping the Watchdog if a HALT code is executed in error. When this occurs, the CPU will be reset when the Watchdog times out or when an external reset is applied.

Figure 34. Crystal Oscillator

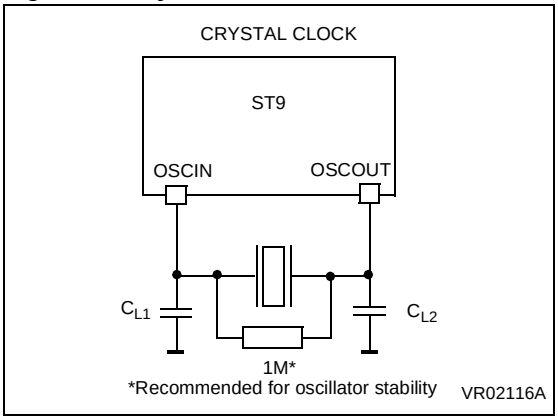


Table 17. Crystal Specification (5V)

Rs max (ohm)	C_{L1}=C_{L2}= 56pF	C_{L1}=C_{L2}= 47pF	C_{L1}=C_{L2}= 22pF
Freq.= 3 MHz	270	350	850
Freq.= 4 MHz	150	200	510
Freq.= 5 MHz	110	120	340

Legend:
C_{L1}, C_{L2}: Maximum Total Capacitances on pins OSCIN and OSCOUT (the value includes the external capacitance tied to the pin CL1 and CL2 plus the parasitic capacitance of the board and of the device).

Note: The tables are relative to the fundamental quartz crystal only (not ceramic resonator).

Figure 35. Internal Oscillator Schematic

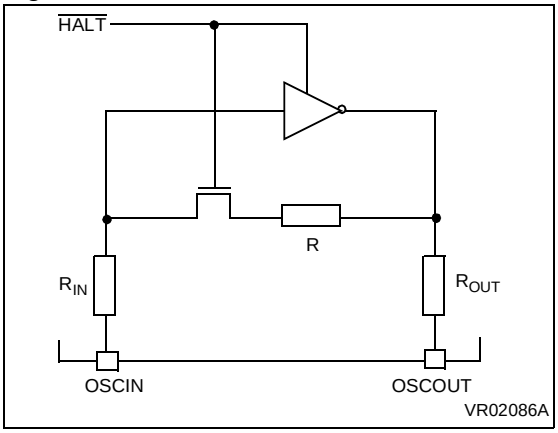
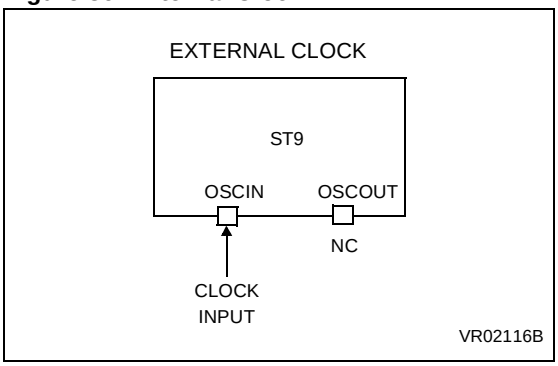


Figure 36. External Clock



5.6 RESET/STOP MANAGER

The Reset/Stop Manager resets the MCU when one of the three following events occurs:

– A Hardware reset, initiated by a low level on the Reset pin.

– A Software reset, initiated by a HALT instruction (when enabled).

– A Watchdog end of count condition.

The Low Voltage Detector (LVD) (see Section 5.8) generates a reset when:

– the power supply, when rising, is under the LVD V_{LVDR} Threshold.

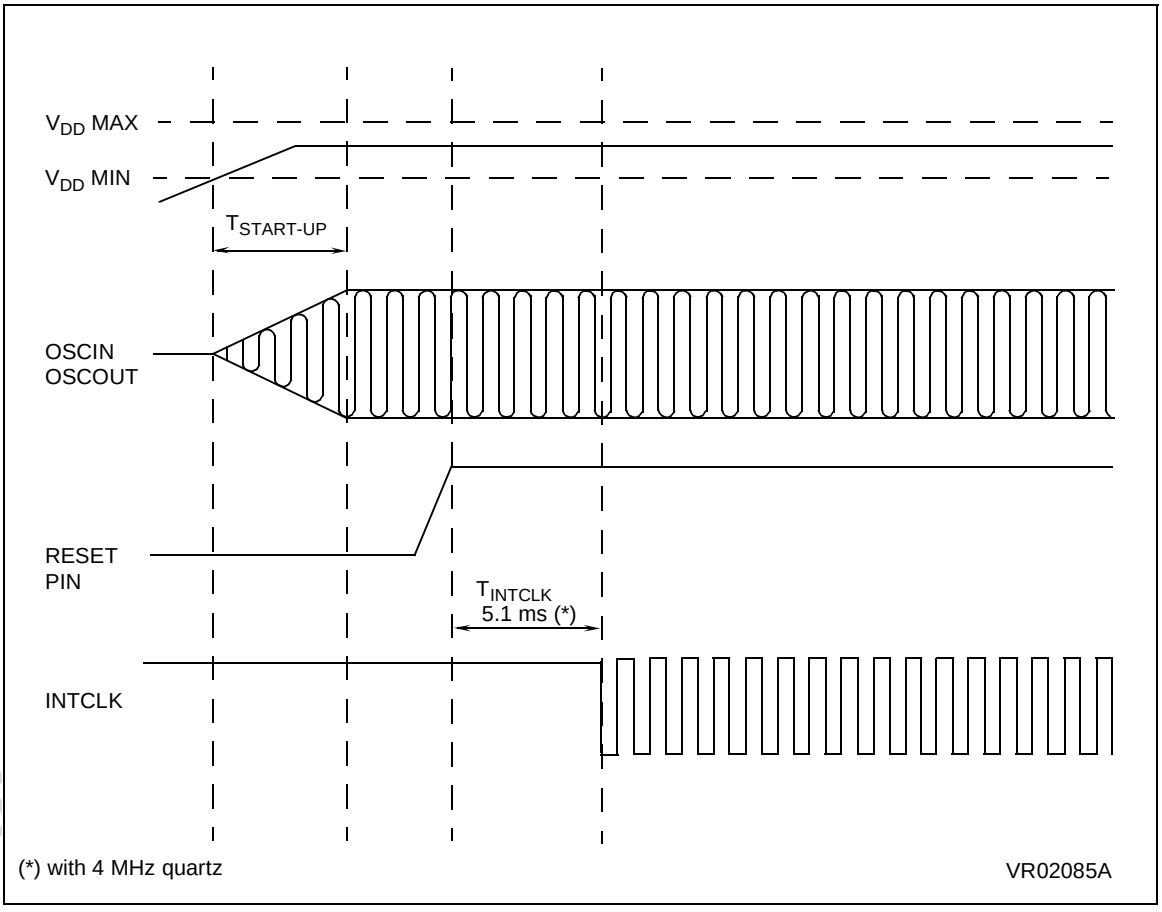
– the power supply, when falling, is under the LVD V_{LVDF} Threshold.

The event which caused the last Reset is flagged in the CLK_FLAG register, by setting the SOFTRES or the WDGRES bits respectively; a hardware initiated reset will leave both these bits reset.

The hardware reset overrides all other conditions and forces the ST9 to the reset state. During Reset, the internal registers are set to their reset values, where these are defined, and the I/O pins are set to the Bidirectional Weak Pull-up mode.

Reset is asynchronous: as soon as the reset pin is driven low, a Reset cycle is initiated.

Figure 37. Oscillator Start-up Sequence and Reset Timing



RESET/STOP MANAGER (Cont'd)

The on-chip Timer/Watchdog generates a reset condition if the Watchdog mode is enabled (WCR.WDEN cleared, R252 page 0), and if the programmed period elapses without the specific code (AAh, 55h) written to the appropriate register. The input pin RESET is not driven low by the on-chip reset generated by the Timer/Watchdog.

When the Reset pin goes high again, a delay occurs before exiting the Reset state. Subsequently a short Boot routine is executed from the device internal Boot ROM, and control then passes to the user program.

The Boot routine sets the device characteristics and loads the correct values in the Memory Management Unit's pointer registers, so that these point to the physical memory areas as mapped in the specific device. The precise duration of this short Boot routine varies from device to device, depending on the Boot ROM contents.

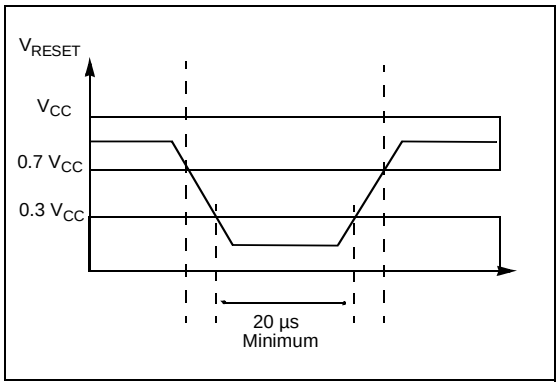
At the end of the Boot routine the Program Counter will be set to the location specified in the Reset Vector located in the lowest two bytes of memory.

5.6.1 Reset Pin Timing

To improve the noise immunity of the device, the Reset pin has a Schmitt trigger input circuit with hysteresis. In addition, a filter will prevent an unwanted reset in case of a single glitch of less than 50 ns on the Reset pin. The device is certain to reset if a negative pulse of more than 20µs is applied. When the reset pin goes high again, a delay of up to 4µs will elapse before the RCCU detects this rising front. From this event on, 20478 (about 5 ms with a 4MHz quartz) oscillator clock cycles (CLOCK1) are counted before exiting the Reset state (+-1 CLOCK1 period depending on the delay between the positive edge the RCCU detects and the first rising edge of CLOCK1)

If the ST9 is a ROMLESS version, without on-chip program memory, the memory interface ports are set to external memory mode (i.e Alternate Function) and the memory accesses are made to external Program memory with wait cycles insertion.

Figure 38. Recommended Signal to be Applied on Reset Pin



5.7 STOP MODE

Under control of the Wake-up Interrupt Management Unit (WUIMU), the Reset/Stop Manager can also stop all oscillators without resetting the device.

In Stop Mode all context information will be preserved. During this condition the internal clock will be frozen in the high state.

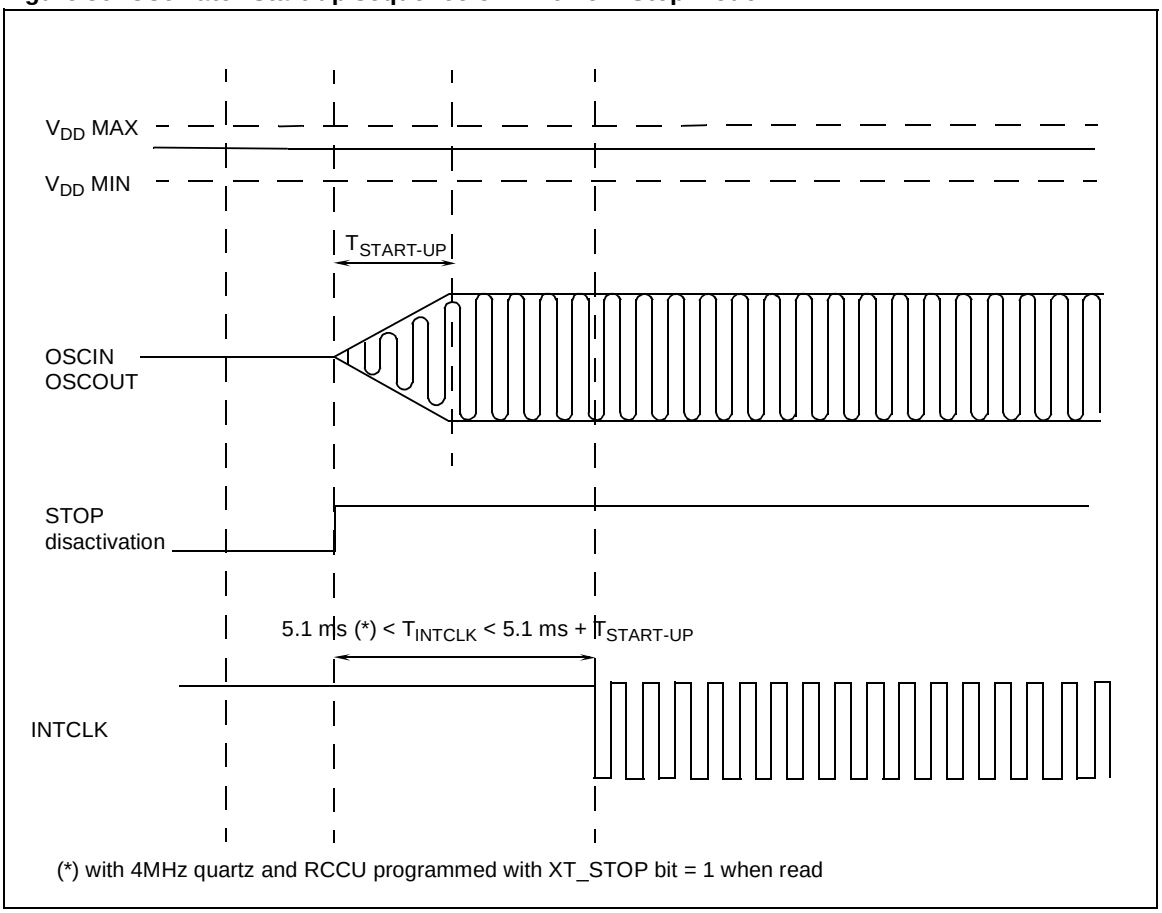
Stop Mode is entered by programming the WUIMU registers (See “WAKE-UP / INTERRUPT LINES MANAGEMENT UNIT (WUIMU)” on page 55.). An active transition on an External Wake Up line, exits the chip from Stop Mode and the MCU resumes execution after a delay of between 10239 CLOCK2 periods and 10239 CLOCK2 periods plus the Oscillator Start Up Time.

On exiting from Stop mode an interrupt is generated and the EX_STP bit in CLK_FLAG will be set, to indicate to the user program that the machine is exiting from Stop mode.

Table 18. Internal Registers Reset Values

Register Number	System Register	Reset Value	Page 0 Register	Reset Value
F	(SSPLR)	undefined	Reserved	
E	(SSPHR)	undefined	(SPICR)	00h
D	(USPLR)	undefined	(SPIDR)	undefined
C	(USPHR)	undefined	(WCR)	7Fh
B	(MODER)	E0h	(WDTCR)	12h
A	(Page Ptr)	undefined	(WDTPR)	undefined
9	(Reg Ptr 1)	undefined	(WDTLR)	undefined
8	(Reg Ptr 0)	undefined	(WDTHR)	undefined
7	(FLAGR)	undefined	(NICR)	00h
6	(CICR)	87h	(EIVR)	x2h
5	(PORT5)	FFh	(EIPLR)	FFh
4	(PORT4)	FFh	(EIMR)	00h
3	(PORT3)	FFh	(EIPR)	00h
2	(PORT2)	FFh	(EITR)	00h
1	(PORT1)	FFh	Reserved	
0	(PORT0)	FFh	Reserved	

Figure 39. Oscillator Start-up sequence on Exit from Stop Mode



(*) with 4MHz quartz and RCCU programmed with XT_STOP bit = 1 when read

5.8 LOW VOLTAGE DETECTOR (LVD)

To allow the integration of power management features in the application, the Low Voltage Detector function (LVD) generates a static reset when the V_{DD} supply voltage is below a V_{LVDf} reference value. This means that it secures the power-up as well as the power-down keeping the ST9 in reset.

The V_{LVDf} reference value for a voltage drop is lower than the V_{LVDr} reference value for power-on in order to avoid a parasitic reset when the MCU starts running and sinks current on the supply (hysteresis).

The LVD Reset circuitry generates a reset when V_{DD} is below:

– V_{LVDr} when V_{DD} is rising
– V_{LVDf} when V_{DD} is falling

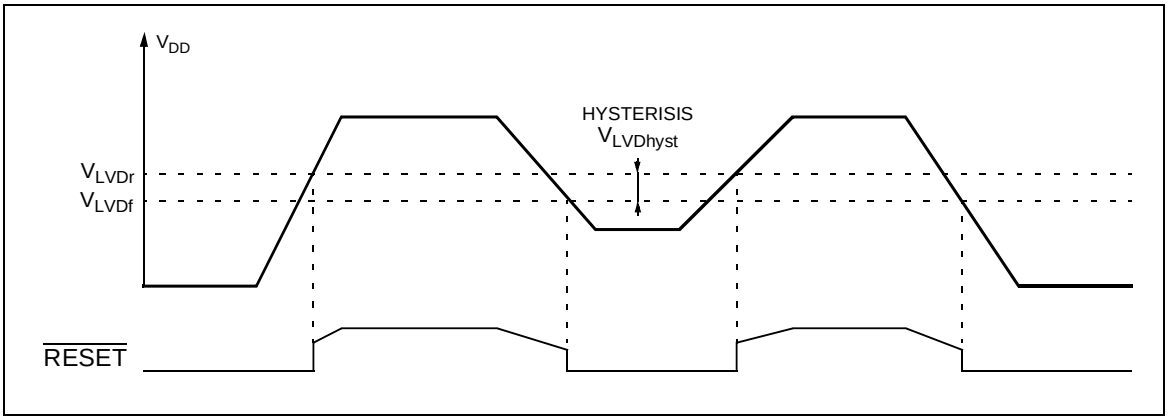
The LVD function is illustrated in Figure 40.

Provided the minimum V_{DD} value (guaranteed for the oscillator frequency) is below V_{LVDf}, the MCU can only be in two modes:

– under full software control
– in static safe reset

In these conditions, secure operation is always ensured for the application without the need for external reset hardware.

Figure 40. Low Voltage Detector vs Reset



6 I/O PORTS

6.1 INTRODUCTION

ST9 devices feature flexible individually programmable multifunctional input/output lines. Refer to the Pin Description Chapter for specific pin allocations. These lines, which are logically grouped as 8-bit ports, can be individually programmed to provide digital input/output and analog input, or to connect input/output signals to the on-chip peripherals as alternate pin functions. All ports can be individually configured as an input, bi-directional, output or alternate function. In addition, pull-ups can be turned off for open-drain operation, and weak pull-ups can be turned on in their place, to avoid the need for off-chip resistive pull-ups. Ports configured as open drain must never have voltage on the port pin exceeding V_{DD} (refer to the Electrical Characteristics section). Depending on the specific port, input buffers are software selectable to be TTL or CMOS compatible, however on Schmitt trigger ports, no selection is possible.

6.2 SPECIFIC PORT CONFIGURATIONS

Refer to the Pin Description chapter for a list of the specific port styles and reset values.

6.3 PORT CONTROL REGISTERS

Each port is associated with a Data register (PxDR) and three Control registers (PxC0, PxC1, PxC2). These define the port configuration and allow dynamic configuration changes during program execution. Port Data and Control registers are mapped into the Register File as shown in Figure 41. Port Data and Control registers are treated just like any other general purpose register. There are no special instructions for port manipulation: any instruction that can address a register, can address the ports. Data can be directly accessed in the port register, without passing through other memory or “accumulator” locations.

Figure 41. I/O Register Map

	GROUP E			GROUP F PAGE 2	GROUP F PAGE 3	GROUP F PAGE 43	
	System Registers		FFh	Reserved	P7DR	P9DR	R255
			FEh	P3C2	P7C2	P9C2	R254
			FDh	P3C1	P7C1	P9C1	R253
			FCh	P3C0	P7C0	P9C0	R252
			FBh	Reserved	P6DR	P8DR	R251
			FAh	P2C2	P6C2	P8C2	R250
			F9h	P2C1	P6C1	P8C1	R249
			F8h	P2C0	P6C0	P8C0	R248
			F7h	Reserved	Reserved		R247
			F6h	P1C2	P5C2		R246
E5h	P5DR	R229	F5h	P1C1	P5C1		R245
E4h	P4DR	R228	F4h	P1C0	P5C0	Reserved	R244
E3h	P3DR	R227	F3h	Reserved	Reserved		R243
E2h	P2DR	R226	F2h	P0C2	P4C2		R242
E1h	P1DR	R225	F1h	P0C1	P4C1		R241
E0h	P0DR	R224	F0h	P0C0	P4C0		R240

PORT CONTROL REGISTERS (Cont'd)

During Reset, ports with weak pull-ups are set in bidirectional/weak pull-up mode and the output Data Register is set to FFh. This condition is also held after Reset, except for Ports 0 and 1 in ROMless devices, and can be redefined under software control.

Bidirectional ports without weak pull-ups are set in high impedance during reset. To ensure proper levels during reset, these ports must be externally connected to either V_{DD} or V_{SS} through external pull-up or pull-down resistors.

Other reset conditions may apply in specific ST9 devices.

6.4 INPUT/OUTPUT BIT CONFIGURATION

By programming the control bits PxC0.n and PxC1.n (see Figure 42) it is possible to configure bit Px.n as Input, Output, Bidirectional or Alternate Function Output, where X is the number of the I/O port, and n the bit within the port (n = 0 to 7).

When programmed as input, it is possible to select the input level as TTL or CMOS compatible by programming the relevant PxC2.n control bit.
This option is not available on Schmitt trigger ports.

The output buffer can be programmed as push-pull or open-drain.

A weak pull-up configuration can be used to avoid external pull-ups when programmed as bidirectional (except where the weak pull-up option has been permanently disabled in the pin hardware assignment).

Each pin of an I/O port may assume software programmable Alternate Functions (refer to the device Pin Description and to Section 6.5 ALTERNATE FUNCTION ARCHITECTURE). To output signals from the ST9 peripherals, the port must be configured as AF OUT. On ST9 devices with A/D Converter(s), configure the ports used for analog inputs as AF IN.

The basic structure of the bit Px.n of a general purpose port Px is shown in Figure 43.

Independently of the chosen configuration, when the user addresses the port as the destination register of an instruction, the port is written to and the data is transferred from the internal Data Bus to the Output Master Latches. When the port is addressed as the source register of an instruction, the port is read and the data (stored in the Input Latch) is transferred to the internal Data Bus.

When Px.n is programmed as an Input:
(See Figure 44).

– The Output Buffer is forced tristate.

– The data present on the I/O pin is sampled into the Input Latch at the beginning of each instruction execution.

– The data stored in the Output Master Latch is copied into the Output Slave Latch at the end of the execution of each instruction. Thus, if bit Px.n is reconfigured as an Output or Bidirectional, the data stored in the Output Slave Latch will be reflected on the I/O pin.

INPUT/OUTPUT BIT CONFIGURATION (Cont'd)

Figure 42. Control Bits

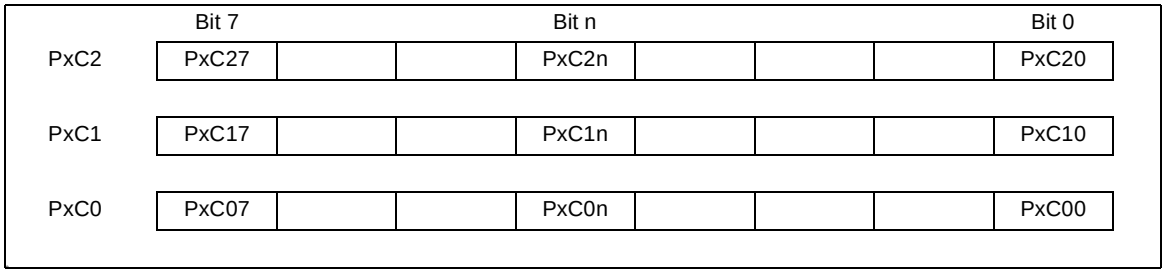


Table 19. Port Bit Configuration Table (n = 0, 1... 7; X = port number)

	General Purpose I/O Pins								A/D Pins
PXC2n PXC1n PXC0n	0 0 0	1 0 0	0 1 0	1 1 0	0 0 1	1 0 1	0 1 1	1 1 1	1 1 1
PXn Configuration	BID	BID	OUT	OUT	IN	IN	AF OUT	AF OUT	AF IN
PXn Output Type	WP OD	OD	PP	OD	HI-Z	HI-Z	PP	OD	HI-Z(1)
PXn Input Type	TTL (or Schmitt Trigger)	TTL (or Schmitt Trigger)	TTL (or Schmitt Trigger)	TTL (or Schmitt Trigger)	CMOS (or Schmitt Trigger)	TTL (or Schmitt Trigger)	TTL (or Schmitt Trigger)	TTL (or Schmitt Trigger)	Analog Input

(1) For A/D Converter inputs.

Legend:

X = Port
n = Bit
AF = Alternate Function
BID = Bidirectional
CMOS= CMOS Standard Input Levels
HI-Z = High Impedance
IN = Input
OD = Open Drain
OUT = Output
PP = Push-Pull
TTL = TTL Standard Input Levels
WP = Weak Pull-up

INPUT/OUTPUT BIT CONFIGURATION (Cont'd)

Figure 43. Basic Structure of an I/O Port Pin

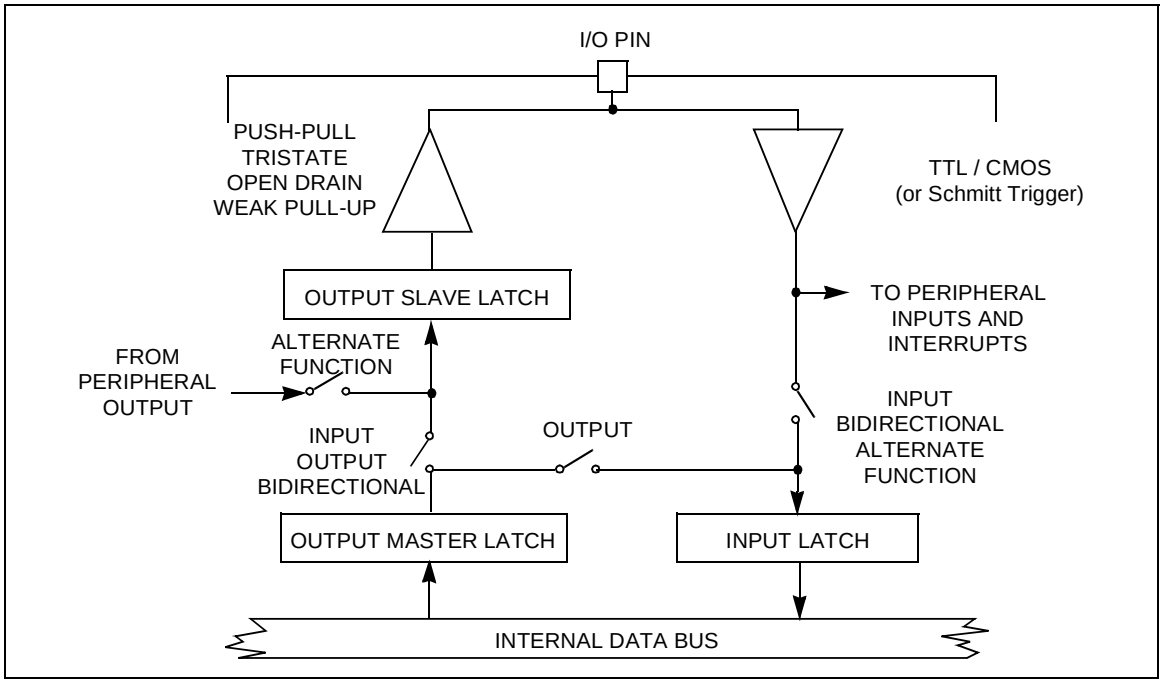


Figure 44. Input Configuration

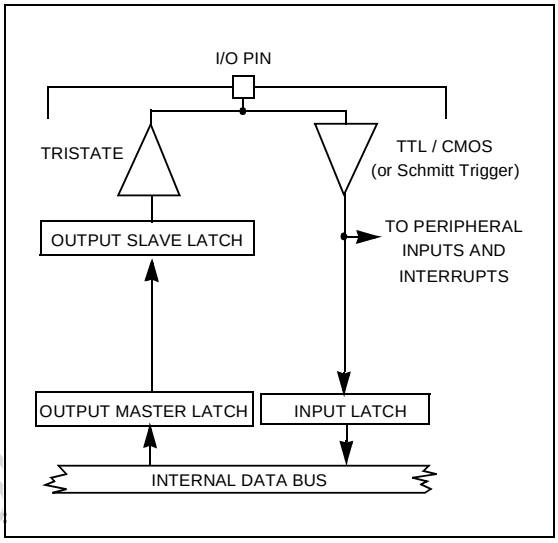
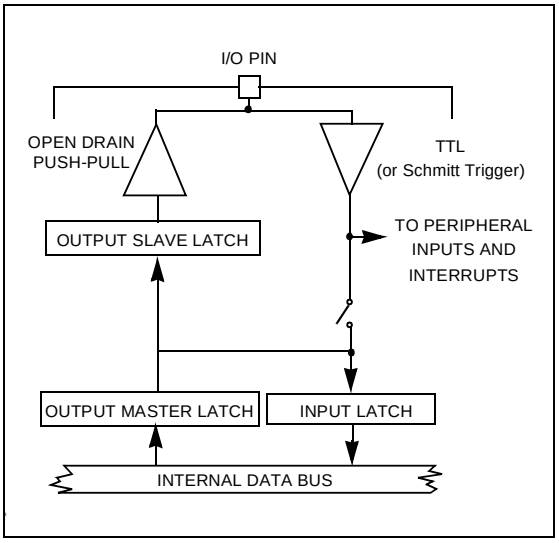


Figure 45. Output Configuration



INPUT/OUTPUT BIT CONFIGURATION (Cont'd)

When Px.n is programmed as an Output:
(Figure 45)

– The Output Buffer is turned on in an Open-drain or Push-pull configuration.

– The data stored in the Output Master Latch is copied both into the Input Latch and into the Output Slave Latch, driving the I/O pin, at the end of the execution of the instruction.

When Px.n is programmed as Bidirectional:
(Figure 46)

– The Output Buffer is turned on in an Open-Drain or Weak Pull-up configuration (except when disabled in hardware).

– The data present on the I/O pin is sampled into the Input Latch at the beginning of the execution of the instruction.

– The data stored in the Output Master Latch is copied into the Output Slave Latch, driving the I/O pin, at the end of the execution of the instruction.

WARNING: Due to the fact that in bidirectional mode the external pin is read instead of the output latch, particular care must be taken with arithmetic/logic and Boolean instructions performed on a bidirectional port pin.

These instructions use a read-modify-write sequence, and the result written in the port register depends on the logical level present on the external pin.

This may bring unwanted modifications to the port output register content.

For example:

Port register content, 0Fh
external port value, 03h
(Bits 3 and 2 are externally forced to 0)

A `bset` instruction on bit 7 will return:

Port register content, 83h
external port value, 83h
(Bits 3 and 2 have been cleared).

To avoid this situation, it is suggested that all operations on a port, using at least one bit in bidirectional mode, are performed on a copy of the port register, then transferring the result with a load instruction to the I/O port.

When Px.n is programmed as a digital Alternate Function Output:
(Figure 47)

– The Output Buffer is turned on in an Open-Drain or Push-Pull configuration.

– The data present on the I/O pin is sampled into the Input Latch at the beginning of the execution of the instruction.

– The signal from an on-chip function is allowed to load the Output Slave Latch driving the I/O pin. Signal timing is under control of the alternate function. If no alternate function is connected to Px.n, the I/O pin is driven to a high level when in Push-Pull configuration, and to a high impedance state when in open drain configuration.

Figure 46. Bidirectional Configuration

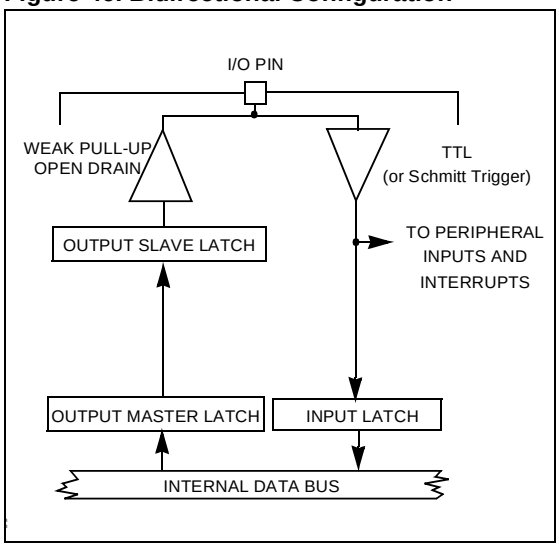
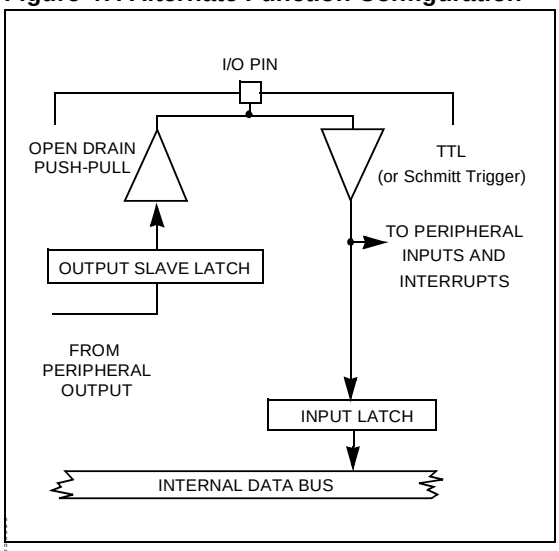


Figure 47. Alternate Function Configuration



6.5 ALTERNATE FUNCTION ARCHITECTURE

Each I/O pin may be connected to three different types of internal signal:

– Data bus Input/Output

– Alternate Function Input

– Alternate Function Output

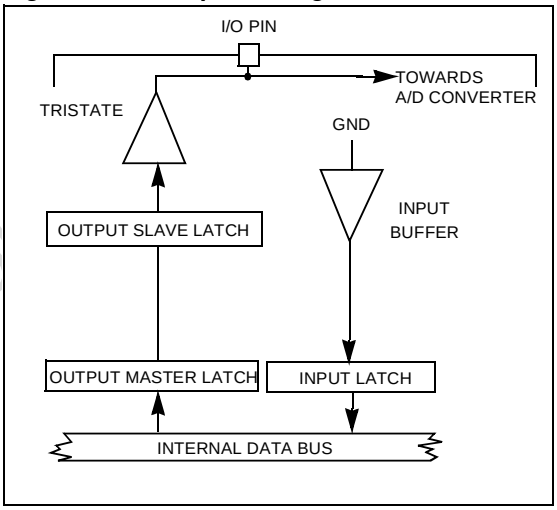
6.5.1 Pin Declared as I/O

A pin declared as I/O, is connected to the I/O buffer. This pin may be an Input, an Output, or a bidirectional I/O, depending on the value stored in (PxC2, PxC1 and PxC0).

6.5.2 Pin Declared as an Alternate Function Input

A single pin may be directly connected to several Alternate Function inputs. In this case, the user must select the required input mode (with the PxC2, PxC1, PxC0 bits) and enable the selected Alternate Function in the Control Register of the peripheral. No specific port configuration is required to enable an Alternate Function input, since the input buffer is directly connected to each alternate function module on the shared pin. As more than one module can use the same input, it is up to the user software to enable the required module as necessary. Parallel I/Os remain operational even when using an Alternate Function input. The exception to this is when an I/O port bit is permanently assigned by hardware as an A/D bit. In this case , after software programming of the bit in AF-OD-TTL, the Alternate function output is forced to logic level 1. The analog voltage level on the corresponding pin is directly input to the A/D (See Figure 48).

Figure 48. A/D Input Configuration



6.5.3 Pin Declared as an Alternate Function Output

The user must select the AF OUT configuration using the PxC2, PxC1, PxC0 bits. Several Alternate Function outputs may drive a common pin. In such case, the Alternate Function output signals are logically ANDed before driving the common pin. The user must therefore enable the required Alternate Function Output by software.

***WARNING**: When a pin is connected both to an alternate function output and to an alternate function input, it should be noted that the output signal will always be present on the alternate function input.*

6.6 I/O STATUS AFTER WFI, HALT AND RESET

The status of the I/O ports during the Wait For Interrupt, Halt and Reset operational modes is shown in the following table. The External Memory Interface ports are shown separately. If only the internal memory is being used and the ports are acting as I/O, the status is the same as shown for the other I/O ports.

Mode	Ext. Mem - I/O Ports		I/O Ports
	P0	P1, P2, P6, P9	
WFI	High Impedance or next address (depending on the last memory operation performed on Port)	Next Address	Not Affected (clock outputs running)
HALT	High Impedance	Next Address	Not Affected (clock outputs stopped)
RESET	Alternate function push-pull (ROMless device)		Bidirectional Weak Pull-up (High impedance when disabled in hardware).

7 ON-CHIP PERIPHERALS

7.1 TIMER/WATCHDOG (WDT)

Important Note: This chapter is a generic description of the WDT peripheral. However depending on the ST9 device, some or all of WDT interface signals described may not be connected to external pins. For the list of WDT pins present on the ST9 device, refer to the device pinout description in the first section of the data sheet.

7.1.1 Introduction

The Timer/Watchdog (WDT) peripheral consists of a programmable 16-bit timer and an 8-bit prescaler. It can be used, for example, to:

– Generate periodic interrupts
– Measure input signal pulse widths
– Request an interrupt after a set number of events
– Generate an output signal waveform
– Act as a Watchdog timer to monitor system integrity

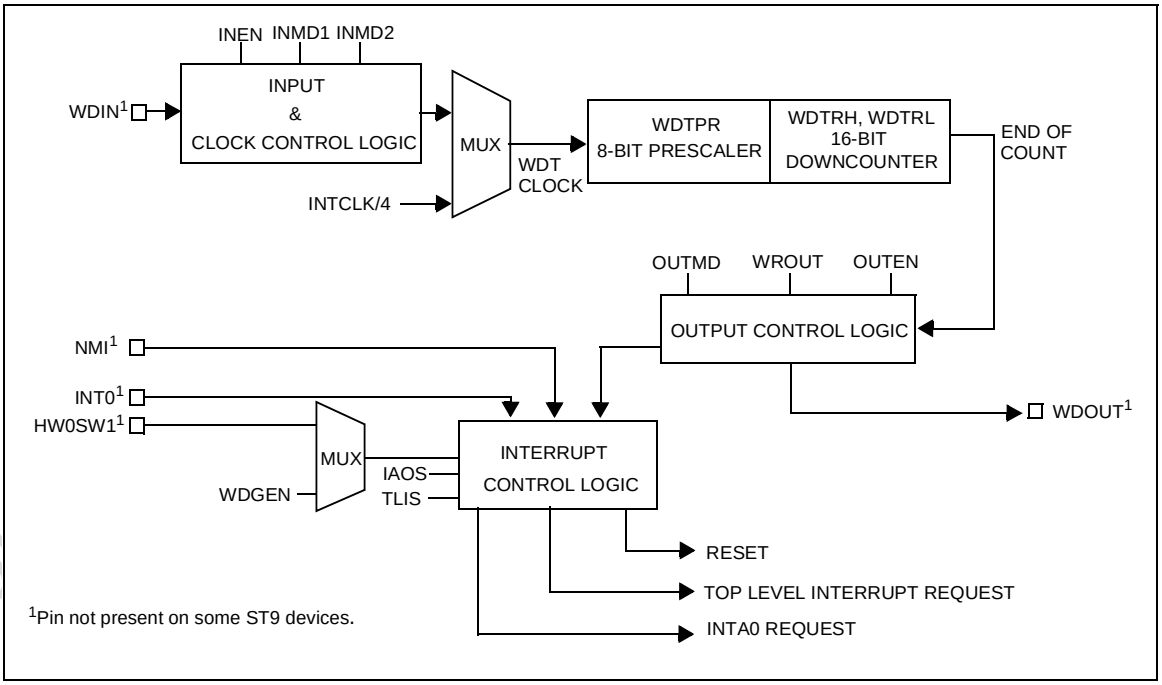
The main WDT registers are:

– Control register for the input, output and interrupt logic blocks (WDTCR)
– 16-bit counter register pair (WDTHR, WDTLR)
– Prescaler register (WDTPR)

The hardware interface consists of up to five signals:

– WDIN External clock input
– WDOUT Square wave or PWM signal output
– INT0 External interrupt input
– NMI Non-Maskable Interrupt input
– HW0SW1 Hardware/Software Watchdog enable.

Figure 49. Timer/Watchdog Block Diagram



TIMER/WATCHDOG (Cont'd)

7.1.2 Functional Description

7.1.2.1 External Signals

The HW0SW1 pin can be used to permanently enable Watchdog mode. Refer to section 7.1.3.1 on page 87.

The WDIN Input pin can be used in one of four modes:

– Event Counter Mode

– Gated External Input Mode

– Triggerable Input Mode

– Retriggerable Input Mode

The WDOUT output pin can be used to generate a square wave or a Pulse Width Modulated signal.

An interrupt, generated when the WDT is running as the 16-bit Timer/Counter, can be used as a Top Level Interrupt or as an interrupt source connected to channel A0 of the external interrupt structure (replacing the INT0 interrupt input).

The counter can be driven either by an external clock, or internally by INTCLK divided by 4.

7.1.2.2 Initialisation

The prescaler (WDTPR) and counter (WDTRL, WDTRH) registers must be loaded with initial values before starting the Timer/Counter. If this is not done, counting will start with reset values.

7.1.2.3 Start/Stop

The ST_SP bit enables downcounting. When this bit is set, the Timer will start at the beginning of the following instruction. Resetting this bit stops the counter.

If the counter is stopped and restarted, counting will resume from the last value unless a new constant has been entered in the Timer registers (WDTRL, WDTRH).

A new constant can be written in the WDTRH, WDTRL, WDTPR registers while the counter is running. The new value of the WDTRH, WDTRL registers will be loaded at the next End of Count (EOC) condition while the new value of the WDTPR register will be effective immediately.

End of Count is when the counter is 0.

When Watchdog mode is enabled the state of the ST_SP bit is irrelevant.

7.1.2.4 Single/Continuous Mode

The S_C bit allows selection of single or continuous mode.This Mode bit can be written with the Timer stopped or running. It is possible to toggle the S_C bit and start the counter with the same instruction.

Single Mode

On reaching the End Of Count condition, the Timer stops, reloads the constant, and resets the Start/Stop bit. Software can check the current status by reading this bit. To restart the Timer, set the Start/Stop bit.

Note: If the Timer constant has been modified during the stop period, it is reloaded at start time.

Continuous Mode

On reaching the End Of Count condition, the counter automatically reloads the constant and restarts. It is stopped only if the Start/Stop bit is reset.

7.1.2.5 Input Section

If the Timer/Counter input is enabled (INEN bit) it can count pulses input on the WDIN pin. Otherwise it counts the internal clock/4.

For instance, when INTCLK = 24MHz, the End Of Count rate is:

2.79 seconds for Maximum Count
(Timer Const. = FFFFh, Prescaler Const. = FFh)

166 ns for Minimum Count
(Timer Const. = 0000h, Prescaler Const. = 00h)

The Input pin can be used in one of four modes:

– Event Counter Mode

– Gated External Input Mode

– Triggerable Input Mode

– Retriggerable Input Mode

The mode is configurable in the WDTCR.

7.1.2.6 Event Counter Mode

In this mode the Timer is driven by the external clock applied to the input pin, thus operating as an event counter. The event is defined as a high to low transition of the input signal. Spacing between trailing edges should be at least 8 INTCLK periods (or 333ns with INTCLK = 24MHz).

Counting starts at the next input event after the ST_SP bit is set and stops when the ST_SP bit is reset.

TIMER/WATCHDOG (Cont'd)

7.1.2.7 Gated Input Mode

This mode can be used for pulse width measurement. The Timer is clocked by INTCLK/4, and is started and stopped by means of the input pin and the ST_SP bit. When the input pin is high, the Timer counts. When it is low, counting stops. The maximum input pin frequency is equivalent to INTCLK/8.

7.1.2.8 Triggerable Input Mode

The Timer (clocked internally by INTCLK/4) is started by the following sequence:

– setting the Start-Stop bit, followed by

– a High to Low transition on the input pin.

To stop the Timer, reset the ST_SP bit.

7.1.2.9 Retriggerable Input Mode

In this mode, the Timer (clocked internally by INTCLK/4) is started by setting the ST_SP bit. A High to Low transition on the input pin causes counting to restart from the initial value. When the Timer is stopped (ST_SP bit reset), a High to Low transition of the input pin has no effect.

7.1.2.10 Timer/Counter Output Modes

Output modes are selected by means of the OUTEN (Output Enable) and OUTMD (Output Mode) bits of the WDTCR register.

No Output Mode
(OUTEN = "0")

The output is disabled and the corresponding pin is set high, in order to allow other alternate functions to use the I/O pin.

Square Wave Output Mode
(OUTEN = "1", OUTMD = "0")

The Timer outputs a signal with a frequency equal to half the End of Count repetition rate on the WDOUT pin. With an INTCLK frequency of 20MHz, this allows a square wave signal to be generated whose period can range from 400ns to 6.7 seconds.

Pulse Width Modulated Output Mode
(OUTEN = "1", OUTMD = "1")

The state of the WROUT bit is transferred to the output pin (WDOUT) at the End of Count, and is held until the next End of Count condition. The user can thus generate PWM signals by modifying the status of the WROUT pin between End of Count events, based on software counters decremented by the Timer Watchdog interrupt.

7.1.3 Watchdog Timer Operation

This mode is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence of operation. The Watchdog, when enabled, resets the MCU, unless the program executes the correct write sequence before expiry of the programmed time period. The application program must be designed so as to correctly write to the WDTLR Watchdog register at regular intervals during all phases of normal operation.

7.1.3.1 Hardware Watchdog/Software Watchdog

The HW0SW1 pin (when available) selects Hardware Watchdog or Software Watchdog.

If HW0SW1 is held low:

– The Watchdog is enabled by hardware immediately after an external reset. (Note: Software reset or Watchdog reset have no effect on the Watchdog enable status).

– The initial counter value (FFFFh) cannot be modified, however software can change the prescaler value on the fly.

– The WDGEN bit has no effect. (Note: it is not forced low).

If HW0SW1 is held high, or is not present:

– The Watchdog can be enabled by resetting the WDGEN bit.

7.1.3.2 Starting the Watchdog

In Watchdog mode the Timer is clocked by INTCLK/4.

If the Watchdog is software enabled, the time base must be written in the timer registers before entering Watchdog mode by resetting the WDGEN bit. Once reset, this bit cannot be changed by software.

If the Watchdog is hardware enabled, the time base is fixed by the reset value of the registers.

Resetting WDGEN causes the counter to start, regardless of the value of the Start-Stop bit.

In Watchdog mode, only the Prescaler Constant may be modified.

If the End of Count condition is reached a System Reset is generated.

TIMER/WATCHDOG (Cont'd)

7.1.3.3 Preventing Watchdog System Reset

In order to prevent a system reset, the sequence AAh, 55h must be written to WDTLR (Watchdog Timer Low Register). Once 55h has been written, the Timer reloads the constant and counting re-starts from the preset value.

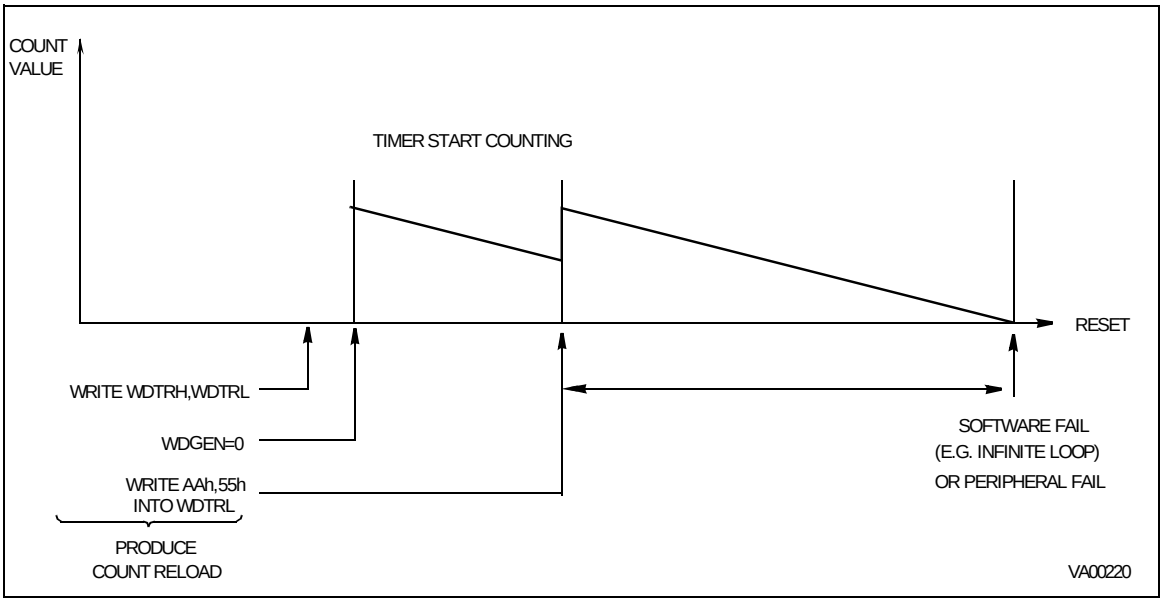
To reload the counter, the two writing operations must be performed sequentially without inserting other instructions that modify the value of the WDTLR register between the writing operations. The maximum allowed time between two reloads of the counter depends on the Watchdog timeout period.

7.1.3.4 Non-Stop Operation

In Watchdog Mode, a `Halt` instruction is regarded as illegal. Execution of the `Halt` instruction stops further execution by the CPU and interrupt acknowledgment, but does not stop INTCLK, CPUCLK or the Watchdog Timer, which will cause a System Reset when the End of Count condition is reached. Furthermore, ST_SP, S_C and the Input Mode selection bits are ignored. Hence, regardless of their status, the counter always runs in Continuous Mode, driven by the internal clock.

The Output mode should not be enabled, since in this context it is meaningless.

Figure 50. Watchdog Timer Mode



TIMER/WATCHDOG (Cont'd)

7.1.4 WDT Interrupts

The Timer/Watchdog issues an interrupt request at every End of Count, when this feature is enabled.

A pair of control bits, IA0S (EIVR.1, Interrupt A0 selection bit) and TLIS (EIVR.2, Top Level Input Selection bit) allow the selection of 2 interrupt sources (Timer/Watchdog End of Count, or External Pin) handled in two different ways, as a Top Level Non Maskable Interrupt (Software Reset), or as a source for channel A0 of the external interrupt logic.

A block diagram of the interrupt logic is given in Figure 51.

Note: Software traps can be generated by setting the appropriate interrupt pending bit.

Table 20 Interrupt Configuration below, shows all the possible configurations of interrupt/reset sources which relate to the Timer/Watchdog.

A reset caused by the watchdog will set bit 6, WDGRES of R242 - Page 55 (Clock Flag Register). See section CLOCK CONTROL REGISTERS.

Figure 51. Interrupt Sources

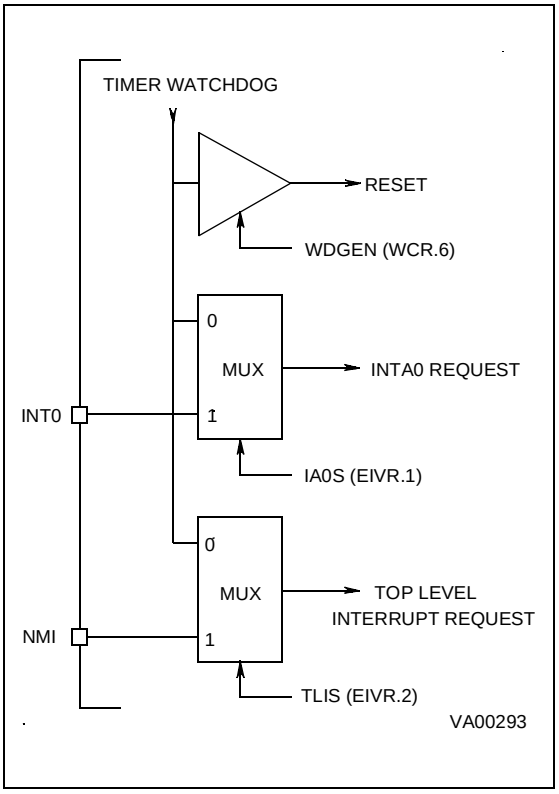


Table 20. Interrupt Configuration

Control Bits			Enabled Sources			Operating Mode
WDGEN	IA0S	TLIS	Reset	INTA0	Top Level	
0	0	0	WDG/Ext Reset	SW TRAP	SW TRAP	Watchdog
0	0	1	WDG/Ext Reset	SW TRAP	Ext Pin	Watchdog
0	1	0	WDG/Ext Reset	Ext Pin	SW TRAP	Watchdog
0	1	1	WDG/Ext Reset	Ext Pin	Ext Pin	Watchdog
1	0	0	Ext Reset	Timer	Timer	Timer
1	0	1	Ext Reset	Timer	Ext Pin	Timer
1	1	0	Ext Reset	Ext Pin	Timer	Timer
1	1	1	Ext Reset	Ext Pin	Ext Pin	Timer

Legend:
WDG = Watchdog function
SW TRAP = Software Trap

Note: If IA0S and TLIS = 0 (enabling the Watchdog EOC as interrupt source for both Top Level and INTA0 interrupts), only the INTA0 interrupt is taken into account.

TIMER/WATCHDOG (Cont'd)

7.1.5 Register Description

The Timer/Watchdog is associated with 4 registers mapped into Group F, Page 0 of the Register File.

WDTHR: Timer/Watchdog High Register

WDTLR: Timer/Watchdog Low Register

WDTPR: Timer/Watchdog Prescaler Register

WDTCR: Timer/Watchdog Control Register

Three additional control bits are mapped in the following registers on Page 0:

Watchdog Mode Enable, (WCR.6)

Top Level Interrupt Selection, (EIVR.2)

Interrupt A0 Channel Selection, (EIVR.1)

Note: The registers containing these bits also contain other functions. Only the bits relevant to the operation of the Timer/Watchdog are shown here.

Counter Register

This 16-bit register (WDTLR, WDTHR) is used to load the 16-bit counter value. The registers can be read or written "on the fly".

TIMER/WATCHDOG HIGH REGISTER (WDTHR)
R248 - Read/Write
Register Page: 0
Reset value: 1111 1111 (FFh)

7							0
R15	R14	R13	R12	R11	R10	R9	R8

Bits 7:0 = **R[15:8]** *Counter Most Significant Bits*.

TIMER/WATCHDOG LOW REGISTER (WDTLR)
R249 - Read/Write
Register Page: 0
Reset value: 1111 1111b (FFh)

7							0
R7	R6	R5	R4	R3	R2	R1	R0

Bits 7:0 = **R[7:0]** *Counter Least Significant Bits.*

TIMER/WATCHDOG PRESCALER REGISTER (WDTPR)
R250 - Read/Write
Register Page: 0
Reset value: 1111 1111 (FFh)

7							0
PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0

Bits 7:0 = **PR[7:0]** *Prescaler value.*
A programmable value from 1 (00h) to 256 (FFh).

Warning: *In order to prevent incorrect operation of the Timer/Watchdog, the prescaler (WDTPR) and counter (WDTRL, WDTRH) registers must be initialised before starting the Timer/Watchdog. If this is not done, counting will start with the reset (un-initialised) values.*

WATCHDOG TIMER CONTROL REGISTER (WDTCR)
R251- Read/Write
Register Page: 0
Reset value: 0001 0010 (12h)

7							0
ST_SP	S_C	INMD1	INMD2	INEN	OUTMD	WROUT	OUTEN

Bit 7 = **ST_SP**: *Start/Stop Bit*.
This bit is set and cleared by software.
0: Stop counting
1: Start counting (see Warning above)

Bit 6 = **S_C**: *Single/Continuous*.
This bit is set and cleared by software.
0: Continuous Mode
1: Single Mode

Bits 5:4 = **INMD[1:2]**: *Input mode selection bits*.
These bits select the input mode:

INMD1	INMD2	INPUT MODE
0	0	Event Counter
0	1	Gated Input (Reset value)
1	0	Triggerable Input
1	1	Retriggerable Input

TIMER/WATCHDOG (Cont'd)

Bit 3 = **INEN**: *Input Enable.*
This bit is set and cleared by software.
0: Disable input section
1: Enable input section

Bit 2 = **OUTMD**: *Output Mode.*
This bit is set and cleared by software.
0: The output is toggled at every End of Count
1: The value of the WROUT bit is transferred to the output pin on every End Of Count if OUTEN=1.

Bit 1 = **WROUT**: *Write Out.*
The status of this bit is transferred to the Output pin when OUTMD is set; it is user definable to allow PWM output (on Reset WROUT is set).

Bit 0 = **OUTEN**: *Output Enable bit.*
This bit is set and cleared by software.
0: Disable output
1: Enable output

WAIT CONTROL REGISTER (WCR)
R252 - Read/Write
Register Page: 0
Reset value: 0111 1111 (7Fh)

7							0
x	WDGEN	x	x	x	x	x	x

Bit 6 = **WDGEN**: *Watchdog Enable* (active low).
Resetting this bit via software enters the Watchdog mode. Once reset, it cannot be set anymore by the user program. At System Reset, the Watchdog mode is disabled.

Note: This bit is ignored if the Hardware Watchdog option is enabled by pin HW0SW1 (if available).

EXTERNAL INTERRUPT VECTOR REGISTER (EIVR)
R246 - Read/Write
Register Page: 0
Reset value: xxxx 0110 (x6h)

7							0
x	x	x	x	x	TLIS	IA0S	x

Bit 2 = **TLIS**: *Top Level Input Selection.*
This bit is set and cleared by software.
0: Watchdog End of Count is TL interrupt source
1: NMI is TL interrupt source

Bit 1 = **IA0S**: *Interrupt Channel A0 Selection.*
This bit is set and cleared by software.
0: Watchdog End of Count is INTA0 source
1: External Interrupt pin is INTA0 source

***Warning**: To avoid spurious interrupt requests, the IA0S bit should be accessed only when the interrupt logic is disabled (i.e. after the DI instruction). It is also necessary to clear any possible interrupt pending requests on channel A0 before enabling this interrupt channel. A delay instruction (e.g. a NOP instruction) must be inserted between the reset of the interrupt pending bit and the IA0S write instruction.*

Other bits are described in the Interrupt section.

7.2 STANDARD TIMER (STIM)

Important Note: This chapter is a generic description of the STIM peripheral. Depending on the ST9 device, some or all of the interface signals described may not be connected to external pins. For the list of STIM pins present on the particular ST9 device, refer to the pinout description in the first section of the data sheet.

7.2.1 Introduction

The Standard Timer includes a programmable 16-bit down counter and an associated 8-bit prescaler with Single and Continuous counting modes capability. The Standard Timer uses an input pin (STIN) and an output (STOUT) pin. These pins, when available, may be independent pins or connected as Alternate Functions of an I/O port bit.

STIN can be used in one of four programmable input modes:

– event counter,

– gated external input mode,

– triggerable input mode,

– retriggerable input mode.

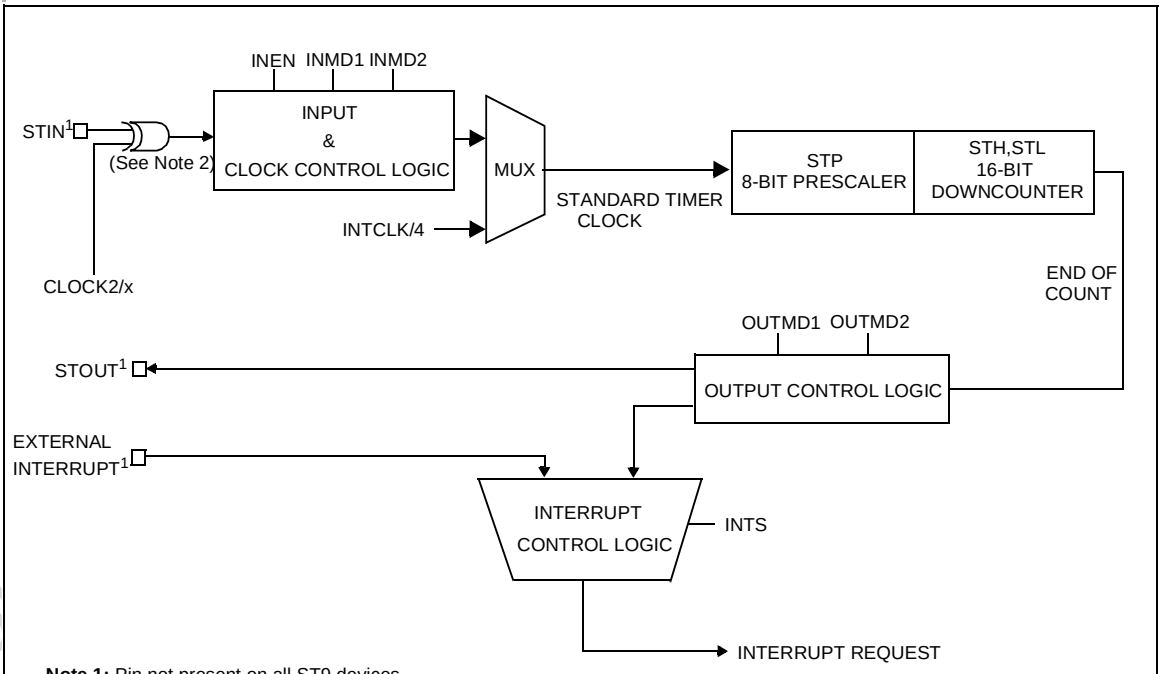
STOUT can be used to generate a Square Wave or Pulse Width Modulated signal.

The Standard Timer is composed of a 16-bit down counter with an 8-bit prescaler. The input clock to the prescaler can be driven either by an internal clock equal to INTCLK divided by 4, or by CLOCK2 derived directly from the external oscillator, divided by device dependent prescaler value, thus providing a stable time reference independent from the PLL programming or by an external clock connected to the STIN pin.

The Standard Timer End Of Count condition is able to generate an interrupt which is connected to one of the external interrupt channels.

The End of Count condition is defined as the Counter Underflow, whenever 00h is reached.

Figure 52. Standard Timer Block Diagram



Note 1: Pin not present on all ST9 devices.

Note 2: Depending on device, the source of the INPUT & CLOCK CONTROL LOGIC block may be permanently connected either to STIN or the RCCU signal CLOCK2/x. In devices without STIN and CLOCK2, the INEN bit must be held at 0.

STANDARD TIMER (Cont'd)

7.2.2 Functional Description

7.2.2.1 Timer/Counter control

Start-stop Count. The ST-SP bit (STC.7) is used in order to start and stop counting. An instruction which sets this bit will cause the Standard Timer to start counting at the beginning of the next instruction. Resetting this bit will stop the counter.

If the counter is stopped and restarted, counting will resume from the value held at the stop condition, unless a new constant has been entered in the Standard Timer registers during the stop period. In this case, the new constant will be loaded as soon as counting is restarted.

A new constant can be written in STH, STL, STP registers while the counter is running. The new value of the STH and STL registers will be loaded at the next End of Count condition, while the new value of the STP register will be loaded immediately.

WARNING: In order to prevent incorrect counting of the Standard Timer, the prescaler (STP) and counter (STL, STH) registers must be initialised before the starting of the timer. If this is not done, counting will start with the reset values (STH=FFh, STL=FFh, STP=FFh).

Single/Continuous Mode.
The S-C bit (STC.6) selects between the Single or Continuous mode.

SINGLE MODE: at the End of Count, the Standard Timer stops, reloads the constant and resets the Start/Stop bit (the user programmer can inspect the timer current status by reading this bit). Setting the Start/Stop bit will restart the counter.

CONTINUOUS MODE: At the End of the Count, the counter automatically reloads the constant and restarts. It is only stopped by resetting the Start/Stop bit.

The S-C bit can be written either with the timer stopped or running. It is possible to toggle the S-C bit and start the Standard Timer with the same instruction.

7.2.2.2 Standard Timer Input Modes (ST9 devices with Standard Timer Input STIN)

Bits INMD2, INMD1 and INEN are used to select the input modes. The Input Enable (INEN) bit enables the input mode selected by the INMD2 and INMD1 bits. If the input is disabled (INEN="0"), the values of INMD2 and INMD1 are not taken into account. In this case, this unit acts as a 16-bit timer (plus prescaler) directly driven by INTCLK/4 and transitions on the input pin have no effect.

Event Counter Mode (INMD1 = "0", INMD2 = "0")

The Standard Timer is driven by the signal applied to the input pin (STIN) which acts as an external clock. The unit works therefore as an event counter. The event is a high to low transition on STIN. Spacing between trailing edges should be at least the period of INTCLK multiplied by 8 (i.e. the maximum Standard Timer input frequency is 3 MHz with INTCLK = 24MHz).

Gated Input Mode (INMD1 = "0", INMD2 = "1")

The Timer uses the internal clock (INTCLK divided by 4) and starts and stops the Timer according to the state of STIN pin. When the status of the STIN is High the Standard Timer count operation proceeds, and when Low, counting is stopped.

Triggerable Input Mode (INMD1 = "1", INMD2 = "0")

The Standard Timer is started by:

a) setting the Start-Stop bit, AND

b) a High to Low (low trigger) transition on STIN.

In order to stop the Standard Timer in this mode, it is only necessary to reset the Start-Stop bit.

Retriggerable Input Mode (INMD1 = "1", INMD2 = "1")

In this mode, when the Standard Timer is running (with internal clock), a High to Low transition on STIN causes the counting to start from the last constant loaded into the STL/STH and STP registers. When the Standard Timer is stopped (ST-SP bit equal to zero), a High to Low transition on STIN has no effect.

7.2.2.3 Time Base Generator (ST9 devices without Standard Timer Input STIN)

For devices where STIN is replaced by a connection to CLOCK2, the condition (INMD1 = "0", INMD2 = "0") will allow the Standard Timer to generate a stable time base independent from the PLL programming.

STANDARD TIMER (Cont'd)

7.2.2.4 Standard Timer Output Modes

OUTPUT modes are selected using 2 bits of the STC register: OUTMD1 and OUTMD2.

No Output Mode (OUTMD1 = "0", OUTMD2 = "0")

The output is disabled and the corresponding pin is set high, in order to allow other alternate functions to use the I/O pin.

Square Wave Output Mode (OUTMD1 = "0", OUTMD2 = "1")

The Standard Timer toggles the state of the STOUT pin on every End Of Count condition. With INTCLK = 24MHz, this allows generation of a square wave with a period ranging from 333ns to 5.59 seconds.

PWM Output Mode (OUTMD1 = "1")

The value of the OUTMD2 bit is transferred to the STOUT output pin at the End Of Count. This allows the user to generate PWM signals, by modifying the status of OUTMD2 between End of Count events, based on software counters decremented on the Standard Timer interrupt.

7.2.3 Interrupt Selection

The Standard Timer may generate an interrupt request at every End of Count.

Bit 2 of the STC register (INTS) selects the interrupt source between the Standard Timer interrupt and the external interrupt pin. Thus the Standard Timer Interrupt uses the interrupt channel and takes the priority and vector of the external interrupt channel.

If INTS is set to "1", the Standard Timer interrupt is disabled; otherwise, an interrupt request is generated at every End of Count.

Note: When enabling or disabling the Standard Timer Interrupt (writing INTS in the STC register) an edge may be generated on the interrupt channel, causing an unwanted interrupt.

To avoid this spurious interrupt request, the INTS bit should be accessed only when the interrupt logic is disabled (i.e. after the DI instruction). It is also necessary to clear any possible interrupt pending requests on the corresponding external interrupt channel before enabling it. A delay instruction (i.e. a NOP instruction) must be inserted between the reset of the interrupt pending bit and the INTS write instruction.

7.2.4 Register Mapping

Depending on the ST9 device there may be up to 4 Standard Timers (refer to the block diagram in the first section of the data sheet).

Each Standard Timer has 4 registers mapped into Page 11 in Group F of the Register File

In the register description on the following page, register addresses refer to STIM0 only.

STD Timer	Register	Register Address
STIM0	STH0	R240 (F0h)
	STL0	R241 (F1h)
	STP0	R242 (F2h)
	STC0	R243 (F3h)
STIM1	STH1	R244 (F4h)
	STL1	R245 (F5h)
	STP1	R246 (F6h)
	STC1	R247 (F7h)
STIM2	STH2	R248 (F8h)
	STL2	R249 (F9h)
	STP2	R250 (FAh)
	STC2	R251 (FBh)
STIM3	STH3	R252 (FCh)
	STL3	R253 (FDh)
	STP3	R254 (FEh)
	STC3	R255 (FFh)

Note: The four standard timers are not implemented on all ST9 devices. Refer to the block diagram of the device for the number of timers.

STANDARD TIMER (Cont'd)

7.2.5 Register Description

COUNTER HIGH BYTE REGISTER (STH)
R240 - Read/Write
Register Page: 11
Reset value: 1111 1111 (FFh)

7							0
ST.15	ST.14	ST.13	ST.12	ST.11	ST.10	ST.9	ST.8

Bits 7:0 = **ST.[15:8]**: *Counter High-Byte.*

COUNTER LOW BYTE REGISTER (STL)
R241 - Read/Write
Register Page: 11
Reset value: 1111 1111 (FFh)

7							0
ST.7	ST.6	ST.5	ST.4	ST.3	ST.2	ST.1	ST.0

Bits 7:0 = **ST.[7:0]**: *Counter Low Byte.*

Writing to the STH and STL registers allows the user to enter the Standard Timer constant, while reading it provides the counter's current value. Thus it is possible to read the counter on-the-fly.

STANDARD TIMER PRESCALER REGISTER (STP)
R242 - Read/Write
Register Page: 11
Reset value: 1111 1111 (FFh)

7							0
STP.7	STP.6	STP.5	STP.4	STP.3	STP.2	STP.1	STP.0

Bits 7:0 = **STP.[7:0]**: *Prescaler.*
The Prescaler value for the Standard Timer is programmed into this register. When reading the STP register, the returned value corresponds to the programmed data instead of the current data.
00h: No prescaler
01h: Divide by 2
FFh: Divide by 256

STANDARD TIMER CONTROL REGISTER (STC)
R243 - Read/Write
Register Page: 11
Reset value: 0001 0100 (14h)

7							0
ST-SP	S-C	INMD1	INMD2	INEN	INTS	OUTMD1	OUTMD2

Bit 7 = **ST-SP**: *Start-Stop Bit.*
This bit is set and cleared by software.
0: Stop counting
1: Start counting

Bit 6 = **S-C**: *Single-Continuous Mode Select.*
This bit is set and cleared by software.
0: Continuous Mode
1: Single Mode

Bits 5:4 = **INMD[1:2]**: *Input Mode Selection.*
These bits select the Input functions as shown in Section 7.2.2.2, when enabled by INEN.

INMD1	INMD2	Mode
0	**0**	Event Counter mode
0	**1**	Gated input mode
1	**0**	Triggerable mode
1	**1**	Retriggerable mode

Bit 3 = **INEN**: *Input Enable.*
This bit is set and cleared by software. If neither the STIN pin nor the CLOCK2 line are present, INEN must be 0.
0: Input section disabled
1: Input section enabled

Bit 2 = **INTS**: *Interrupt Selection.*
0: Standard Timer interrupt enabled
1: Standard Timer interrupt is disabled and the external interrupt pin is enabled.

Bits 1:0 = **OUTMD[1:2]**: Output Mode Selection.
These bits select the output functions as described in Section 7.2.2.4.

OUTMD1	OUTMD2	Mode
0	**0**	No output mode
0	**1**	Square wave output mode
1	**x**	PWM output mode

7.3 EXTENDED FUNCTION TIMER (EFT)

7.3.1 Introduction

The timer consists of a 16-bit free-running counter driven by a programmable prescaler.

It may be used for a variety of purposes, including pulse length measurement of up to two input signals (*input capture*) or generation of up to two output waveforms (*output compare* and *PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the INTCLK prescaler.

7.3.2 Main Features

- Programmable prescaler: INTCLK divided by 2, 4 or 8.
- Overflow status flag and maskable interrupts
- External clock input (must be at least 4 times slower than the INTCLK clock speed) with the choice of active edge
- Output compare functions with
  - 2 dedicated 16-bit registers
  - 2 dedicated programmable signals
  - 2 dedicated status flags
  - 1 dedicated maskable interrupt
- Input capture functions with
  - 2 dedicated 16-bit registers
  - 2 dedicated active edge selection signals
  - 2 dedicated status flags
  - 1 dedicated maskable interrupt
- Pulse width modulation mode (PWM)
- One pulse mode
- 5 alternate functions on I/O ports*
- Up to 3 separate Timer interrupts or a global interrupt (depending on device) mapped on external interrupt channels:
  - ICI: Timer Input capture interrupt.
  - OCI: Timer Output compare interrupt.
  - TOI: Timer Overflow interrupt.
  - EFTI: Timer Global interrupt (replaces ICI, OCI and TOI).

The Block Diagram is shown in Figure 53.

Table 21. EFT Pin Naming conventions

Function	EFT0	EFT1	EFTn
Input Capture 1 - ICAP1	ICAPA0	ICAPA1	ICAPAn
Input Capture 2 - ICAP2	ICAPB0	ICAPB1	ICAPBn
Output Compare 1 - OCMP1	OCMPA0	OCMPA1	OCMPAn
Output Compare 2 - OCMP2	OCMPB0	OCMPB1	OCMPBn

***Note 1:** Some external pins are not available on all devices. Refer to the device pin out description.

***Note 2:** Refer to the device interrupt description, to see if a single timer interrupt is used, or three separate interrupts.

7.3.3 Functional Description

7.3.3.1 Counter

The principal block of the Programmable Timer is a 16-bit free running counter and its associated 16-bit registers:

Counter Registers

- Counter High Register (CHR) is the most significant byte (MSB).
- Counter Low Register (CLR) is the least significant byte (LSB).

Alternate Counter Registers

- Alternate Counter High Register (ACHR) is the most significant byte (MSB).
- Alternate Counter Low Register (ACLR) is the least significant byte (LSB).

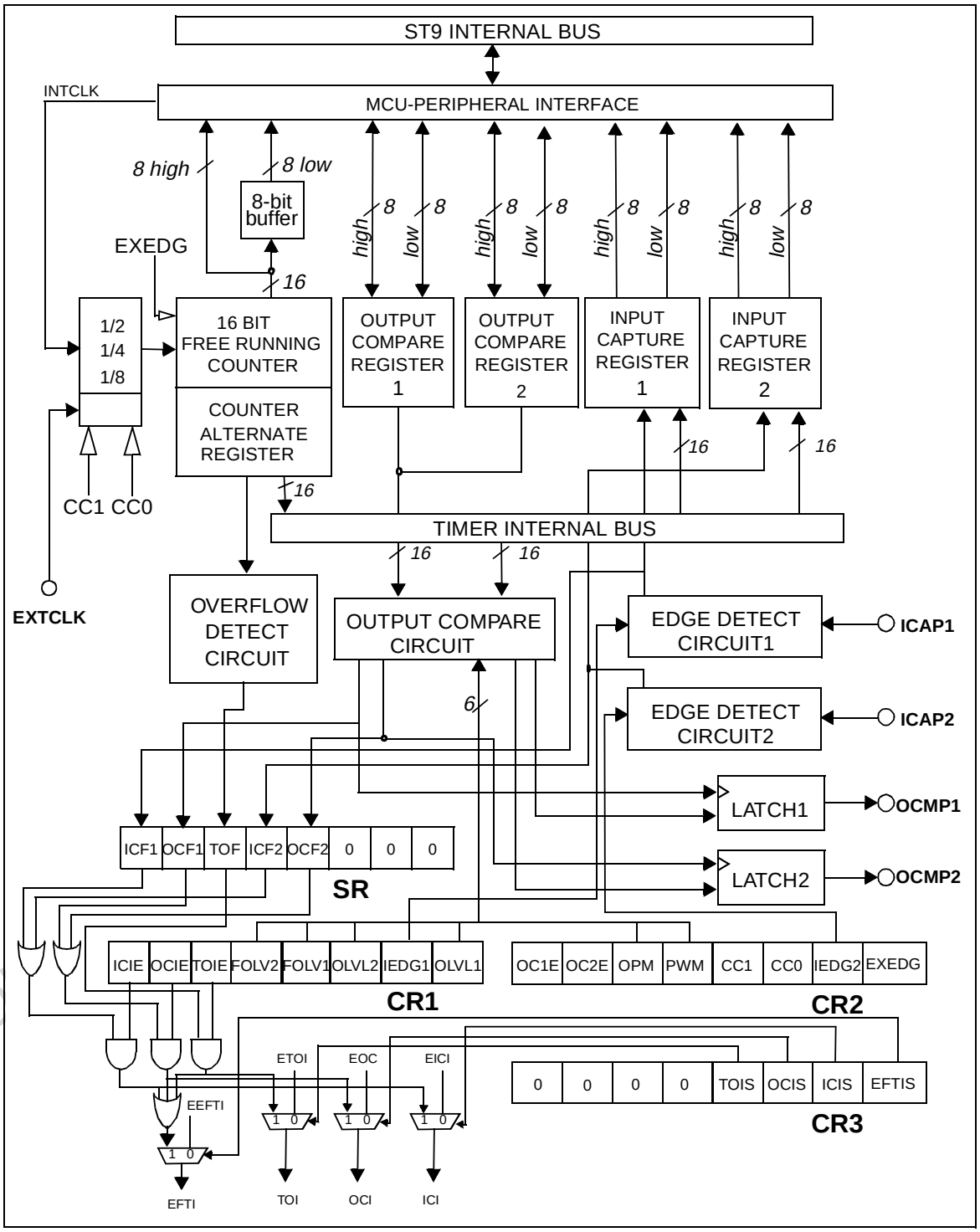
These two read-only 16-bit registers contain the same value but with the difference that reading the ACLR register does not clear the TOF bit (overflow flag), (see note page 98).

Writing in the CLR register or ACLR register resets the free running counter to the FFFCh value.

The timer clock depends on the clock control bits of the CR2 register, as illustrated in Table 22 Clock Control Bits. The value in the counter register repeats every 131.072, 262.144 or 524.288 INTCLK cycles depending on the CC1 and CC0 bits.

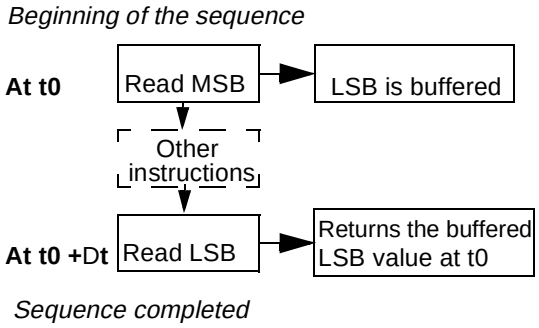
EXTENDED FUNCTION TIMER (Cont'd)

Figure 53. Timer Block Diagram



EXTENDED FUNCTION TIMER (Cont'd)

16-bit read sequence: (from either the Counter Register or the Alternate Counter Register).



The user must read the MSB first, then the LSB value is buffered automatically.

This buffered value remains unchanged until the 16-bit read sequence is completed, even if the user reads the MSB several times.

After a complete reading sequence, if only the CLR register or ACLR register are read, they return the LSB of the count value at the time of the read.

An overflow occurs when the counter rolls over from FFFFh to 0000h then:

– The TOF bit of the SR register is set.

– A timer interrupt is generated if:

  – TOIE bit of the CR1 register is set

  – TOIS bit of the CR3 register is set (or EFTIS bit if only global interrupt is available).

If one of these conditions is false, the interrupt remains pending to be issued as soon as they are both true.

Clearing the overflow interrupt request is done by:

1. Reading the SR register while the TOF bit is set.
2. An access (read or write) to the CLR register.

Notes: The TOF bit is not cleared by accesses to ACLR register. This feature allows simultaneous use of the overflow function and reads of the free running counter at random times (for example, to measure elapsed time) without the risk of clearing the TOF bit erroneously.

The timer is not affected by WAIT mode.

In HALT mode, the counter stops counting until the mode is exited. Counting then resumes from the reset count (MCU awakened by a Reset).

7.3.3.2 External Clock

The external clock (where available) is selected if CC0=1 and CC1=1 in CR2 register.

The status of the EXEDG bit determines the type of level transition on the external clock pin EXTCLK that will trigger the free running counter.

The counter is synchronised with the falling edge of INTCLK.

At least four falling edges of the INTCLK must occur between two consecutive active edges of the external clock; thus the external clock frequency must be less than a quarter of the INTCLK frequency.

EXTENDED FUNCTION TIMER (Cont'd)

Figure 54. Counter Timing Diagram, INTCLK divided by 2

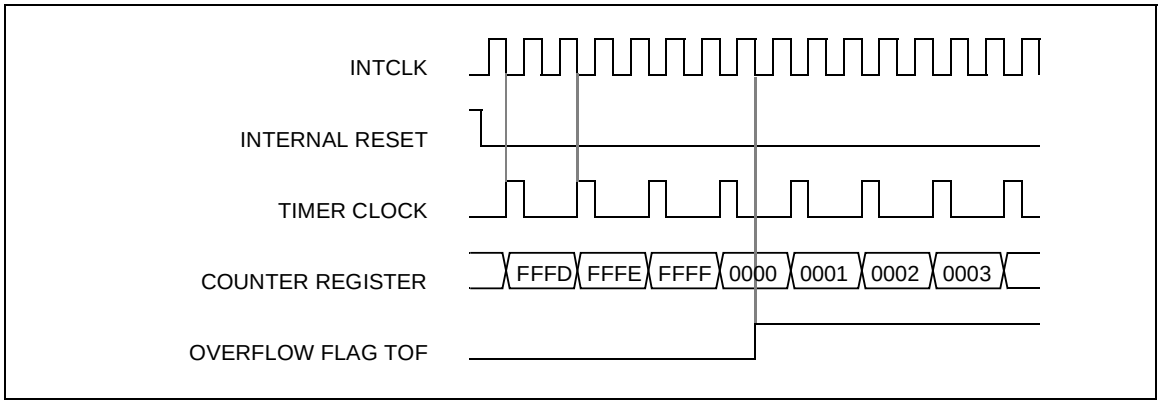


Figure 55. Counter Timing Diagram, INTCLK divided by 4

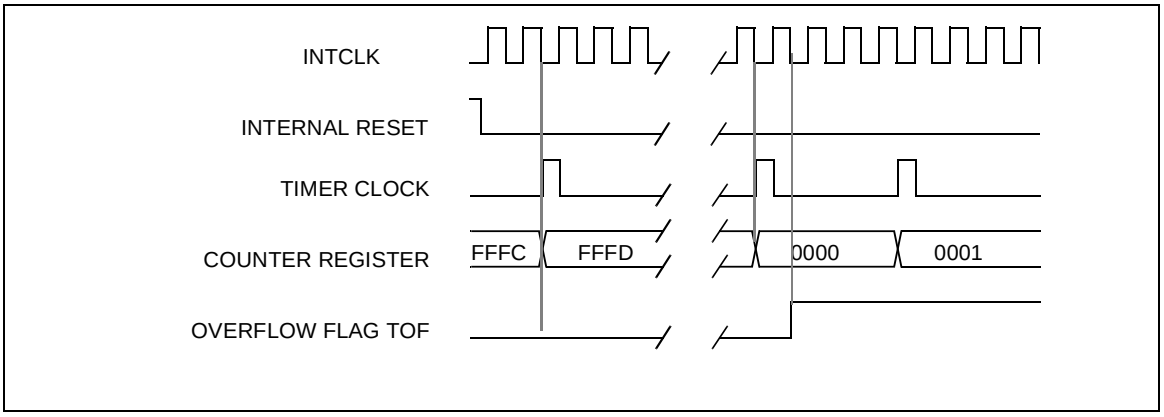
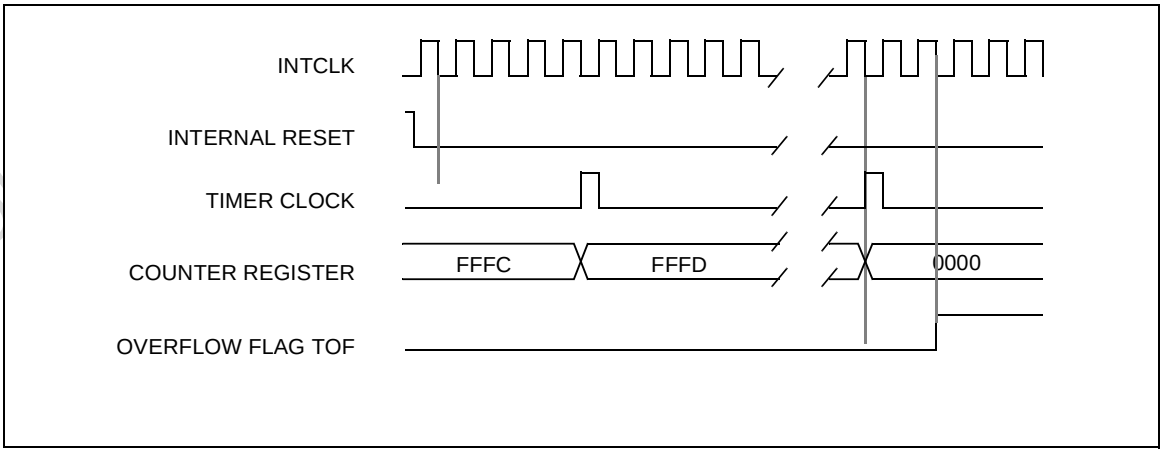


Figure 56. Counter Timing Diagram, INTCLK divided by 8



EXTENDED FUNCTION TIMER (Cont'd)

7.3.3.3 Input Capture

In this section, the index, *i*, may be 1 or 2.

The two input capture 16-bit registers (IC1R and IC2R) are used to latch the value of the free running counter after a transition detected by the ICAP*i* pin (see figure 5).

	MS Byte	LS Byte
IC*i*R	**IC*i*HR**	**IC*i*LR**

IC*i* Rregister is a read-only register.

The active transition is software programmable through the IEDG*i* bit of the Control Register (CR*i*).

Timing resolution is one count of the free running counter: ($INTCLK_{/CC[1:0]}$).

Procedure

To use the input capture function select the following in the CR2 register:

– Select the timer clock (CC[1:0] (see Table 22 Clock Control Bits).

– Select the edge of the active transition on the ICAP2 pin with the IEDG2 bit.

And select the following in the CR1 register:

– Set the ICIE bit to generate an interrupt after an input capture.

– Select the edge of the active transition on the ICAP1 pin with the IEDG1 bit.

When an input capture occurs:

– ICF*i* bit is set.

– The IC*i*R register contains the value of the free running counter on the active transition on the ICAP*i* pin (see Figure 58).

– A timer interrupt is generated if the ICIE bit is set and the ICIS bit (or EFTIS bit if only global interrupt is available) is set. Otherwise, the interrupt remains pending until both conditions become true.

Clearing the Input Capture interrupt request is done by:

1. Reading the SR register while the ICF*i* bit is set.

2. An access (read or write) to the IC*i*LR register.

Note: After reading the IC*i*HR register, transfer of input capture data is inhibited until the IC*i*LR register is also read.

The IC*i*R register always contains the free running counter value which corresponds to the most recent input capture.

EXTENDED FUNCTION TIMER (Cont'd)

Figure 57. Input Capture Block Diagram

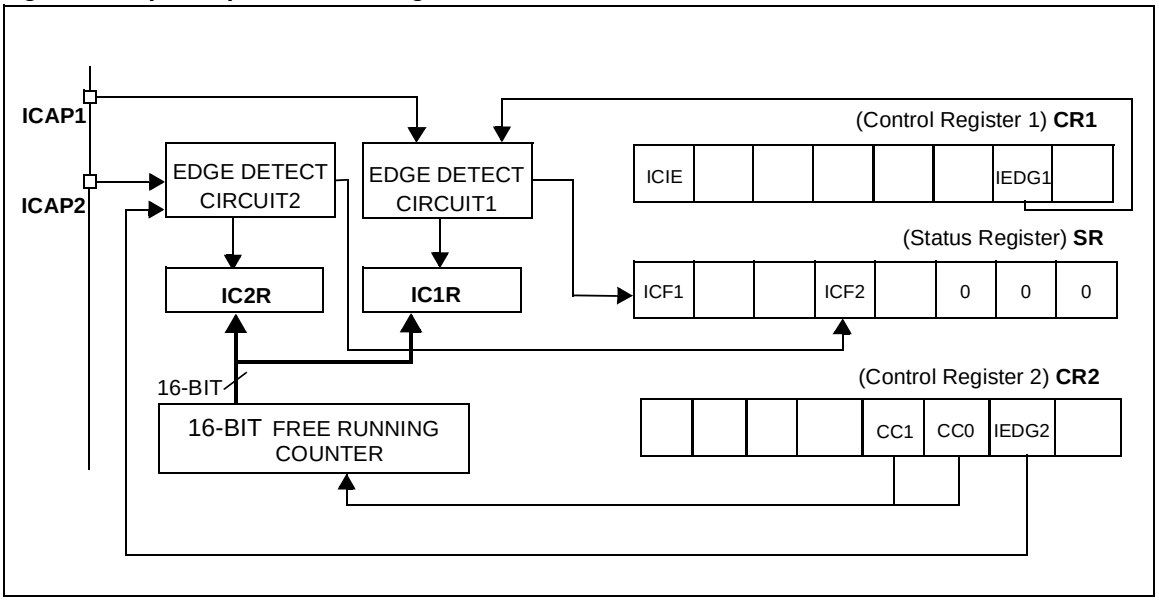
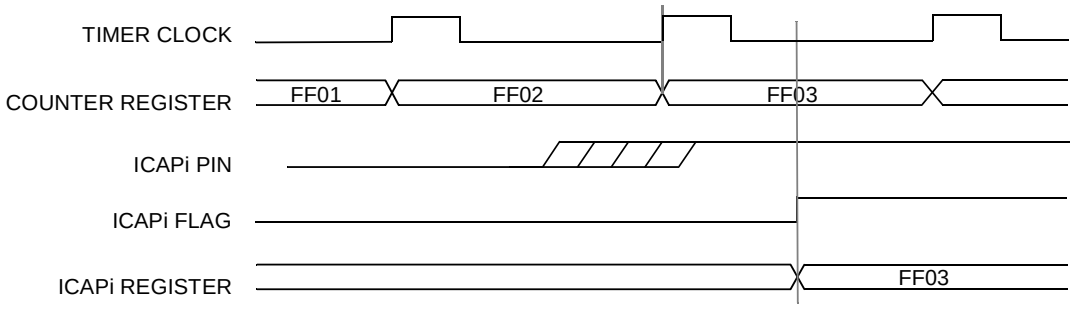


Figure 58. Input Capture Timing Diagram



Note: Active edge is rising edge.

EXTENDED FUNCTION TIMER (Cont'd)

7.3.3.4 Output Compare

In this section, the index, *i*, may be 1 or 2.

This function can be used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the Output Compare register and the free running counter, the output compare function:

– Assigns pins with a programmable value if the OC*i*E bit is set

– Sets a flag in the status register

– Generates an interrupt if enabled

Two 16-bit registers Output Compare Register 1 (OC1R) and Output Compare Register 2 (OC2R) contain the value to be compared to the free running counter each timer clock cycle.

	MS Byte	LS Byte
OC*i*R	OC*i*HR	OC*i*LR

These registers are readable and writable and are not affected by the timer hardware. A reset event changes the OC*i*R value to 8000h.

Timing resolution is one count of the free running counter: ($INTCLK_{/CC[1:0]}$).

Procedure

To use the output compare function, select the following in the CR2 register:

– Set the OC*i*E bit if an output is needed then the OCMP*i* pin is dedicated to the output compare *i* function.

– Select the timer clock CC[1:0] (see Table 22 Clock Control Bits).

And select the following in the CR1 register:

– Select the OLVL*i* bit to applied to the OCMP*i* pins after the match occurs.

– Set the OCIE and OCIS bits (or EFTIS bit if only global interrupt is available) to generate an interrupt if it is needed.

When match is found:

– OCF*i* bit is set.

– The OCMP*i* pin takes OLVL*i* bit value (OCMP*i* pin latch is forced low during reset and stays low until valid compares change it to OLVL*i* level).

– A timer interrupt is generated if the OCIE bit is set in the CR2 register and OCIS bit (or EFTIS bit if only global interrupt is available) is set in the CR3 register.

Clearing the output compare interrupt request is done by:

3. Reading the SR register while the OCF*i* bit is set.

4. An access (read or write) to the OC*i*LR register.

Note: After a processor write cycle to the OC*i*HR register, the output compare function is inhibited until the OC*i*LR register is also written.

If the OC*i*E bit is not set, the OCMP*i* pin is a general I/O port and the OLVL*i* bit will not appear when match is found but an interrupt could be generated if the OCIE bit is set.

The value in the 16-bit OC*i*R register and the OLVL*i* bit should be changed after each successful comparison in order to control an output waveform or establish a new elapsed timeout.

The OC*i*R register value required for a specific timing application can be calculated using the following formula:

$$\Delta \text{ OC}i\text{R} = \frac{\Delta t * \text{INTCLK}}{(\text{CC1.CC0})}$$

Where:

Δt = Desired output compare period (in seconds)

INTCLK = Internal clock frequency

CC1-CC0 = Timer clock prescaler

The following procedure is recommended to prevent the OCF*i* bit from being set between the time it is read and the write to the OC*i*R register:

– Write to the OC*i*HR register (further compares are inhibited).

– Read the SR register (first step of the clearance of the OCF*i* bit, which may be already set).

– Write to the OC*i*LR register (enables the output compare function and clears the OCF*i* bit).

EXTENDED FUNCTION TIMER (Cont'd)

Figure 59. Output Compare Block Diagram

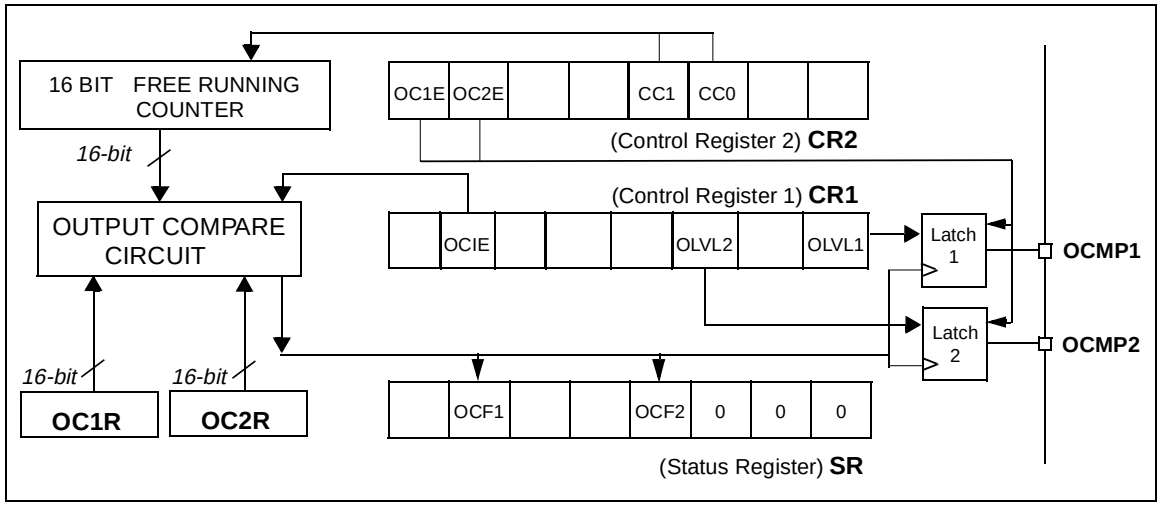
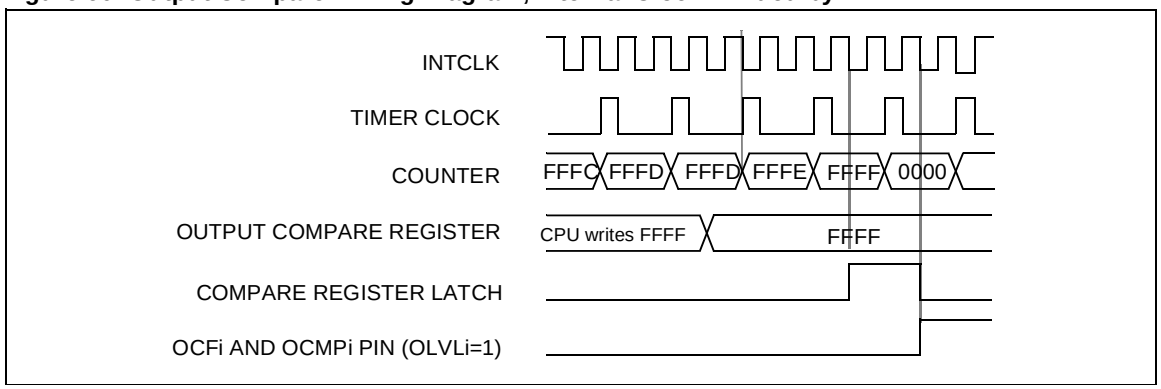


Figure 60. Output Compare Timing Diagram, Internal Clock Divided by 2



EXTENDED FUNCTION TIMER (Cont'd)

7.3.3.5 Forced Compare Mode

In this section *i* may represent 1 or 2.

The following bits of the CR1 register are used:

			FOLV2	FOLV1	OLVL2		OLVL1

When the FOLV*i* bit is set, the OLVL*i* bit is copied to the OCMP*i* pin. The OLVL*i* bit has to be toggled in order to toggle the OCMP*i* pin when it is enabled (OC*i*E bit=1).

The OCF*i* bit is not set, and thus no interrupt request is generated.

7.3.3.6 One Pulse Mode

One Pulse mode enables the generation of a pulse when an external event occurs. This mode is selected via the OPM bit in the CR2 register.

The one pulse mode uses the Input Capture1 function and the Output Compare1 function.

Procedure

To use one pulse mode, select the following in the the CR1 register:

– Using the OLVL1 bit, select the level to be applied to the OCMP1 pin after the pulse.

– Using the OLVL2 bit, select the level to be applied to the OCMP1 pin during the pulse.

– Select the edge of the active transition on the ICAP1 pin with the IEDG1 bit.

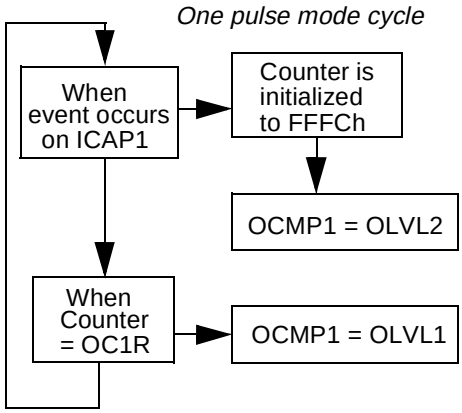
And select the following in the CR2 register:

– Set the OC1E bit, the OCMP1 pin is then dedicated to the Output Compare 1 function.

– Set the OPM bit.

– Select the timer clock CC[1:0] (see Table 22 Clock Control Bits).

Load the OC1R register with the value corresponding to the length of the pulse (see the formula in Section 7.3.3.7).



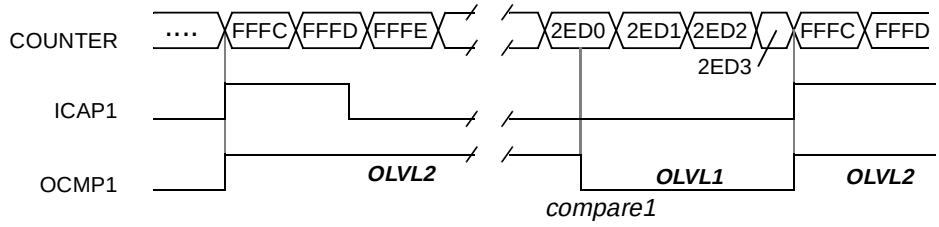
Then, on a valid event on the ICAP1 pin, the counter is initialized to FFFCh and OLVL2 bit is loaded on the OCMP1 pin. When the value of the counter is equal to the value of the contents of the OC1R register, the OLVL1 bit is output on the OCMP1 pin, (See Figure 61).

Note: The OCF1 bit cannot be set by hardware in one pulse mode but the OCF2 bit can generate an Output Compare interrupt.

The ICF1 bit is set when an active edge occurs and can generate an interrupt if the ICIE bit is set.

When the Pulse Width Modulation (PWM) and One Pulse Mode (OPM) bits are both set, the PWM mode is the only active one.

Figure 61. One Pulse Mode Timing



Note: IEDG1=1, OC1R=2ED0h, OLVL1=0, OLVL2=1

EXTENDED FUNCTION TIMER (Cont'd)

7.3.3.7 Pulse Width Modulation Mode

Pulse Width Modulation mode enables the generation of a signal with a frequency and pulse length determined by the value of the OC1R and OC2R registers.

The pulse width modulation mode uses the complete Output Compare 1 function plus the OC2R register.

Procedure

To use pulse width modulation mode select the following in the CR1 register:

– Using the OLVL1 bit, select the level to be applied to the OCMP1 pin after a successful comparison with OC1R register.

– Using the OLVL2 bit, select the level to be applied to the OCMP1 pin after a successful comparison with OC2R register.

And select the following in the CR2 register:

– Set OC1E bit: the OCMP1 pin is then dedicated to the output compare 1 function.

– Set the PWM bit.

– Select the timer clock CC[1:0] bits (see Table 22 Clock Control Bits).

Load the OC2R register with the value corresponding to the period of the signal.

Load the OC1R register with the value corresponding to the length of the pulse if (OLVL1=0 and OLVL2=1).

If OLVL1=1 and OLVL2=0 the length of the pulse is the difference between the OC2R and OC1R registers.

The OC*i*R register value required for a specific timing application can be calculated using the following formula:

$$\text{OC}i\text{R Value} = \frac{t * \text{INTCLK}}{\text{CC[1:0]}} - 5$$

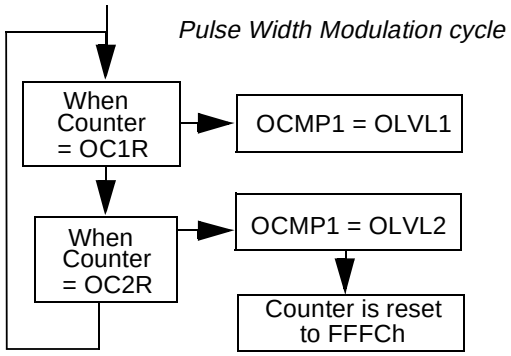
Where:

– t = Desired output compare period (seconds)

– INTCLK = Internal clock frequency

– CC1-CC0 = Timer clock prescaler

The Output Compare 2 event causes the counter to be initialized to FFFCh (See Figure 62).

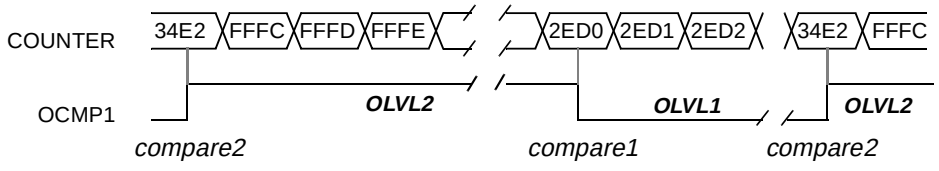


Note: After a write instruction to the OC*i*HR register, the output compare function is inhibited until the OC*i*LR register is also written.

The OCF1 and OCF2 bits cannot be set by hardware in PWM mode therefore the Output Compare interrupt is inhibited. The Input Capture interrupts are available.

When the Pulse Width Modulation (PWM) and One Pulse Mode (OPM) bits are both set, the PWM mode is the only active one.

Figure 62. Pulse Width Modulation Mode Timing



Note: OC1R=2ED0h, OC2R=34E2, OLVL1=0, OLVL2= 1

EXTENDED FUNCTION TIMER (Cont'd)

7.3.4 Interrupt Management

The interrupts of the Extended Function Timer are mapped on the eight external interrupt channels of the microcontroller (refer to the "Interrupts" chapter).
Depending on device specification, one of the following configurations can occur:

– The three interrupt sources are mapped on three different interrupt channels (to use this feature, the EFTIS bit must be reset)

– The three interrupt sources are mapped on the same interrupt channel (to use this feature, the EFTIS bit must be set)

Each External Interrupt Channel has:

– A trigger control bit in the EITR register (R242 - Page 0)

– A pending bit in the EIPR register (R243 - Page 0)

– A mask bit in the EIMR register (R244 - Page 0)

Program the interrupt priority level using the EIPLR register (R245 - Page 0). For a description of these registers refer to the "Interrupts" and "DMA" chapters.

Use of three interrupt channels

To use the interrupt features, for each interrupt channel used, perform the following sequence:

– Set the priority level of the interrupt channel(s) used for the Extended Function Timer (EIPRL register)

– Select the interrupt trigger edge(s) as rising edge (set the corresponding bit(s) in the EITR register)

– Set the OCIS and/or ICIS and/or TOIS bit(s) of the CR3 register to select the peripheral interrupt source(s)

– Set the OCIE and/or ICIE and/or TOIE bit(s) of the CR1 register to enable the peripheral to perform interrupt requests on the desiderate events

– In the EIPR register, reset the pending bit(s) of the interrupt channels used by the peripheral interrupts to avoid any spurious interrupt requests being performed when the mask bit(s) is/are set

– Set the mask bit(s) of the interrupt channel(s) used to enable the MCU to acknowledge the interrupt requests of the peripheral.

Use of one external interrupt channel for all the interrupts

To use the interrupt features, perform the following sequence:

– Set the priority level of the interrupt channel used (EIPRL register)

– Select the interrupt trigger edge as rising edge (set the corresponding bit in the EITR register)

– Set the EFTIS bit of the CR3 register to select the peripheral interrupt sources

– Set the OCIE and/or ICIE and/or TOIE bit(s) of the CR1 register to enable the peripheral to perform interrupt requests on the wanted events

– In the EIPR register, reset the pending bit of the interrupt channel used by the peripheral interrupts to avoid any spurious interrupt requests being performed when the mask bits is set

– Set the mask bits of the interrupt channels used to enable the MCU to acknowledge the interrupt requests of the peripheral.

Caution: Care should be taken when using only one of the input capture pins, as both capture interrupts are enabled by the ICIE bit in the CR1 register. If only ICAP1 is used (for example), an interrupt can still be generated by the ICAP2 pin when this pin toggles, even if it is configured as a standard output. If this case, the interrupt capture status bits in the SR register should handled in polling mode.

Caution:

1. It is mandatory to clear all EFT interrupt flags simultaneously at least once before exiting an EFT timer interrupt routine (the SR register must = 00h at some point during the interrupt routine), otherwise no interrupts can be issued on that channel anymore.
   Refer to the following assembly code for an interrupt sequence example.

2. Since a loop statement is needed inside the IT routine, the user must avoid situations where an interrupt event period is narrower than the duration of the interrupt treatment. Otherwise nested interrupt mode must be used to serve higher priority requests.

EXTENDED FUNCTION TIMER (Cont'd)

Note: A single access (read/write) to the SR register at the beginning of the interrupt routine is the first step needed to clear all the EFT interrupt flags. In a second step, the lower bytes of the data registers must be accessed if the corresponding flag is set. It is not necessary to access the SR register between these instructions, but it can done.

```
; INTERRUPT ROUTINE EXAMPLE
            push R234          ; Save current page
            spp #28            ; Set EFT page
L6:
            cp R254,#0         ; while E0_SR is not cleared
            jxz L7
            tm R254,#128       ; Check Input Capture 1 flag
            jxz L2             ; else go to next test
            ld r1,R241         ; Dummy read to clear IC1LR
            ; Insert your code here
L2:
            tm R254,#16       ; Check Input Capture 2 flag
            jxz L3             ; else go to next test
            ld r1,R243         ; Dummy read to clear IC2LR
            ; Insert your code here
L3:
            tm R254,#64       ; Check Input Compare 1 flag
            jxz L4             ; else go to next test
            ld r1,R249         ; Dummy read to clear OC1LR
            ; Insert your code here
L4:
            tm R254,#8        ; Check Input Compare 2 flag
            jxz L5             ; else go to next test
            ld r1,R251         ; Dummy read to clear OC1LR
            ; Insert your code here
L5:
            tm R254,#32        ; Check Input Overflow flag
            jxz L6             ; else go to next test
            ld r1,R245         ; Dummy read to clear Overflow flag
            ; Insert your code here
            jx L6
L7:
            pop R234           ; Restore current page
iret
```

EXTENDED FUNCTION TIMER (Cont'd)

7.3.5 Register Description

Each Timer is associated with three control and one status registers, and with six pairs of data registers (16-bit values) relating to the two input captures, the two output compares, the counter and the alternate counter.

Notes:

1. In the register description on the following pages, register and page numbers are given using the example of Timer 0. On devices with more than one timer, refer to the device register map for the adresses and page numbers.

2. To work correctly with register pairs, it is strongly recommended to use single byte instructions. Do not use word instructions to access any of the 16-bit registers.

INPUT CAPTURE 1 HIGH REGISTER (IC1HR)

R240 - Read Only
Register Page: 28
Reset Value: Undefined

This is an 8-bit read only register that contains the high part of the counter value (transferred by the input capture 1 event).

7							0
MSB							LSB

INPUT CAPTURE 1 LOW REGISTER (IC1LR)

R241 - Read Only
Register Page: 28
Reset Value: Undefined

This is an 8-bit read only register that contains the low part of the counter value (transferred by the input capture 1 event).

7							0
MSB							LSB

INPUT CAPTURE 2 HIGH REGISTER (IC2HR)

R242 - Read Only
Register Page: 28
Reset Value: Undefined

This is an 8-bit read only register that contains the high part of the counter value (transferred by the Input Capture 2 event).

7							0
MSB							LSB

INPUT CAPTURE 2 LOW REGISTER (IC2LR)

R243 - Read Only
Register Page: 28
Reset Value: Undefined

This is an 8-bit read only register that contains the low part of the counter value (transferred by the Input Capture 2 event).

7							0
MSB							LSB

EXTENDED FUNCTION TIMER (Cont'd)

COUNTER HIGH REGISTER (CHR)

R244 - Read Only
Register Page: 28
Reset Value: 1111 1111 (FFh)

This is an 8-bit register that contains the high part of the counter value.

7							0
MSB							LSB

COUNTER LOW REGISTER (CLR)

R245 - Read/Write
Register Page: 28
Reset Value: 1111 1100 (FCh)

This is an 8-bit register that contains the low part of the counter value. A write to this register resets the counter. An access to this register after accessing the SR register clears the TOF bit.

7							0
MSB							LSB

ALTERNATE COUNTER HIGH REGISTER (ACHR)

R246 - Read Only
Register Page: 28
Reset Value: 1111 1111 (FFh)

This is an 8-bit register that contains the high part of the counter value.

7							0
MSB							LSB

ALTERNATE COUNTER LOW REGISTER (ACLR)

R247 - Read/Write
Register Page: 28
Reset Value: 1111 1100 (FCh)

This is an 8-bit register that contains the low part of the counter value. A write to this register resets the counter. An access to this register after an access to SR register does not clear the TOF bit in SR register.

7							0
MSB							LSB

EXTENDED FUNCTION TIMER (Cont'd)

OUTPUT COMPARE 1 HIGH REGISTER (OC1HR)

R248 - Read/Write
Register Page: 28
Reset Value: 1000 0000 (80h)

This is an 8-bit register that contains the high part of the value to be compared to the CHR register.

7							0
MSB							LSB

OUTPUT COMPARE 2 HIGH REGISTER (OC2HR)

R250 - Read/Write
Register Page: 28
Reset Value: 1000 0000 (80h)

This is an 8-bit register that contains the high part of the value to be compared to the CHR register.

7							0
MSB							LSB

OUTPUT COMPARE 1 LOW REGISTER (OC1LR)

R249 - Read/Write
Register Page: 28
Reset Value: 0000 0000 (00h)

This is an 8-bit register that contains the low part of the value to be compared to the CLR register.

7							0
MSB							LSB

OUTPUT COMPARE 2 LOW REGISTER (OC2LR)

R251 - Read/Write
Register Page: 28
Reset Value: 0000 0000 (00h)

This is an 8-bit register that contains the low part of the value to be compared to the CLR register.

7							0
MSB							LSB

EXTENDED FUNCTION TIMER (Cont'd)

CONTROL REGISTER 1 (CR1)

R252 - Read/Write
Register Page: 28
Reset Value: 0000 0000 (00h)

7							0
ICIE	OCIE	TOIE	FOLV2	FOLV1	OLVL2	IEDG1	OLVL1

Bit 7 = **ICIE** *Input Capture Interrupt Enable.*
0: Interrupt is inhibited.
1: A timer interrupt is generated whenever the ICF1 or ICF2 bit of the SR register is set.

Bit 6 = **OCIE** *Output Compare Interrupt Enable.*
0: Interrupt is inhibited.
1: A timer interrupt is generated whenever the OCF1 or OCF2 bit of the SR register is set.

Bit 5 = **TOIE** *Timer Overflow Interrupt Enable.*
0: Interrupt is inhibited.
1: A timer interrupt is enabled whenever the TOF bit of the SR register is set.

Bit 4 = **FOLV2** *Forced Output Compare 2.*
0: No effect.
1: Forces the OLVL2 bit to be copied to the OCMP2 pin.

Bit 3 = **FOLV1** *Forced Output Compare 1.*
0: No effect.
1: Forces OLVL1 to be copied to the OCMP1 pin.

Bit 2 = **OLVL2** *Output Level 2.*
This bit is copied to the OCMP2 pin whenever a successful comparison occurs with the OC2R register and OC2E is set in the CR2 register. This value is copied to the OCMP1 pin in One Pulse Mode and Pulse Width Modulation mode.

Bit 1 = **IEDG1** *Input Edge 1.*
This bit determines which type of level transition on the ICAP1 pin will trigger the capture.
0: A falling edge triggers the capture.
1: A rising edge triggers the capture.

Bit 0 = **OLVL1** *Output Level 1.*
The OLVL1 bit is copied to the OCMP1 pin whenever a successful comparison occurs with the OC1R register and the OC1E bit is set in the CR2 register.

EXTENDED FUNCTION TIMER (Cont'd)

CONTROL REGISTER 2 (CR2)

R253 - Read/Write
Register Page: 28
Reset Value: 0000 0000 (00h)

7							0
OC1E	OC2E	OPM	PWM	CC1	CC0	IEDG2	EXEDG

Bit 7 = **OC1E** *Output Compare 1 Enable.*
0: Output Compare 1 function is enabled, but the OCMP1 pin is a general I/O.
1: Output Compare 1 function is enabled, the OCMP1 pin is dedicated to the Output Compare 1 capability of the timer.

Bit 6 = **OC2E** *Output Compare 2 Enable.*
0: Output Compare 2 function is enabled, but the OCMP2 pin is a general I/O.
1: Output Compare 2 function is enabled, the OCMP2 pin is dedicated to the Output Compare 2 capability of the timer.

Bit 5 = **OPM** *One Pulse Mode.*
0: One Pulse Mode is not active.
1: One Pulse Mode is active, the ICAP1 pin can be used to trigger one pulse on the OCMP1 pin; the active transition is given by the IEDG1 bit. The length of the generated pulse depends on the contents of the OC1R register.

Bit 4 = **PWM** *Pulse Width Modulation.*
0: PWM mode is not active.
1: PWM mode is active, the OCMP1 pin outputs a programmable cyclic signal; the length of the pulse depends on the value of OC1R register; the period depends on the value of OC2R register.

Bit 3, 2 = **CC[1:0]** *Clock Control.*

The value of the timer clock depends on these bits:

Table 22. Clock Control Bits

CC1	CC0	Timer Clock
0	0	INTCLK / 4
0	1	INTCLK / 2
1	0	INTCLK / 8
1	1	External Clock (where available)

Bit 1 = **IEDG2** *Input Edge 2.*
This bit determines which type of level transition on the ICAP2 pin will trigger the capture.
0: A falling edge triggers the capture.
1: A rising edge triggers the capture.

Bit 0 = **EXEDG** *External Clock Edge.*
This bit determines which type of level transition on the external clock pin EXTCLK will trigger the free running counter.
0: A falling edge triggers the free running counter.
1: A rising edge triggers the free running counter.

EXTENDED FUNCTION TIMER (Cont'd)

STATUS REGISTER (SR)

R254 - Read Only
Register Page: 28
Reset Value: 0000 0000 (00h)

The three least significant bits are not used.

7							0
ICF1	OCF1	TOF	ICF2	OCF2	0	0	0

Bit 7 = **ICF1** *Input Capture Flag 1.*
0: No input capture (reset value).
1: An input capture has occurred. To clear this bit, first read the SR register, then read or write the low byte of the IC1R (IC1LR) register.

Bit 6 = **OCF1** *Output Compare Flag 1.*
0: No match (reset value).
1: The content of the free running counter has matched the content of the OC1R register. To clear this bit, first read the SR register, then read or write the low byte of the OC1R (OC1LR) register.

Bit 5 = **TOF** *Timer Overflow.*
0: No timer overflow (reset value).
1: The free running counter rolled over from FFFFh to 0000h. To clear this bit, first read the SR register, then read or write the low byte of the CR (CLR) register.

Note: Reading or writing the ACLR register does not clear TOF.

Bit 4 = **ICF2** *Input Capture Flag 2.*
0: No input capture (reset value).
1: An input capture has occurred. To clear this bit, first read the SR register, then read or write the low byte of the IC2R (IC2LR) register.

Bit 3 = **OCF2** *Output Compare Flag 2.*
0: No match (reset value).
1: The content of the free running counter has matched the content of the OC2R register. To clear this bit, first read the SR register, then read or write the low byte of the OC2R (OC2LR) register.

Bit 2-0 = Reserved, forced by hardware to 0.

CONTROL REGISTER 3 (CR3)

R255 - Read/Write
Register Page: 28
Reset Value: 0000 0000 (00h)

7							0
0	0	0	0	TOIS	OCIS	ICIS	EFTIS

Bit 7-4 = **Unused** *Read as 0.*

Bit 3 = **TOIS** *Timer Overflow Interrupt Selection.*
0: Select External interrupt.
1: Select Timer Overflow Interrupt.

Bit 2 = **OCIS** *Output Compare Interrupt Selection.*
0: Select External interrupt.
1: Select Timer Output Compare Interrupt.

Bit 1 = **ICIS** *Input Capture Interrupt Selection.*
0: Select External interrupt.
1: Select Timer Input Capture Interrupt.

Bit 0 = **EFTIS** *Global Timer Interrupt Selection.*
0: Select External interrupt.
1: Select Global Timer Interrupt.

EXTENDED FUNCTION TIMER (Cont'd)

Table 23. Extended Function Timer Register Map

Address (Dec.)	Register Name	7	6	5	4	3	2	1	0
R240	**IC1HR**	MSB							LSB
	Reset Value	x	x	x	x	x	x	x	x
R241	**IC1LR**	MSB							LSB
	Reset Value	x	x	x	x	x	x	x	x
R242	**IC2HR**	MSB							LSB
	Reset Value	x	x	x	x	x	x	x	x
R243	**IC2LR**	MSB							LSB
	Reset Value	x	x	x	x	x	x	x	x
R244	**CHR**	MSB							LSB
	Reset Value	1	1	1	1	1	1	1	1
R245	**CLR**	MSB							LSB
	Reset Value	1	1	1	1	1	1	0	0
R246	**ACHR**	MSB							LSB
	Reset Value	1	1	1	1	1	1	1	1
R247	**ACLR**	MSB							LSB
	Reset Value	1	1	1	1	1	1	0	0
R248	**OC1HR**	MSB							LSB
	Reset Value	1	0	0	0	0	0	0	0
R249	**OC1LR**	MSB							LSB
	Reset Value	0	0	0	0	0	0	0	0
R250	**OC2HR**	MSB							LSB
	Reset Value	1	0	0	0	0	0	0	0
R251	**OC2LR**	MSB							LSB
	Reset Value	0	0	0	0	0	0	0	0
R252	**CR1**	OC1E	OC2E	OPM	PWM	CC1	CC0	IEDG2	EXEDG
	Reset Value	0	0	0	0	0	0	0	0
R253	**CR2**	ICIE	OCIE	TOIE	FOLV2	FOLV1	OLVL2	IEDG1	OLVL1
	Reset Value	0	0	0	0	0	0	0	0
R254	**SR**	ICF1	OCF1	TOF	ICF2	OCF2	0	0	0
	Reset Value	0	0	0	0	0	0	0	0
R255	**CR3**	0	0	0	0	TOIS	OCIS	ICIS	EFTIS
	Reset Value	0	0	0	0	0	0	0	0

EXTENDED FUNCTION TIMER (Cont'd)

Table 24. Extended Function Timer Page Map

Timer number	Page (hex)
EFT0	**1C**

7.4 3-PHASE INDUCTION MOTOR CONTROLLER (IMC)

7.4.1 Introduction

The IMC controller is designed for variable speed motor control applications. Three PWM outputs are available for controlling a three-phase motor drive. Rotor speed feedback is provided by capturing a tachogenerator input signal.

7.4.2 Main Features

- 10-bit PWM Up/Down Counter
- Classical and zerocentered PWM operating modes
- Full-scale PWM generation
- 6-bit dead time generator
- Rotor speed measurement
- 8 interrupt sources + 1 NMI

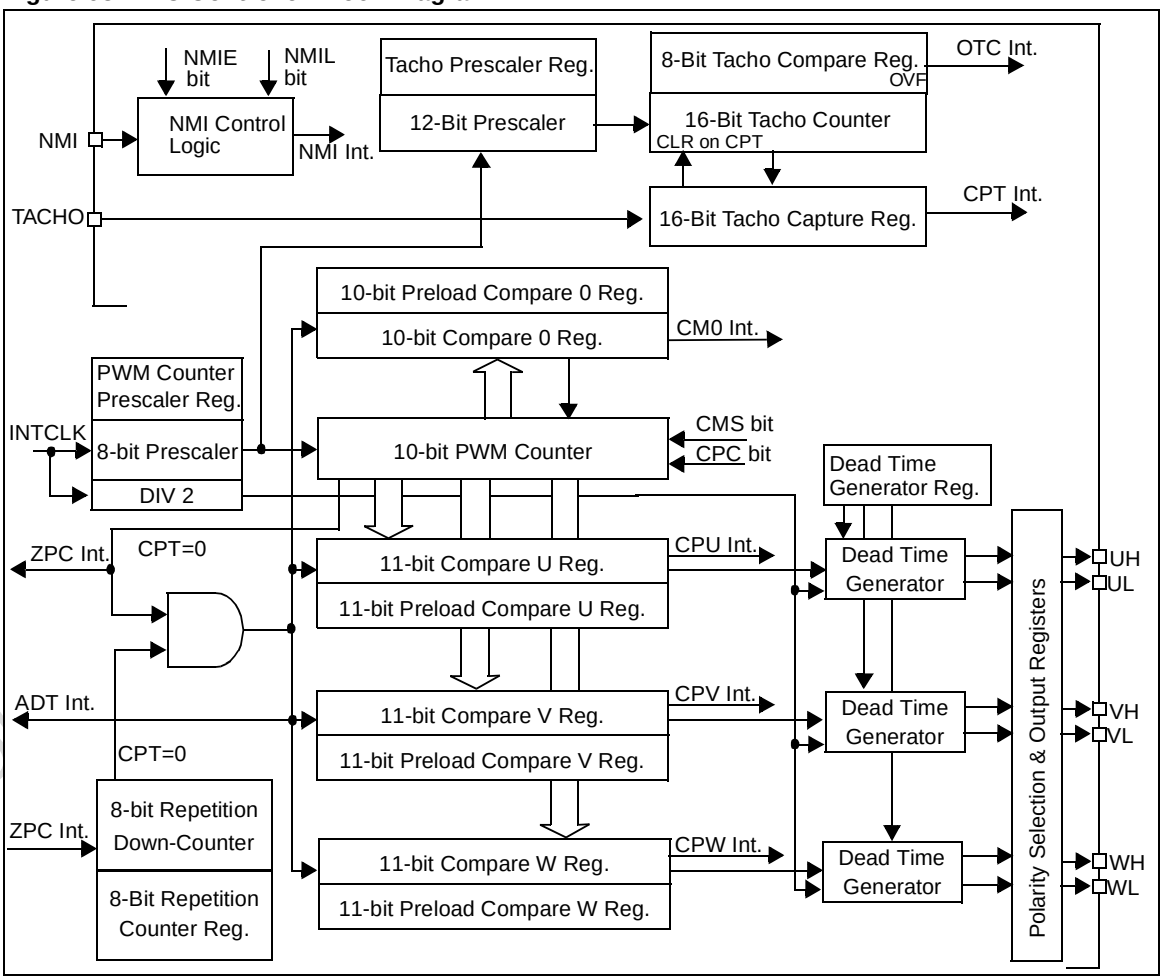
7.4.3 Functional Description

The IMC controller consists of the following function blocks:

– Input and Output pins
– Rotor speed measurement
– 3-Phase PWM signal generation
– 6-bit Dead Time generation
– Polarity selection
– Interrupt generation

The block diagram is shown in Figure 63.

Figure 63. IMC Controller Block Diagram



INDUCTION MOTOR CONTROLLER (Cont'd)

7.4.3.1 Input and Output pins

– Input Pin

TACHO: Signal input from a tachogenerator for measuring the rotor speed.

NMI: Input signal for disabling the IMC output and sending an interrupt request to the ST9 core.

– Output Pins

UH, UL, VH, VL, WH, WL: 3-Phase PWM signals and complementary signals (dedicated pins, refer to device Pin Description).

Note: The INTCLK signal is the internal clock of the ST9 microcontroller (system clock).

7.4.3.2 Rotor Speed Measurement

The TACHO signal is input from a Schmitt trigger port. When a rising and/or falling edge occurs (programmable edge sensitivity), the IMC controller does the following:

– Captures the 16-bit Tacho Counter

– Clears the Tacho Counter (if CCPT bit is set)

– Generates a CPT interrupt

The 16-bit Tacho Counter clock is derived from the clock used by the PWM Counter, through a 12-bit prescaler. The 12-bit prescaler divides by 1, 2, 3, ..., 4096.

If no edge occurs on the TACHO signal or the event sensitivity is disabled (see Table 25) and the 16-bit counter is running, an OTC overflow interrupt will be issued when the MSB (Most Significant Byte) of the Tacho Counter reaches the Tacho Compare register value.

7.4.3.3 Three-Phase PWM Generator

The 3-Phase PWM signal is generated using a 10-bit PWM Counter and three 11-bit Compare registers one for each phase (U, V, W).

The 10-bit PWM Counter clock is supplied through a 8-bit prescaler (dividing by 1, 2, 3, .., 256).

It can work in Zerocentered mode or in Classical mode. The mode is selected by the CMS bit in the PCR0 register:

Zerocentered Mode

In this operating mode, the PWM Counter counts up to the value loaded in the 10-bit Compare 0 register then counts down until it reaches zero and restarts counting up.

Classical Mode

In this operating mode, the PWM Counter counts up to the value loaded in the 10-bit Compare Register. Then the PWM Counter is cleared and it restarts counting up.

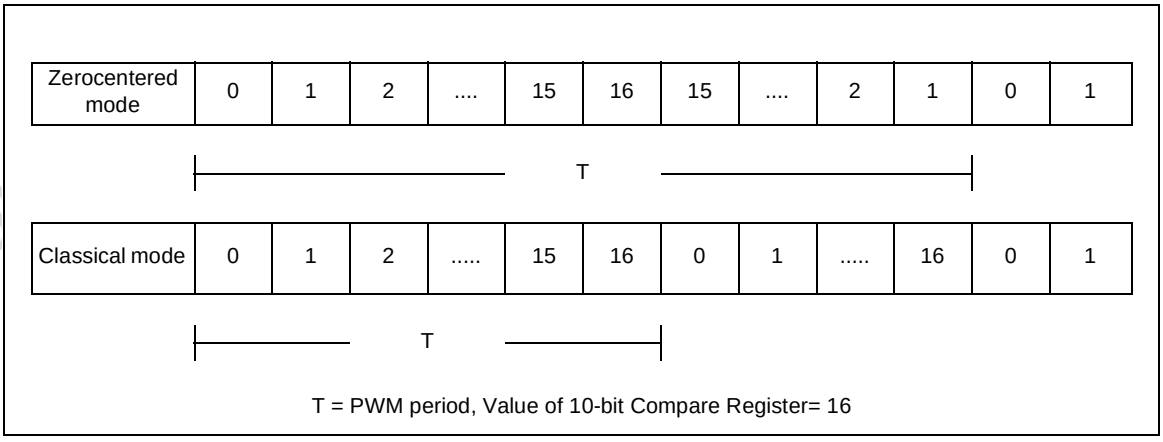
Figure 64 shows the counting sequence in Classical and Zerocentered mode.

PWM signal generation in Zerocentered mode

In this mode, all three PWM signals are set to '0' when the PWM Counter reaches, in up-counting, the corresponding 11-bit Compare register value and they are set to '1' when the PWM Counter reaches the 11-bit Compare value again in down-counting.

The comparison is performed between the PWM Counter value extended to 11 bits and the 11-bit Compare register (either in Zerocentered or in Classical mode).

Figure 64. Counting sequence in Zerocentered and Classical mode



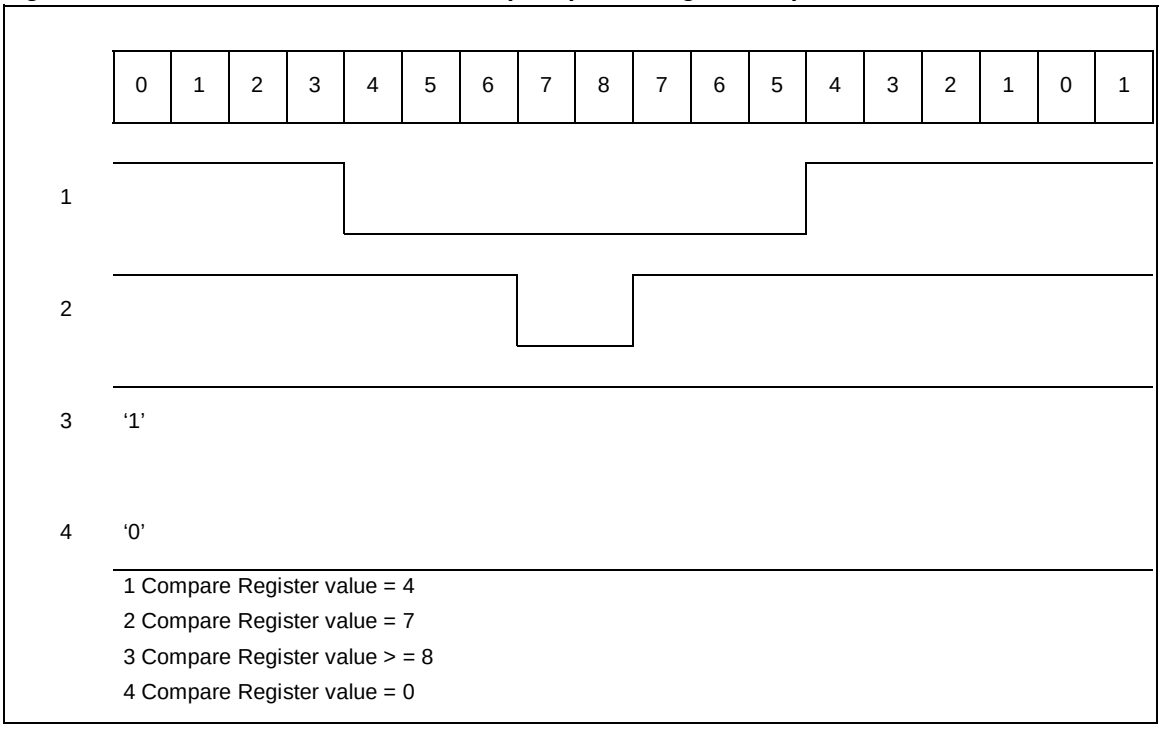
INDUCTION MOTOR CONTROLLER (Cont'd)

If the 11-bit Compare register value is greater than the extended Compare 0 Register (the 11th bit is set to '0'), the corresponding PWM output signal is held at '1'.

If the 11-bit Compare register value is 0, the corresponding PWM output signal is held at '0'.

Figure 65 shows some Zerocentered PWM waveforms in an example where the Compare 0 register value = 8.

Figure 65. Zerocentered PWM Waveforms (Compare 0 Register = 8)



INDUCTION MOTOR CONTROLLER (Cont'd)

PWM signals generation in Classical mode

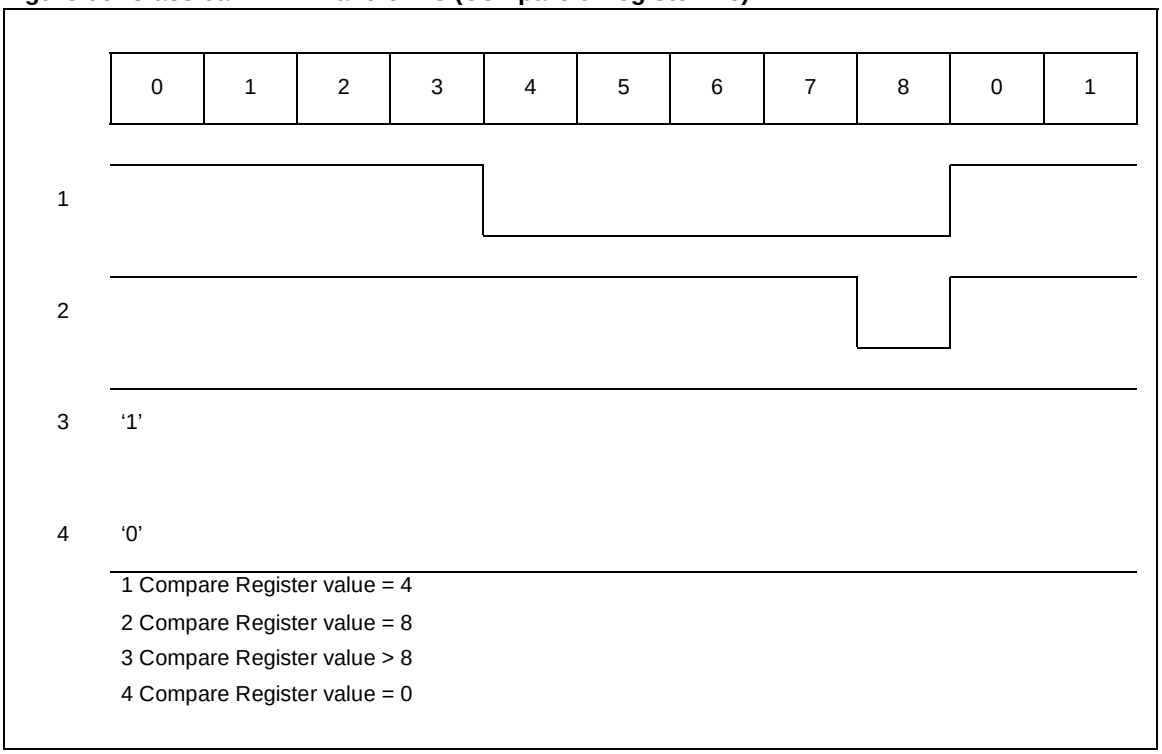
In this mode, each of the three PWM signals set to '0' when the PWM Counter reaches, in up-counting, the corresponding 11-bit Compare register value and they are set to '1' when the PWM Counter is cleared.

If the 11-bit Compare register value is greater than the extended Compare 0 register (the 11th bit is set to '0'), the corresponding PWM output signal is held at '1'.

If the 11-bit Compare register value = 0, the corresponding PWM output signal is held at '0'.

Figure 66 shows some Classical PWM waves in an example where the Compare 0 register value = 8.

Figure 66. Classical PWM Waveforms (Compare 0 Register = 8)



INDUCTION MOTOR CONTROLLER (Cont'd)

Repetition Down-Counter

Both in Zerocentered and Classical working mode, the four Compare registers (one Compare 0 and three for the U, V and W phases) are updated when the PWM counter value is zero and the 8-bit Repetition Down-Counter has reached zero value by counting or by software programming (see PCR2 register).

This means that data transits from the Preload Compare registers to the Compare registers every N cycles of the PWM Counter, where N is the value of the 8-bit Repetition register (N=1, 2, .., 256).

7.4.3.4 Dead Time Generator

For each phase there is one 6-bit Dead Time generator.

It generates two output signals: h and l.

The h output signal is the same as the input phase signal except for the rising edge, which is delayed relatively to the input signal rising edge.

The l output signal is the opposite of the input phase signal except the rising edge which is delayed relatively to the input signal falling edge.

The delay is the same for each phase (U, V, W) and its value is:

delay = T x N

where T is the period of the Dead Time Generator input clock (INTCLK divided by 2) and N is the 6-bit number in the Dead Time register.

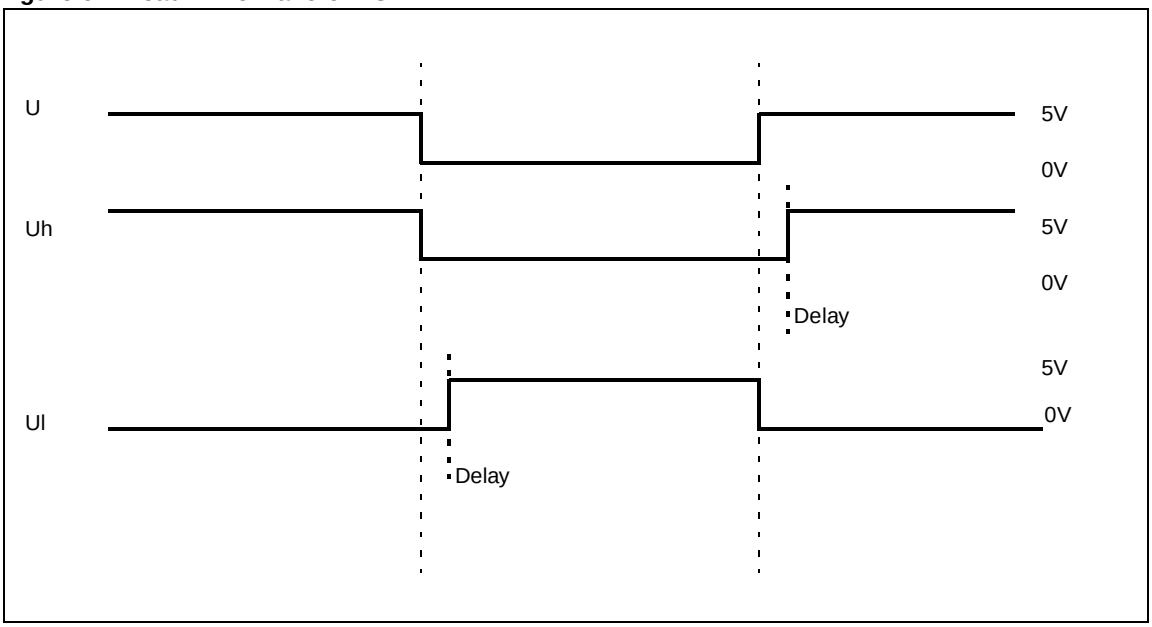
If the DTE bit in PCR0 register is reset, the Dead Time Generator is disabled. This means that no delays are added to the l complemented outputs.

Figure 67 shows an example waveform of the U phase.

If the delay is greater than the width of the active phase (l or h) then the corresponding pulse is not generated.

See Figure 68 and Figure 69.

Figure 67. Dead Time waveforms



INDUCTION MOTOR CONTROLLER (Cont'd)

Figure 68. Dead Time waveforms with delay greater than the negative PWM pulse

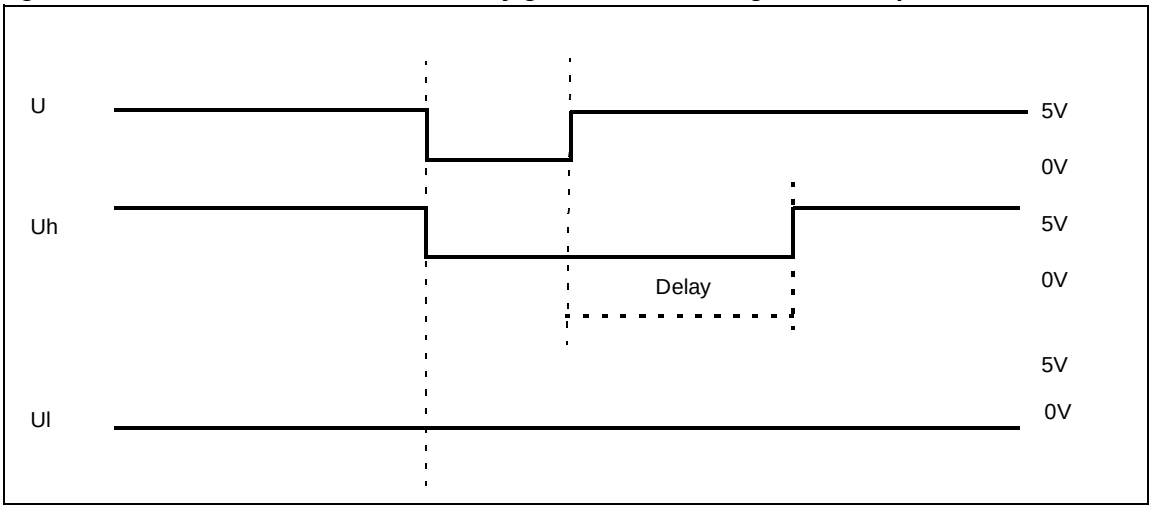
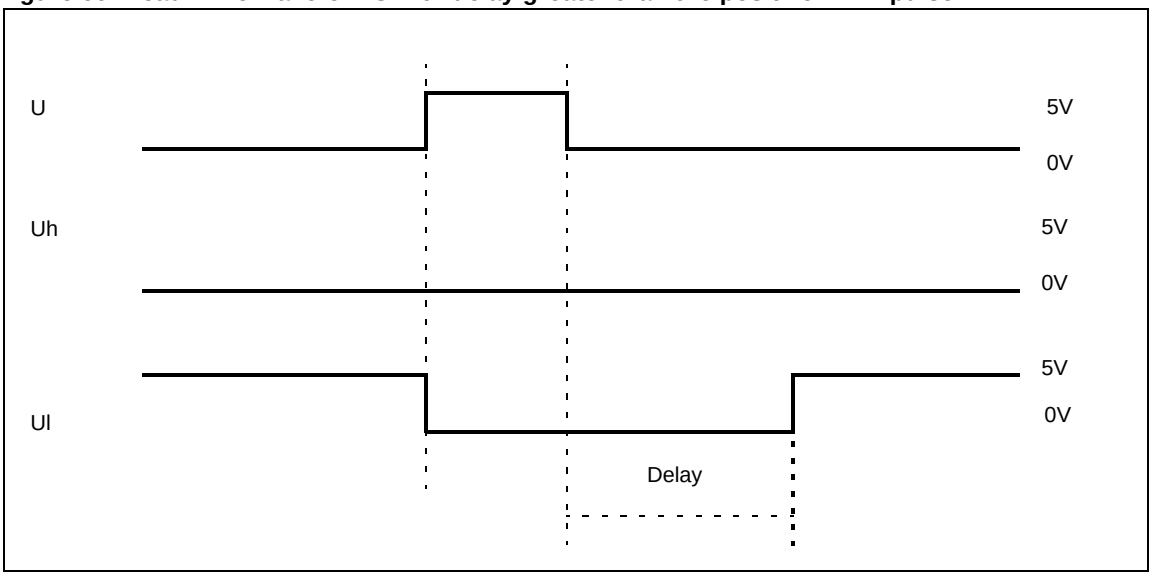


Figure 69. Dead Time waveforms with delay greater than the positive PWM pulse



INDUCTION MOTOR CONTROLLER (Cont'd)

7.4.3.5 Polarity Selection

The Polarity Selection performs a logical complement of the input signals (Uh, Ul, Vh, Vl, Wh, Wl) as programmed in the Polarity Selection register.

7.4.3.6 Interrupts

The IMC controller generates 8 interrupt requests and 1 NMI. Each interrupt request has a separate vector address. The NMI interrupt is managed by the ST9 as a Top Level Interrupt.The interrupt priority is fixed by hardware as listed below:

Priority	Interrupt Source
CPU Top Level Int. priority	**NMI:** Event on external pin
0 High Priority	**ADT:** Data transfer (when data is transferred from the preload registers to the compare registers)
1	**ZPC:** PWM Counter Zero Event
2	**CM0:** PWM Counter Compare 0 Event
3	**CPT:** Tacho Counter Capture Event
4	**CPU:** Compare U Event (PWM counter reached U compare value)
5	**CPV:** Compare V Event (PWM counter reached V compare value)
6	**CPW:** Compare W Event (PWM counter reached W compare value)
7 Low Priority	**OTC:** Tacho Counter Overflow

7.4.4 Tacho Counter Operating mode

The Tacho Counter can work in One Shot mode or in Continuous mode.

In both Continuous or One Shot mode the Capture event can be generated by hardware (TACHO Pin) or by software (STC bit in the PCR1 register) according to the value of the TES bit in the PCR1 register.

When the CTC bit in the PCR0 register is set, the TACHO Counter is cleared (this bit is reset by hardware).

7.4.4.1 Tacho Counter in One Shot mode

In this operating mode (TCB bit = 1 in the PCR1 register) the Counter does the following:

– Counting is started by setting the TCE bit in the PCR0 register.

– When a Capture event occurs, counting is stopped (TCE bit is cleared), the value is captured and a CPT interrupt is generated (if the CCPT bit in the PCR1 register is set, the Counter is cleared).

– When the MSB of Tacho Counter reaches the Tacho Compare register value, the Counter is stopped (TCE bit is cleared) and the OTC interrupt is generated.

7.4.4.2 Tacho Counter in Continuous mode

In this operating mode (TCB bit = 0 in the PCR1 register) the Counter does the following:

– Counting is started by setting the TCE bit in the PCR0 register.

– Every Capture event, the value is captured and a CPT interrupt is generated (if the CCPT bit in the PCR1 register is set, the Counter is cleared).

– When the MSB of Tacho Counter reaches the Tacho Compare register value, an OTC interrupt is generated.

7.4.5 IMC Operating mode

The IMC controller can work in two different modes:

– Hardware Operating mode (DTS bit = 0 in PRCR2 register)

– Software Operating mode (DTS bit = 1 in PRCR2 register)

In both modes, when the corresponding event occurs, the ADT and the other interrupts are generated.

When the CPC bit in the PCR0 register is set, the PWM Counter is cleared (this bit is reset by software).

7.4.5.1 IMC Hardware Operating mode

After system reset, the Compare U, V, W and Compare 0 register values are all "0".

When the PWM Counter is enabled (by setting the PCE bit in the PCR0 register) and every time the Repetition Counter and the PWM Counter reach "0" value, the Repetition Counter is loaded, the preload registers are loaded into the Compare registers and an ADT interrupt is generated.

Note: If an ADT (or any other interrupt) is generated and the previous one is not completed, the last one will be lost without any error condition being issued.

INDUCTION MOTOR CONTROLLER (Cont'd)

7.4.5.2 IMC Software Operating mode

In this operating mode, the Repetition register and any Compare register can be independently updated by software by setting the SDT bit in the PCR2 register (this bit will be reset by hardware) and the corresponding enable bit in the same register.

No hardware loading is performed when an ADT interrupt is generated.

Note: The Repetition Counter is decremented immediately when the Repetition Counter is updated.

7.4.6 IMC Output selection

The IMC Output can be selected from the following sources:

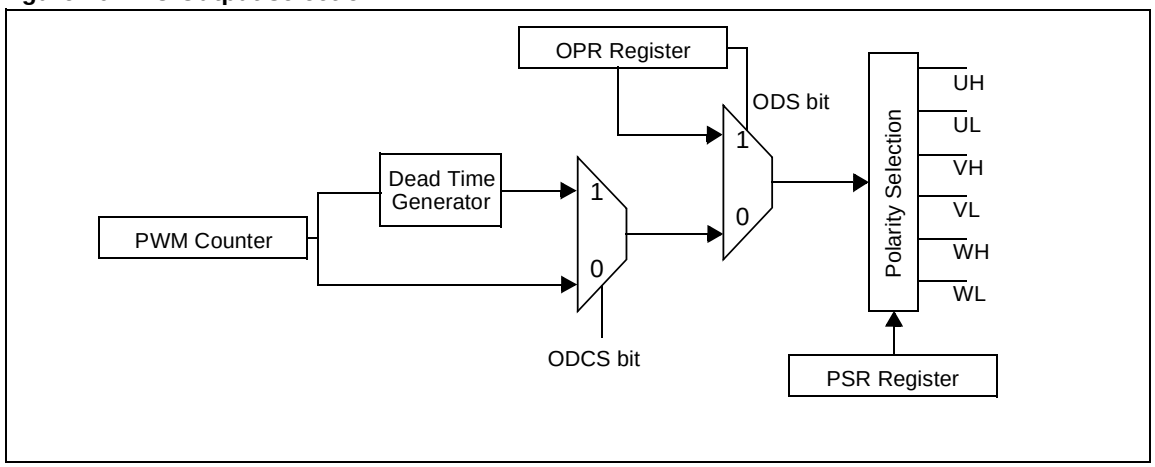
– OPR register (bit 5:0), by setting the ODS bit in the OPR register.

– Dead Time Generator outputs, by setting the ODCS bit in PCR0 register.

– PWM Counter outputs (h and l) are not complemented when the ODCS bit is reset.

Figure 70 shows the IMC output selection.

Figure 70. IMC Output selection.



INDUCTION MOTOR CONTROLLER (Cont'd)

7.4.7 NMI management

Figure 71 shows how the external input NMI signal is managed by the IMC peripheral.

After an ST9 reset, the NMIE bit in the PCR1 register is cleared, which means that NMI signal coming from the external pin is sent, as is, to the ST9 core without affecting the IMC peripheral.

If the NMIE bit is set, when an NMI event occurs on the external pin, it will be acknowledged (depending on the value of the NMIL bit in the PSR register). In this case:

– The NMI bit in the IMCIVR register is set

– The dedicated output pins of the IMC are put in high impedance

– A high level signal is sent to the ST9 Top Level interrupt

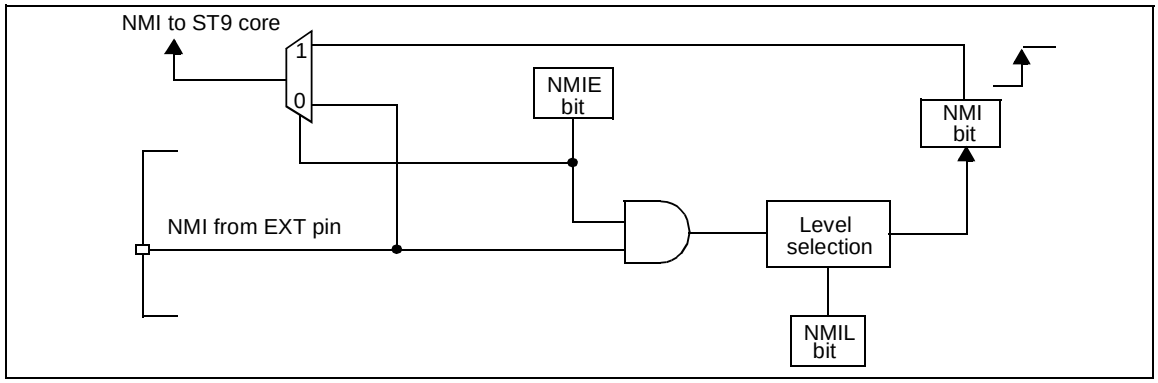
– The OPE bit is cleared (as the NMI interrupt signal is no longer active)

Notes:

1. Because the signal to the ST9 top level interrupt is active high, the TLTEV bit in the EIVR register must be set.

2. When the user wants to leave the NMI interrupt routine, it is strongly recommended to verify, before leaving the routine, that the NMI pending bit (bit 3 of IMCIVR) is really at "0". To do this, the user can try to write the NMI pending bit to "0" repeatedly until it has successfully been cleared. The NMI pending bit in IMCIVR register can be written to "0" only if the external NMI signal is no longer active. This makes sure that no NMI event will be lost. If the user leaves the NMI interrupt routine without clearing the NMI pending bit, no other NMI interrupt can be issued afterwards because the ST9 top level interrupt is edge sensitive.

Figure 71. NMI management by the IMC peripheral



INDUCTION MOTOR CONTROLLER (Cont'd)

7.4.8 Register Description

TACHO CAPTURE REGISTER HIGH (TCPTH)
R240 - Read Only
Register Page: 51
Reset Value: undefined

7							0
TCH7	TCH6	TCH5	TCH4	TCH3	TCH2	TCH1	TCH0

Bit 7:0 = **TCH[7:0]** *Most Significant Byte of Tacho Capture register.*

TACHO CAPTURE REGISTER LOW (TCPTL)
R241 - Read Only
Register Page: 51
Reset Value: undefined

7							0
TCL7	TCL6	TCL5	TCL4	TCL3	TCL2	TCL1	TCL0

Bit 7:0 = **TCL[7:0]** *Low Byte of Tacho Capture register.*

TACHO COMPARE REGISTER (TCMP)
R242 - Read/Write
Register Page: 51
Reset Value: 1111 1111 (FFh)

7							0
TCM7	TCM6	TCM5	TCM4	TCM3	TCM2	TCM1	TCM0

Bit 7:0 = **TCM[7:0]** *Byte of Tacho Compare register.*

When the Most Significant Byte of the Tacho Counter reaches TCMP value, the Tacho Counter is cleared and an OTC interrupt is generated both in Continuous and One Shot mode.

INDUCTION MOTOR CONTROLLER (Cont'd)

INTERRUPT PENDING REGISTER (IPR)
R243 - Read/Write
Register Page: 51
Reset Value: 0000 0000 (00h)

7							0
CM0	CPT	OTC	ADT	ZPC	CPU	CPV	CPW

Bit 7 = **CM0:** *Compare 0 of PWM pending bit.*
This bit is set by hardware when the PWM counter reaches the value in the Compare 0 register while CM0E=1. The CM0 bit must be cleared by software.
0: No CMP0 interrupt occurred
1: CMP0 interrupt pending

Bit 6 = **CPT:** *Capture of Tacho counter pending bit.*
This bit is set by hardware when a Tacho signal event occurs while CPTE=1. The CPT bit must be cleared by software.
0: No CPT interrupt occurred
1: CPT interrupt pending

Bit 5 = **OTC:** *Overflow of Tacho counter pending bit.*

This bit is set by hardware on a Tacho counter overflow while OTCE=1. The OTC bit must be cleared by software.
0: No OTC interrupt occurred
1: OTC interrupt pending

Bit 4 = **ADT:** *Automatic Data Transfer pending bit.*
This bit is set by hardware when data is transferred from the preload registers to the compare registers while ADTE=1. The ADT bit must be cleared by software.
0: No ADT interrupt occurred
1: ADT interrupt pending

Bit 3 = **ZPC:** *Zero of PWM counter pending bit.*

This bit is set by hardware when the PWM counter reaches zero while ZPCE=1. The ZPC bit must be cleared by software.
0: No ZPC interrupt occurred
1: ZPC interrupt pending

Bit 2 = **CPU:** *Compare U pending bit.*
In Classical Mode (CMS bit = 0), this bit is set by hardware when the PWM Counter reaches the Compare U register value while CPUE=1.

In Zerocentered Mode (CMS bit =1), this bit is set by hardware when the PWM Counter reaches the Compare U register value while CPUE=1 in up or downcounting (depending on the UDIS bit in the PSR register). The CPU bit must be cleared by software.
0: No CPU interrupt occurred
1: CPU interrupt pending

Bit 1 = **CPV:** *Compare V pending bit.*
In Classical Mode (CMS bit = 0), this bit is set by hardware when the PWM Counter reaches the Compare V register value while CPVE=1.

In Zerocentered Mode (CMS bit =1), this bit is set by hardware when the PWM Counter reaches the Compare V register value while CPVE=1 in up or downcounting (depending on the UDIS bit in the PSR register). The CPPRS register). The CPV bit must be cleared by software.
0: No CPV interrupt occurred
1: CPV interrupt pending

Bit 0 = **CPW:** *Compare W pending bit.*
In Classical Mode (CMS bit = 0), this bit is set by hardware when the PWM Counter reaches the Compare W register value while CPWE=1.

In Zerocentered Mode (CMS bit =1), this bit is set by hardware when the PWM Counter reaches the Compare W register value while CPWE=1 in up or downcounting (depending on the UDIS bit in the PSR register). The CPW bit must be cleared by software.

0: No CPW interrupt occurred
1: CPW interrupt pending

Note 1: None of the bits in the IPR register can be set by software, they can only be cleared.

Note 2: To clear the bits in the IPR register, the user must not use direct addressing bit instructions such as AND, OR, BRES, etc. because some interrupts may not be generated as expected. To avoid this, do the following:
To clear one pending bit of the IPR register, load the register with the mask corresponding to the bit to be cleared.

For example:

To clear the CPW bit, use the instruction
LD R243,#1111 1110
rather than the instruction
AND R243,#1111 1110.

INDUCTION MOTOR CONTROLLER (Cont'd)

TACHO PRESCALER REGISTER HIGH (TPRSH)
R244 - Read/Write
Register Page: 51
Reset Value: 0000 0000 (00h)

7							0
-	-	-	-	TPRH 3	TPRH 2	TPRH 1	TPRH 0

Bit 7:4 = Reserved.

Bit 3:0 = **TPRH[3:0]** *Most Significant Bits of tacho prescaler value (N).*

TACHO PRESCALER REGISTER LOW (TPRSL)
R245 - Read/Write
Register Page: 51
Reset Value: 0000 0000 (00h)

7							0
TPRL 7	TPRL 6	TPRL 5	TPRL 4	TPRL 3	TPRL 2	TPRL 1	TPRL 0

Bit 7:0 = **TPRL[7:0]** *Low byte of tacho prescaler value (N).*

If N = 0 Tacho Prescaler divides by 1.

PWM COUNTER PRESCALER REGISTER (CPRS)
R246 - Read/Write
Register Page: 51
Reset Value: 0000 0000 (00h)

7							0
CPR7	CPR6	CPR5	CPR4	CPR3	CPR2	CPR1	CPR0

Bit 7:0 = **CPR[7:0]** *PWM counter prescaler value (N).*

This value divides the INTCLK frequency by (N +1), i.e. if N = 0, INTCLK is divided by 1.

REPETITION COUNTER REGISTER (REP)
R247 - Read/Write
Register Page: 51
Reset Value: 0000 0000 (00h)

7							0
REP7	REP6	REP5	REP4	REP3	REP2	REP1	REP0

Bit 7:0 = **REP[7:0]** *Repetition counter value (N).*

If N = 0, each time the PWM Counter reaches zero, the Compare registers are updated and an ADT interrupt is generated.

INDUCTION MOTOR CONTROLLER (Cont'd)

COMPARE PHASE W PRELOAD REGISTER HIGH (CPWH)
R248 - Read/Write
Register Page: 51
Reset Value: 0000 0000 (00h)

7							0
CPWH 7	CPWH 6	CPWH 5	CPWH 4	CPWH 3	CPWH 2	CPWH 1	CPWH 0

Bit 7:0 = **CPWH[7:0]** *Most Significant Byte of phase W preload value*

COMPARE PHASE W PRELOAD REGISTER LOW (CPWL)
R249 - Read/Write
Register Page: 51
Reset Value: 0000 0000 (00h)

7							0
CPWL 7	CPWL 6	CPWL 5	-	-	-	-	-

Bit 7:5 = **CPWL[7:5]** *Low bits of phase W preload value*.

Bit 4:0 = Reserved.

COMPARE PHASE V PRELOAD REGISTER HIGH (CPVH)
R250 - Read/Write
Register Page: 51
Reset Value: 0000 0000 (00h)

7							0
CPVH7	CPVH6	CPVH5	CPVH4	CPVH3	CPVH2	CPVH1	CPVH0

Bit 7:0 = **CPVH[7:0]** *Most Significant Byte of phase V preload value*

COMPARE PHASE V PRELOAD REGISTER LOW (CPVL)
R251 - Read/Write
Register Page: 51
Reset Value: 0000 0000 (00h)

7							0
CPVL7	CPVL6	CPVL5	-	-	-	-	-

Bit 7:5 = **CPVL[7:5]** *Low bits of phase V preload value*.

Bit 4:0 = Reserved.

INDUCTION MOTOR CONTROLLER (Cont'd)

COMPARE PHASE U PRELOAD REGISTER HIGH (CPUH)
R252 - Read/Write
Register Page: 51
Reset Value: 0000 0000 (00h)

7							0
CPUH 7	CPUH 6	CPUH 5	CPUH 4	CPUH 3	CPUH 2	CPUH 1	CPUH 0

Bit 7:0 = **CPUH[7:0]** *Most Significant Byte of phase U preload value*

COMPARE PHASE U PRELOAD REGISTER LOW (CPUL)
R253 - Read/Write
Register Page: 51
Reset Value: 0000 0000 (00h)

7							0
CPUL7	CPUL6	CPUL5	-	-	-	-	-

Bit 7:5 = **CPUL[7:5]** *Low bits of phase U preload value*.

Bit 4:0 = Reserved.

COMPARE 0 PRELOAD REGISTER HIGH (CP0H)
R254 - Read/Write
Register Page: 51
Reset Value: 0000 0000 (00h)

7							0
-	-	-	-	-	-	CP0H1	CP0H0

Bit 7:2 = Reserved.

Bit 1:0 = **CP0H[1:0]** *Most Significant Bits of Compare 0 preload value.*

COMPARE 0 PRELOAD REGISTER LOW (CP0L)
R255 - Read/Write
Register Page: 51
Reset Value: 0000 0000 (00h)

7							0
CP0L7	CP0L6	CP0L5	CP0L4	CP0L3	CP0L2	CP0L1	CP0L0

Bit 7:0 = **CP0L[7:0]** *Low byte of Compare 0 preload value*.

Note: CP0[9:0] value must be greater than 1.

INDUCTION MOTOR CONTROLLER (Cont'd)

PERIPHERAL CONTROL REGISTER 0 (PCR0)
R248 - Read/Write
Register Page: 48
Reset Value: 1000 0011 (83h)

7							0
DTE	TCE	PCE	CTC	CPC	CMS	UDCS	ODCS

Bit 7 = **DTE:** *Dead Time Counter Enable.*
0: Stop and bypass the Dead Time counter
1: Enable the Dead Time counter

Bit 6 = **TCE:** *Tacho Counter Enable.*
0: Stop Tacho counter and prescaler
1: Start Tacho counter and prescaler

Note: This bit is reset by the counter overflow or by the Tacho capture when the IMC controller is in one shot mode.

Bit 5 = **PCE:** *PWM Counter Enable.*
0: Stop PWM Counter and prescaler
1: Start PWM Counter and prescaler

Bit 4 = **CTC:** *Clear of Tacho Counter.*
0: No effect
1: Clear the Tacho Counter (this bit is reset by hardware)

Bit 3 = **CPC:** *Clear of PWM Counter.*
0: No effect.
1: Clear the PWM Counter (his bit is reset by hardware)

Bit 2 = **CMS:** *PWM Counter Mode Selection.*
0: Classical mode.
1: Zerocentered mode

Bit 1 = **UDCS:** *Up/Down - status (read only).*
This bit is set and cleared by hardware.
0: The PWM Counter is counting down.
1: The PWM Counter is counting up.

Bit 0 = **ODCS:** *Output Dead Time counter Selection*
0: Select the same signal for both (h, l) outputs
1: Select complementary signal for output (Dead Time Generator outputs)

PERIPHERAL CONTROL REGISTER 1 (PCR1)
R249 - Read/Write
Register Page: 48
Reset Value: 0000 0000 00h

7							0
-	NMIE	CCPT	TES	STC	TCB	TIN1	TIN0

Bit 7 = Reserved.

Bit 6 = **NMIE:** *Non Maskable Interrupt Enable*
0: When an NMI event occurs on the external pin, it is sent as is (independently of the NMIL value) to the ST9 core and has no effect on the IMC controller.
1: When an NMI event occurs on the external pin, if it is acknowledged (depending on the NMIL bit) an interrupt request is sent to the ST9 core (a high level signal), the NMI pending bit (NMI bit in IMCIVR register) is set and the OPE bit is cleared.

Bit 5 = **CCPT:** *Clear on Capture of tacho counter*
0: no clear on capture
1: clear on capture

Bit 4 = **TES:** *Tacho Event Selection.*
0: Select capture by tacho event signal
1: Select capture by software (STC bit)

Bit 3 = **STC:** *Software tacho capture*
0: No effect
1: Capture the Tacho counter (while TES=1). This bit is reset by hardware

Bit 2 = **TCB:** *Tacho Counter Mode*
0: Select continuous mode.
1: Select one shot mode (counting starts when TCE bit is set and stops when a capture or an overflow event occurs).

Bit 1:0 = **TIN[1:0]** *Tacho Signal Event Sensitivity*

These bits select which Tacho signal event triggers the Tacho Capture register.

Table 25. Tacho Signal Event Sensitivity

TIN1	TIN0	Event
0	0	No operation
0	1	Falling edge
1	0	Rising edge
1	1	Rising and falling edges

INDUCTION MOTOR CONTROLLER (Cont'd))

PERIPHERAL CONTROL REGISTER 2 (PCR2)
R250 - Read/Write
Register Page: 48
Reset Value: 0000 0000 (00h)

7							0
GPIE	RSE	CWSE	CVSE	CUSE	C0SE	SDT	DTS

Bit 7 = **GPIE:** *Global Peripheral Interrupt Enable.*
0: Disable all IMC controller interrupts.
1: Enable all IMC controller interrupts.

Bit 6 = **RSE:** *Enable Software Data Transfer to Repetition register.*
0: Disable loading of Repetition register by SDT bit
1: Enable loading of Repetition register by SDT bit

Bit 5 = **CWSE:** *Enable Software Data Transfer to Compare W.*
0: Disable load of Compare W register by SDT bit
1: Enable load of Compare W register by SDT bit

Bit 4 = **CVSE:** *Enable Software Data Transfer to Compare V register.*
0: Disable loading of Compare V register by SDT bit
1: Enable loading of Compare V register by SDT bit

Bit 3 = **CUSE:** *Enable Software Data Transfer to Compare U register.*
0: Disable loading of Compare U register by SDT bit
1: Enable loading of Compare U register by SDT bit

Bit 2 = **C0SE:** *Enable Software Data Transfer to Compare 0 register.*
0: Disable loading of Compare 0 register by SDT bit
1: Enable loading of Compare 0 register by SDT bit

Bit 1 = **SDT:** *Software Data Transfer*
0: No effect
1: Transfer Data from preload to compare register (while DTS=1) (This bit is reset by hardware).

Bit 0 = **DTS:** *Data Transfer Mode Selection.*
0: Hardware transfer using Repetition counter
1: Software transfer using SDT bit.

POLARITY SELECTION REGISTER (PSR)
R251 - Read/Write
Register Page: 48
Reset Value: 0000 0000 (00h)

7							0
NMIL	UDIS	PUH	PUL	PVH	PVL	PWH	PWL

Bit 7 = **NMIL:** *Non Maskable Interrupt Level.*
0: Low level of NMI event is acknowledged.
1: High level of NMI event is acknowledged.

Bit 6 = **UDIS:** *Up-Down Interrupt Select.*

When the PWM Counter is working in Zerocentered Mode the meaning is:
0: The Compare interrupts (CPU, CPV, CPW) are issued when the Counter is counting up.
1: The Compare interrupts (CPU, CPV, CPW) are issued when the Counter is counting down.

This bit has no effect when the Counter is working in Classical Mode

Bit 5 = **PUH:** *Polarity of Uh phase.*
0: Positive logical level.
1: Complemented logical level.

Bit 4 = **PUL:** *Polarity of Ul phase.*
0: Positive logical level.
1: Complemented logical level.

Bit 3 = **PVH:** *Polarity of Vh phase.*
0: Positive logical level.
1: Complemented logical level.

Bit 2 = **PVL:** *Polarity of Vl phase.*
0: Positive logical level.
1: Complemented logical level.

Bit 1 = **PWH:** *Polarity of Wh phase.*
0: Positive logical level.
1: Complemented logical level.

Bit 0: **PWL:** *Polarity of Wl phase.*
0: Positive logical level.
1: Complemented logical level.

INDUCTION MOTOR CONTROLLER (Cont'd)

OUTPUT PERIPHERAL REGISTER (OPR)
R252 - Read/Write
Register Page: 48
Reset Value: 0000 0000 (00h)

7							0
OPE	ODS	UH	UL	VH	VL	WH	WL

Bit 7 = **OPE:** *Output Port Enable.*

This bit can be set by software only if the NMI bit is cleared.
0: Output port is in high impedance (without pull-up and pull-down resistor).
1: Output port is available for data transfer.

Bit 6 = **ODS:** *Output Data Selection.*
0: Dead time generator data.
1: Select the Bit 5:0 data

Bit 5:0 = *Uh, Ul, Vh, Vl, Wh, Wl phases*

These bits can be sent out through the output port.

INTERRUPT MASK REGISTER (IMR)
R253 - Read/Write
Register Page: 48
Reset Value: 0000 0000 (00h)

7							0
CM0E	CPTE	OTCE	ADTE	ZPCE	CPUE	CPVE	CPWE

Bit 7 = **CM0E:** *Compare 0 of PWM counter enable.*
0: Disabled.
1: Enabled.

Bit 6 = **CPTE:** *Capture of Tacho counter Interrupt enable.*
0: Disabled.
1: Enabled.

Bit 5 = **OTCE:** *Overflow of Tacho counter Interrupt enable.*
0: Disabled.
1: Enabled.

Bit 4 = **ADTE:** *Automatic data transfer Interrupt enable.*
0: Disabled.
1: Enabled.

Bit 3 = **ZPCE:** *Zero of PWM counter interrupt enable*.
0: Disabled.
1: Enabled.

Bit 2:0 = **CPUE, CPVE, CPWE:** *Compare U, V, W interrupt enable*.
0: Disabled.
1: Enabled.

INDUCTION MOTOR CONTROLLER (Cont'd)

DEAD TIME GENERATOR REGISTER (DTG)
R254 - Read/Write
Register Page: 48
Reset Value: 0011 1111 (3Fh)

7							0
-	-	DTG5	DTG4	DTG3	DTG2	DTG1	DTG0

Bit 7:6 = Reserved.

Bit 5:0 = **DTG[5:0]** *Dead time generator value (N).*
The delay is N x INTCLK period multiplied by 2.
If N = 0 the delay is 0.

IMC INTERRUPT VECTOR REGISTER (IMCIVR)
R255 - Read/Write
Register Page: 48
Reset Value: undefined (17h)

7							0
V3	V2	V1	V0	NMI	PL2	PL1	PL0

Bit 7:4 = **V[3:0]:** *Interrupt Vector Base Address.*
User programmable interrupt vector bits.
The most significant nibble of the interrupt vector address is given by V[3:0]. The other nibble is given by W[3:0] where W[0] is forced to '0' and W[3:1] are set by hardware according to the Table 26 Interrupt Source Address.

Table 26. Interrupt Source Address

W[3:1]	Interrupt Source
000	ADT Data transfer
001	ZPC Zero event of PWM counter
010	CM0 PWM counter Compare 0
011	CPT Tacho capture
100	CPU Compare U
101	CPV Compare V
110	CPW Compare W
111	OTC Tacho counter overflow

Bit 3 = **NMI:** *Non Maskable Interrupt pending bit.*
This bit is set by hardware when an NMI event occurs and the NMIE bit = 1. The NMI bit can be cleared by software. It cannot be set by software. As long as this bit is '1' the NMI signal to ST9 is kept active (high level).
0: No NMI pending
1: NMI pending

Note: As long as the external NMI signal is active the NMI bit can not be reset.

Bit 2:0 = **PL[2:0]**: Priority level for the peripheral interrupt.

INDUCTION MOTOR CONTROLLER (Cont'd)

Table 27. IMC Controller Register Map

Page 51 Register No.	Register Name	7	6	5	4	3	2	1	0
R240	**TCPTH**	TCH7	TCH6	TCH5	TCH4	TCH3	TCH2	TCH1	TCH0
R241	**TCPTL**	TCL7	TCL6	TCL5	TCL4	TCL3	TCL2	TCL1	TCL0
R242	**TCMP**	TCP7	TCP6	TCP5	TCP4	TCP3	TCP2	TCP1	TCP0
R243	**IPR**	CM0	CPT	OTC	ADT	ZPC	CPU	CPV	CPW
R244	**TPRSH**	-	-	-	-	TPH3	TPH2	TPH1	TPH0
R245	**TPRSL**	TPRL7	TPRL6	TPRL5	TPRL4	TPRL3	TPRL2	TPRL1	TPRL0
R246	**CPRS**	CPR7	CPR6	CPR5	CPR4	CPR3	CPR2	CPR1	CPR0
R247	**REP**	REP7	REP6	REP5	REP4	REP3	REP2	REP1	REP0
R248	**CPWH**	CPWH7	CPWH6	CPWH5	CPWH4	CPWH3	CPWH2	CPWH1	CPWH0
R249	**CPWL**	CPWL7	CPWL6	CPWL5	-	-	-	-	-
R250	**CPVH**	CPVH7	CPVH6	CPVH5	CPVH4	CPVH3	CPVH2	CPVH1	CPVH0
R251	**CPVL**	CPVL7	CPVL6	CPVL5	-	-	-	-	-
R252	**CPUH**	CPUH7	CPUH6	CPUH5	CPUH4	CPUH3	CPUH2	CPUH1	CPUH0
R253	**CPUL**	CPU7L	CPUL6	CPUL5	-	-	-	-	-
R254	**CP0H**	-	-	-	-	-	-	CP0H1	CP0H0
R255	**CP0L**	CP0L7	CP0L6	CP0L5	CP0L4	CP0L3	CP0L2	CP0L1	CP0L0
Page 48 Register No.	**Register Name**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
R248	**PCR0**	DTE	TCE	PCE	CTC	CPC	CMS	UDCS	ODCS
R249	**PCR1**	-	NMIE	CCPT	TES	STC	TCB	TIN1	TIN0
R250	**PCR2**	GPIE	RSE	CWSE	CVSE	CUSE	COSE	SDT	DTS
R251	**PSR**	NMIL	UDIS	PUH	PUL	PVH	PVL	PWH	PWL
R252	**OPR**	OPE	ODS	UH	UL	VH	VL	WH	WL
R253	**IMR**	CM0E	CPTE	OTCE	ADTE	ZPCE	CPUE	CPVE	CPWE
R254	**DTG**	-	-	DTG5	DTG4	DTG3	DTG2	DTG1	DTG0
R255	**IMCIVR**	V3	V2	V1	V0	NMI	PL2	PL1	PL0

7.5 SERIAL PERIPHERAL INTERFACE (SPI)

7.5.1 Introduction

The Serial Peripheral Interface (SPI) allows full-duplex, synchronous, serial communication with external devices. An SPI system may consist of a master and one or more slaves or a system in which devices may be either masters or slaves.

The SPI is normally used for communication between the microcontroller and external peripherals or another Microcontroller.

Refer to the Pin Description chapter for the device-specific pin-out.

7.5.2 Main Features

- Full duplex, three-wire synchronous transfers
- Master or slave operation
- Four master mode frequencies
- Maximum slave mode frequency = INTCLK/2.
- Fully programmable 3-bit prescaler for a wide range of baud rates, plus a programmable divider by 2
- Programmable clock polarity and phase
- End of transfer interrupt flag
- Write collision flag protection
- Master mode fault protection capability.

7.5.3 General Description

The SPI is connected to external devices through 4 alternate pins:

- MISO: Master In Slave Out pin
- MOSI: Master Out Slave In pin
- SCK: Serial Clock pin
- $\overline{SS}$: Slave select pin

To use any of these alternate functions (input or output), the corresponding I/O port must be programmed as alternate function output.

A basic example of interconnections between a single master and a single slave is illustrated on Figure 72.

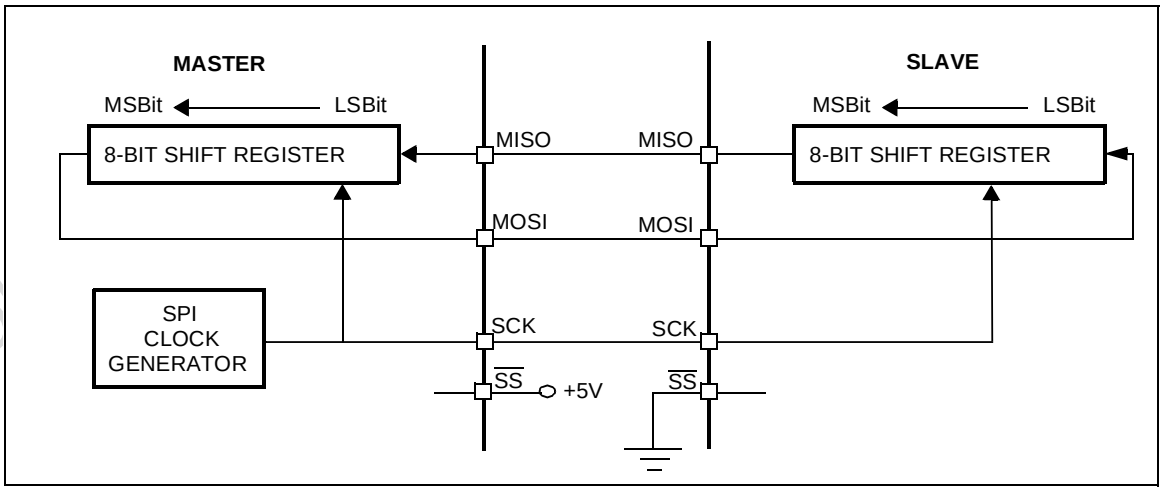
The MOSI pins are connected together as are MISO pins. In this way data is transferred serially between master and slave (most significant bit first).

When the master device transmits data to a slave device via MOSI pin, the slave device responds by sending data to the master device via the MISO pin. This implies full duplex transmission with both data out and data in synchronized with the same clock signal (which is provided by the master device via the SCK pin).

Thus, the byte transmitted is replaced by the byte received and eliminates the need for separate transmit-empty and receiver-full bits. A status flag is used to indicate that the I/O operation is complete.

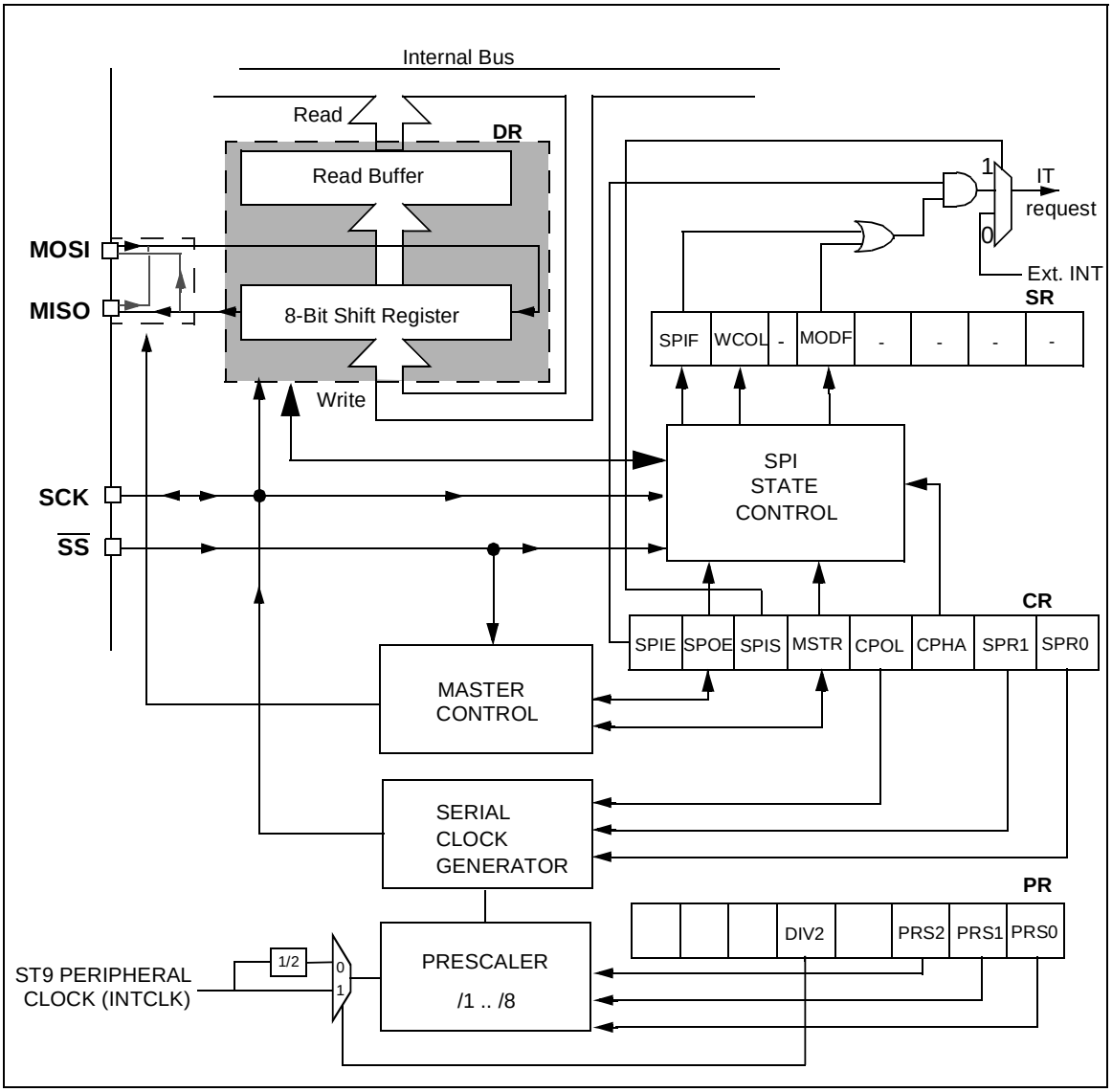
Four possible data/clock timing relationships may be chosen (see Figure 75) but master and slave must be programmed with the same timing mode.

Figure 72. Serial Peripheral Interface Master/Slave



SERIAL PERIPHERAL INTERFACE (Cont'd)

Figure 73. Serial Peripheral Interface Block Diagram



SERIAL PERIPHERAL INTERFACE (Cont'd)

7.5.4 Functional Description

Figure 73 shows the serial peripheral interface (SPI) block diagram.

This interface contains 4 dedicated registers:

- A Control Register (CR)
- A Prescaler Register (PR)
- A Status Register (SR)
- A Data Register (DR)

Refer to the CR, PR, SR and DR registers in Section 7.5.6for the bit definitions.

7.5.4.1 Master Configuration

In a master configuration, the serial clock is generated on the SCK pin.

Procedure

- Define the serial clock baud rate by setting/resetting the DIV2 bit of PR register, by writing a prescaler value in the PR register and programming the SPR0 & SPR1 bits in the CR register.
- Select the CPOL and CPHA bits to define one of the four relationships between the data transfer and the serial clock (see Figure 75).
- The $\overline{SS}$ pin must be connected to a high level signal during the complete byte transmit sequence.
- The MSTR and SPOE bits must be set (they remain set only if the $\overline{SS}$ pin is connected to a high level signal).

In this configuration the MOSI pin is a data output and to the MISO pin is a data input.

Transmit Sequence

The transmit sequence begins when a byte is written the DR register.

The data byte is parallel loaded into the 8-bit shift register (from the internal bus) during a write cycle and then shifted out serially to the MOSI pin most significant bit first.

When data transfer is complete:

- The SPIF bit is set by hardware
- An interrupt is generated if the SPIS and SPIE bits are set.

During the last clock cycle the SPIF bit is set, a copy of the data byte received in the shift register is moved to a buffer. When the DR register is read, the SPI peripheral returns this buffered value.

Clearing the SPIF bit is performed by the following software sequence:

1. An access to the SR register while the SPIF bit is set
2. A write or a read of the DR register.

Note: While the SPIF bit is set, all writes to the DR register are inhibited until the SR register is read.

SERIAL PERIPHERAL INTERFACE (Cont'd)

7.5.4.2 Slave Configuration

In slave configuration, the serial clock is received on the SCK pin from the master device.

The value of the PR register and SPR0 & SPR1 bits in the CR is not used for the data transfer.

Procedure

- For correct data transfer, the slave device must be in the same timing mode as the master device (CPOL and CPHA bits). See Figure 75.
- The $\overline{SS}$ pin must be connected to a low level signal during the complete byte transmit sequence.
- Clear the MSTR bit and set the SPOE bit to assign the pins to alternate function.

In this configuration the MOSI pin is a data input and the MISO pin is a data output.

Transmit Sequence

The data byte is parallel loaded into the 8-bit shift register (from the internal bus) during a write cycle and then shifted out serially to the MISO pin most significant bit first.

The transmit sequence begins when the slave device receives the clock signal and the most significant bit of the data on its MOSI pin.

When data transfer is complete:

- The SPIF bit is set by hardware
- An interrupt is generated if the SPIS and SPIE bits are set.

During the last clock cycle the SPIF bit is set, a copy of the data byte received in the shift register is moved to a buffer. When the DR register is read, the SPI peripheral returns this buffered value.

Clearing the SPIF bit is performed by the following software sequence:

1. An access to the SR register while the SPIF bit is set.
2. A write or a read of the DR register.

Notes: While the SPIF bit is set, all writes to the DR register are inhibited until the SR register is read.

The SPIF bit can be cleared during a second transmission; however, it must be cleared before the second SPIF bit in order to prevent an overrun condition (see Section 7.5.4.6).

Depending on the CPHA bit, the $\overline{SS}$ pin has to be set to write to the DR register between each data byte transfer to avoid a write collision (see Section 7.5.4.4).

SERIAL PERIPHERAL INTERFACE (Cont'd)

7.5.4.3 Data Transfer Format

During an SPI transfer, data is simultaneously transmitted (shifted out serially) and received (shifted in serially). The serial clock is used to synchronize the data transfer during a sequence of eight clock pulses.

The $\overline{SS}$ pin allows individual selection of a slave device; the other slave devices that are not selected do not interfere with the SPI transfer.

Clock Phase and Clock Polarity

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits.

The CPOL (clock polarity) bit controls the steady state value of the clock when no data is being transferred. This bit affects both master and slave modes.

The combination between the CPOL and CPHA (clock phase) bits selects the data capture clock edge.

Figure 75, shows an SPI transfer with the four combinations of the CPHA and CPOL bits. The diagram may be interpreted as a master or slave timing diagram where the SCK pin, the MISO pin, the MOSI pin are directly connected between the master and the slave device.

The $\overline{SS}$ pin is the slave device select input and can be driven by the master device.

The master device applies data to its MOSI pin-clock edge before the capture clock edge.

CPHA Bit is Set

The second edge on the SCK pin (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set) is the MSBit capture strobe. Data is latched on the occurrence of the first clock transition.

No write collision should occur even if the $\overline{SS}$ pin stays low during a transfer of several bytes (see Figure 74).

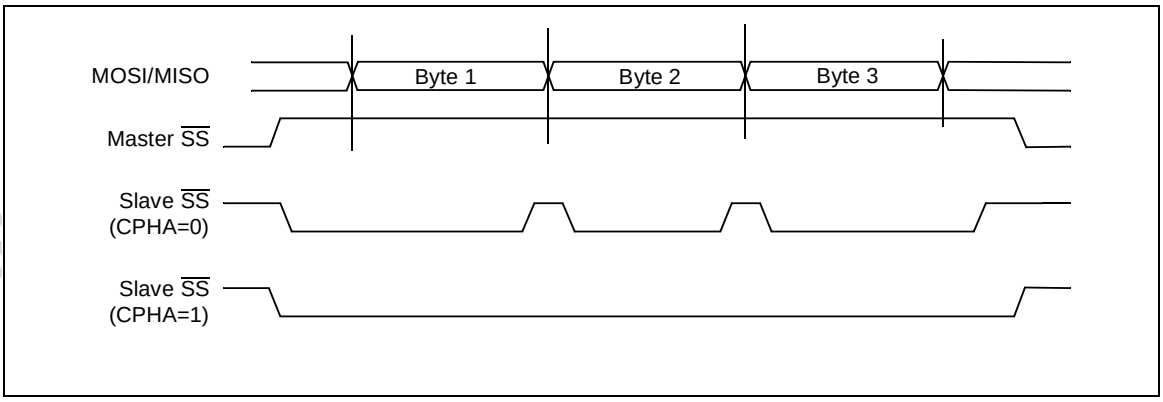
CPHA Bit is Reset

The first edge on the SCK pin (falling edge if CPOL bit is set, rising edge if CPOL bit is reset) is the MSBit capture strobe. Data is latched on the occurrence of the second clock transition.

This pin must be toggled high and low between each byte transmitted (see Figure 74).

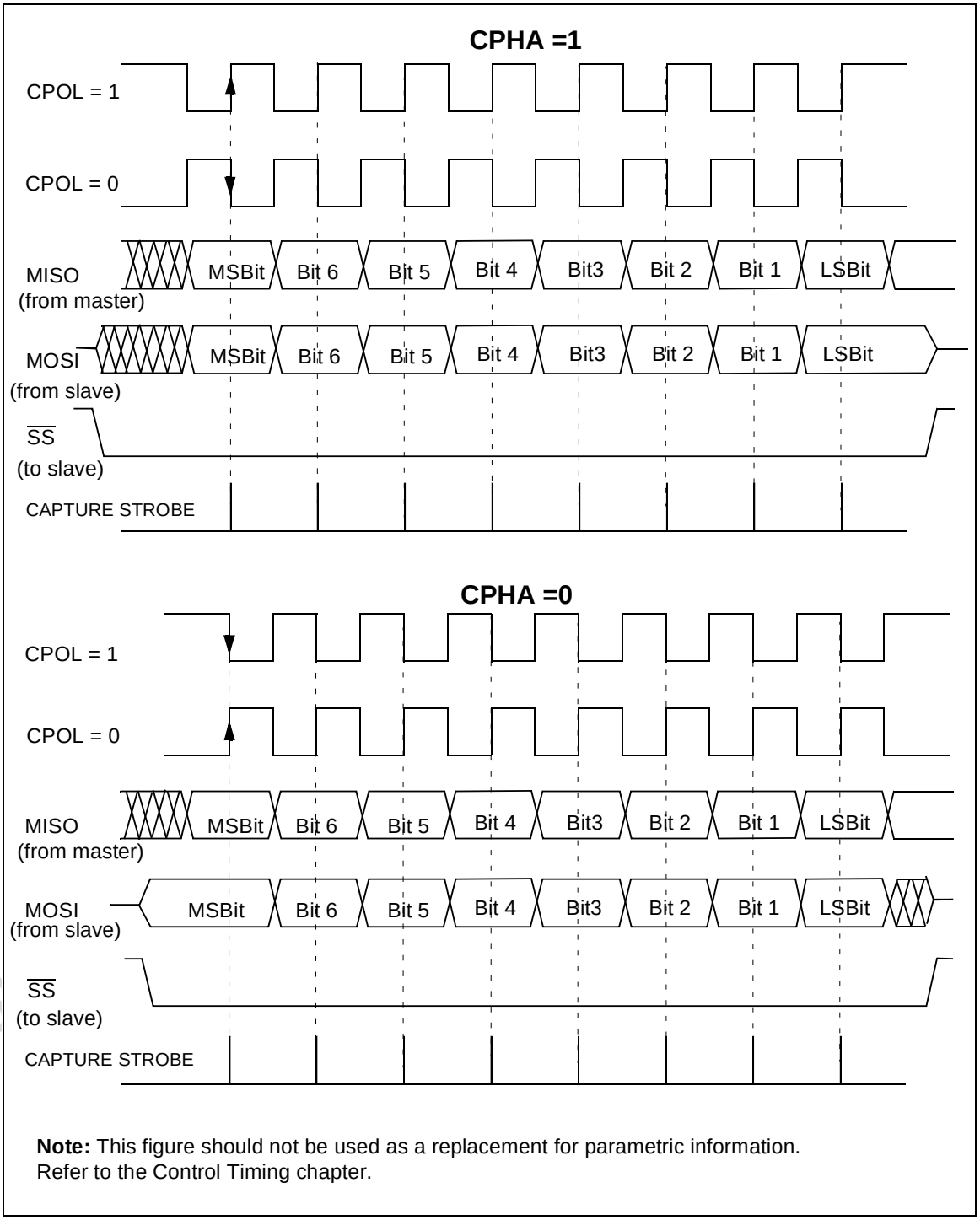
To protect the transmission from a write collision a low value on the $\overline{SS}$ pin of a slave device freezes the data in its DR register and does not allow it to be altered. Therefore the $\overline{SS}$ pin must be high to write a new data byte in the DR without producing a write collision.

Figure 74. CPHA / $\overline{SS}$ Timing Diagram



SERIAL PERIPHERAL INTERFACE (Cont'd)

Figure 75. Data Clock Timing Diagram



Note: This figure should not be used as a replacement for parametric information. Refer to the Control Timing chapter.

SERIAL PERIPHERAL INTERFACE (Cont'd)

7.5.4.4 Write Collision Error

A write collision occurs when the software tries to write to the DR register while a data transfer is taking place with an external device. When this happens, the transfer continues uninterrupted and the software write will be unsuccessful.

Write collisions can occur both in master and slave mode.

Note: a "read collision" will never occur since the received data byte is placed in a buffer in which access is always synchronous with the MCU operation.

In Slave mode

When the CPHA bit is set:

The slave device will receive a clock (SCK) edge prior to the latch of the first data transfer. This first clock edge will freeze the data in the slave device DR register and output the MSBit on to the external MISO pin of the slave device.

The $\overline{SS}$ pin low state enables the slave device but the output of the MSBit onto the MISO pin does not take place until the first data transfer clock edge.

When the CPHA bit is reset:

Data is latched on the occurrence of the first clock transition. The slave device does not have any way of knowing when that transition will occur; therefore, the slave device collision occurs when software attempts to write the DR register after its $\overline{SS}$ pin has been pulled low.

For this reason, the $\overline{SS}$ pin must be high, between each data byte transfer, to allow the CPU to write in the DR register without generating a write collision.

In Master mode

Collision in the master device is defined as a write of the DR register while the internal serial clock (SCK) is in the process of transfer.

The $\overline{SS}$ pin signal must be always high on the master device.

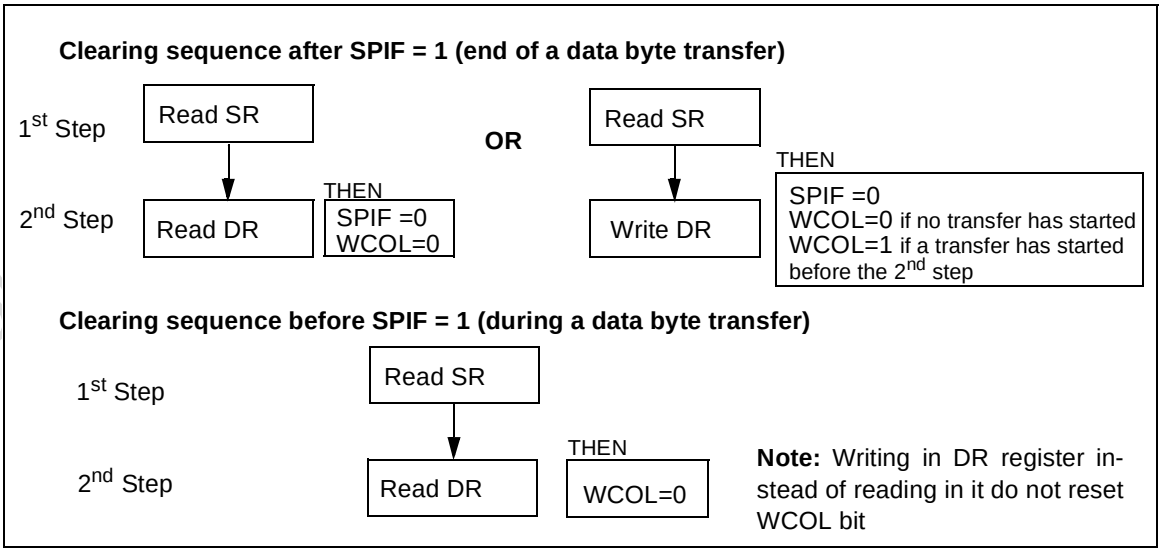
WCOL Bit

The WCOL bit in the SR register is set if a write collision occurs.

No SPI interrupt is generated when the WCOL bit is set (the WCOL bit is a status flag only).

Clearing the WCOL bit is done through a software sequence (see Figure 76).

Figure 76. Clearing the WCOL bit (Write Collision Flag) Software Sequence



SERIAL PERIPHERAL INTERFACE (Cont'd)

7.5.4.5 Master Mode Fault

Master mode fault occurs when the master device has its $\overline{SS}$ pin pulled low, then the MODF bit is set.

Master mode fault affects the SPI peripheral in the following ways:

- The MODF bit is set and an SPI interrupt is generated if the SPIE bit is set.
- The SPOE bit is reset. This blocks all output from the device and disables the SPI peripheral.
- The MSTR bit is reset, thus forcing the device into slave mode.

Clearing the MODF bit is done through a software sequence:

1. A read or write access to the SR register while the MODF bit is set.
2. A write to the CR register.

Notes: To avoid any multiple slave conflicts in the case of a system comprising several MCUs, the $\overline{SS}$ pin must be pulled high during the clearing sequence of the MODF bit. The SPOE and MSTR bits may be restored to their original state during or after this clearing sequence.

Hardware does not allow the user to set the SPOE and MSTR bits while the MODF bit is set except in the MODF bit clearing sequence.

In a slave device the MODF bit can not be set, but in a multi master configuration the device can be in slave mode with this MODF bit set.

The MODF bit indicates that there might have been a multi-master conflict for system control and allows a proper exit from system operation to a reset or default system state using an interrupt routine.

7.5.4.6 Overrun Condition

An overrun condition occurs, when the master device has sent several data bytes and the slave device has not cleared the SPIF bit issuing from the previous data byte transmitted.

In this case, the receiver buffer contains the byte sent after the SPIF bit was last cleared. A read to the DR register returns this byte. All other bytes are lost.

This condition is not detected by the SPI peripheral.

SERIAL PERIPHERAL INTERFACE (Cont'd)

7.5.4.7 Single Master and Multimaster Configurations

There are two types of SPI systems:

– Single Master System

– Multimaster System

Single Master System

A typical single master system may be configured, using an MCU as the master and four MCUs as slaves (see Figure 77).

The master device selects the individual slave devices by using four pins of a parallel port to control the four $\overline{SS}$ pins of the slave devices.

The $\overline{SS}$ pins are pulled high during reset since the master device ports will be forced to be inputs at that time, thus disabling the slave devices.

Note: To prevent a bus conflict on the MISO line the master allows only one slave device during a transmission.

For more security, the slave device may respond to the master with the received data byte. Then the master will receive the previous byte back from the slave device if all MISO and MOSI pins are connected and the slave has not written its DR register.

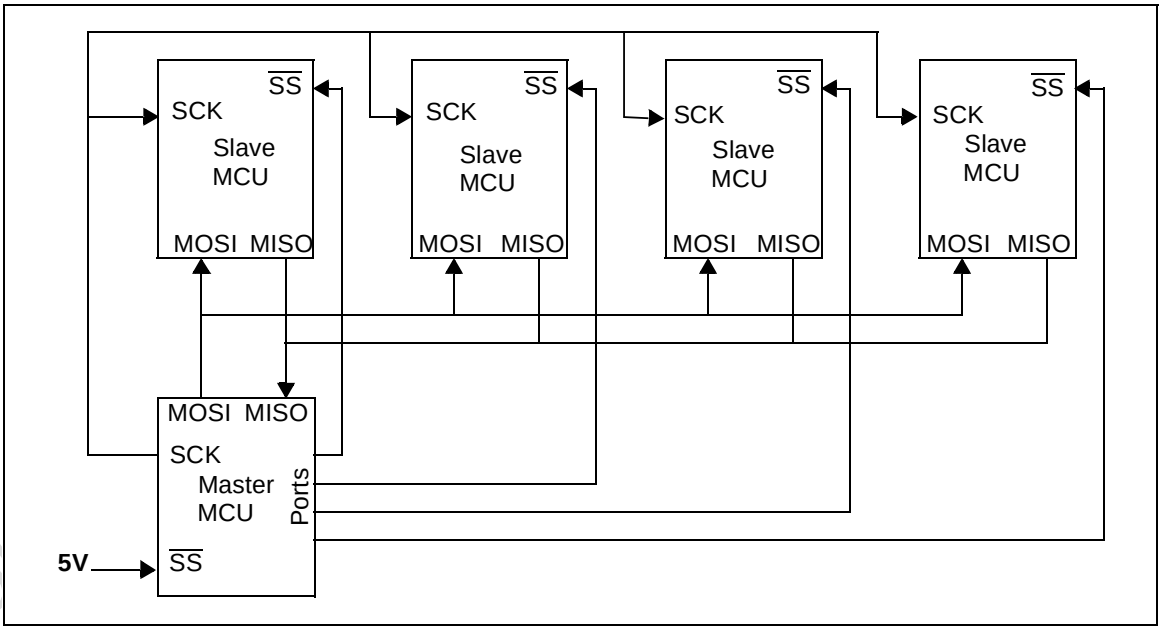
Other transmission security methods can use ports for handshake lines or data bytes with command fields.

Multi-Master System

A multi-master system may also be configured by the user. Transfer of master control could be implemented using a handshake method through the I/O ports or by an exchange of code messages through the serial peripheral interface system.

The multi-master system is principally handled by the MSTR bit in the CR register and the MODF bit in the SR register.

Figure 77. Single Master Configuration



SERIAL PERIPHERAL INTERFACE (Cont'd)

7.5.5 Interrupt Management

The interrupt of the Serial Peripheral Interface is mapped on one of the eight External Interrupt Channels of the microcontroller (refer to the "Interrupts" chapter).

Each External Interrupt Channel has:

– A trigger control bit in the EITR register (R242 - Page 0),

– A pending bit in the EIPR register (R243 - Page0),

– A mask bit in the EIMR register (R244 - Page 0).

Program the interrupt priority level using the EIPLR register (R245 - Page 0). For a description of these registers refer to the "Interrupts" and "DMA" chapters.

To use the interrupt feature, perform the following sequence:

– Set the priority level of the interrupt channel used for the SPI (EIPRL register)

– Select the interrupt trigger edge as rising edge (set the corresponding bit in the EITR register)

– Set the SPIS bit of the CR register to select the peripheral interrupt source

– Set the SPIE bit of the CR register to enable the peripheral to perform interrupt requests

– In the EIPR register, reset the pending bit of the interrupt channel used by the SPI interrupt to avoid any spurious interrupt requests being performed when the mask bit is set

– Set the mask bit of the interrupt channel used to enable the MCU to acknowledge the interrupt requests of the peripheral.

Note: In the interrupt routine, reset the related pending bit to avoid the interrupt request that was just acknowledged being proposed again.
Then, after resetting the pending bit and before the IRET instruction, check if the SPIF and MODF interrupt flags in the SR register) are reset; otherwise jump to the beginning of the routine. If, on return from an interrupt routine, the pending bit is reset while one of the interrupt flags is set, no interrupt is performed on that channel until the flags are set. A new interrupt request is performed only when a flag is set with the other not set.

7.5.5.1 Register Map

Depending on the device, one or two Serial Peripheral interfaces can be present. The previous table summarizes the position of the registers of the two peripherals in the register map of the microcontroller.

	Address	Page	Name
SPI0	R240 (F0h)	7	DR0
	R241 (F1h)	7	CR0
	R242 (F2h)	7	SR0
	R243 (F3h)	7	PR0
SPI1	R248 (F8h)	7	DR1
	R249 (F9h)	7	CR1
	R250 (FAh)	7	SR1
	R251 (FBh)	7	PR1

SERIAL PERIPHERAL INTERFACE (Cont'd)

7.5.6 Register Description

DATA REGISTER (SPDR)
R240 - Read/Write
Register Page: 7
Reset Value: 0000 0000 (00h)

7							0
D7	D6	D5	D4	D3	D2	D1	D0

The DR register is used to transmit and receive data on the serial bus. In the master device only a write to this register will initiate transmission/reception of another byte.

Notes: During the last clock cycle the SPIF bit is set, a copy of the received data byte in the shift register is moved to a buffer. When the user reads the serial peripheral data register, the buffer is actually being read.

Warning: A write to the DR register places data directly into the shift register for transmission.

A read to the DR register returns the value located in the buffer and not the content of the shift register (see Figure 73).

CONTROL REGISTER (SPCR)
R241 - Read/Write
Register Page: 7
Reset Value: 0000 0000 (00h)

7							0
SPIE	SPOE	SPIS	MSTR	CPOL	CPHA	SPR1	SPR0

Bit 7 = **SPIE** *Serial peripheral interrupt enable.*
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An SPI interrupt is generated whenever either SPIF or MODF are set in the SR register while the other flag is 0.

Bit 6 = **SPOE** *Serial peripheral output enable.*
This bit is set and cleared by software. It is also cleared by hardware when, in master mode, $\overline{SS}$=0 (see Section 7.5.4.5 Master Mode Fault).
0: SPI alternate functions disabled (MISO, MOSI and SCK can only work as input)
1: SPI alternate functions enabled (MISO, MOSI and SCK can work as input or output depending on the value of MSTR)

Note: To use the MISO, MOSI and SCK alternate functions (input or output), the corresponding I/O port must be programmed as alternate function output.

Bit 5 = **SPIS** *Interrupt Selection.*
This bit is set and cleared by software.
0: Interrupt source is external interrupt
1: Interrupt source is SPI

Bit 4 = **MSTR** *Master.*
This bit is set and cleared by software. It is also cleared by hardware when, in master mode, $\overline{SS}$=0 (see Section 7.5.4.5 Master Mode Fault).
0: Slave mode is selected
1: Master mode is selected, the function of the SCK pin changes from an input to an output and the functions of the MISO and MOSI pins are reversed.

Bit 3 = **CPOL** *Clock polarity.*
This bit is set and cleared by software. This bit determines the steady state of the serial Clock. The CPOL bit affects both the master and slave modes.
0: The steady state is a low value at the SCK pin.
1: The steady state is a high value at the SCK pin.

Bit 2 = **CPHA** *Clock phase.*
This bit is set and cleared by software.
0: The first clock transition is the first data capture edge.
1: The second clock transition is the first capture edge.

Bit 1:0 = **SPR[1:0]** *Serial peripheral rate.*
These bits are set and cleared by software. They select one of four baud rates to be used as the serial clock when the device is a master.

These 2 bits have no effect in slave mode.

Table 28. Serial Peripheral Baud Rate

INTCLK Clock Divide	SPR1	SPR0
2	0	0
4	0	1
16	1	0
32	1	1

SERIAL PERIPHERAL INTERFACE (Cont'd)

STATUS REGISTER (SPSR)
R242 - Read Only
Register Page: 7
Reset Value: 0000 0000 (00h)

7							0
SPIF	WCOL	-	MODF	-	-	-	-

Bit 7 = **SPIF** *Serial Peripheral data transfer flag.*
This bit is set by hardware when a transfer has been completed. An interrupt is generated if SPIE=1 in the CR register. It is cleared by a software sequence (an access to the SR register followed by a read or write to the DR register).
0: Data transfer is in progress or has been approved by a clearing sequence.
1: Data transfer between the device and an external device has been completed.

Note: While the SPIF bit is set, all writes to the DR register are inhibited.

Bit 6 = **WCOL** *Write Collision status.*

This bit is set by hardware when a write to the DR register is done during a transmit sequence. It is cleared by a software sequence (see Figure 76).
0: No write collision occurred
1: A write collision has been detected

Bit 5 = Unused.

Bit 4 = **MODF** *Mode Fault flag.*
This bit is set by hardware when the $\overline{SS}$ pin is pulled low in master mode (see Section 7.5.4.5 Master Mode Fault). An SPI interrupt can be generated if SPIE=1 in the CR register. This bit is cleared by a software sequence (An access to the SR register while MODF=1 followed by a write to the CR register).
0: No master mode fault detected
1: A fault in master mode has been detected

Bits 3:0 = Unused.

PRESCALER REGISTER (SPPR)
R243 - Read/Write
Register Page: 7
Reset Value: 0000 0000 (00h)

7							0
0	0	0	DIV2	0	PRS2	PRS1	PRS0

Bits 7:5 = Reserved, forced by hardware to 0**.**

Bit 4 = **DIV2** *Divider enable.*
This bit is set and cleared by software.
0: Divider by 2 enabled.
1: Divider by 2 disabled.

Bit 3 = Reserved. forced by hardware to 0.

Bits 2:0 = **PRS[2:0]** *Prescaler Value.*
These bits are set and cleared by software. The baud rate generator is driven by INTCLK/(n1*n2*n3) where n1= PRS[2:0]+1, n2 is the value of the SPR[1:0] bits, n3 = 1 if DIV2=1 and n3= 2 if DIV2=0. Refer to Figure 73.

These bits have no effect in slave mode.

Table 29. Prescaler Baud Rate

Prescaler Division Factor	PRS2	PRS1	PRS0
1 (no division)	0	0	0
2	0	0	1
...			
8	1	1	1

7.6 ANALOG TO DIGITAL CONVERTER (ADC)

Important Note: This chapter is a generic description of the ADC peripheral. However depending on the ST9 device, some or all of the interface signals described may not be connected to external pins. For the list of ADC pins present on the ST9 device, refer to the device pinout description in the first section of the data sheet.

7.6.1 Introduction

The Analog to Digital Converter (ADC) comprises an input multiplex channel selector feeding a successive approximation converter.

The conversion time depends on the INTCLK frequency and the prescaler factor stored in the PR[2:0] bits of the CRR register (R252).

The minimum conversion time is 138 INTCLK and with the maximum prescaling factor it becomes about 16 times longer.

For instance, with INTCLK at 20 MHz and PR[2:0] equal to "111", conversion of the selected channel requires 6.9µs. It requires 27.45µs if PR[2:0] equals "110" and so on. Refer to Table 30 for the list of conversion and sampling times.

The 6.9µs time includes the 4µs required by the built-in Sample and Hold circuitry, which minimizes the need for external components and allows quick sampling of the signal to minimise warping and conversion error.

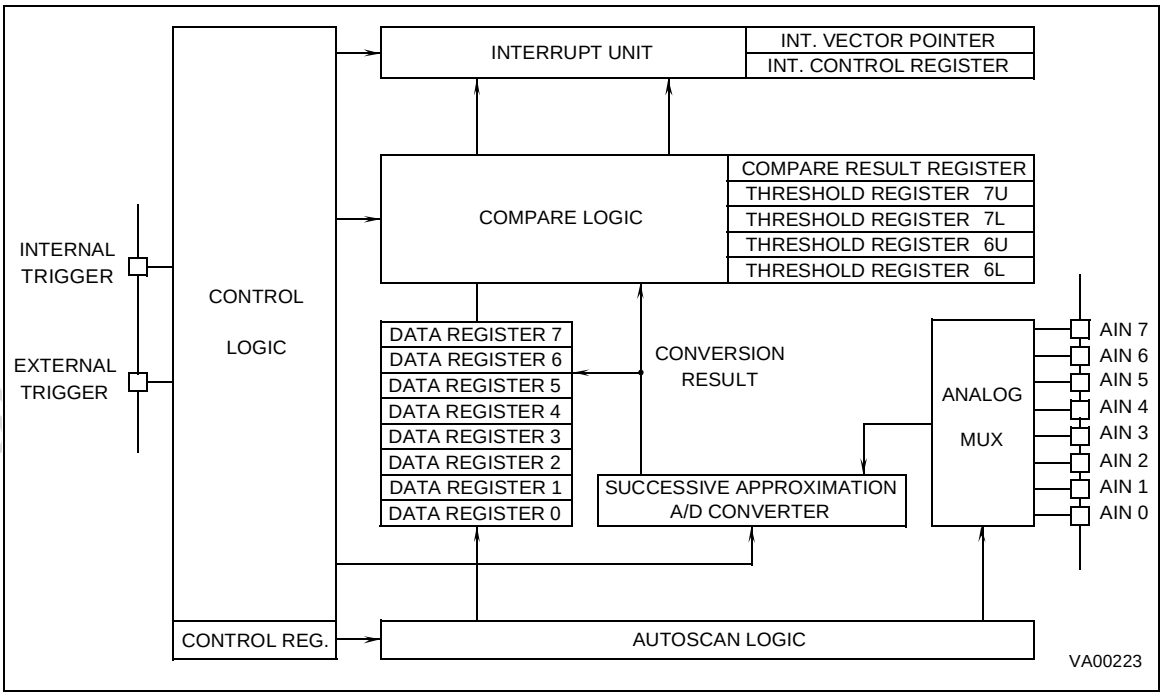
Conversion resolution is 8 bits, with ±1 LSB maximum error in the input range between V_{SS} and the analog V_{DD} reference.

The converter uses a fully differential analog input configuration for the best noise immunity and precision performance. Two separate supply references are provided to ensure the best possible supply noise rejection. In fact, the converted digital value, is referred to the analog reference voltage which determines the full scale converted value. Naturally**,** Analog and Digital V_{SS} MUST be common. If analog supplies are not present, input reference voltages are referred to the digital ground and supply.

Up to 8 multiplexed Analog Inputs are available, depending on the specific device type. A group of signals can be converted sequentially by simply programming the starting address of the first analog channel to be converted and with the AUTOSCAN feature.

Two Analog Watchdogs are provided, allowing continuous hardware monitoring of two input channels. An Interrupt request is generated whenever the converted value of either of these two analog inputs is outside the upper or lower programmed threshold values. The comparison result is stored in a dedicated register.

Figure 78. Block Diagram



ANALOG TO DIGITAL CONVERTER (Cont'd)

Single and continuous conversion modes are available. Conversion may be triggered by an external signal or, internally, by the Multifunction Timer.

Conversion Time

The maximum conversion time is as follows:

138 * FDF * INTCLK

The minimum sample time is:

84 * FDF * INTCLK

where FDF is the Frequency Division Factor (refer to the PR bit description in the CCR register).

For instance, if PR[2:0] = 100 -> FDF = 8

then the conversion time is:

138 * 8 * INTCLK = 1104 and the sample time is 84 * 8 * INTCLK = 672 INTCLK.

A Power-Down programmable bit allows the ADC to be set in low-power idle mode.

The ADC's Interrupt Unit provides two maskable channels (Analog Watchdog and End of Conversion) with hardware fixed priority, and up to 7 programmable priority levels.

CAUTION: ADC INPUT PIN CONFIGURATION

The input Analog channel is selected by using the I/O pin Alternate Function setting (PXC2, PXC1, PXC0 = 1,1,1) as described in the I/O ports section. The I/O pin configuration of the port connected to the A/D converter is modified in order to prevent the analog voltage present on the I/O pin from causing high power dissipation across the input buffer. Deselected analog channels should also be maintained in Alternate function configuration for the same reason.

7.6.2 Functional Description

7.6.2.1 Operating Modes

Two operating modes are available: Continuous Mode and Single Mode. To enter one of these modes it is necessary to program the CONT bit of the Control Logic Register. The Continuous Mode is selected when CONT is set, while Single Mode is selected when CONT is reset.

Both modes operate in AUTOSCAN configuration, allowing sequential conversion of the input channels. The number of analog inputs to be converted may be set by software, by setting the number of the first channel to be converted into the Control Register (SC2, SC1, SC0 bits). As each conversion is completed, the channel number is automatically incremented, up to channel 7. For example, if SC2, SC1, SC0 are set to 0,1,1, conversion will proceed from channel 3 to channel 7, whereas, if SC2, SC1, SC0 are set to 1,1,1, only channel 7 will be converted.

When the ST bit of the Control Logic Register is set, either by software or by hardware (by an internal or external synchronisation trigger signal), the analog inputs are sequentially converted (from the first selected channel up to channel 7) and the results are stored in the relevant Data Registers.

In **Single Mode** (CONT = "0"), the ST bit is reset by hardware following conversion of channel 7; an End of Conversion (ECV) interrupt request is issued and the ADC waits for a new start event.

In **Continuous Mode** (CONT = "1"), a continuous conversion flow is initiated by the start event. When conversion of channel 7 is complete, conversion of channel 's' is initiated (where 's' is specified by the setting of the SC2, SC1 and SC0 bits); this will continue until the ST bit is reset by software. In all cases, an ECV interrupt is issued each time channel 7 conversion ends.

When channel 'i' is converted ('s' <'i' <7), the related Data Register is reloaded with the new conversion result and the previous value is lost. The End of Conversion (ECV) interrupt service routine can be used to save the current values before a new conversion sequence (so as to create signal sample tables in the Register File or in Memory).

7.6.2.2 Triggering and Synchronisation

In both modes, conversion may be triggered by internal or external conditions; externally this may be tied to EXTRG, as an Alternate Function input on an I/O port pin, and internally, it may be tied to INTRG, generated by a Multifunction Timer peripheral. Both external and internal events can be separately masked by programming the EXTG/ INTG bits of the Control Logic Register (CLR). The events are internally ORed, thus avoiding potential hardware conflicts. However, the correct procedure is to enable only one alternate synchronisation condition at any time.

The effect either of these synchronisation modes is to set the ST bit by hardware. This bit is reset, in Single Mode only, at the end of each group of conversions. In Continuous Mode, all trigger pulses after the first are ignored.

The synchronisation sources must be at a logic low level for at least the duration of one INTCLK cycle and, in Single Mode, the period between trigger pulses must be greater than the total time required for a group of conversions. If a trigger occurs when the ST bit is still set, i.e. when conversion is still in progress, it will be ignored.

ANALOG TO DIGITAL CONVERTER (Cont'd)

On devices where two A/D Converters are present they can be triggered from the same source.

Converter	External Trigger	On Chip Event (Internal trigger)
A/D 0	EXTRG pin	MFT 0
A/D 1		

7.6.2.3 Analog Watchdogs

Two internal Analog Watchdogs are available for highly flexible automatic threshold monitoring of external analog signal levels.

Analog channels 6 and 7 monitor an acceptable voltage level window for the converted analog inputs. The external voltages applied to inputs 6 and 7 are considered normal while they remain below their respective Upper thresholds, and above or at their respective Lower thresholds.

When the external signal voltage level is greater than, or equal to, the upper programmed voltage limit, or when it is less than the lower programmed voltage limit, a maskable interrupt request is generated and the Compare Results Register is updated in order to flag the threshold (Upper or Lower) and channel (6 or 7) responsible for the interrupt. The four threshold voltages are user programmable in dedicated registers (08h to 0Bh) of the ADC register page. Only the 4 MSBs of the Compare Results Register are used as flags, each of the four MSBs being associated with a threshold condition.

Following a reset, these flags are reset. During normal ADC operation, the CRR bits are set, in order to flag an out of range condition and are automatically reset by hardware after a software reset of the Analog Watchdog Request flag in the ICR Register.

7.6.2.4 Power Down Mode

Before enabling an A/D conversion, the POW bit of the Control Logic Register must be set; this must be done at least 60µs before the first conversion start, in order to correctly bias the analog section of the converter circuitry.

When the ADC is not required, the POW bit may be reset in order to reduce the total power consumption. This is the reset configuration, and this state is also selected automatically when the ST9 is placed in Halt Mode (following the execution of the `halt` instruction).

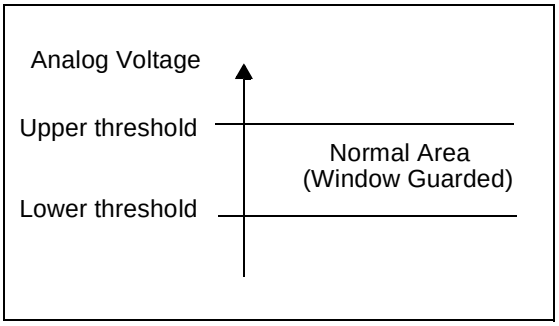
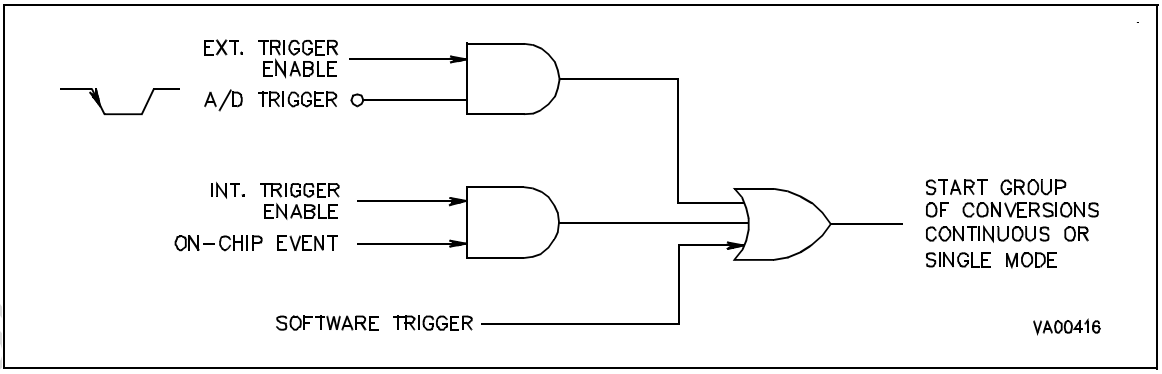
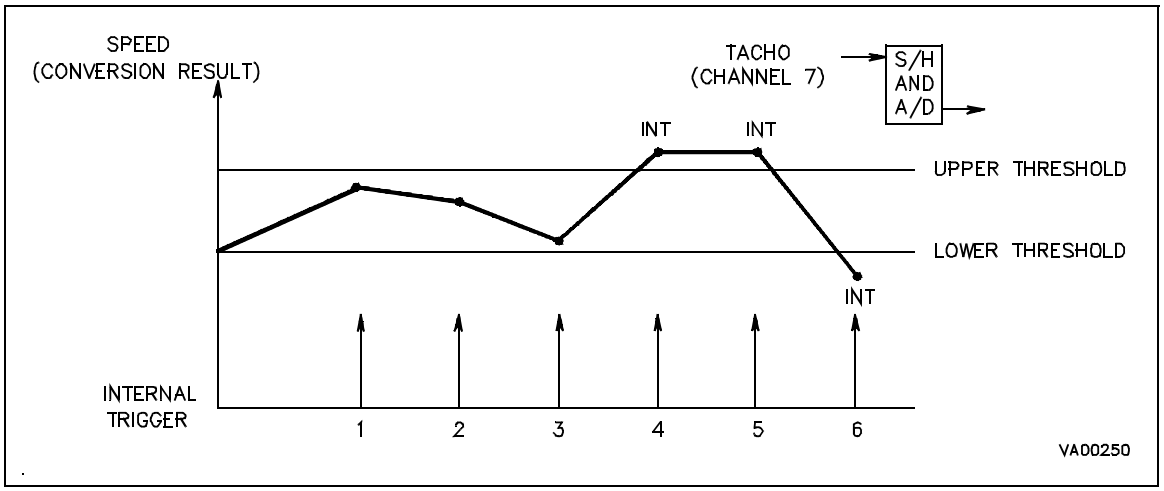


Figure 79. A/D Trigger Source



ANALOG TO DIGITAL CONVERTER (Cont'd)

Figure 80. Application Example: Analog Watchdog used in Motorspeed Control



7.6.3 Interrupts

The ADC provides two interrupt sources:

– End of Conversion

– Analog Watchdog Request

The A/D Interrupt Vector Register (IVR) provides hardware generated flags which indicate the interrupt source, thus allowing automatic selection of the correct interrupt service routine.

	7							0	
Analog Watchdog Request	X	X	X	X	X	X	0	0	Lower Word Address

	7							0	
End of Conv. Request	X	X	X	X	X	X	1	0	Upper Word Address

The A/D Interrupt vector should be programmed by the User to point to the first memory location in the Interrupt Vector table containing the base address of the four byte area of the interrupt vector table in which the address of the A/D interrupt service routines are stored.

The Analog Watchdog Interrupt Pending bit (AWD, ICR.6), is automatically set by hardware whenever any of the two guarded analog inputs go out of range. The Compare Result Register (CRR) tracks the analog inputs which exceed their programmed thresholds.

When two requests occur simultaneously, the Analog Watchdog Request has priority over the End of Conversion request, which is held pending.

The Analog Watchdog Request requires the user to poll the Compare Result Register (CRR) to determine which of the four thresholds has been exceeded. The threshold status bits are set to flag an out of range condition, and are automatically reset by hardware after a software reset of the Analog Watchdog Request flag in the ICR Register. The interrupt pending flags, ECV and AWD, should be reset by the user within the interrupt service routine. Setting either of these two bits by software will cause an interrupt request to be generated.

7.6.4 Register Description

DATA REGISTERS (DiR)

The conversion results for the 8 available channels are loaded into the 8 Data registers following conversion of the corresponding analog input.

CHANNEL 0 DATA REGISTER (D0R)
R240 - Read/Write
Register Page: 63
Reset Value: undefined

7							0
D0.7	D0.6	D0.5	D0.4	D0.3	D0.2	D0.1	D0.0

Bit 7:0 = D0.[7:0]: Channel 0 Data

CHANNEL 1 DATA REGISTER (D1R)
R241 - Read/Write
Register Page: 63
Reset Value: undefined

7							0
D1.7	D1.6	D1.5	D1.4	D1.3	D1.2	D1.1	D1.0

Bit 7:0 = D1.[7:0]: Channel 1 Data

CHANNEL 2 DATA REGISTER (D2R)
R242 - Read/Write
Register Page: 63
Reset Value: undefined

7							0
D2.7	D2.6	D2.5	D2.4	D2.3	D2.2	D2.1	D2.0

Bit 7:0 = D2.[7:0]: Channel 2 Data

CHANNEL 3 DATA REGISTER (D3R)
R243 - Read/Write
Register Page: 63
Reset Value: **undefined**

7							0
D3.7	D3.6	D3.5	D3.4	D3.3	D3.2	D3.1	D3.0

Bit 7:0 = D3.[7:0]: Channel 3 Data

CHANNEL 4 DATA REGISTER (D4R)
R244 - Read/Write
Register Page: 63
Reset Value: undefined

7							0
D4.7	D4.6	D4.5	D4.4	D4.3	D4.2	D4.1	D4.0

Bit 7:0 = D4.[7:0]: Channel 4 Data

CHANNEL 5 DATA REGISTER (D5R)
R245 - Read/Write
Register Page: 63
Reset Value: undefined

7							0
D5.7	D5.6	D5.5	D5.4	D5.3	D5.2	D5.1	D5.0

Bit 7:0 = D5.[7:0]: Channel 5 Data

CHANNEL 6 DATA REGISTER (D6R)
R246 - Read/Write
Register Page: 63
Reset Value: undefined

7							0
D6.7	D6.6	D6.5	D6.4	D6.3	D6.2	D6.1	D6.0

Bit 7:0 = D6.[7:0]: Channel 6 Data

CHANNEL 7 DATA REGISTER (D7R)
R247 - Read/Write
Register Page: 63
Reset Value: undefined

7							0
D7.7	D7.6	D7.5	D7.4	D7.3	D7.2	D7.1	D7.0

Bit 7:0 = D7.[7:0]: Channel 7 Data

REGISTER DESCRIPTION (Cont'd)

LOWER THRESHOLD REGISTERS (LTiR)

The two Lower Threshold registers are used to store the user programmable lower threshold 8-bit values, to be compared with the current conversion results, thus setting the lower window limit.

CHANNEL 6 LOWER THRESHOLD REGISTER (LT6R)
R248 - Read/Write
Register Page: 63
Reset Value: undefined

7							0
LT6.7	LT6.6	LT6.5	LT6.4	LT6.3	LT6.2	LT6.1	LT6.0

Bit 7:0 = LT6.[7:0]: Channel 6 Lower Threshold

CHANNEL 7 LOWER THRESHOLD REGISTER (LT7R)
R249 - Read/Write
Register Page: 63
Reset Value: undefined

7							0
LT7.7	LT6.7	LT7.5	LT7.4	LT7.3	LT7.2	LT7.1	LT7.0

Bit 7:0 = LT7.[7:0]**:** *Channel 7 Lower Threshold*

UPPER THRESHOLD REGISTERS (UTIR)

The two Upper Threshold registers are used to store the user programmable upper threshold 8-bit values, to be compared with the current conversion results, thus setting the upper window limit.

CHANNEL 6 UPPER THRESHOLD REGISTER (UT6R)
R250 - Read/Write
Register Page: 63
Reset Value: undefined

7							0
UT6.7	UT6.6	UT6.5	UT6.4	UT6.3	UT6.2	UT6.1	UT6.0

Bit 7:0 = **UT6.[7:0]**: *Channel 6 Upper Threshold value*

CHANNEL 7 UPPER THRESHOLD REGISTER (UT7R)
R251 - Read/Write
Register Page: 63
Reset Value: undefined

7							0
UT7.7	UT6.7	UT7.5	UT7.4	UT7.3	UT7.2	UT7.1	UT7.0

Bit 7:0 = **UT7.[7:0]**: *Channel 7 Upper Threshold value*

REGISTER DESCRIPTION (Cont'd)

COMPARE RESULT REGISTER (CRR)
R252 - Read/Write
Register Page: 63
Reset Value: 0000 1111 (0Fh)

7							0
C7U	C6U	C7L	C6L	X	PR2	PR1	PR0

The result of the comparison between the current value of data registers 6 and 7 and the threshold registers is stored in the 4 most significant bits of this register.

Bit 7 = **C7U**: *Compare Reg 7 Upper threshold*
Set when converted data is greater than or equal to the threshold value. Not affected otherwise.

Bit 6 = **C6U**: *Compare Reg 6 Upper threshold*
Set when converted data is greater than or equal to the threshold value. Not affected otherwise.

Bit 5 = **C7L**: *Compare Reg 7 Lower threshold*
Set when converted data is less than the threshold value. Not affected otherwise.

Bit 4 = **C6L**: *Compare Reg 6 Lower threshold*
Set when converted data is less than the threshold value. Not affected otherwise.

These bits should be reset at the end of the "Out of Range" interrupt service routine.

Note: Any software reset request of the ICR, will also cause all the compare status bits to forced by hardware to zero, in order to prevent possible overwriting if an interrupt request occurs between reset and the Interrupt request software reset.

Bit 3 = undefined, return '1' when red.

Bit 2:0 = **PR[2:0]**: *Clock divider bits*
These bits enable a frequency division factor depending on the value stored:

Table 30. Frequency Division Factors

PR[2:0] bits	Freq. Div. Factor (FDF)	Max. conversion time (INTCLK)	Min. sampling time (INTCLK)
1 1 1	1	138	84
1 1 0	4	552	336
1 0 1	6	828	504
1 0 0	8	1104	672
0 1 1	10	1380	840
0 1 0	12	1656	1008
0 0 1	14	1932	1176
0 0 0	16	2208	1344

Warning: If the prescaler programming value is changed during a conversion, the user has to restart the conversion (i.e. simply rewriting the CLR register with the same value).

REGISTER DESCRIPTION (Cont'd)

CONTROL LOGIC REGISTER (CLR)

The Control Logic Register (CLR) manages the ADC's logic. Writing to this register will cause the current conversion to be aborted and the autoscan logic to be re-initialized. CLR is programmable as follows:

CONTROL LOGIC REGISTER (CLR)
R253 - Read/Write
Register Page: 63
Reset Value: 0000 0000 (00h)

7							0
SC2	SC1	SC0	EXTG	INTG	POW	CONT	ST

Bit 7:5 = **SC[2:0]:** *Start Conversion Address*.
These 3 bits define the starting analog input channel (Autoscan mode). The first channel addressed by SC[2:0] is converted, then the channel number is incremented for the successive conversion, until channel 7 (111) is converted. When SC2, SC1 and SC0 are all set, only channel 7 will be converted.

Bit 4 = **EXTG**: *External Trigger Enable*.
This bit is set and cleared by software.
0: External trigger disabled.
1: External trigger enabled. Allows a conversion sequence to be started on the subsequent edge of the external signal applied to the EXTRG pin (when enabled as an Alternate Function).

Bit 3 = **INTG**: *Internal Trigger Enable*.
This bit is set and cleared by software.
0: Internal trigger disabled.
1: Internal trigger enabled. Allows a conversion sequence to be started, synchronized by an internal signal (On-chip Event signal) from a Multifunction Timer peripheral.

Both External and Internal Trigger inputs are internally ORed, thus avoiding Hardware conflicts; however, the correct procedure is to enable only one alternate synchronization input at a time.

Note: The effect of either synchronization mode is to set the START/STOP bit, which is reset by hardware when in SINGLE mode, at the end of each sequence of conversions.

Requirements: The External Synchronisation Input must receive a low level pulse wider than an INTCLK period and, for both External and On-Chip Event synchronisation, the repetition period must be greater than the time required for the selected sequence of conversions.

Bit 2 = **POW**: *Power Up/Power Down.*
This bit is set and cleared by software.
0: Power down mode: all power-consuming logic is disabled, thus selecting a low power idle mode.
1: Power up mode: the A/D converter logic and analog circuitry is enabled.

Bit 1 = **CONT**: *Continuous/Single*.
0: Single Mode: a single sequence of conversions is initiated whenever an external (or internal) trigger occurs, or when the ST bit is set by software.
1: Continuous Mode: the first sequence of conversions is started, either by software (by setting the ST bit), or by hardware (on an internal or external trigger, depending on the setting of the INTG and EXTG bits); a continuous conversion sequence is then initiated.

Bit 0 = **ST**: *Start/Stop.*
0: Stop conversion. When the A/D converter is running in Single Mode, this bit is hardware reset at the end of a sequence of conversions.
1: Start a sequence of conversions.

REGISTER DESCRIPTION (Cont'd)

INTERRUPT CONTROL REGISTER (AD_ICR)

The Interrupt Control Register contains the three priority level bits, the two source flags, and their bit mask:

INTERRUPT CONTROL REGISTER (AD_ICR)
R254 - Read/Write
Register Page: 63
Reset Value: 0000 1111 (0Fh)

7							0
ECV	AWD	ECI	AWDI	X	PL2	PL1	PL0

Bit 7 = **ECV**: *End of Conversion.*
This bit is automatically set by hardware after a group of conversions is completed. It must be reset by the user, before returning from the Interrupt Service Routine. Setting this bit by software will cause a software interrupt request to be generated.
0: No End of Conversion event occurred
1: An End of Conversion event occurred

Bit 6 = **AWD**: *Analog Watchdog.*
This is automatically set by hardware whenever either of the two monitored analog inputs goes out of bounds. The threshold values are stored in registers F8h and FAh for channel 6, and in registers F9h and FBh for channel 7 respectively. The Compare Result Register (CRR) keeps track of the analog inputs exceeding the thresholds.

The AWD bit must be reset by the user, before returning from the Interrupt Service Routine. Setting this bit by software will cause a software interrupt request to be generated.
0: No Analog Watchdog event occurred
1: An Analog Watchdog event occurred

Bit 5 = **ECI**: *End of Conversion Interrupt Enable.*
This bit masks the End of Conversion interrupt request.
0: Mask End of Conversion interrupts
1: Enable End of Conversion interrupts

Bit 4 = **AWDI**: *Analog Watchdog Interrupt Enable*.
This bit masks or enables the Analog Watchdog interrupt request.
0: Mask Analog Watchdog interrupts
1: Enable Analog Watchdog interrupts

Bit 3 = Reserved.

Bit 2:0 = **PL[2:0]:** *A/D Interrupt Priority Level*.
These three bits allow selection of the Interrupt priority level for the ADC.

INTERRUPT VECTOR REGISTER (AD_IVR)
R255 - Read/Write
Register Page: 63
Reset Value: xxxx xx10 (x2h**)**

7							0
V7	V6	V5	V4	V3	V2	W1	0

Bit 7:2 = **V[7:2]:** *A/D Interrupt Vector.*
This vector should be programmed by the User to point to the first memory location in the Interrupt Vector table containing the starting addresses of the A/D interrupt service routines.

Bit 1 = **W1**: *Word Select.*
This bit is set and cleared by hardware, according to the A/D interrupt source.
0: Interrupt source is the Analog Watchdog, pointing to the lower word of the A/D interrupt service block (defined by V[7:2]).
1:Interrupt source is the End of Conversion interrupt, thus pointing to the upper word.

Note: When two requests occur simultaneously, the Analog Watchdog Request has priority over the End of Conversion request, which is held pending.

Bit 0 = Reserved. Forced by hardware to 0.

8 ELECTRICAL CHARACTERISTICS

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

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

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

$T_J = T_A + P_D \times RthJA$

Where: T_A = Ambient Temperature.

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

$P_D = P_{INT} + P_{PORT}$.

$P_{INT} = I_{DD} \times V_{DD}$ (chip internal power).

P_{PORT} = Port power dissipation (determined by the user)

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_{DD}	Supply Voltage	– 0.3 to 6.5	V
AV_{DD}	A/D Converter Analog Reference	up to V_{DD} + 0.3 (1)	V
AV_{SS}	A/D Converter V_{SS}	V_{SS}	
V_{IN}	Input Voltage (standard I/O pins)	– 0.3 to V_{DD} + 0.3	V
V_{AIN}	Analog Input Voltage (A/D Converter)	AV_{SS} to AV_{DD}	V
ESD	ESD susceptibility	2000	V
T_{STG}	Storage Temperature	– 55 to +150	°C
I_{INJ}	Pin Injection Current - Digital and Analog Input	+/- 10	mA
	Maximum Accumulated Pin Injection Current	+/- 100	mA

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. All voltages are referenced to V_{SS}=0.

(1) AV_{DD} can be shut down while the A/D Converter is not in use.

THERMAL CHARACTERISTICS

Symbol	Package	Value	Unit
RthJA	PDIP32Sh	60	°C/W
	PSO34Sh	75	

RECOMMENDED OPERATING CONDITIONS

Symbol	Parameter	Value		Unit
		Min	Max	
T_A	Operating Temperature	–40	85	°C
V_{DD}	Operating Supply Voltage(1)	4.5	5.5	V
AV_{DD}	Analog Supply Voltage	0	V_{DD} + 0.3	V
f_{INTCLK}	Internal Clock Frequency @ 4.5V - 5.5V	0(2)	25	MHz

Note:

(1) Device is reset whenever Supply Voltage is below LVD Thresholds

(2) 1MHz when A/D is used with its internal Prescaler programmed to 1.

DC ELECTRICAL CHARACTERISTICS

(V_{DD} = 5V ± 10%, T_A = –40°C to +85°C, unless otherwise specified)

Symbol	Parameter	Comment	Value			Unit
			Min	Typ(1)	Max	
V_{IH}	Input High Level Standard Schmitt Trigger P5[7:2]		$0.7V_{DD}$		V_{DD} + 0.3	V
	Input High Level High Hyst. Schmitt Trigger P5[1:0]-P3[6:0]-TACHO		$0.8V_{DD}$		V_{DD} + 0.3	V
V_{IL}	Input Low Level Standard Schmitt Trigger P5[7:2]		– 0.3		0.8	V
	Input Low Level High Hyst. Schmitt Trigger P5[1:0]-P3[6:0]-TACHO		– 0.3		0.8	V
V_{HYS}	Input Hysteresis (2) Standard Schmitt Trigger P5[7:2]			0.5		V
	Input Hysteresis (2) High Hyst. Schmitt Trigger P5[1:0]-P3[6:0]-TACHO			1.5		V
V_{OH}	Output High Level High Current Pins P3.5-P3.6-P5.0-P5.2	Push Pull, I_{OH}= – 2mA EMR1 Register - BSZ bit = 0 (3)	V_{DD} – 0.8			V
		Psh Pull, I_{OH}= – 8mA EMR1 Register - BSZ bit = 1 (3)	V_{DD} – 0.8			V
	Output High Level Standard Current Pins P3[4:0]-P3.7-P5.1-P5[7:3]-UH-UL-VH-VL-WH-WL	Push Pull, I_{OH}= – 2mA	V_{DD} – 0.8			V
V_{OL}	Output Low Level High Current Pins P3.5-P3.6-P5.0-P5.2	Push Pull, I_{OL}=2mA EMR1 Register - BSZ bit = 0 (3)			0.4	V
		Push Pull, I_{OL}=8mA, EMR1 Register - BSZ bit = 1 (3)			0.4	V
		Push Pull, I_{OL}=20mA, EMR1 Register - BSZ bit = 1 (3)			3	V
	Output Low Level Standard Current Pins P3[4:0]-P3.7-P5.1-P5[7:3]-UH-UL-VH-VL-WH-WL	Push Pull, I_{OL}=2mA			0.4	V
I_{WPU}	Weak Pull-up Current P5[7:0]-P3[6:0]	Bidirectional Weak Pull-up V_{OL} = 0V	50		600	µA
I_{LKIO}	I/O Pin Input Leakage	Input/Tri-State, 0V < V_{IN} < V_{DD}	– 10		+ 10	µA
$I_{LKA/D}$	A/D Conv. Input Leakage		– 1		+ 1	µA

Note:
(1) Unless otherwise stated, typical data are based on T_A=25°C and V_{DD}=5V. They are only reported for design guidelines - not tested in production.
(2) Data based on characterization results - not tested in production.
(3) For a description of the EMR1 Register - BSZ bit refer to the Device Configuration Registers Chapter.

LOW VOLTAGE DETECTOR DC CHARACTERISTICS

(V_{DD} = 5V ± 10%, T_A = –40°C to +85°C, unless otherwise specified)

Symbol	Parameter	Test Conditions	Min	Typ(1)	Max	Unit
V_{LVDR}	Reset release Threshold	V_{DD} rise - Halt Mode			4.2	V
V_{LVDF}	Reset generation Threshold	V_{DD} fall - Halt Mode	3.4			V
$V_{LVDHyst}$	Hysteresis(2)	Halt Mode		200		mV
I_{DDLVD}	Supply Current	Halt Mode			250	µA

Note:
(1) Unless otherwise stated, typical data are based on T_A=25°C and V_{DD}=5V. They are only reported for design guidelines - not tested in production.
(2) Data based on characterization results - not tested in production

AC ELECTRICAL CHARACTERISTICS

(V_{DD} = 5V ± 10%, T_A = –40°C to +85°C, unless otherwise specified)

Symbol	Parameter	INTCLK	Typ(1)	Max	Unit
I_{DDRUN}	Run Mode Current(2)	25 MHz		50	mA
I_{DDWFI}	WFI Mode Current	25 MHz		15	mA
$I_{DDLPWFI}$	Low Power WFI Mode Current	125 kHz		2	mA
I_{DDHALT}	HALT Mode Current	-		250	µA

Note:
All I/O Ports are configured to a static value of VDD or VSS, external clock pin (OSCIN) is driven by square wave external clock.
(1) Unless otherwise stated, typical data are based on T_A=25°C and V_{DD}=5V. They are only reported for design guidelines - not tested in production.
(2) CPU running with memory access, all peripherals switched off.

EXTERNAL INTERRUPT TIMING TABLE

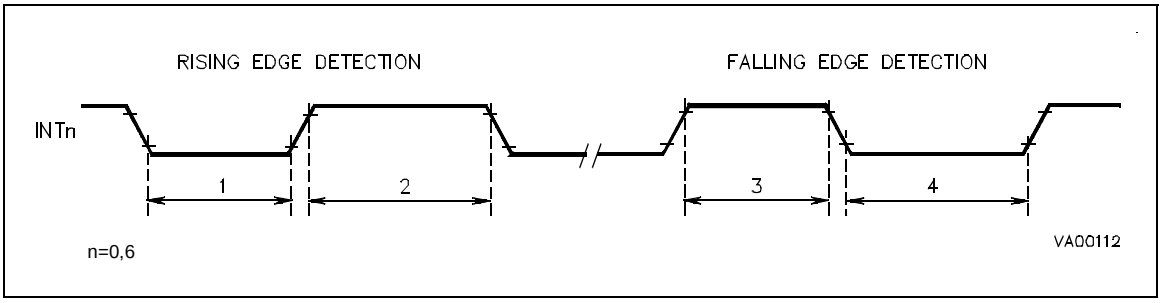
(V_{DD} = 5V ± 10%, T_A = –40°C to +85°C, C_{Load} = 50pF, f_{INTCLK} = 25MHz, unless otherwise specified)

N°	Symbol	Parameter	Value (Note)			Unit
			Formula(1)	Min	Max	
1	TwINTLR	Low Level Pulse Width in Rising Edge Mode	≥ Tck+10	50		ns
2	TwINTHR	High Level Pulse Width in Rising Edge Mode	≥ Tck+10	50		ns
3	TwINTHF	High Level Pulse Width in Falling Edge Mode	≥ Tck+10	50		ns
4	TwINTLF	Low Level Pulse Width in Falling Edge Mode	≥ Tck+10	50		ns

Note: The value in the left hand column shows the formula used to calculate the timing minimum or maximum from the oscillator clock period. The value in the right hand two columns show the timing minimum and maximum for an internal clock at 25MHz (INTCLK).
Measurement points are taken with reference to V_{IH}-V_{IH} for positive pulse and V_{IL}-V_{IL} for negative pulse
(1) Formula guaranteed by design.

Legend:
Tck = INTCLK period = OSCIN period when OSCIN is not divided by 2;
2 x OSCIN period when OSCIN is divided by 2;
OSCIN period x PLL factor when the PLL is enabled.

EXTERNAL INTERRUPT TIMING



WAKE-UP MANAGEMENT TIMING TABLE

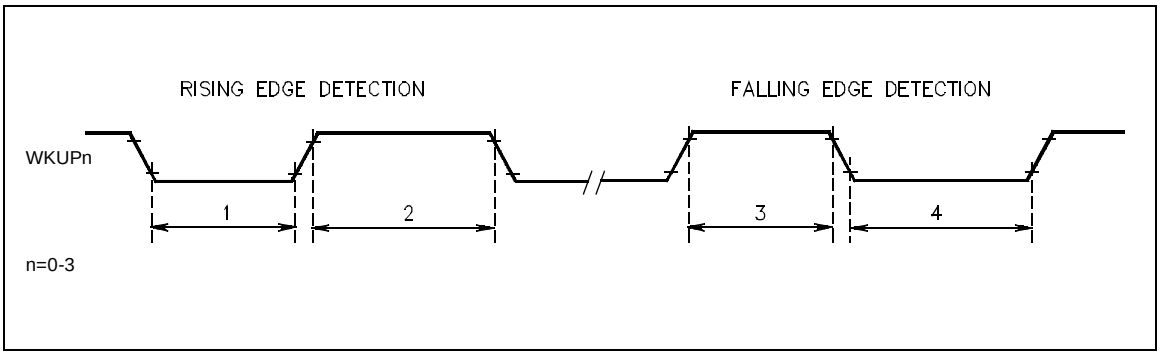
(V_{DD} = 5V ± 10%, T_A = –40°C to +85°C, C_{Load} = 50pF, f_{INTCLK} = 25MHz, unless otherwise specified)

N°	Symbol	Parameter	Value (Note)			Unit
			Formula(1)	Min	Max	
1	TwWKPLR	Low Level Pulse Width in Rising Edge Mode	≥ Tck+10	50		ns
2	TwWKPHR	High Level Pulse Width in Rising Edge Mode	≥ Tck+10	50		ns
3	TwWKPHF	High Level Pulse Width in Falling Edge Mode	≥ Tck+10	50		ns
4	TwWKPLF	Low Level Pulse Width in Falling Edge Mode	≥ Tck+10	50		ns

Note: The value in the left hand column shows the formula used to calculate the timing minimum or maximum from the oscillator clock period. The value in the right hand two columns show the timing minimum and maximum for an internal clock at 25MHz (INTCLK).
The given data are related to Wake-up Management Unit used in External Interrupt mode.
Measurement points are taken with reference to V_{IH}-V_{IH} for positive pulse and V_{IL}-V_{IL} for negative pulse
(1) Formula guaranteed by design.

Legend:
Tck = INTCLK period = OSCIN period when OSCIN is not divided by 2;
2 x OSCIN period when OSCIN is divided by 2;
OSCIN period x PLL factor when the PLL is enabled.

WAKE-UP MANAGEMENT TIMING



RCCU CHARACTERISTICS

(V_{DD} = 5V ± 10%, T_A = –40°C to +85°C, C_{Load} = 50pF, f_{INTCLK} = 25MHz, unless otherwise specified)

Symbol	Parameter	Comment	Value (Note)			Unit
			Min	Typ(1)	Max	
V_{IHRS}	$\overline{\text{RESET}}$ Input High Level		$0.7V_{DD}$		V_{DD} + 0.3	V
V_{ILRS}	$\overline{\text{RESET}}$ Input Low Level		– 0.3		$0.3V_{DD}$	V
V_{HYRS}	$\overline{\text{RESET}}$ Input Hysteresis(2)			900		mV
I_{LKRS}	$\overline{\text{RESET}}$ Pin Input Leakage	$0V < V_{IN} < V_{DD}$	– 10		+ 10	µA

Note:
(1) Unless otherwise stated, typical data are based on T_A=25°C and V_{DD}=5V. They are only reported for design guidelines - not tested in production.
(2) Data based on characterization results - not tested in production

RCCU TIMING TABLE

(V_{DD} = 5V ± 10%, T_A = –40°C to +85°C, C_{Load} = 50pF, f_{INTCLK} = 25MHz, unless otherwise specified)

Symbol	Parameter	Comment	Value (Note)			Unit
			Min	Typ	Max	
T_{FRS}	$\overline{\text{RESET}}$ Input Filtered Pulse				50	ns
T_{NFRS}	$\overline{\text{RESET}}$ Input not Filtered Pulse		20			µs
T_{RSPH}(1)	RESET Phase duration			20478 x T_{osc}		µs
T_{STR}	STOP Restart duration	DIV2 = 0 DIV2 = 1		10239 x T_{osc} 20478 x T_{osc}		µs

Note:
(1) Depending on the delay between rising edge of $\overline{\text{RESET}}$ pin and the first rising edge of CLOCK1, the value can differ from the typical value for +/- 1 CLOCK1 cycle.
Legend: T_{osc} = OSCIN period

PLL CHARACTERISTICS

(V_{DD} = 5V ± 10%, T_A = –40°C to +85°C, C_{Load} = 50pF, f_{INTCLK} = 25MHz, unless otherwise specified)

Symbol	Parameter	Comment	Value (Note)			Unit
			Min	Typ	Max	
F_{VCO}	VCO Operating Frequency		6		25	MHz
T_{PLK}	Lock-in Time				1000 x Tosc	µs
	PLL Jitter		0		1.2 (1)	ns

Note:
(1) Measured with PLL output clock frequency equal to 25MHz. Data based on characterization results - not tested in production.
Legend: Tosc = OSCIN period

OSCILLATOR CHARACTERISTICS

(V_{DD} = 5V ± 10%, T_A = –40°C to +85°C, C_{Load} = 50pF, f_{INTCLK} = 25MHz, unless otherwise specified)

Symbol	Parameter	Comment	Value (Note)			Unit
			Min	Typ(1)	Max	
F_{OSC}	Crystal Frequency	Fundamental mode crystal only	3		5	MHz
g_m	Oscillator		0.6	1.4	2.5	mA/V
V_{IHCK}	Clock Input High Level	External Clock	$0.8V_{DD}$		V_{DD} + 0.3	V
V_{ILCK}	Clock Input Low Level	External Clock	– 0.3		$0.2V_{DD}$	V
I_{LKOS}	OSCIN/OSCOUT Pins Input Leakage	$0V < V_{IN} < V_{DD}$ (HALT/STOP)	– 10		+ 10	µA
T_{STUP}	Oscillator Start-up Time			6(2)		ms

Note:
(1) Unless otherwise stated, typical data are based on T_A=25°C and V_{DD}=5V. They are only reported for design guidelines - not tested in production.
(2) Typical value with OSCIN=5MHz, CL=33pF on OSCIN-OSCOUT, T_A=25°C, V_{DD}=5V. The value depends on resonator quality as well.

WATCHDOG TIMING TABLE

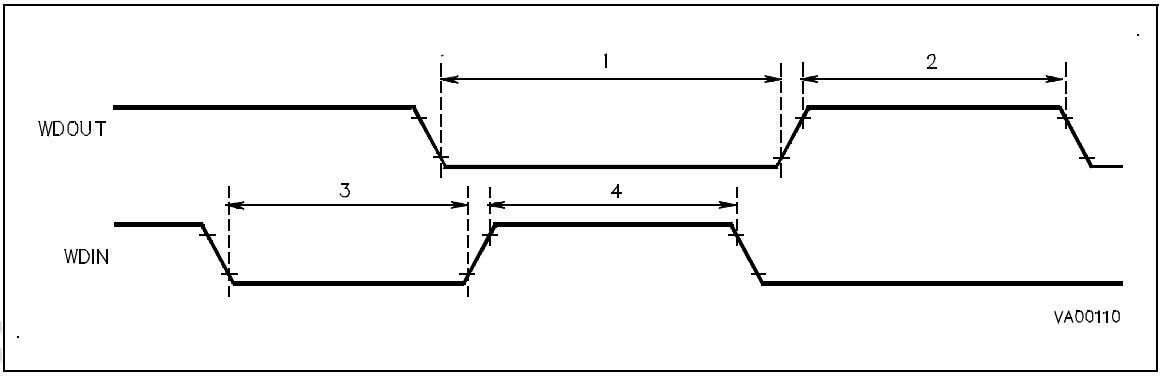
(V_{DD} = 5V ± 10%, T_A = –40°C to +85°C, C_{Load} = 50pF, f_{INTCLK} = 25MHz, Push-pull output configuration, unless otherwise specified)

N°	Symbol	Parameter	Value (Note)			Unit
			Formula[1]	Min	Max	
1	TwWDOL	WDOUT Low Pulse Width	4 x (Psc+1) x (Cnt+1) x Tck	160	2.69	ns s
			(Psc+1) x (Cnt+1) x T_{WDIN} with $T_{WDIN} \geq 8$ x Tck	320		ns
2	TwWDOH	WDOUT High Pulse Width	4 x (Psc+1) x (Cnt+1) x Tck	160	2.69	ns s
			(Psc+1) x (Cnt+1) x T_{WDIN} with $T_{WDIN} \geq 8$ x Tck	320		ns
3	TwWDIL	WDIN High Pulse Width	≥ 4 x Tck + 10	170		ns
4	TwWDIH	WDIN Low Pulse Width	≥ 4 x Tck +10	170		ns

Note: The value in the left hand column shows the formula used to calculate the timing minimum or maximum from the oscillator clock period, watchdog prescaler and counter programmed values.
The value in the right hand two columns show the timing minimum and maximum for an internal clock (INTCLK) at 25MHz, with minimum and maximum prescaler value and minimum and maximum counter value.
Measurement points are taken with reference to V_{IH}-V_{IH} / V_{OH}-V_{OH} for positive pulse and V_{IL}-V_{IL} / V_{OL}-V_{OL} for negative pulse
(1) Formula guaranteed by design.

Legend:
Tck = INTCLK period = OSCIN period when OSCIN is not divided by 2;
2 x OSCIN period when OSCIN is divided by 2;
OSCIN period x PLL factor when the PLL is enabled.
Psc = Watchdog Prescaler Register content (WDTPR): from 0 to 255
Cnt = Watchdog Counter Registers content (WDTRH,WDTRL): from 0 to 65535
T_{WDIN} = Watchdog Input signal period (WDIN)

WATCHDOG TIMING



STANDARD TIMER TIMING TABLE

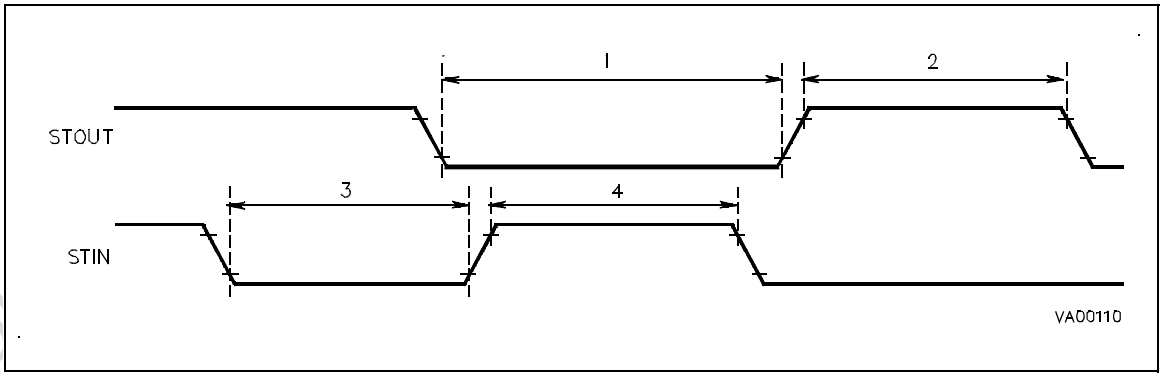
(V_{DD} = 5V ± 10%, T_A = –40°C to +85°C, C_{Load} = 50pF, f_{INTCLK} = 25MHz, Push-pull output configuration, unless otherwise specified)

N°	Symbol	Parameter	Value (Note)			Unit
			Formula[1]	Min	Max	
1	TwSTOL	STOUT Low Pulse Width	4 x (Psc+1) x (Cnt+1) x Tck	160	2.69	ns s
			(Psc+1) x (Cnt+1) x T_{STIN} with $T_{STIN} \geq 8$ x Tck	320		ns
2	TwSTOH	STOUT High Pulse Width	4 x (Psc+1) x (Cnt+1) x Tck	160	2.69	ns s
			(Psc+1) x (Cnt+1) x T_{STIN} with $T_{STIN} \geq 8$ x Tck	320		ns
3	TwSTIL	STIN High Pulse Width	≥ 4 x Tck + 10	170		ns
4	TwSTIH	STIN Low Pulse Width	≥ 4 x Tck + 10	170		ns

Note: The value in the left hand column shows the formula used to calculate the timing minimum or maximum from the oscillator clock period, standard timer prescaler and counter programmed values.
The value in the right hand two columns show the timing minimum and maximum for an internal clock (INTCLK) at 25MHz, with minimum and maximum prescaler value and minimum and maximum counter value.
Measurement points are taken with reference to V_{IH}-V_{IH} / V_{OH}-V_{OH} for positive pulse and V_{IL}-V_{IL} / V_{OL}-V_{OL} for negative pulse
(1) Formula guaranteed by design.

Legend:
Tck = INTCLK period = OSCIN period when OSCIN is not divided by 2;
2 x OSCIN period when OSCIN is divided by 2;
OSCIN period x PLL factor when the PLL is enabled.
Psc = Standard Timer Prescaler Register content (STP): from 0 to 255
Cnt = Standard Timer Counter Registers content (STH,STL): from 0 to 65535
T_{STIN} = Standard Timer Input signal period (STIN).

STANDARD TIMER TIMING



EXTENDED FUNCTION TIMER EXTERNAL TIMING TABLE

(V_{DD} = 5V ± 10%, T_A = –40°C to +105°C, C_{Load} = 50pF, f_{INTCLK} = 25MHz, unless otherwise specified)

N°	Symbol	Parameter	Value		Unit
			Formula(1)	Min	
1	Tw_{PEWL}	External Clock low pulse width (EXTCLK)	2 x Tck + 10	90	ns
2	Tw_{PEWH}	External Clock high pulse width (EXTCLK)	2 x Tck + 10	90	ns
3	Tw_{PIWL}	Input Capture low pulse width (ICAPx)	2 x Tck + 10	90	ns
4	Tw_{PIWH}	Input Capture high pulse width (ICAPx)	2 x Tck + 10	90	ns
5	Tw_{ECKD}	Distance between two active edges on EXTCLK	≥ 4 x Tck + 10	170	ns
6	Tw_{EICD}	Distance between two active edges on ICAPx	2 x Tck x Prsc +10	170	ns

Note: The value in the left hand column shows the formula used to calculate the timing minimum or maximum from the oscillator clock period, standard timer prescaler and counter programmed values.
The value in the right hand two columns show the timing minimum and maximum for an internal clock (INTCLK) at 25MHz, and minimum prescaler factor (=2).

Measurement points are taken with reference to V_{IH}-V_{IH} for positive pulse and V_{IL}-V_{IL} for negative pulse

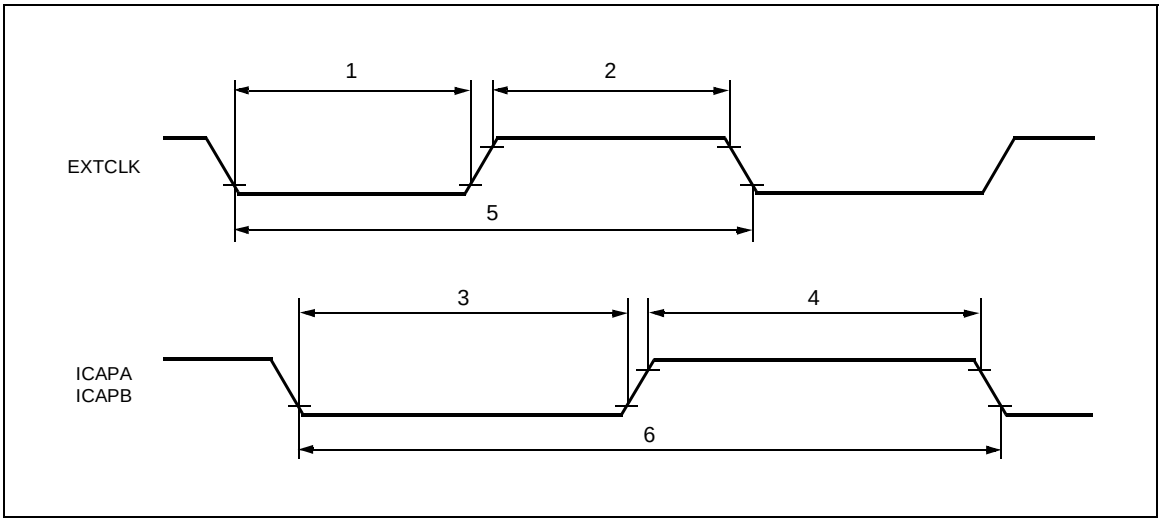
(1) Formula guaranteed by design.

Legend:

Tck = INTCLK period = OSCIN period when OSCIN is not divided by 2;
2 x OSCIN period when OSCIN is divided by 2;
OSCIN period x PLL factor when the PLL is enabled.

Prsc = Prescaler factor defined by Extended Function Timer Clock Control bits (CC1,CC0) on control register CR2 (values: 2,4,8).

EXTENDED FUNCTION TIMER EXTERNAL TIMING



SPI TIMING TABLE

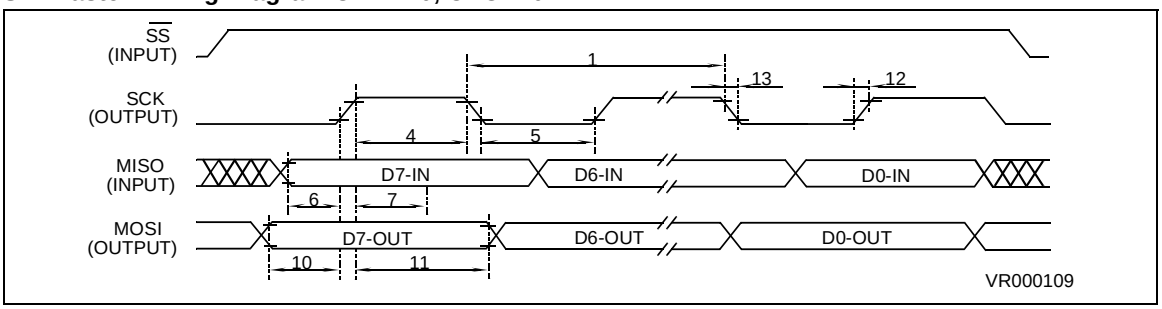
(V_{DD} = 5V ± 10%, T_A = –40°C to +85°C, C_{Load} = 50pF, f_{INTCLK} = 25MHz, unless otherwise specified)

N°	Symbol	Parameter	Condition	Value(1)		Unit
				Min	Max	
	f_{SPI}	SPI frequency	Master Slave	f_{INTCLK} / 128 0	f_{INTCLK} / 4 f_{INTCLK} / 2	MHz
1	t_{SPI}	SPI clock period	Master Slave	4 x Tck 2 x Tck		ns
2	t_{Lead}	Enable lead time	Slave	40		ns
3	t_{Lag}	Enable lag time	Slave	40		ns
4	t_{SPI_H}	Clock (SCK) high time	Master Slave	80 90		ns
5	t_{SPI_L}	Clock (SCK) low time	Master Slave	80 90		ns
6	t_{SU}	Data set-up time	Master Slave	40 40		ns
7	t_H	Data hold time (inputs)	Master Slave	40 40		ns
8	t_A	Access time (time to data active from high impedance state)	Slave	0	120	ns
9	t_{Dis}	Disable time (hold time to high impedance state)			240	ns
10	t_V	Data valid	Master (before capture edge) Slave (after enable edge)	Tck / 4	 120	ns ns
11	t_{Hold}	Data hold time (outputs)	Master (before capture edge) Slave (after enable edge)	Tck / 4 0		ns ns
12	t_{Rise}	Rise time (20% V_{DD} to 70% V_{DD}, C_L = 200pF)	Outputs: SCK,MOSI,MISO Inputs: SCK,MOSI,MISO,$\overline{SS}$		100 100	ns µs
13	t_{Fall}	Fall time (70% V_{DD} to 20% V_{DD}, C_L = 200pF)	Outputs: SCK,MOSI,MISO Inputs: SCK,MOSI,MISO,$\overline{SS}$		100 100	ns µs

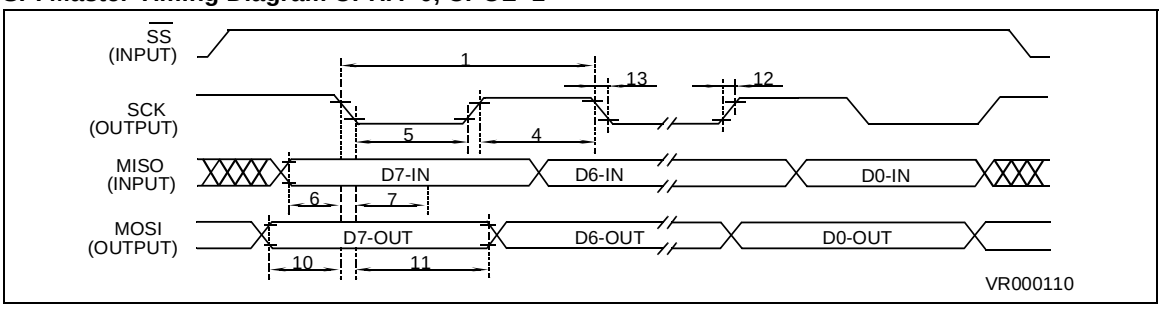
Note:
Measurement points are taken with reference to V_{IH}-V_{IH} / V_{OH}-V_{OH} for positive pulse and V_{IL}-V_{IL} / V_{OL}-V_{OL} for negative pulse
(1) Values guaranteed by design.

Legend:
Tck = INTCLK period = OSCIN period when OSCIN is not divided by 2;
2 x OSCIN period when OSCIN is divided by 2;
OSCIN period x PLL factor when the PLL is enabled.

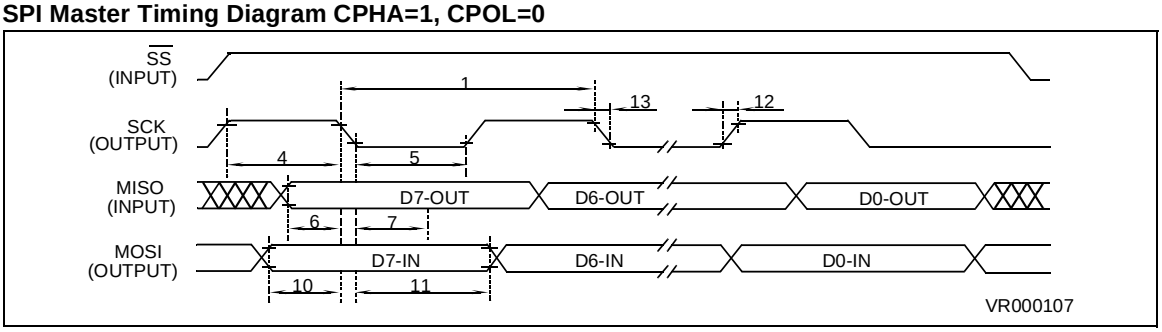
SPI Master Timing Diagram CPHA=0, CPOL=0



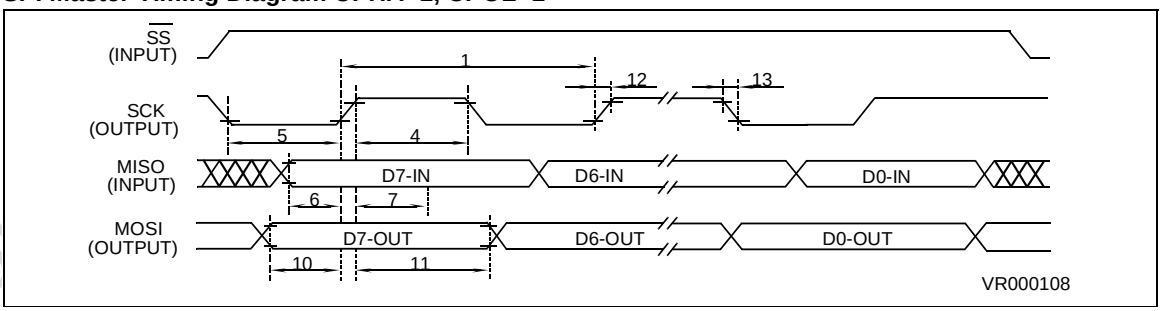
SPI Master Timing Diagram CPHA=0, CPOL=1



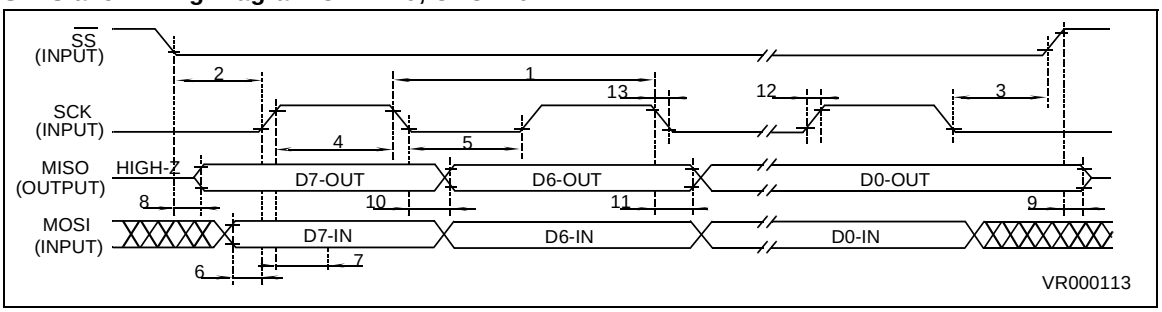
SPI Master Timing Diagram CPHA=1, CPOL=0



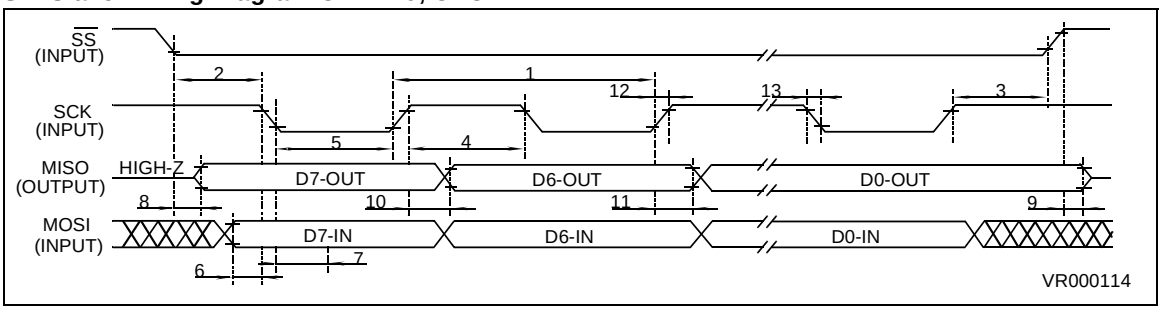
SPI Master Timing Diagram CPHA=1, CPOL=1



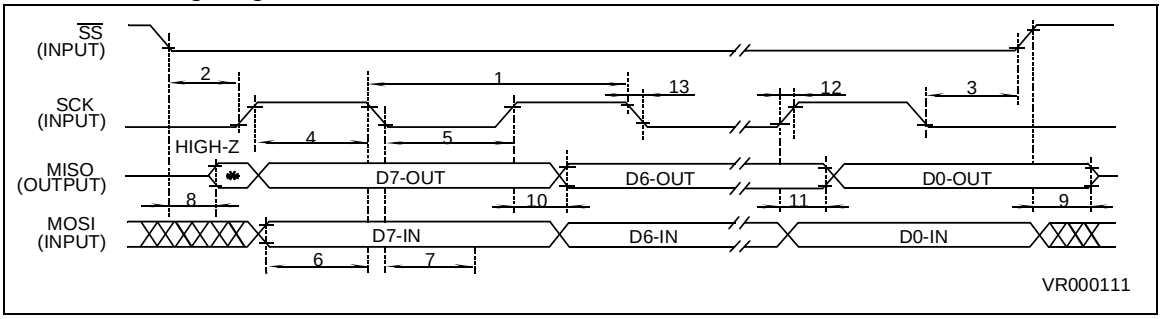
SPI Slave Timing Diagram CPHA=0, CPOL=0



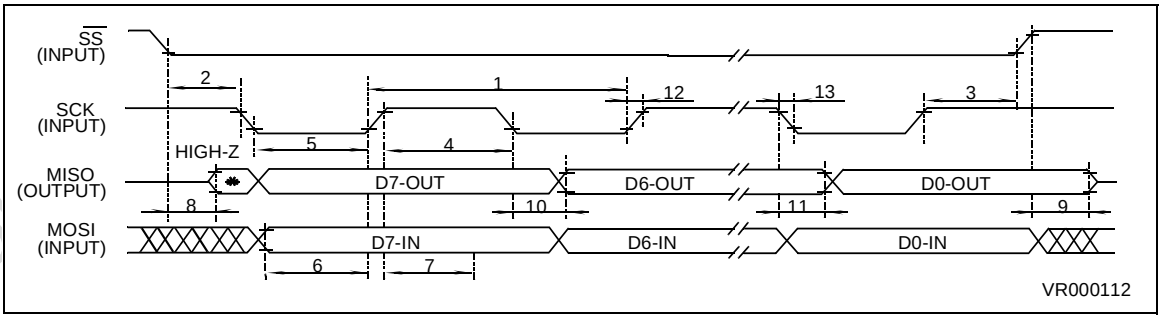
SPI Slave Timing Diagram CPHA=0, CPOL=1



SPI Slave Timing Diagram CPHA=1, CPOL=0



SPI Slave Timing Diagram CPHA=1, CPOL=1



A/D EXTERNAL TRIGGER TIMING TABLE

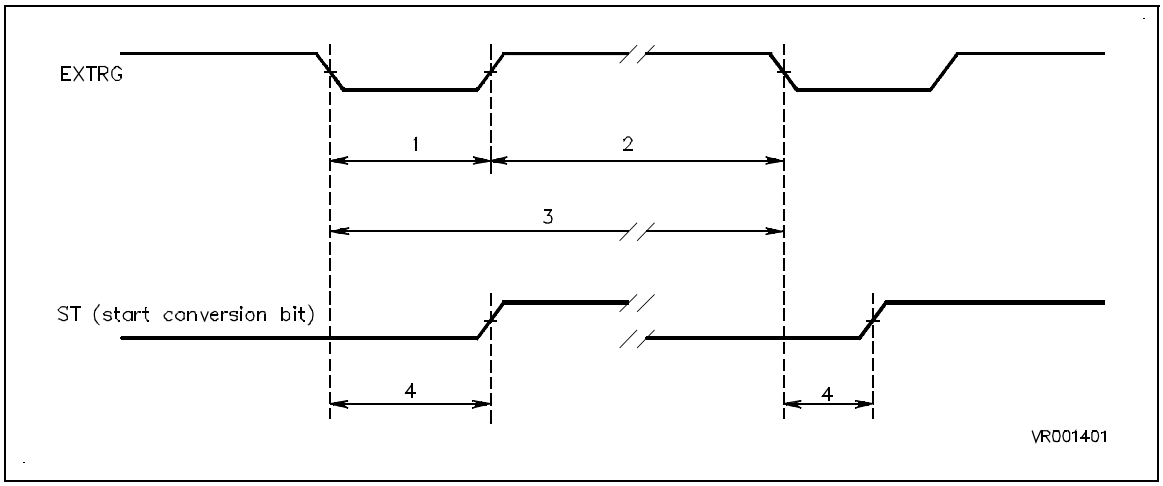
(V_{DD} = 5V ± 10%, T_A = –40°C to +85°C, C_{Load} = 50pF, f_{INTCLK} = 25MHz, unless otherwise specified)

N°	Symbol	Parameter	Value (Note)			Unit
			Formula(1)	Min.	Max.	
1	Tw_{LOW}	External trigger pulse width	≥ 1.5 x Tck	60	-	ns
2	Tw_{HIGH}	External trigger pulse distance	≥ 1.5 x Tck	60	-	ns
3	Tw_{EXT}	External trigger active edges distance	≥ 138 x n x FDF x Tck	n x FDF x 5.52	-	µs
4	Td_{STR}	EXTRG falling edge and first conversion start	0.5 x Tck 1.5 x Tck	20	60	ns

Note: The value in the left hand column shows the formula used to calculate the timing minimum or maximum from the oscillator clock period, standard timer prescaler and counter programmed values.
The value in the right hand two columns show the timing minimum and maximum for an internal clock (INTCLK) at 25MHz.
Measurement points are taken with reference to V_{IH}-V_{IH} for positive pulse and V_{IL}-V_{IL} for negative pulse
(1) Formula guaranteed by design.

Legend:
Tck = INTCLK period = OSCIN period when OSCIN is not divided by 2;
2*OSCIN period when OSCIN is divided by 2;
OSCIN period / PLL factor when the PLL is enabled.
n = number of autoscanned channels (1 ≤ n ≤ 8)
FDF = Frequency Division Factor (ADC prescaler factor), refer to section 7.6.1 on page 147

A/D EXTERNAL TRIGGER TIMING



A/D ANALOG SPECIFICATIONS

(V_{DD} = 5V, T_A = 25°C, f_{INTCLK} = 25MHz, unless otherwise specified)

Parameter	Typical	Minimum	Maximum	Units (1)	Notes
Conversion time		138		INTCLK	(2)(6)
Sample time		85		INTCLK	(6)
Power-up time		60		µs	(6)
Resolution	8	8		bits	
Monotonicity	GUARANTEED				
No missing codes	GUARANTEED				
Zero input reading		00		Hex	(6)
Full scale reading			FF	Hex	(6)
Offset error			1	LSBs	(1)(4)(6)
Gain error			1	LSBs	(4)(6)
Diff. Non Linearity error (DNL)			1	LSBs	(4)(6)
Int. Non Linearity error (INL)			1	LSBs	(4)(6)
Absolute Accuracy			2	LSBs	(4)(6)
Input Resistance	2.7			kΩ	(3)(5)(6)
Hold Capacitance		1.4		pF	(5)(6)
Input Leakage			±1	µA	(6)

Note:

(1) "1LSBideal" has a value of $AV_{DD}/256$

(2) Including sample time

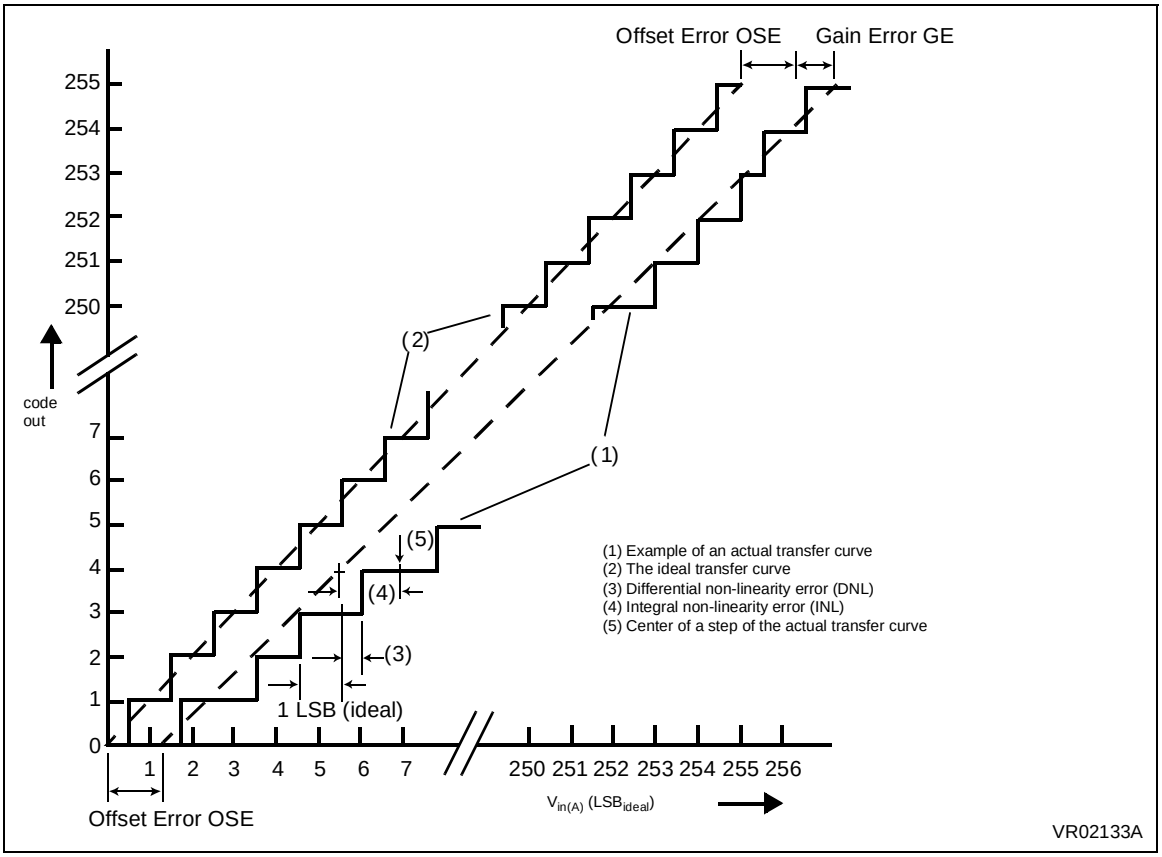
(3) This is the internal series resistance before the sampling capacitor

(4) This is a typical expected value, but not a tested production parameter.
If V(i) is the value of the i-th transition level ($0 \le i \le 254$), the performance of the A/D converter has been evaluated as follows:
OFFSET ERROR= deviation between the actual V(0) and the ideal V(0) (=1/2 LSB)
GAIN ERROR= deviation between the actual V(254) and the ideal V(254) - V(0) (ideal V(254)=AV_{DD}-3/2 LSB)
DNL ERROR= max {[V(i) - V(i-1)]/LSB - 1}
INL ERROR= max {[V(i) - V(0)]/LSB - i}
ABS. ACCURACY= overall max conversion error

(5) Simulated value.

(6) The specified values are guaranteed only if an overload condition occurs on a maximum of 2 non-selected analog input pins and the absolute sum of input overload currents on all analog input pins does not exceed ±10 mA.

Figure 81. A/D Conversion Characteristics



IMC TIMING TABLE

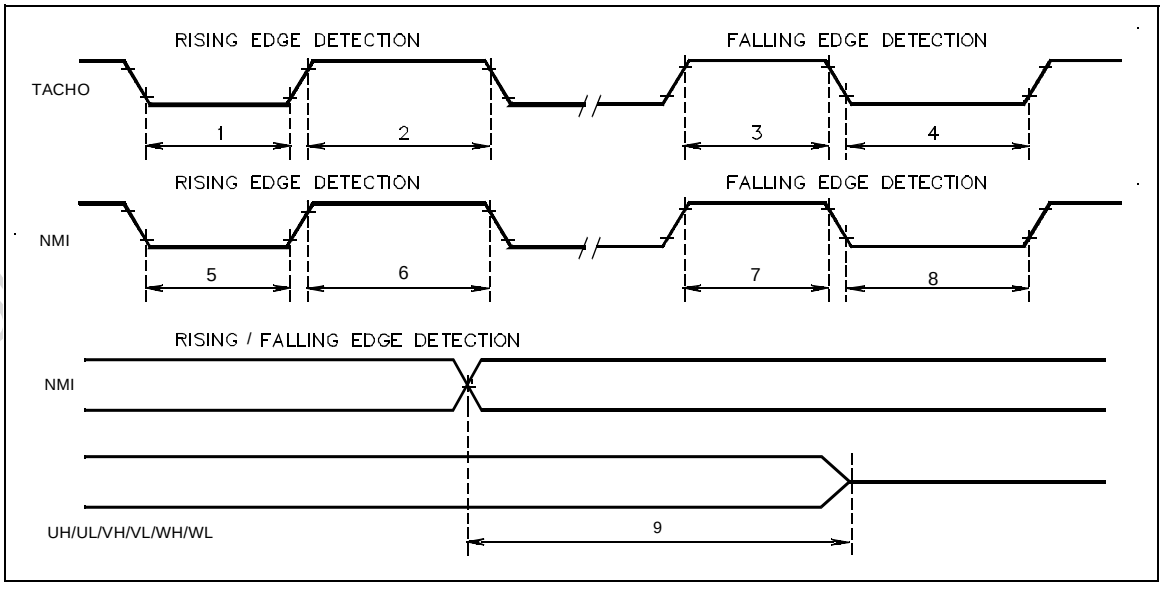
V_{DD} = 5V ± 10%, T_A = –40°C to +85°C, C_{Load} = 50pF, f_{INTCLK} = 25MHz, unless otherwise specified)

N°	Symbol	Parameter	Value (Note)			Unit
			Formula(1)	Min	Max	
1	TwTACLR	Tacho Low Level Minimum Pulse Width in Rising Edge Mode	Tck	40		ns
2	TwTACHR	Tacho High Level Minimum Pulse Width in Rising Edge Mode	Tck	40		ns
3	TwTACHF	Tacho High Level Minimum Pulse Width in Falling Edge Mode	Tck	40		ns
4	TwTACLF	Tacho Low Level Minimum Pulse Width in Falling Edge Mode	Tck	40		ns
5	TwNMILR	NMI Low Level Minimum Pulse Width in Rising Edge Mode		1000		ns
6	TwNMIHR	NMI High Level Minimum Pulse Width in Rising Edge Mode		1000		ns
7	TwNMIHF	NMI High Level Minimum Pulse Width in Falling Edge Mode		1000		ns
8	TwNMILF	NMI Low Level Minimum Pulse Width in Falling Edge Mode		1000		ns
9	TdPHZ	Delay from NMI to Phases in High Impedance		1000		ns

Note: The value in the left hand column shows the formula used to calculate the minimum or maximum timing from the oscillator clock period. The value in the right hand two columns show the minimum and maximum timing for an internal clock at 25MHz (INTCLK).
Measurement points are taken with reference to V_{IH}-V_{IH} for positive pulse and V_{IL}-V_{IL} for negative pulse
(1) Formula guaranteed by design.

Legend:
Tck = INTCLK period = OSCIN period when OSCIN is not divided by 2;
2 x OSCIN period when OSCIN is divided by 2;
OSCIN period x PLL factor when the PLL is enabled.

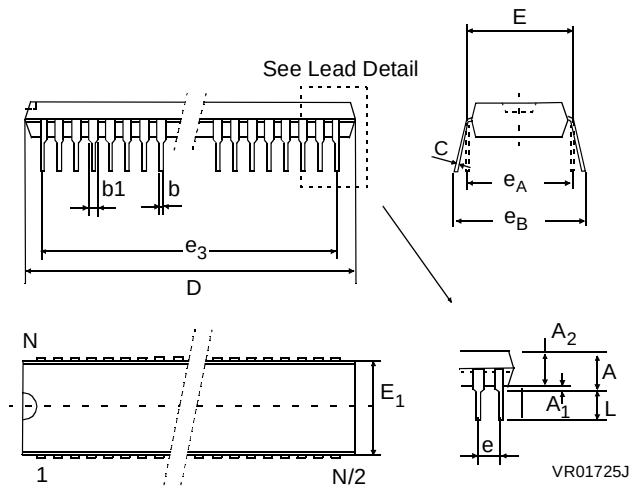
IMC TIMING



9 GENERAL INFORMATION

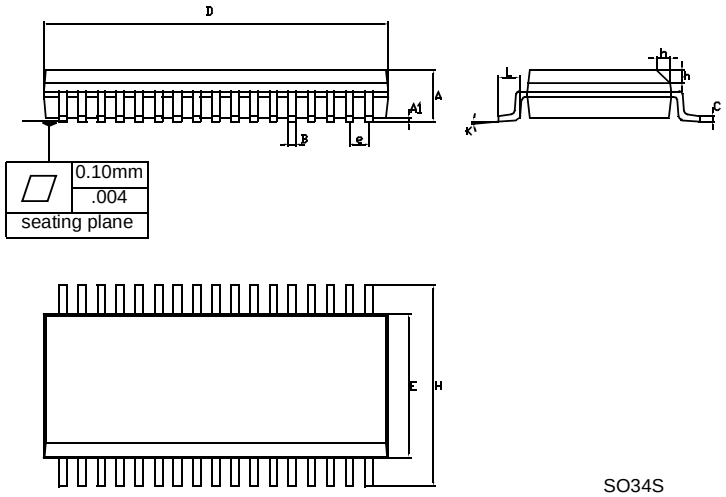
9.1 PACKAGE MECHANICAL DATA

32-PIN SHRINK PLASTIC DUAL IN LINE PACKAGE



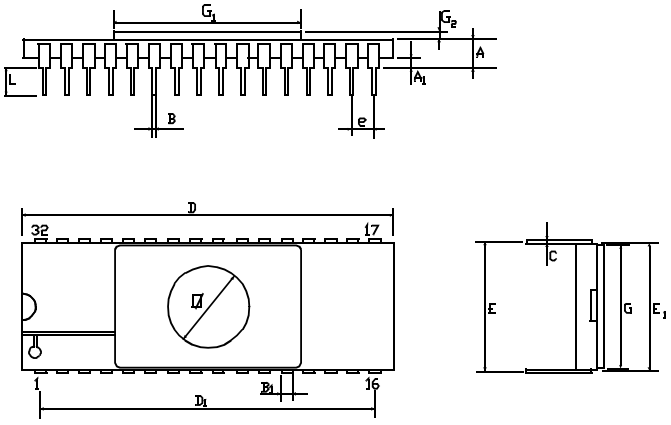
Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A	3.56	3.76	5.08	0.140	0.148	0.200
A1	0.51			0.020		
A2	3.05	3.56	4.57	0.120	0.140	0.180
b	0.36	0.46	0.58	0.014	0.018	0.023
b1	0.76	1.02	1.40	0.030	0.040	0.055
C	0.20	0.25	0.36	0.008	0.010	0.014
D	27.43	27.94	28.45	1.080	1.100	1.120
E	9.91	10.41	11.05	0.390	0.410	0.435
E1	7.62	8.89	9.40	0.300	0.350	0.370
e		1.78			0.070	
eA		10.16			0.400	
eB			12.70			0.500
L	2.54	3.05	3.81	0.100	0.120	0.150
	Number of Pins					
N	32					

34-PIN PLASTIC SMALL OUTLINE PACKAGE



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A	2.46		2.64	0.097		0.104
A1	0.13		0.29	0.005		0.0115
B	0.36		0.48	0.014		0.019
C	0.23		0.32	0.0091		0.0125
D	17.73		18.06	0.698		0.711
E	7.42		7.59	0.292		0.299
e		1.02			0.040	
H	10.16		10.41	0.400		0.410
h	0.64		0.74	0.025		0.029
K				0°		8°
L	0.61		1.02	0.024		0.040
	Number of Pins					
N	34					

32-PIN SHRINK CERAMIC DUAL IN-LINE PACKAGE



CDIP32SW

Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A			3.63			0.143
A1	0.38			0.015		
B	0.36	0.46	0.58	0.014	0.018	0.023
B1	0.64	0.89	1.14	0.025	0.035	0.045
C	0.20	0.25	0.36	0.008	0.010	0.014
D	29.41	29.97	30.53	1.158	1.180	1.202
D1		26.67			1.050	
E		10.16			0.400	
E1	9.45	9.91	10.36	0.372	0.390	0.408
e		1.78			0.070	
G		9.40			0.370	
G1		14.73			0.580	
G2		1.12			0.044	
L		3.30			0.130	
Ø		7.37			0.290	
	Number of Pins					
N	32					

9.2 ORDERING INFORMATION

The following section deals with the procedure for transfer of customer codes to STMicroelectronics.

9.3 Transfer of Customer Code

Customer code is made up of the FASTROM (Factory Advanced Service Technique ROM) contents. The FASTROM contents are to be sent on diskette, or by electronic means, with the hexadecimal file generated by the development tool. All unused bytes must be set to FFh.

The customer code should be communicated to STMicroelectronics with the correctly completed OPTION LIST appended.

The STMicroelectronics Sales Organization will be pleased to provide detailed information on contractual points.

Figure 82. Sales Type Coding Rules

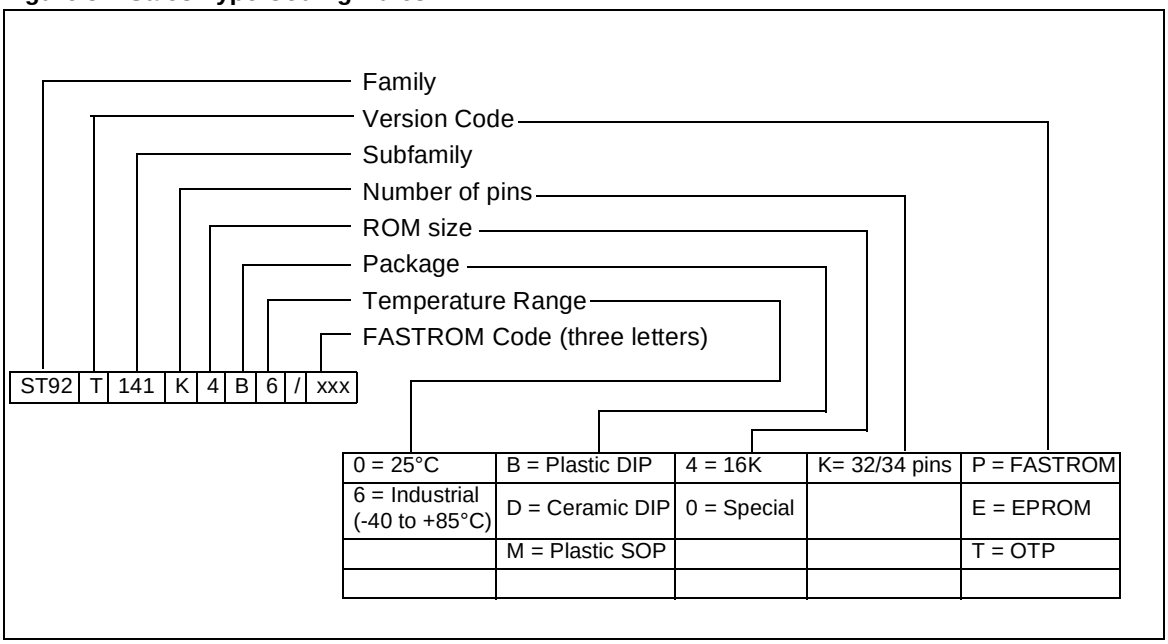


Table 31. Ordering Information

Sales Type [1]	Program Memory (bytes)	RAM (bytes)	Package
ST92P141K4B6/xxx	16K FASTROM	512	PSDIP32
ST92P141K4M6/xxx			SO34
ST92E141K4D0	16K EPROM	512	CSDIP32W
ST92T141K4B6	16K OTP	512	PSDIP32
ST92T141K4M6			SO34

Note 1: xxx stands for the FASTROM code name assigned by STMicroelectronics.

Table 32. Development Tools

Development Tool	Sales Type	Remarks
Real time emulator	ST92141-EMU2	
EPROM Programming Board	ST92E141-EPB/EU	220V Power Supply
	ST92E141-EPB/US	110V Power Supply
Gang Programmer	Third Party product available from Leap at www.leap.com.tw	SDIP32 package SO34 package
C Hiware Compiler and Debugger	ST9P-SWC/PC	for PC

STMicroelectronics OPTION LIST

ST92P141 MICROCONTROLLER FAMILY (FASTROM DEVICE)

Customer: . . . . . .
Address: . . . . . .
. . . . . .
Contact: . . . . . .
Phone No: . . . . . .
Reference/FASTROM Code* :. . . . . .
*The FASTROM code name is assigned by STMicroelectronics.

STMicroelectronics reference:
Device (PSDIP32): [ ] ST92P141K4B6/xxx*

Device (SO34): [ ] ST92P141K4M6/xxx*

Conditioning: [ ] Tube
[ ] Tape & Reel (not available for SDIP packages)

*xxx = FASTROM code name

Software Development: [ ] STMicroelectronics
[ ] Customer
[ ] External laboratory

Special Marking: [ ] No [ ] Yes "_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _"
For marking, one line is possible with maximum 10 characters for PSDIP32 and 16 characters for SO34.
Authorized characters are letters, digits, '.', '-', '/' and spaces only.

We have checked the FASTROM code verification file returned to us by STMicroelectronics. It conforms exactly with the FASTROM code file orginally supplied. We therefore authorize STMicroelectronics to proceed with device manufacture.

Signature . . . . . .

Date . . . . . .

10 SUMMARY OF CHANGES

Description of the changes between the current release of the specification and the previous one.

Rev.	Main Changes	Date
1.7	Added paragraph in section “PLL Clock Multiplier Programming” on page 66 about mandatory use of the divide-by-two prescaler for PLL operation. Updated Option List	Oct 01

Notes: