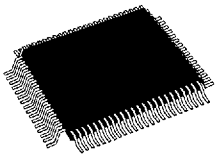
SGS-THOMSON
MICROELECTRONICS

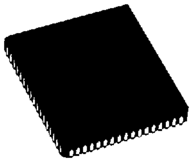
ST90R91

ROMLESS HCMOS MCU WITH BANKSWITCH AND A/D CONVERTER

PRELIMINARY DATA

- Register oriented 8/16 bit CORE with RUN, WFI and HALT modes
- Minimum instruction cycle time : 500ns (12MHz internal)
- ROMless to allow maximum external memory flexibility
- 1536 bytes of internal RAM
- 224 general purpose registers available as RAM, accumulators or index pointers (register file)
- Bankswitch logic allowing a maximum addressing capability of up to 2Mbytes for Program and Dataspace
- 80-pin PQFP package for ST90R91Q
- 68-pin PLCC package for ST90R91C
- DMA controller, Interrupt handler and a Serial Peripheral Interface as standard features
- Up to 32 fully programmable I/O ports
- Up to 7 external plus 1 non-maskable interrupts
- 16-bit Timer with 8 bit Prescaler, able to be used as a Watchdog Timer
- 16-bit Slice Timer with 8 bit Prescaler
- 16-bit Multifunction Timer, with 8 bit prescaler and 13 operating modes
- 4 channel 8 bit Analog to Digital Converter, with Analog Watchdogs and external references
- Rich Instruction Set with 14 Addressing modes
- Division-by-Zero trap generation
- Versatile Development Tools, including assembler, linker, C-compiler, archiver, graphic oriented debugger and hardware emulators
- Real Time Operating System


PQFP80


PLCC68

(Ordering Information at the end of the Datasheet)

Figure 1. 80 Pin PQFP Package

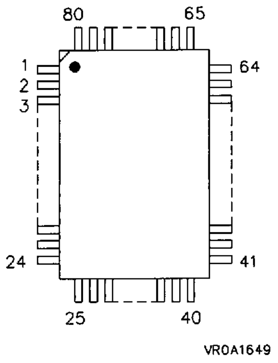


Table 1. ST90R91Q Pin Description

Pin	Name
1	NC
2	P62/A2
3	P61/A1
4	P60/A0
5	P23/BS3
6	P22/BS2
7	P21/BS1
8	P20/BS0
9	P07/AD7
10	P06/AD6
11	P05/AD5
12	P00/AD0
13	P04/AD4
14	P01/AD1
15	P03/AD3
16	P02/AD2
17	V_{CC}
18	$\overline{AS}$
19	$\overline{DS}$
20	R/$\overline{W}$
21	NC
22	NC
23	P50/W/$\overline{R}$
24	P51
25	P52
26	P53/P/$\overline{D}$
27	P54
28	P55
29	P56
30	P57
31	P40
32	P41/WDIN
33	P42/WDOUT
34	P43/T0INPA
35	P44/T0INPB
36	P45/T0OUTA /ADTRG
37	P46/T0OUTB
38	P47/AIN4
39	P80/AIN5
40	NC
64	P75/SDO
63	P74/INT6 /BSH_EN1/SLOUT
62	P73
61	NC
60	P72/INT3 /SLIN
59	P71/INT7
58	P70/INT1/$\overline{WAIT}$
57	P27/BS7
56	P26/BS6
55	P25/BS5
54	P24/BS4
53	P87/NMI
52	P86/INT5
51	P85/INT4
50	P84
49	P83/T0OUTA
48	V_{SS2}
47	$\overline{RESET}$
46	OSCIN
45	V_{SS}
44	OSCOUT
43	P82/AIN7
42	P81/AIN6
41	NC
80	NC
79	P63/A3
78	P64/A4
77	P65/A5
76	P66/A6
75	P67/A7
74	P12/A10
73	P13/A11
72	P11/A9
71	P10/A8
70	P15/A13
69	P14/A12
68	P16/A14
67	P17/A15
66	P77/SDO/SDI
65	P76/SCK/INT2 /BSL_EN1

Note. NC = Not Connected

Figure 2. 68 Pin PLCC Package

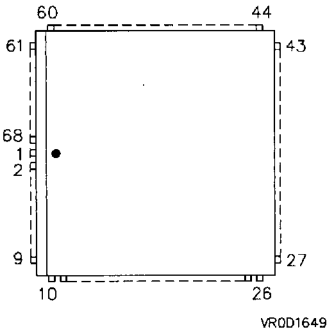


Table 2. ST90R91C Pin Description

Pin	Name
61	P74/INT6 /BSH_EN1/SLOUT
62	P75/SDO
63	P76/SCK/INT2 /BSL_EN1
64	P77/SDO/SDI
65	P17/A15
66	P16/A14
67	P14/A12
68	P15/A13
● 1	P10/A8
2	P11/A9
3	P13/A11
4	P12/A10
5	P67/A7
6	P66/A6
7	P65/A5
8	P64/A4
9	P63/A3
10	P62/A2
11	P61/A1
12	P60/A0
13	P21/BS1
14	P20/BS0
15	P07/AD7
16	P06/AD6
17	P05/AD5
18	P00/AD0
19	P04/AD4
20	P01/AD1
21	P03/AD3
22	P02/AD2
23	V_{CC}
24	$\overline{AS}$
25	$\overline{DS}$
26	R/$\overline{W}$
43	P80/AIN5
42	P47/AIN4
41	P46/T0OUTB
40	P45/T0OUTA /ADTRG
39	P44/T0INPB
38	P43/T0INPA
37	P42/WDOUT
36	P41/WDIN
35	P40
34	P57
33	P56
32	P55
31	P54
30	P53/P/$\overline{D}$
29	P52
28	P51
27	P50/W/$\overline{R}$
60	P72/INT3/SLIN
59	P71/INT7
58	P70/INT1/WAIT
57	P25/BS5
56	P24/BS4
55	P87/NMI
54	P86/INT5
53	P85/INT4
52	P84
51	P83/T0OUTA
50	V_{SS2}
49	$\overline{RESET}$
48	OSCIN
47	V_{SS}
46	OSCOUT
45	P82/AIN7
44	P81/AIN6

1.1 GENERAL DESCRIPTION

The ST90R91 is a Romless member of the ST9 family of microcontrollers, completely developed and produced by SGS-THOMSON Microelectronics using a proprietary n-well HCMOS process.

The Romless part may be used for the prototyping and pre-production phases of development, and offers the maximum in program flexibility

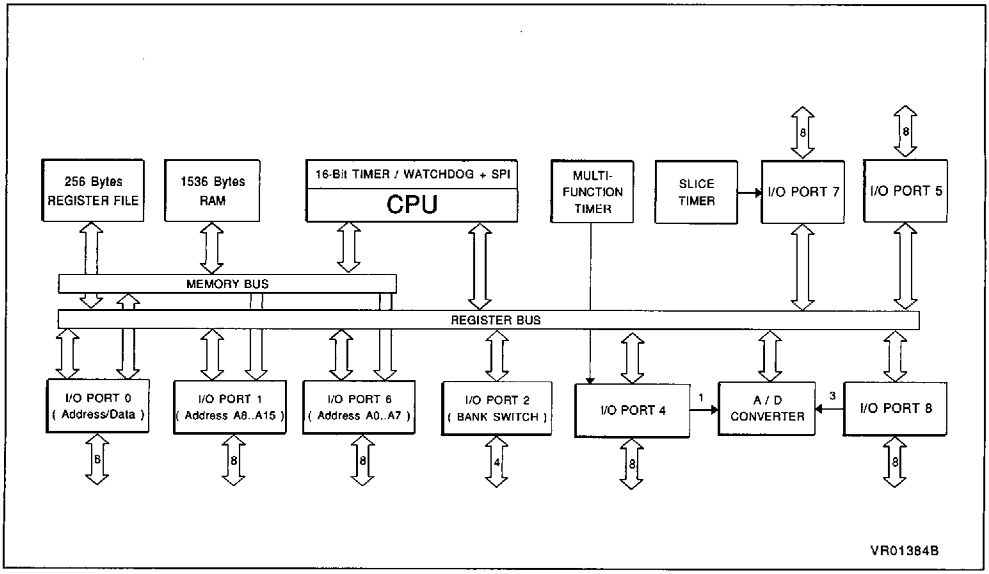
The nucleus of the ST90R91 is the advanced Core which includes the Central Processing Unit (CPU), the Register File, a 16 bit Timer/Watchdog with 8 bit Prescaler, a Serial Peripheral Interface supporting S-bus, I^{2}C-bus and IM-bus Interface, plus two 8 bit I/O ports. The Core has independent memory and register buses allowing a high degree of pipelining to add to the efficiency of the code execution speed of the extensive instruction set. The powerful I/O capabilities demanded by microcontroller applications are fulfilled by the ST90R91 with up to 32 I/O lines dedicated to digital Input/Output. These lines are grouped into nine 8 bit I/O Ports and can be configured on a bit basis under software control to provide timing, status signals, an address/data bus for interfacing to the external memory, timer inputs and outputs, analog inputs, external interrupts and serial or parallel I/O.

Three basic memory spaces are avalable to support this wide range of configurations: Program Memory, Data Memory and the internal Register File, which includes the control and status registers of the on-chip peripherals.

A 16 bit MultiFunction Timer, with an 8 bit Prescaler and 13 operating modes allows simple use for complex waveform generation and measurement, PWM functions and many other system timing functions by the usage of the two associated DMA channels for each timer. In addition there is an 4 channel Analog to Digital Converter with integral sample and hold, fast 11µs conversion time and 8 bit resolution. An Analog Watchdog feature is included for two input channels.

Completing the device is the Slice Timer, capable of generating PWM signals and simple timing functions.

Figure 1-3. ST90R91 Block Diagram



1.2 PIN DESCRIPTION

$\overline{\text{AS}}$. *Address Strobe (output, active low, 3-state).* Address Strobe is pulsed low once at the beginning of each memory cycle. The rising edge of $\overline{\text{AS}}$ indicates that address, Read/Write (R/$\overline{\text{W}}$), and Data Memory signals are valid for program or data memory transfers. Under program control, $\overline{\text{AS}}$ can be placed in a high-impedance state along with Port 0 and Port 1, Data Strobe ($\overline{\text{DS}}$) and R/$\overline{\text{W}}$.

$\overline{\text{DS}}$. *Data Strobe (output, active low, 3-state).* Data Strobe provides the timing for data movement to or from Port 0 for each memory transfer. During a write cycle, data out is valid at the leading edge of $\overline{\text{DS}}$. During a read cycle, Data In must be valid prior to the trailing edge of $\overline{\text{DS}}$. When the ST90R91 accesses on-chip memory, $\overline{\text{DS}}$ is held high during the whole memory cycle. It can be placed in a high impedance state along with Port 0, Port 1, $\overline{\text{AS}}$ and R/$\overline{\text{W}}$.

R/$\overline{\text{W}}$. *Read/Write (output, 3-state).* Read/Write determines the direction of data transfer for external memory transactions. R/$\overline{\text{W}}$ is low when writing to external program or data memory, and high for all other transactions. It can be placed in a high impedance state along with Port 0, Port 1, Port 6, $\overline{\text{AS}}$ and $\overline{\text{DS}}$.

$\overline{\text{RESET}}$. *Reset (input, active low).* The ST9 is initialised by the Reset signal. With the deactivation of $\overline{\text{RESET}}$, program execution begins from the Program memory location pointed to by the vector contained in program memory locations 00h and 01h.

OSCIN, OSCOUT. *Oscillator (input and output).* These pins connect a parallel-resonant crystal (24MHz maximum), or an external source to the on-chip clock oscillator and buffer. OSCIN is the input of the oscillator inverter and internal clock generator; OSCOUT is the output of the oscillator inverter.

V_{DD}. Main Power Supply Voltage (5V± 10%)

V_{SS}., V_{SS2} Digital Circuit Ground.

P0.0-P0.7, P1.0-P1.7, P6.0-P6.7 *(Input/Output, TTL or CMOS compatible).* 8 lines grouped into I/O ports of 8 bits providing the external memory interface to address the external program memory.

P2.0-P2.7, P4.0-P4.7, P5.0-P5.7, P7.0-P7.7, P8.0-P8.7 *I/O Port Lines (Input/Output, TTL or CMOS compatible).* 8 lines grouped into I/O ports of 8 bits, bit programmable under program control as general purpose I/O or as alternate functions.

1.2.1 I/O Port Alternate Functions

Each pin of the I/O ports of the ST90R91 may assume software programmable Alternative Functions as shown in the Pin Configuration Drawings. Table 3 shows the Functions allocated to each I/O Port pins.

PIN DESCRIPTION (Continued)

Table 3. ST90R91 I/O Port Alternate Function Summary

I/O PORT	Name	Function IN/OUT	Alternate Function	Pin Assignment	
Port.bit				PLCC	PQFP
P0.0	A0/D0	I/O	Address/Data bit 0 mux	18	12
P0.1	A1/D1	I/O	Address/Data bit 1 mux	20	14
P0.2	A2/D2	I/O	Address/Data bit 2 mux	22	16
P0.3	A3/D3	I/O	Address/Data bit 3 mux	21	15
P0.4	A4/D4	I/O	Address/Data bit 4 mux	19	13
P0.5	A5/D5	I/O	Address/Data bit 5 mux	17	11
P0.6	A6/D6	I/O	Address/Data bit 6 mux	16	10
P0.7	A7/D7	I/O	Address/Data bit 7 mux	15	9
P1.0	A8	O	Address bit 8	1	71
P1.1	A9	O	Address bit 9	2	72
P1.2	A10	O	Address bit 10	4	74
P1.3	A11	O	Address bit 11	3	73
P1.4	A12	O	Address bit 12	67	69
P1.5	A13	O	Address bit 13	68	70
P1.6	A14	O	Address bit 14	66	68
P1.7	A15	O	Address bit 15	65	67
P2.0	BS0	O	Bank Switch Address 0	14	8
P2.1	BS1	O	Bank Switch Address 1	13	7
P2.2	BS2	O	Bank Switch Address 2	-	6
P2.3	BS3	O	Bank Switch Address 3	-	5
P2.4	BS4	O	Bank Switch Address 4	56	54
P2.5	BS5	O	Bank Switch Address 5	57	55
P2.6	BS6	O	Bank Switch Address 6	-	56
P2.7	BS7	O	Bank Switch Address 7	-	57
P4.0		I/O		35	31
P4.1	WDIN	I	T/WD input	36	32
P4.2	WDOUT	O	T/WD output	37	33
P4.3	T0INA	I	MF Timer 0 Input A	38	34
P4.4	T0INB	I	MF Timer 0 Input B	39	35
P4.5	T0OUTA	O	MF Timer 0 output A	40	36
P4.5	ADTRG	I	A/D Conversion Trigger	40	36
P4.6	T0OUTB	O	MF Timer 0 output B	41	37
P4.7	AIN4	I	A/D Analog Input 4	42	38

PIN DESCRIPTION (Continued)

Table 3. ST90R91 I/O Port Alternate Function Summary(Continued)

I/O PORT	Name	Function IN/OUT	Alternate Function	Pin Assignment	
Port.bit				PLCC	PQFP
P5.0	W/$\overline{R}$	O	Write/$\overline{Read}$ Strobe	27	23
P5.1		I/O		28	24
P5.2		I/O		29	25
P5.3	P/$\overline{D}$	O	Program/Data Space Select	30	26
P5.4		I/O		31	27
P5.5		I/O		32	28
P5.6		I/O		33	29
P5.7		I/O		34	30
P6.0	A0	O	Address bit 0 (non mux)	12	4
P6.1	A1	O	Address bit 1 (non mux)	11	3
P6.2	A2	O	Address bit 2 (non mux)	10	2
P6.3	A3	O	Address bit 3 (non mux)	9	79
P6.4	A4	O	Address bit 4 (non mux)	8	78
P6.5	A5	O	Address bit 5 (non mux)	7	77
P6.6	A6	O	Address bit 6 (non mux)	6	76
P6.7	A7	O	Address bit 7 (non mux)	5	75
P7.0	INT1	I	External Interrupt 1	58	58
P7.0	$\overline{WAIT}$	I	External Wait Input	58	58
P7.1	INT7	I	External Interrupt 7	59	59
P7.2	INT3	I	External Interrupt 3	60	60
P7.2	SLIN	I	Slice Timer Input	60	60
P7.3		I/O		-	62
P7.4	INT6	I	External Interrupt 6	61	63
P7.4	BSH_EN1	I	Bankswitch High Nibble Enable	61	63
P7.4	SLOUT	O	Slice Timer Output	61	63
P7.5	SDO	O	SPI Serial Data Output	62	64
P7.6	SCK	O	SPI Serial Clock	63	65
P7.6	INT2	I	External Interrupt 2	63	65
P7.6	BSL_EN1	I	Bankswitch Low Nibble Enable	63	65
P7.7	SDO	O	SPI Serial Data Output	64	66
P7.7	SDI	I	SPI Serial Data Input	64	66
P8.0	AIN5	I	A/D Analog Input 5	43	39

PIN DESCRIPTION (Continued)

Table 3. ST90R91 I/O Port Alternate Function Summary(Continued)

I/O PORT	Name	Function IN/OUT	Alternate Function	Pin Assignment	
Port.bit				PLCC	PQFP
P8.1	AIN6	I	A/D Analog Input 6	44	42
P8.2	AIN7	I	A/D Analog Input 7	45	43
P8.3	T0OUTA	O	MF Timer 0 output A	51	49
P8.4		I/O		52	50
P8.5	INT4	I	External interrupt 4	53	51
P8.6	INT5	I	External interrupt 5	54	52
P8.7	NMI	I	Non-Maskable Interrupt	55	53

2 CORE ARCHITECTURE

2.1 CORE ARCHITECTURE

The Core or Central Processing Unit (CPU) of the ST9 includes the 8 bit Arithmetic Logic Unit and the 16 bit Program Counter, System and User Stack Pointers. The microcoded Instruction Set is highly optimised for both byte (8 bit) and word (16 bit) data, BCD and Boolean data types, with 14 addressing modes.

Three independent buses are controlled by the Core, a 16 bit Memory bus, an 8 bit Register addressing bus and a 6 bit Interrupt/DMA bus connected to the interrupt and DMA controllers in the on-chip peripherals and the Core. This multiple bus architecture allows a high degree of pipelining and parallel operation, giving the ST9 its efficiency in both numerical calculations and communication with the on-chip peripherals.

2.2 ADDRESS SPACES

The ST9 has three separate address spaces:

- Register File: 240 8-bit registers plus up to 64 pages of 16 bytes each, located in the on-chip peripherals.
- Data memory with up to 64K (65536) bytes
- Program memory with up to 64K (65536) bytes

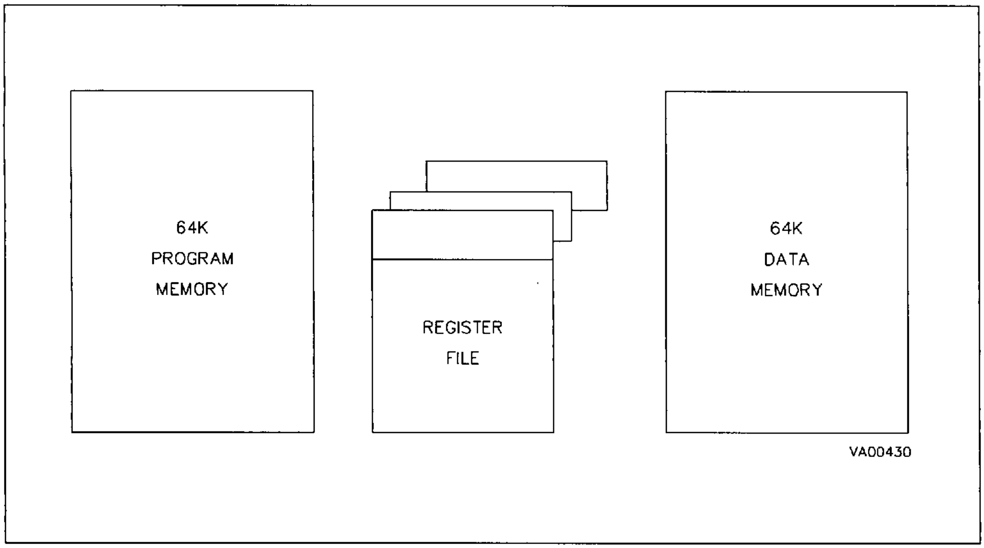
The Data and Program memory spaces will be addressed in further detail in the next section.

2.2.1 Register File

The Register File consists of:

- 224 general purpose registers R0 to R223
- 16 system registers in the System Group (R224 to R239).
- I/O pages depending on the configuration of the ST9, each containing up to 16 registers, with paging facilities based on the top group (R240 to R255).

Figure 2-1. Address Spaces



ADDRESS SPACES (Continued)

Figure 2-2. Register Grouping

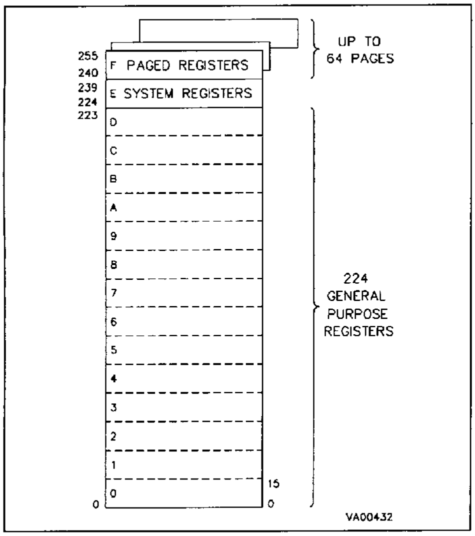


Figure 2-3. Page Pointer Configuration

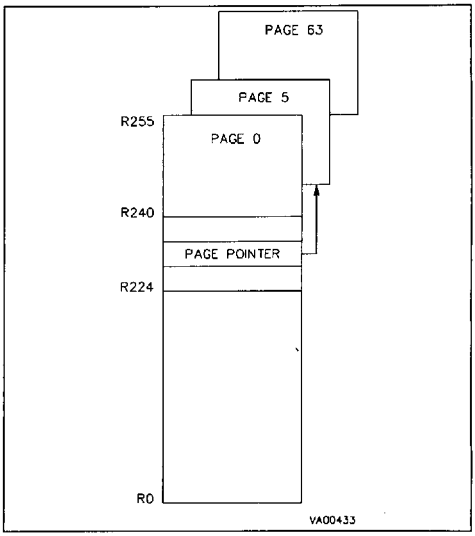
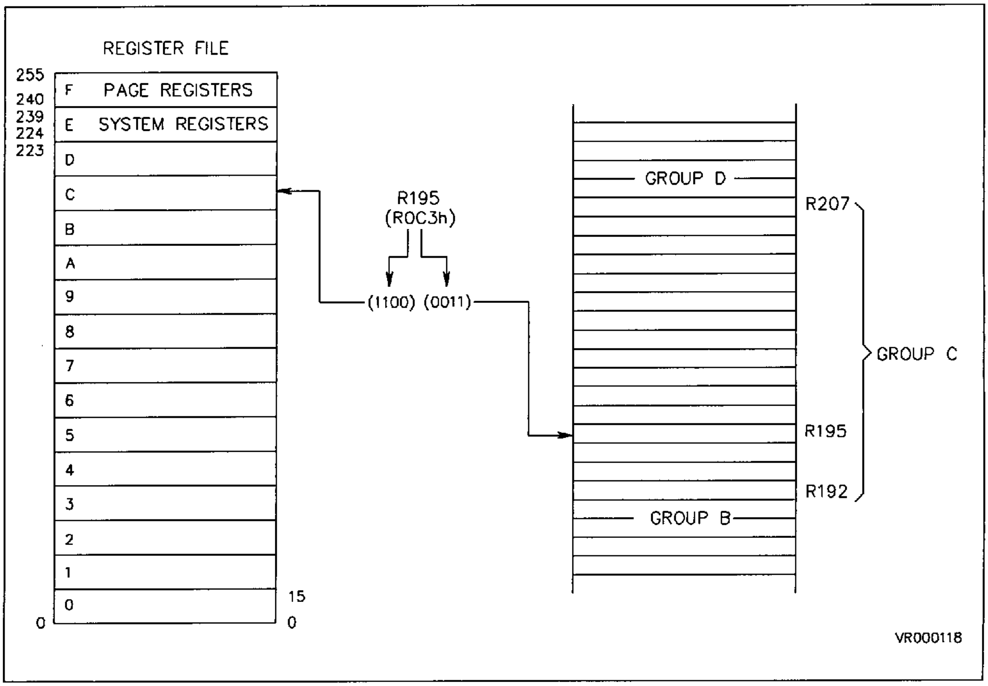


Figure 2-4. Addressing the Register File



ADDRESS SPACES (Continued)

2.2.2 Addressing Registers

All registers in the Register File and pages can be specified by using a decimal, hex or binary address, e.g. R231, RE7h or R11100111b is the same register.
The registers can be referred to by their hexadecimal group address, so that registers R0-R15 form group 0, R160-R175 form group A and so on.

Working Register Addresses

The 8-bit register address is formed by 2 nibbles, for example, for register R195 or RC3h or R11000011, 1100 specifies the 13th group (i.e. group C) and 0011 specifies the 3rd register in that group.

Working registers are addressed by supplying the least significant nibble in the instruction and adding it to the most significant nibble found in the Register Pointer (R233). Working register addressing is shown in Figures 2-4.

System Registers

The 16 system registers at addresses R224 to R239 form Group E.
The system registers are addressable using any of the 4 register addressing modes and the most significant nibble will, in all cases, be 14 (0Eh).

Paged Registers

There are a maximum of 64 pages each containing 16 registers. These are addressed using the register addressing modes with the addition of the Page Pointer register, R234. This register selects the page to be addressed in 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 (R242) of page 5.

These paged registers hold data and control registers related to the on-chip peripherals, and thus the configuration depends upon the peripheral organisation of each ST9 family member. i.e. pages only exist if the peripheral exists.

Available pages are shown in Table 2-2.

2.2.3 Input/Output Ports

The Input/Output ports are located in two areas. The port registers for Ports 0-5 are located at the bottom of the System register group in locations R224 to R229.

Each Port has three associated Control registers, which determine the individual pin modes (I/O, Open-Drain etc). These registers are located in pages 2 and 3.

Table 2-1. 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

ADDRESS SPACES (Continued)

Table 2-2. Group F Peripheral Organization

Applicable for ST90R91

	DEC	00	02	03	09	10	11	43	63
DEC	HEX	00	02	03	09	0A	0B	2B	3F
R255	RFF	RES.	RES.	PORT 7	RES.	MFT 0	RES.	RES.	A/D
R254	RFE	MSPI	RES.	PORT 7	RES.	MFT 0	RES.	RES.	A/D
R253	RFD	MSPI	RES.	PORT 7	RES.	MFT 0	RES.	RES.	A/D
R252	RFC	WCR	RES.	PORT 7	RES.	MFT 0	RES.	RES.	A/D
R251	RFB	T/WD	RES.	RES.	RES.	MFT 0	RES.	PORT8	A/D
R250	RFA	T/WD	PORT 2	RES.	RES.	MFT 0	RES.	PORT8	A/D
R249	RF9	T/WD	PORT 2	RES.	RES.	MFT 0	RES.	PORT8	A/D
R248	RF8	T/WD	PORT 2	RES.	RES.	MFT 0	RES.	PORT8	A/D
R247	RF7	EXT INT	RES.	RES.	RES.	MFT 0	RES.	RES.	A/D
R246	RF6	EXT INT	PORT1	PORT 5	RES.	MFT 0	RES.	RES.	A/D
R245	RF5	EXT INT	PORT1	PORT 5	RES.	MFT 0	RES.	RES.	A/D
R244	RF4	EXT INT	PORT1	PORT 5	RES.	MFT 0	RES.	RES.	A/D
R243	RF3	EXT INT	RES.	RES	MFT0	MFT 0	SLT	RES.	A/D
R242	RF2	EXT INT	PORT 0	PORT 4	MFT0	MFT 0	SLT	RES.	A/D
R241	RF1	MIRROR	PORT 0	PORT 4	MFT0	MFT 0	SLT	RES.	A/D
R240	RF0	RES	PORT 0	PORT 4	MFT0	MFT 0	SLT	RES.	A/D

2.3 SYSTEM REGISTERS

Following is the description of System Registers. For PORT0 to PORT5 Registers, please refer to I/O Port Chapter.

Figure 2-5. System Registers

R239 (EFh)	SYS. STACK POINTER LOW
R238 (EEh)	SYS. STACK POINTER HIGH
R237 (EDh)	USER STACK POINTER LOW
R236 (ECh)	USER STACK POINTER HIGH
R235 (EBh)	MODE REGISTER
R234 (EAh)	PAGE POINTER
R233 (E9h)	REGISTER POINTER 1
R232 (E8h)	REGISTER POINTER 0
R231 (E7h)	FLAGS
R230 (E6h)	CENTRAL INT. CNTL REG
R229 (E5h)	PORT5
R228 (E4h)	PORT4
R227 (E3h)	RESERVED
R226 (E2h)	PORT2
R225 (E1h)	PORT1
R224 (E0h)	PORT0

2.3.1 Central Interrupt Control Register

This Register CICR is located in the system Register Group at the address R230 (E6h). Please refer to "INTERRUPT" and "DMA" chapters in order to get the background of the ST9 interrupt philosophy.

CICR R230 (E6h) System Read/Write
Central Interrupt Control Register
Reset Value : 1000 0111

7							0
GCEN	TLIP	TLI	IEN	IAM	CPL2	CPL1	CPL0

b7 = **GCEN:** *Global Counter Enable.* This bit is the Global Counter Enable of the Multifunction Timers. The GCEN bit is ANDed with the CE (Counter Enable) bit of the Timer Control Register (explained in the Timer chapter) in order to enable the Timers when both bits are set. This bit is set after the Reset cycle.

b6 = **TLIP:** *Top Level Interrupt Pending.* This bit is automatically set when a Top Level Interrupt Request is recognized. This bit can also be set by Software in order to simulate a Top Level Interrupt Request.

b5 = **TLI:** *Top Level Interrrupt bit.* When this bit is set, a Top Level interrupt request is acknowledged depending on the IEN bit and the TLNM bit (in Nested Interrupt Control Register). If the TLM bit is reset the top level interrupt acknowledgement depends on the TLNM alone.

b4 = **IEN:** *Enable Interrupt.* This bit, (when set), allows interrupts to be accepted. When reset no interrupts other than the NMI can be acknowledged. It is cleared by interrupt acknowledgement for concurrent mode and set by interrupt return (`iret`). It can be managed by hardware and software (`ei` and `di` instruction).

b3 = **IAM:** *Interrupt Arbitration Mode.* This bit covers the selection of the two arbitration modes, the Concurrent Mode being indicated by the value "0" and the Fully Automatic Nested Mode by the value "1". This bit is under software control.

b2-b0 = **CPL2-CPL0:** *Current Priority Level.* These three bits record the priority level of the interrupt presently under service (i.e. the Current Priority Level, CPL). For these priority levels 000 is the highest priority and 111 is the lowest priority. The CPL bits can be set by hardware or software and give the reference by which following interrupts are either left pending or able to interrupt the current interrupt. When the present interrupt is replaced by one of a greater priority, the current priority value is automatically stored until required.

SYSTEM REGISTERS (Continued)

2.3.2 Flag Register

The Flag Register contains 8 flags indicating the status of the ST9. During an interrupt the flag register is automatically stored in the system stack area and recalled at the end of the interrupt service routine so that the ST9 is returned to the original status. This occurs for all interrupts and, when operating in the nested mode, up to seven versions of the flag register may be stored.

FLAGR R231 (E7h) System Read/Write Flag Register

Reset value: undefined

7							0
C	Z	S	V	DA	H	UF	DP

b7 = **C:** *Carry Flag.* The carry flag C is affected by the following instructions:

Addition (`add, addw, adc, adcw`),
Subtraction (`sub, subw, sbc, sbcw`),
Compare (`cp, cpw`),
Shift Right Arithmetic (`sra, sraw`),
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 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 (changed to "0" if "1", and vice versa) by the Complement Carry Flag (`ccf`) instruction.

b6 = **Z:** *Zero Flag.* The Zero flag is affected by the following instructions:

Addition (`add, addw, adc, adcw`),
Subtraction (`sub, subw, sbc, sbcw`),
Compare (`cp, cpw`),
Shift Right Arithmetic (`sra, sraw`),
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 register being used as an accumulator register is zero, following one of the above operations.

b5 = **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 byte operation) or bit 15 (for word operation) of the register used as an accumulator is one.

b4 = **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 twos-complement notation.

b3 = **DA:** *Decimal Adjust Flag.* The Decimal Adjust 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 Decimal Adjust flag cannot normally be used as a test condition by the programmer.

b2 = **H:** *Half Carry Flag.* The Half Carry 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 Half Carry 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 Decimal Adjust flag, this flag is not normally accessed by the user.

b1 = **UF:** *User Flag.* Bit 1 in the flag register (UF) is available to the user, but it must be set or cleared by an instruction.

b0 = **DP:** *Data/Program Memory Flag.* This bit in the flag register indicates which memory area is addressed. Its value is affected by the Set Data Memory (`sdm`) and Set Program Memory (`spm`) instructions.

If the bit is set, the ST9 addresses the Data Memory Area; when the bit is cleared, the ST9 addresses the Program Memory Area. By reading this bit, the user can verify in which memory area the processor is working. The user writes this bit with the `sdm` or `spm` instructions.

SYSTEM REGISTERS (Continued)

2.3.3 Register Pointing Techniques

Two registers, R232 and R233, within the system register group, are available for register pointing. R232 and R233 may be used together as a single pointer for a 16 register working space or separately for two 8 register spaces, in which case R232 becomes Register Pointer 0 (RP0) and R233 becomes Register Pointer 1 (RP1).

The instructions `srp`, `srp0` and `srp1` (the Set Register Pointer instructions) automatically inform the ST9 whether the Register File is to operate with a single 16-register group or two 8-register groups. The `srp0` and `srp1` instructions automatically set the twin 8-register group mode while the `srp` instruction sets the single 16-register group mode. There is no limitation on the order or positions of these chosen register groups other than they must be on 8 or 16 register boundaries.

The addressing of working registers involves use of the Register Pointer value plus an offset value given by the number of the addressed working register.

When addressing a register, the most significant nibble (bits 4-7) gives the group address and the least significant nibble (bits 0-3) gives the register within that group.

REGISTER POINTER 0

RP0 R232 (E8h) System Read/Write
Register Pointer 0

Reset Value : undefined

7							0
RG7	RG6	RG5	RG4	RG3	RPS	D1	D0

b7-b3 = **RG7-RG3:** *Register Group number.* These bits contain the number (from 0 to 31) of the group of working registers indicated in the instructions `srp0` or `srp`. When using a 16-register group, a number between 0 and 31 must be used in the `srp` instruction indicating one of the two adjacent 8-register group of working registers used. RG7 is the MSB.

b2 = **RPS:** *Register Pointer Selector.* This bit is set by the instructions `srp0` and `srp1` to indicate that a double register pointing mode is used. Otherwise, the instruction `srp` resets the RPS bit to zero to indicate that a single register pointing mode is used.

b1,b0 = **D1,D0:** These bits are fixed by hardware to zero and are not affected by any writing instruction trying to modify their value.

REGISTER POINTER 1

RP1 R233 (E9h) System Read/Write
Register Pointer 1

Reset Value : undefined

7							0
RG7	RG6	RG5	RG4	RG3	RPS	D1	D0

This register is used only with double register pointing mode; otherwise, using single register pointing mode, the RP1R register has to be considered as reserved and not usable as a general purpose register.

b7-b3 = **RG7-RG3:** *Register Group number.* These bits contain the number (from 0 to 31) of the group of 8 working registers indicated in the instructions `srp1`. Bit 7 is the MSB.

b2 = **RPS:** *Register Pointer Selector.* This bit is automatically set by the instructions `srp0` and `srp1` to indicate that a double register pointing mode is used. Otherwise the instruction `srp` reset the RPS bit to zero to indicate that a single register pointing mode is used.

b1,b0 = **D1,D0:** These bits are hardware fixed to zero and are not affected by any writing instruction trying to modify their value.

Note. If working in twin 8-register group mode but only using `srp0` (i.e. only using one 8-register group) the unused register (R233) is to be considered as reserved and not usable as a general purpose register.

The group of registers immediately below the system registers (i.e. group D, R208-R223) can only be accessed via the Register Pointers. To address group D then, it is necessary to set the Register Pointer to group D and then use the addressing procedure for working registers. The programmer is required to remember that the group D should be used as a stacking area. This point is also covered in the Stack Pointers paragraph.

SYSTEM REGISTERS (Continued)

EXAMPLES

Using the Single 16 Register Group

When the system is operating in the single 16-register group mode, the registers are referred to as r0-r15. In this mode, the offset value (i.e. the number of the working register referred to) is supplied in the address (preceded by a small r, e.g. r5) and is added to the Register Pointer 0 value to give the absolute address.

For example, if the Register Pointer contains the value 70h, then working register r7 would have the absolute address, R77h.

In this mode, the single 16-registers group will always start from the lowest even number equal or lower to the number given in the instruction.

Example: `srp #3` is equivalent to `srp #2`.

Using the Twin 8-Register Group

When working in the twin working group mode, the registers pointed by Register Pointer 0 (RP0R), are referred as r0-r7 and those pointed by Register Pointer 1 (RP1R), are referred to as r8-r15, regardless of their absolute addresses. In this mode, when operating with the first 8 working registers (i.e. r0 - r7) the working register number acts as an offset which is added to the value in Register Pointer 0.

So if Register Pointer 0 contains the value 96, then working register 0 has the absolute address 96, working register 5 has the absolute address 101, and so on. The second group of working registers, r8-r15, has the offset values 0 to 7 respectively (i.e. r8 has the offset value 0, r9 has the offset value 1, and so on), this offset value being added to the value in Register Pointer 1.

For example, given that the value in Register Pointer 1 is 32, then working register 12 supplies an offset value of 4 (given by 12 minus 8) to the value in Register Pointer 1 to give an absolute address of 36.

Figure 2-6. Single 16 Register pointing Mode

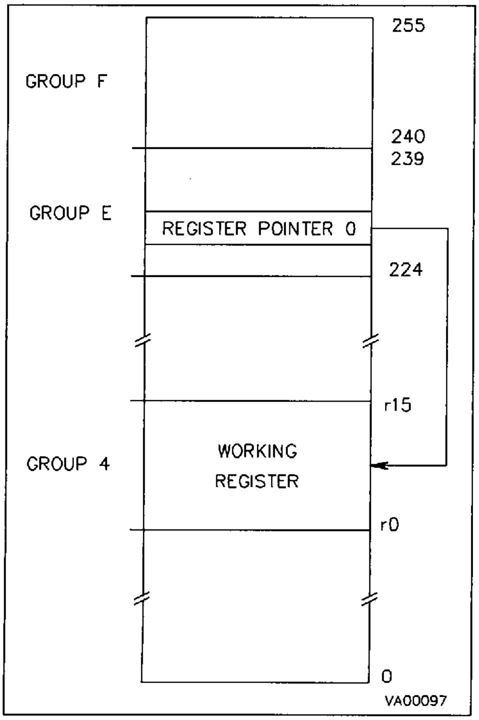
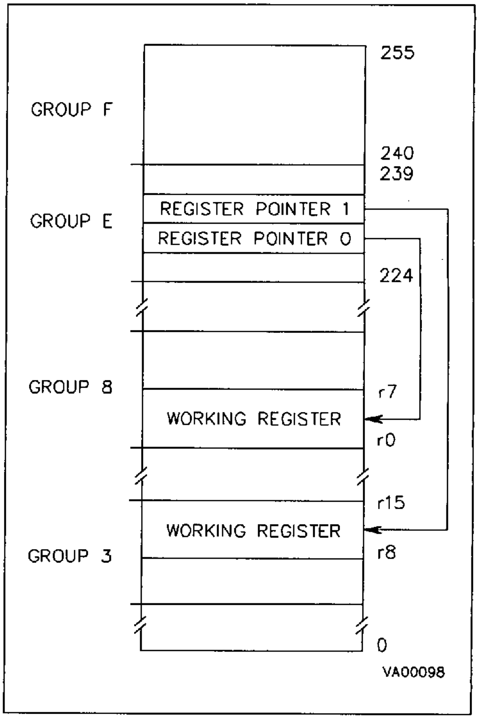


Figure 2-7. Double Register pointing Mode



7929237 0057964 757

SYSTEM REGISTERS (Continued)

2.3.4 Page Configuration

The pages are available to be used for the storage of control information (such as interrupt vector pointers) relevant to particular peripherals. There are up to 64 pages (each with 16 registers) based on registers R240-R255. These paged registers are addressable via the page pointer register (PPR), which is system register R234.

To address a paged register the page pointer register (R234) must be loaded with the relevant page number using the `spp` instruction (Set Page Pointer) and subsequently any address from the top (F) group (R240-R255) will be referred to that page.

For example if register 23 contains the value 44, the following sequence loads the third register R242 on page 5 with the value 44.

```
spp 5
ld R242, R23
```

PPR R234 (EAh) System Read/Write
Page Pointer Register

Reset value : undefined

7							0
PP7	PP6	PP5	PP4	PP3	PP2	D1	D0

b7-b2 = **PP7-PP2:** *Page Pointer.* These bits contain the number (between 0 to 63) of the page chosen by the instruction `ssp` (Set Page Pointer). PP7 is the MSB of the page address. Once the page pointer has been set, there is no need to refresh it unless a different page is required.

b1-b0 = **D1,D0:** These bits are fixed by hardware to zero and are not affected by any writing instruction trying to modify their value.

PAGE 0 contains the control registers of:

- the external interrupt
- the watchdog timer
- the wait logic states
- the serial peripheral interface (SPI)

2.3.5 Mode Registers

This register MODER is located in the System Register Group at the address 235.

Using this register it is possible:

- to select either internal or external System and User Stack area,
- to manage the clock frequency
- to enable the Bus request and Wait signals when interfacing external memory.

MODER R235 (EBh) System Read/Write
Mode Register

Reset value : 1110 0000

7							0
SSP	USP	DIV2	PRS2	PRS1	PRS0	BRQEN	HIMP

b7 = **SSP:** *System Stack Pointer.* This bit selects internal (in the Register File) or external (in the external Data Memory) System Stack area, logical "1" for internal, and logical "0" for external. After Reset the value of this bit is "1".

b6 = **USP:** *User Stack Pointer.* Same as bit 7 for the User Stack Pointer;

b5 = **DIV2:** *OSCIN Clock Divided by 2.* This bit controls the divide by 2 circuit which operates on the OSCIN Clock. A logical "1" value means that the OSCIN clock is internally divided by 2, and a logical "0" value means that no division of the OSCIN Clock occurs.

b4-b2 = **PRS2-PRS0:** *ST9 CPUCLK Prescaler.* These bits load the prescaling module of the internal clock (INTCLK). The prescaling value selects the frequency of the ST9 clock, which can be divided by 1 to 8. See Clock chapter for more information.

b1 = **BRQEN:** *Bus Request Enable.* This bit is a software enable of an External Bus Request. When set to "1", it enables a Bus Request on the BUSREQ pin.

b0 = **HIMP:** *High Impedance Enable.* When Port 0 and/or Port 1 are programmed as multiplexed address and Data lines to interface external Program and/or Data Memory, these lines and the Memory interface control lines ($\overline{AS}$, $\overline{DS}$, R/$\overline{W}$) can be forced into the High Impedance state by setting to "1" the HIMP bit. When this bit is reset, it has no effect on P0 and P1 lines.

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

SYSTEM REGISTERS (Continued)

2.3.6 Stack Pointers

There are two separate, double register stack pointers available (named System Stack Pointer and 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”.

For example, the register address space is selected for a stack and the corresponding stack pointer register contains 220. When a byte of data is “pushed” into the stack, the stack pointer register is decremented to 219, then the data byte is “loaded” into register 219. Conversely, if a stack pointer register contains 189 and a byte of data is “popped” out, the byte of data is then extracted from the stack and then the stack pointer register is incremented to 190.

The `push` and `pop` commands used to manage the system stack area are made applicable to the user stack by adding the suffix U, while to use a stack instruction for a word a W is added.

For example `push` inserts data into the system stack, but an added U indicates the user stack and W means a word, so the instruction `pushuw` loads a word into the bottom of the user stack.

If the User Stack Pointer register contains 223 (working in register space) the instruction `pushuw` will decrement User Stack Pointer register to 222 and then load a word into register R222 and R221.

When bytes (or words) are “popped out” the values in those registers are left unchanged until fresh data is loaded into those locations. Thus when data is “popped” out from a stack area, the stack content remains unchanged.

Note. Stacks must not be located in the pages or the system register area.

The System Stack area and The System Stack Pointer

The System Stack area is used for the storage of temporarily suspended system and/or control registers, i.e. the Flag register and the Program counter, while interrupts are being serviced. For subroutine execution only the Program Counter needs to be saved in the System stack area.

There are two situations when this occurs automatically, one being when an interrupt occurs and the other when the instruction call subroutine is used. When the system stack area is in the Register File, the stack pointer, which points to the bottom of the stack, only needs one byte for addressing, in which case the System Stack Pointer Low Register (R239) is sufficient for addressing purposes. As a result the System Stack Pointer High Register (R238) becomes redundant BUT must be considered as reserved (please refer also to “spurious” memory access section). Clearly when the stack is external a full word address is necessary and so both registers are used to point, the even register providing the MSB and the odd register providing the LSB.

The User Stack area and User Stack Pointer

The User Stack area is completely free from all interference from automatic operations and so it provides a totally user controlled stacking area, that area being in any part of the memory which is of a RAM nature, or the first 14 groups of the general Register File i.e. not in the System register or Paged group.

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

SYSTEM REGISTERS (Continued)

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. This will also benefit programmers who may locate the stacks in group D using, for example the instruction `ld R237, #223` which loads the value 223 into the User Stack Pointer Low Register. The Programmer will not need to remember to set the Register Pointer to 208 to gain access to registers in the D-group, a problem outlined in Register Pointing Techniques paragraph.
Stacks may be located anywhere in the first 14 groups of the Register File (internal stacks) or the data memory (external stacks). It is not necessary to set the data memory using the instruction `sdm` as external stack instructions automatically use the data memory.

Figure 2-8. System and/or User Stack in Register Stack Mode

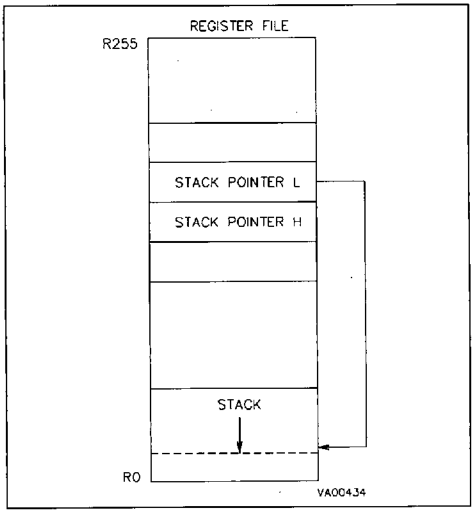
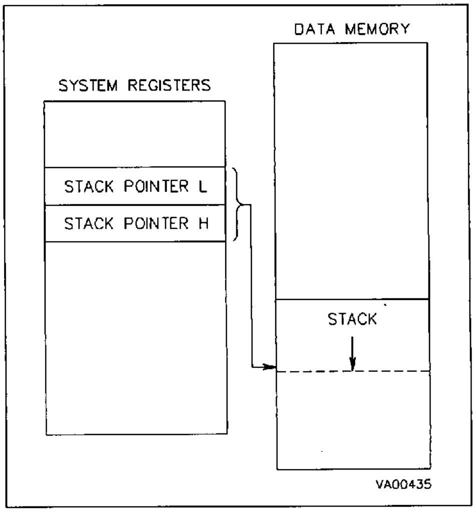


Figure 2-9. System and/or User Stack in Memory Stack Mode



USP R236 (ECh) Read/Write
User Stack Pointer High Byte
Reset value: undefined

7							0
USP15	USP14	USP13	USP12	USP11	USP10	USP9	USP8

SSP R238 (EEh) Read/Write
System Stack Pointer High Byte
Reset value: undefined

7							0
SSP15	SSP14	SSP13	SSP12	SSP11	SSP10	SSP9	SSP8

USP R237 (EDh) Read/Write
User Stack Pointer Low Byte
Reset value: undefined

7							0
USP7	USP6	USP5	USP4	USP3	USP2	USP1	USP0

SSP R239 (EFh) Read/Write
System Stack Pointer Low Byte
Reset value: undefined

7							0
SSP7	SSP6	SSP5	SSP4	SSP3	SSP2	SSP1	SSP0

3 MEMORY

3.1 INTRODUCTION

The memory of the ST9 is divided into two spaces:

- Data memory with up to 64K (65536) bytes
- Program memory with up to 64K (65536) bytes

Thus there is a total of 128K bytes of directly addressable memory space. The Bankswitch logic allows the expansion of the two 64K byte spaces into a maximum of 8M bytes in each space by a paging mechanism of the top 32K bytes of each memory space.

The Romless part uses off-chip memory, addressed using the multiplexed address and data bus (Port 0), high byte address bus (Port 1) and optionally the non-multiplexed addressed bus on Port 6 (see External Memory Interface chapter).

The Program and Data spaces are separated by the external decoding of the Program/Data select pin (P/$\overline{D}$) available as an Alternate function output, allowing the full 128Kbyte memory.

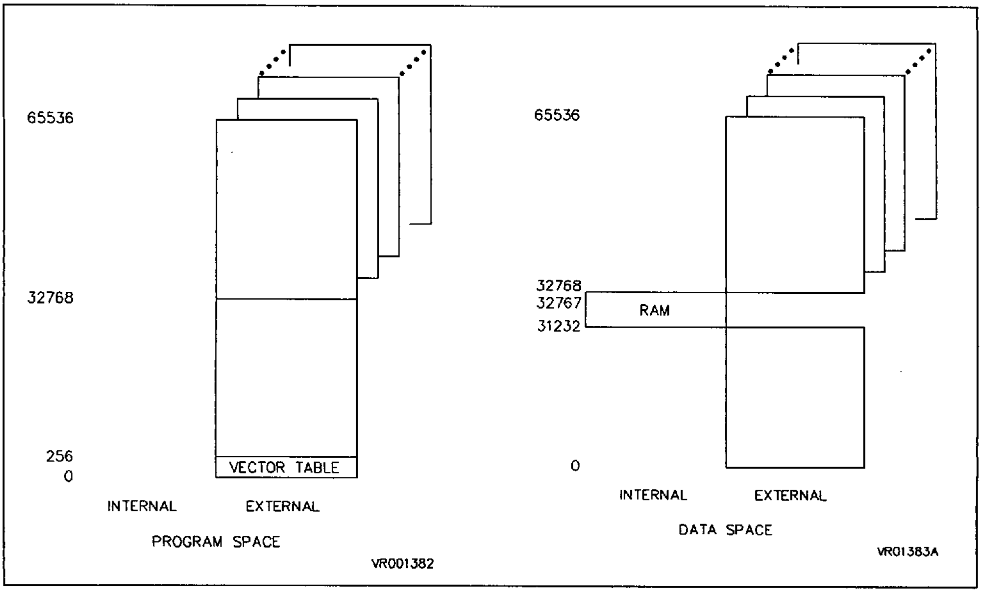
The memory spaces are selected by the execution of the `sdm` and `spm` instructions (Set Data Memory and Set Program Memory, respectively). There is no need to use either of these instructions again until the memory area required is to be changed. This requirement is not necessary in two cases: first, when operating with external stacks (the Data memory is automatically selected) and, secondly, when using the memory indirect to memory indirect post-increment addressing mode (the memory types are specified in the instructions: `ldpp, ldpd, lddp, lddd`).

Program instructions and data in the immediate addressing mode are always read from the Progam space.

Either the Data Memory or the Program Memory, can be addressed using any of the memory addressing modes.

The 16 bit memory address may be supplied directly using the absolute memory location address or indirectly using a pair of registers. In addition the address can be given by an indexed mode when a short (byte) or long (word) offset is added to an indirect base word address.

Figure 3-1. Program and Data Spaces



3.2 PROGRAM SPACE DEFINITION

The Program memory space of the ST90R91 64K bytes of directly addressable off-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.

Table 3-1. 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

Each vector is contained in two consecutive byte locations, the high order address held in the lower (even) byte, the low order address held in the upper (odd) byte, forming the address which is loaded into the Program Counter when selected by the interrupt vector provided by the interrupt source. This should point to the relevant Interrupt Service routine provided by the user for immediate response to the interrupt.

3.3 DATA SPACE DEFINITION

The external Data memory is addressed through the External Memory Interface when decoded with the P/$\overline{D}$ pin. The maximum directly addressable size is 64K bytes and the space has exactly the same addresses and addressing modes as the Program memory, the spaces being distinguished by the use of the memory setting command (`sdm`, Set Data Memory). All subsequent operand and stack memory references will access the Data Space.

When a separate Data Space is not provided, data may also be stored in external RAM or ROM memory within the Program Space. The on-chip general purpose registers may be used as RAM memory.

3.4 BANKSWITCH LOGIC

3.4.1 Introduction

The Bankswitch (BS) logic of the ST9 family is an address expander allowing the ST9 to extend its addressing range from 64K Program space + 64K Data space to up to 8 Megabytes of Program memory plus 8M bytes of Data memory. The Bankswitch memory is organised in 32K byte segments mapped into the address range 8000h to FFFFh, and a 32K byte common segment in the address range 0000h to 7FFFh.

The common segment, segment 0, allows for the direct access to interrupt service routines and page change routines, and other common subroutines, while the paged segments (1 to 256) of 32K bytes may contain additional program code, database entries, printer fonts, buffer space, or any other function requiring a large amount of memory.

The memory expansion is achieved by using the 8-bit data presented on the Bankswitch port (I/O Port 2) as address bits A23:A16, and internally using address bit A15 as the control signal to control the common segment (seg0) selection.

The data present on the Bankswitch port (i.e. Addresses A23 to A16) changes according to the following conditions:

- the address is in Program memory
- the address is in Data memory
- a DMA transaction is being made with Program memory
- a DMA transaction is being made with Data memory
- the common segment is being addressed

The Bankswitch logic includes 4 registers, *whose contents are defined by the user,* as the segment numbers to be used with these memory addressing conditions as shown in the following table:

- BS_PSR when A15 = “1” and the segment is in Program Memory
- BS_DSR when A15 = “1” and the segment is in Data Memory
- BS_PDSR when A15 = “1” and DMA is using Program Memory
- BS_DDSR when A15 = “1” and DMA is using Data Memory

also

- BS Port = FEh whenever A15 = “0”

BANKSWITCH LOGIC (Continued)

3.4.2 Bankswitch Port Programming

Note. The LST9 ST9 Incremental Linker supports the paging mechanism of the Bankswitch and is able to allocate program and data code into specific segments if required.

The Bankswitch Port functionality can be nibble (4bits or half a byte) programmed as Bankswitch outputs or I/O by latching (in the Reset cycle) the state of the input pins BSH_EN1 and BSL_EN1. They must be maintained in the desired state for a minimum of 7 crystal periods after the rising edge.

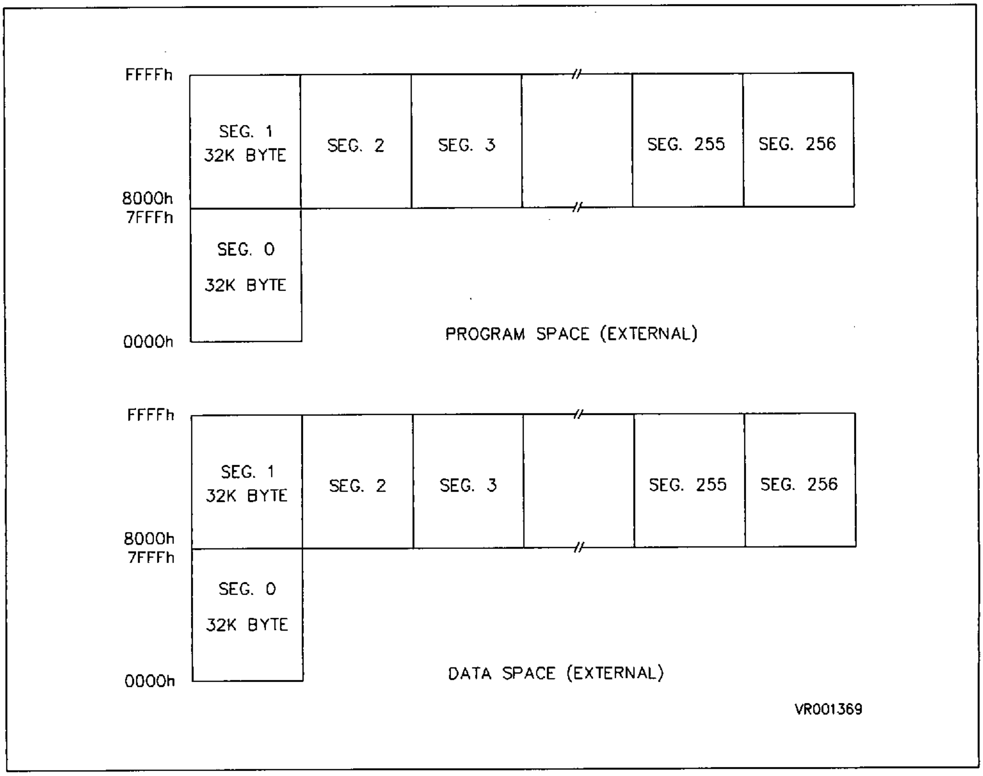
These port pins are set to INPUT CMOS status with a weak pull-up (100kΩ) during the Reset cycle so programming is made through the use of external pull-up or pull-down resistors. After the Reset cycle both port pins can be independently reprogrammed as any other I/O pin.

The programming configurations 01 and 10 both enable the lower nibble as active Bankswitch outputs.

Table 3-2. Port 2 Nibble Programming For Bank Switch and I/O

BSH_EN1	BSL_EN1	BS Port Nibble		BS Port Reset Value	
		High	Low	Prog	Data
0	0	I/O	I/O	FFh	FFh
0	I	I/O	BS	FEh	FDh
I	0	I/O	BS	FEh	FDh
I	I	BS	BS	0Eh	0Dh

Figure 3-2. Bank Switch Memory Maps



BANKSWITCH LOGIC (Continued)

3.4.3 Bankswitch Register Mapping

When the Bankswitch port is used as an I/O port, the data and control registers are mapped as I/O Port 2 in the System Page E (for the Data Register) and Page 2 (for the Control Registers). Refer to Figure 3-3 for the Bankswitch Register Mapping.

When the Bankswitch function is enabled, the four Bankswitch Registers are mapped into Page 2 (BS_PDSR and BS_DDSR) and in System Group E (BS_PSR and BS_DSR). The mapping of the Program and Data Memory Bankswitch Registers into Group E optimises the software overhead during memory segment changes, while the mapping of the DMA Bankswitch registers in an I/O page does not give a great overhead as, once initialized, the values are used automatically and do not need a constant service.

An alternative use of the Bankswitch Port is to use the bits directly as chip selects to external memory. For this reason, the Bankswitch Port has the value FEh after the Reset State and whenever address bit A15 is low (indicating the common segment). This allows the initialisation program and common subroutines to be held in the external memory mapped to bit 0 of the Bankswitch output port.

WARNING: *BS_PSR and P2C2 share the same physical register, so that when the Bankswitch port is nibble programmed, caution must be taken when writing to this register. If the lower nibble is programmed as Bankswitch and the upper as I/O, programming of the Bankswitch register with a byte value can cause the erroneous reconfiguration of the I/O nibble.*

Table 3-3. Bankswitch Register Mapping

Bankswitch Disabled

		0FFh	
		0FEh	
		0FDh	
		0FCh	
		0FBh	
		0FAh	P2C2
		0F9h	P2C1
		0F8h	P2C0
		0F7h	
		0F6h	
	0E5h	0F5h	
	0E4h	0F4h	
RESERVED	0E3h	0F3h	
P2	0E2h	0F2h	
	0E1h	0F1h	
	0E0h	0F0h	

Bankswitch Enabled

		0FFh	
		0FEh	
		0FDh	
		0FCh	
		0FBh	
		0FAh	RESERVED
		0F9h	BS_PDSR
		0F8h	BS_DDSR
		0F7h	
		0F6h	
	0E5h	0F5h	
	0E4h	0F4h	
BS_PSR	0E3h	0F3h	
BS_DSR	0E2h	0F2h	
	0E1h	0F1h	
	0E0h	0F0h	

4 INTERRUPTS

4.1 INTRODUCTION

The ST9 responds to peripheral events and external events through its Interrupt channels. When such an event occurs, if previously enabled and according to a priority mechanism, the current program execution can be suspended to allow the ST9 to execute a specific response routine. If the event generates an interrupt request, the current program status is saved after the current instruction is completed and the CPU control passes to the Interrupt Service Routine.

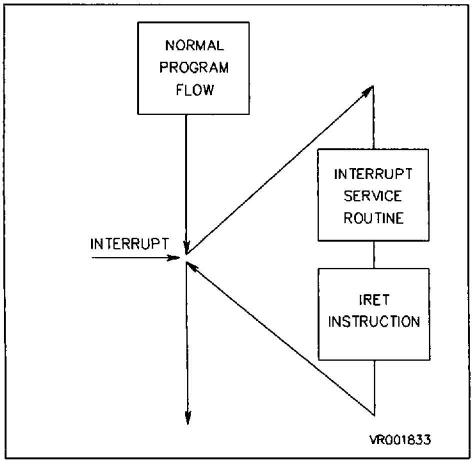
The ST9 CPU can receive requests from the following type of sources:

- 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 depending 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 pin NMI (to provide a Non-Maskable-Interrupt) or to the Timer/Watchdog. Interrupt service routines are addressed through a vector table mapped in Program Memory.

Figure 4-1. Interrupt Flow



4.2 INTERRUPT VECTORIZATION

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

When the 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 list of the addresses of the Interrupt Service Routines, is located in the first 256 locations of the Program Memory. The first 6 locations of the Program Memory are reserved for:

Address Content

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

With one Interrupt Vector register, it is possible to address more interrupt service routines; in fact, several peripherals 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 inside the vector table in the program memory, the least significant bits are controlled by the interrupt module in hardware to select the specific vector.

Note: The first 256 locations of the program memory can contain program code. Other than the Reset vector, they are not exclusively reserved to the vector table.

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 acknowledge routine must end with the* `RET` *instruction.*

INTERRUPT VECTORIZATION (Continued)

Figure 4-2. Vectors and Associated Routines

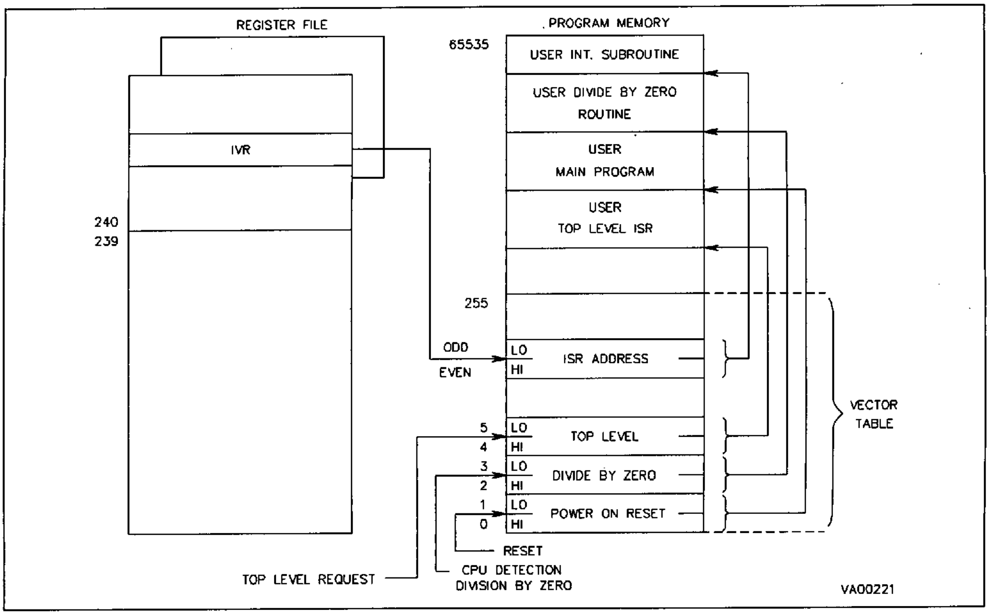
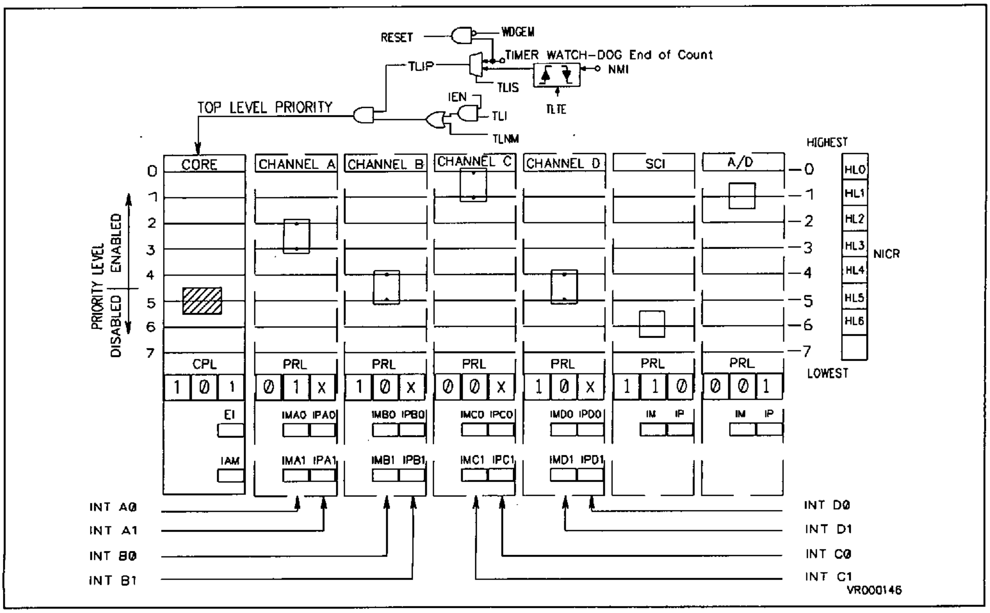


Figure 4-3. Interrupt Architecture, Example of priority Allocations



7929237 0057973 76T

4.3 INTERRUPT PRIORITY LEVEL ARCHITECTURE

The ST9 supports a fully programmable interrupt priority structure. Figure 4-4 shows a conceptual description.

9 priority levels are available to define the channel priority relationship. Each channel has a 3 bit field, PRL (Priority Level), that defines its priority level among 8 programmable levels. The ninth level (Top Level Priority) is reserved for the Timer/Watchdog or the External Pseudo Non-Maskable Interrupt. The On-chip peripheral channel and the eight external interrupt sources can be programmed within eight priority levels: level 7 has the lowest priority, level 0 has the highest priority.

If several units are located at the same priority level, an internal daisy chain, fixed for each ST9 device, defines the priority relationship within that level.

The PRL bits are used to define the priority level for interrupt requests.

Top level priority interrupt (highest) can be assigned either to the external Pseudo Non-Maskable interrupt or to the internal Timer/Watch-Dog. 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).

4.4 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.

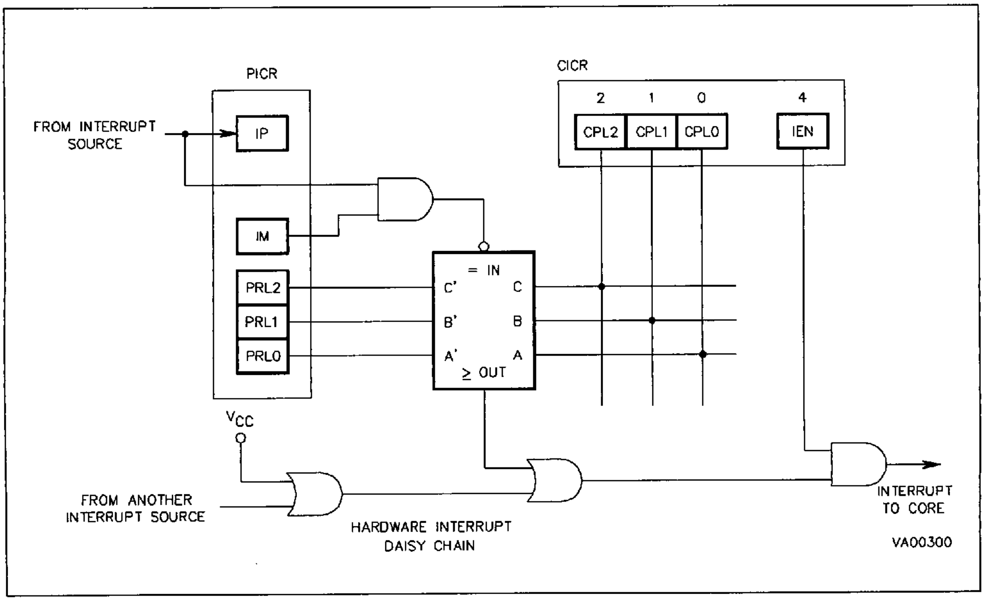
During every instruction an arbitration phase is made between every channel capable of generating an Interrupt, each priority level is compared to all the other requests. If the highest priority request is an interrupt, it must be *higher* than the CPL value in order to be acknowledged.

The priority of the Top Level Interrupt overrides every other priority.

If two or more requests occur at the same instant of time and at the same priority level, an on-chip daisy chain, specific to every ST9 version, selects the channel with the highest position in the chain. The position in the chain is shown in table 4-1.

ST9 provides two interrupt arbitration modes: Concurrent and Nested modes. The Concurrent mode is the standard interrupt arbitration mode while the Nested mode improves the effective interrupt response time when a nesting of the service routines is required according to the request priority levels.

Figure 4-4. Interrupt Logic



PRIORITY LEVEL ARBITRATION (Continued)

The control bit IAM (CICR.3) selects the Concurrent Arbitration mode (when reset to "0") or the Nested Arbitration Mode (when set to "1").

Table 4-1. Daisy Chain Priorities

Applicable for ST90R91

Highest Position	INTA0
	INTA1
	INTB0
	INTB1
	INTC0
	INTC1
	INTD0
	INTD1
	A/D
Lowest Position	TIMER0

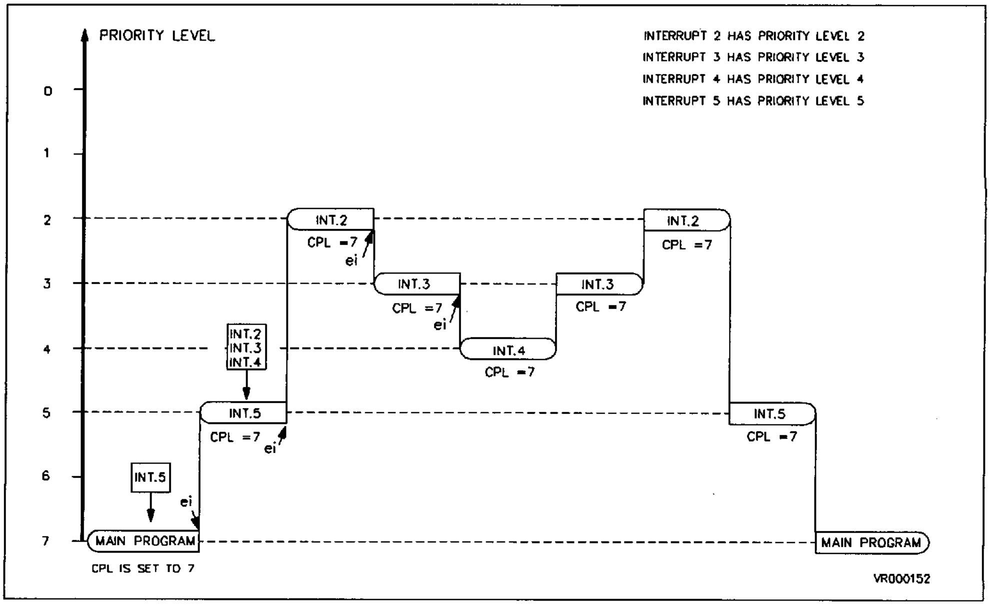
4.4.1 Concurrent Mode

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.

If the highest priority request is an interrupt request and its priority value is higher than the Current Priority Value CICR.2,1,0 (R230.2,1,0), the interrupt request will be acknowledged at the end of the current instruction. The interrupt Machine Cycle performs the following steps:

- 1. Disables all the maskable interrupt requests by clearing CICR.IEN
- 2. Pushes the PC low byte into the system stack
- 3. Pushes the PC high byte into the system stack
- 4. Pushes the Flag register into the system stack
- 5. Loads the PC with the 16-bit vector stored in the Vector Table, pointed to by the Interrupt Vector Register (IVR).

Figure 4-5. Example of a Sequence of Interrupt Requests with :
- Concurrent mode
- EI set to 1 during the interrupt routine execution



PRIORITY LEVEL ARBITRATION (Continued)

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

- 1. Pops off the Flag register from the system Stack
- 2. Pops off PC high byte from the system Stack
- 3. Pops off PC low byte from the system Stack
- 4. Enables all the un-masked Interrupts, by setting the CICR.IEN bit

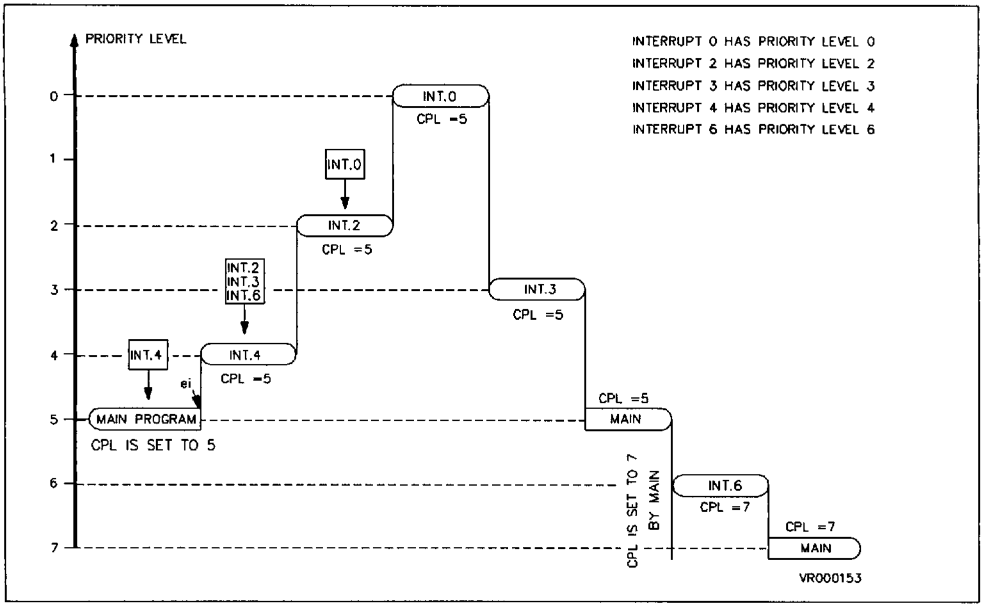
The suspended program execution is thus recovered at the interrupted instruction. All pending interrupts existing, or having occurred during the interrupt service routine execution, remain pending until the Enable Interrupt instruction (even if it is executed during the interrupt service routine).

NOTE: When Concurrent mode is selected, the source priority level is meaningful only during the arbitration phase, where it is compared to all the other priority levels and the CPL, but no trace is kept of its value during the Interrupt Service Routine. Therefore, if other requests are issued, once the global CICR.IEN is enabled again, they will be acknowledged regardless of the Interrupt Service Routine priority value; if no care is taken by the programmer, unpleasant side effects can take place.

A typical case is the following: 3 pending requests with different priority levels (ie 2,3,4) generate requests at the same time (because the associated events occurred during the same instruction). The three interrupt service routines set Interrupt Enable (IEN, CICR.4) by the `ei` instruction at the beginning of the routine to avoid a high interrupt response time to requests with a priority higher than the one under service (usually, the higher the priority, the sooner the routine must be executed). Unfortunately, what will happen in this case is that the three interrupt servicing routines will be executed exactly in the opposite order of their priority. Interrupt routine level 2 will be acknowledged first, then, when the `ei` instruction is executed, it will be interrupted by interrupt routine level 3, which itself will be interrupted by interrupt routine level 4. When interrupt routine level 4 is completed, interrupt routine level 3 will be recovered and finally, interrupt routine level 2.

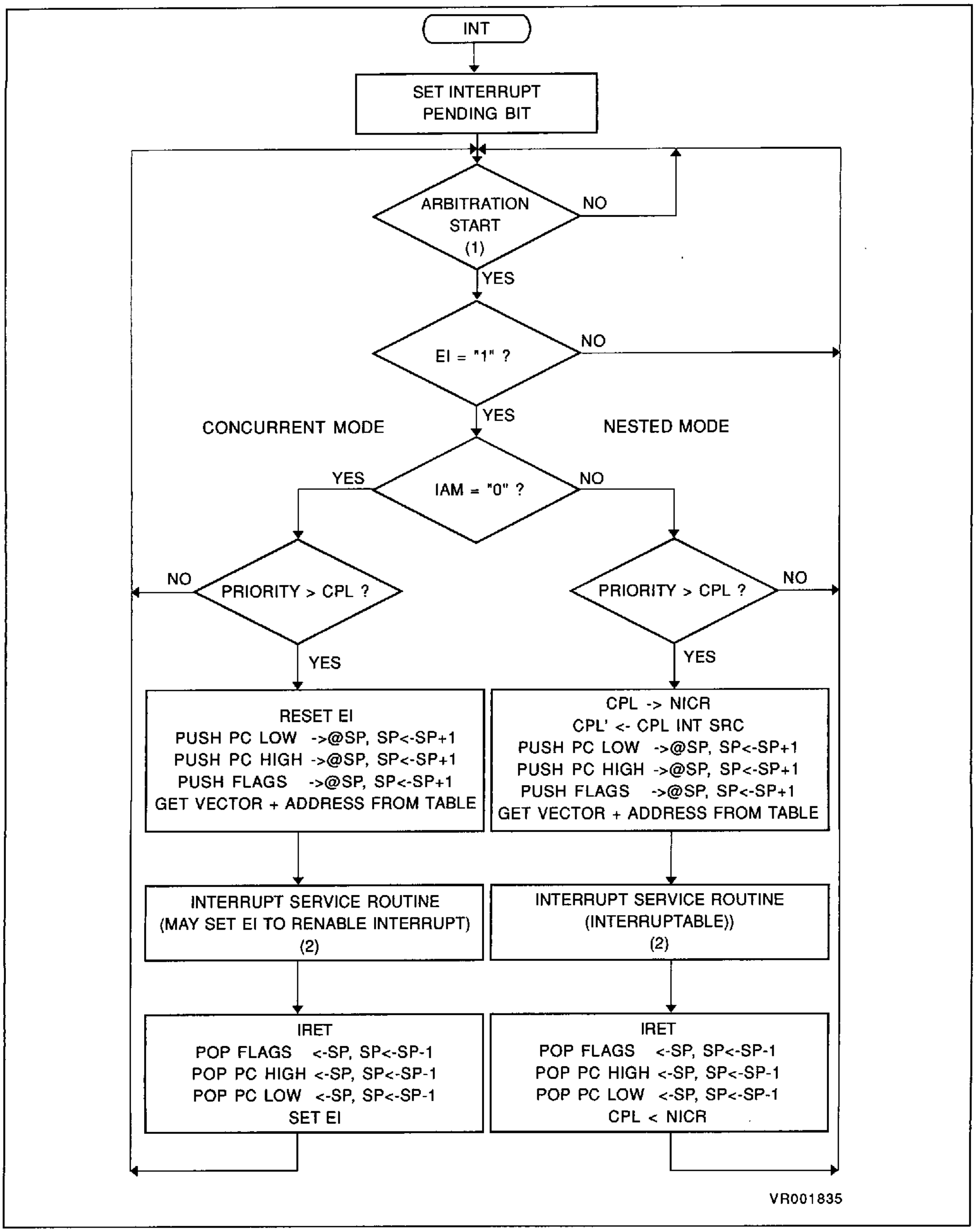
Therefore, **it is recommended, in concurrent mode, to avoid the insertion of the `ei` instruction in the interrupt subroutine**, which can trigger this LIFO (Last In, First Out) sequence of interrupt processing.

Figure 4-6. Example of a Sequence of Interrupt Requests with :
- Concurrent mode
- EI unchanged by the interrupt routines



PRIORITY LEVEL ARBITRATION (Continued)

Figure 4-7. Interrupt Mode Flow-Chart


INT
SET INTERRUPT PENDING BIT
ARBITRATION START (1)
NO
YES
EI = "1" ?
NO
YES
CONCURRENT MODE
NESTED MODE
YES
IAM = "0" ?
NO
NO
PRIORITY > CPL ?
PRIORITY > CPL ?
NO
YES
YES
RESET EI
PUSH PC LOW ->@SP, SP<-SP+1
PUSH PC HIGH ->@SP, SP<-SP+1
PUSH FLAGS ->@SP, SP<-SP+1
GET VECTOR + ADDRESS FROM TABLE
CPL -> NICR
CPL' <- CPL INT SRC
PUSH PC LOW ->@SP, SP<-SP+1
PUSH PC HIGH ->@SP, SP<-SP+1
PUSH FLAGS ->@SP, SP<-SP+1
GET VECTOR + ADDRESS FROM TABLE
INTERRUPT SERVICE ROUTINE (MAY SET EI TO RENABLE INTERRUPT) (2)
INTERRUPT SERVICE ROUTINE (INTERRUPTABLE)) (2)
IRET
POP FLAGS <-SP, SP<-SP-1
POP PC HIGH <-SP, SP<-SP-1
POP PC LOW <-SP, SP<-SP-1
SET EI
IRET
POP FLAGS <-SP, SP<-SP-1
POP PC HIGH <-SP, SP<-SP-1
POP PC LOW <-SP, SP<-SP-1
CPL < NICR
VR001835

Notes:
1. The interrupt arbitration starts 6 CPUCLK cycles before the end of execution of each instruction (5 cycles during WFI).
2. Clear interrupt pending bit

PRIORITY LEVEL ARBITRATION (Continued)

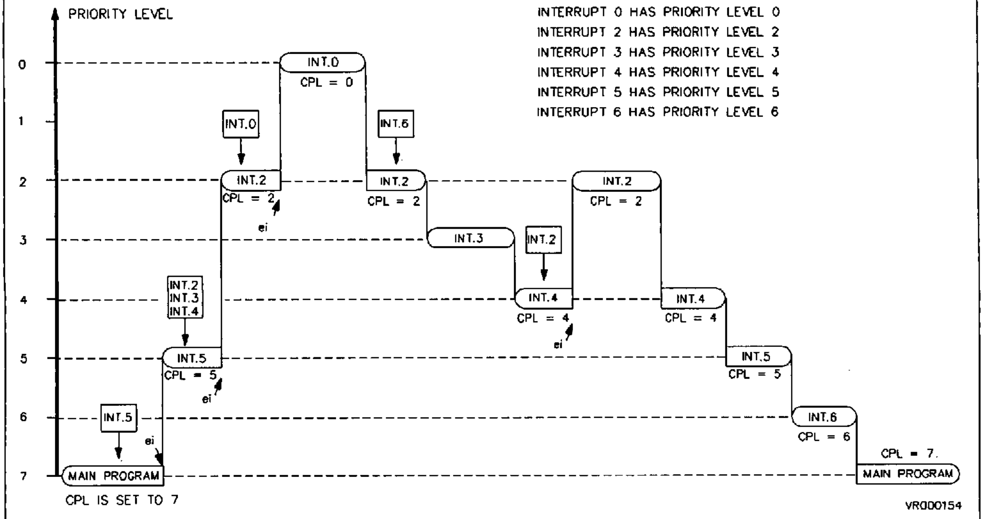
4.4.2 Nested Mode

The difference of the Nested mode to the Concurrent mode consists of the modification of the CPL value during the interrupt processing. The arbitration phase is basically identical to the concurrent Mode, however once the request is acknowledged, the current CPL value is saved in the Nested Interrupt Control Register (NICR, R247 page 0) by setting the NICR bit corresponding to the CPL value (i.e. if the CPL is 3, NICR.3 bit will be set). The CPL value is then updated with the Priority value of the request just acknowledged, in this way the next arbitration cycle will be performed against the priority level of the Service Routine in progress.

The Interrupt Machine Cycle will perform the following steps:

- Disable all the maskable interrupts by clearing IEN
- Save the CPL value into the special stack NICR to hold the priority level of the suspended routine
- Store in CPL the priority level of the acknowledged routine, so that the next request priority will be compared with the one of the routine under service
- Push the PC-low byte into the System Stack
- Push the PC-high byte into the System Stack
- Push the Flag Register into the System Stack

Figure 4-8. Example of a Sequence of Interrupt Requests with :
- Nested mode
- EI set to 1 during the interrupt routine execution



PRIORITY LEVEL ARBITRATION (Continued)

- Load the PC with the vector pointed by IVR.

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

- 1. Pop off the Flag Register from the System Stack
- 2. Pop off the PC-high byte from the System Stack
- 3. Pop off the PC-low byte from the System Stack
- 4. Enable all the unmasked interrupts by setting the IEN bit
- 5. Recover the interrupted routine priority level by popping the value from the special register (NICR) and by copying it into CPL.

The suspended execution is thus recovered at the interrupted instruction.

REMARKS

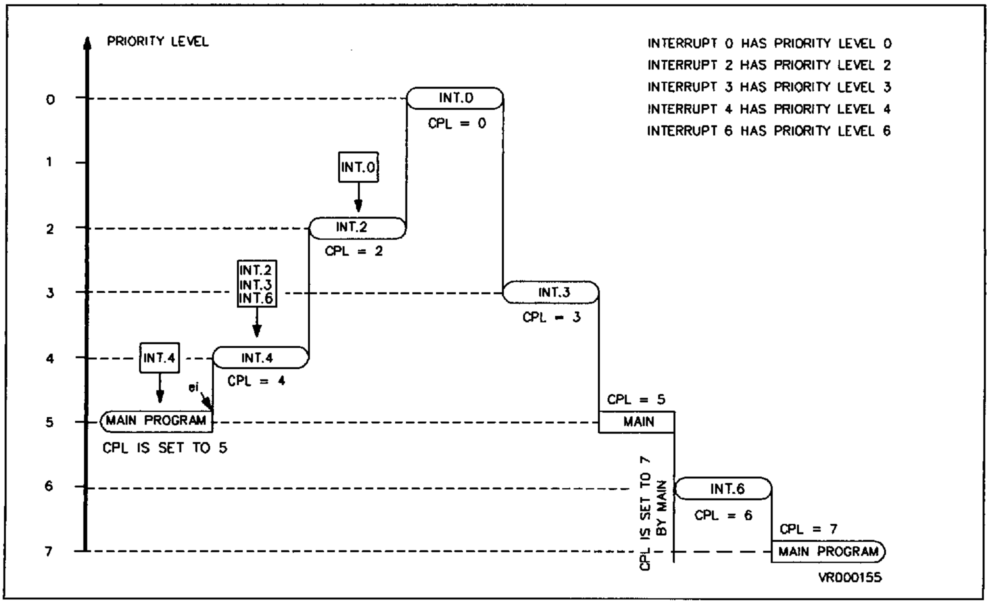
1) Dynamic priority level modification: the main program and routines can be specifically prioritized. Since CPL is represented by 3 bits in a read/write register, it is possible to modify dynamically the current priority value during the program execution. This means that a critical section can have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

2) Maximum number of nestings: 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) Priority level 7: Interrupt requests at level 7 cannot be acknowledged as their priority cannot be higher than the CPL value. This can be of use in a fully polled interrupt environment.

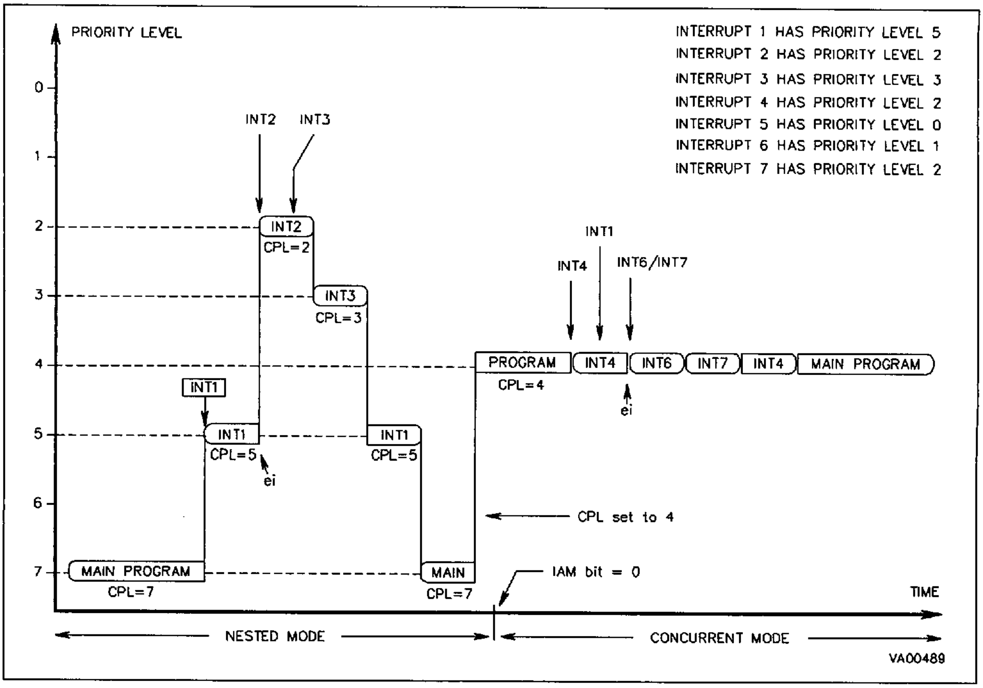
A nested/concurrent mode sequence is given on Figure 4-10. This example clearly shows that Nested and Concurrent modes are defined by the user. Note that here the Y axis is referenced by CPL, instead of the source priority level, and *that Interrupt 1 stays pending*, having a priority level lower than CPL.

Figure 4-9. Example of a Sequence of Interrupt Requests with :
- Nested mode
- EI unchanged by the interrupt routiness



PRIORITY LEVEL ARBITRATION (Continued)

Figure 4-10. Example of a Nested and Concurrent Mode Sequence


PRIORITY LEVEL
INTERRUPT 1 HAS PRIORITY LEVEL 5
INTERRUPT 2 HAS PRIORITY LEVEL 2
INTERRUPT 3 HAS PRIORITY LEVEL 3
INTERRUPT 4 HAS PRIORITY LEVEL 2
INTERRUPT 5 HAS PRIORITY LEVEL 0
INTERRUPT 6 HAS PRIORITY LEVEL 1
INTERRUPT 7 HAS PRIORITY LEVEL 2
0
1
2
3
4
5
6
7
INT2
INT3
INT2
CPL=2
INT3
CPL=3
INT1
INT1
CPL=5
ei
INT1
CPL=5
MAIN PROGRAM
CPL=7
MAIN
CPL=7
PROGRAM
CPL=4
INT1
INT4
INT6/INT7
INT4
INT6
INT7
INT4
MAIN PROGRAM
ei
CPL set to 4
IAM bit = 0
TIME
NESTED MODE
CONCURRENT MODE
VA00489

4.5 EXTERNAL INTERRUPTS

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

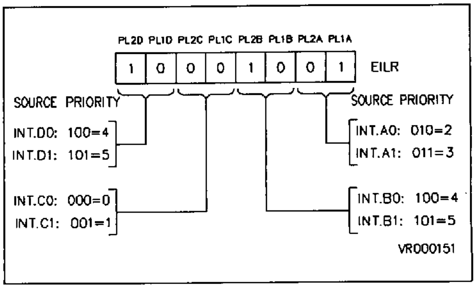
Table 4-2. External Interrupt Channel Grouping

External Interrupt	Channel
INT7 INT6	INTD1 INTD0
INT5 INT4	INTC1 INTC0
INT3 INT2	INTB1 INTB0
INT1 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 4-12.

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 4-11 shows an example of priority levels.

Figure 4-11. Priority Level Examples



- 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 B0 can be selected between the external pin INT2 (when (SPEN,BMS)=(0,0)) or the on-chip SPI peripheral.

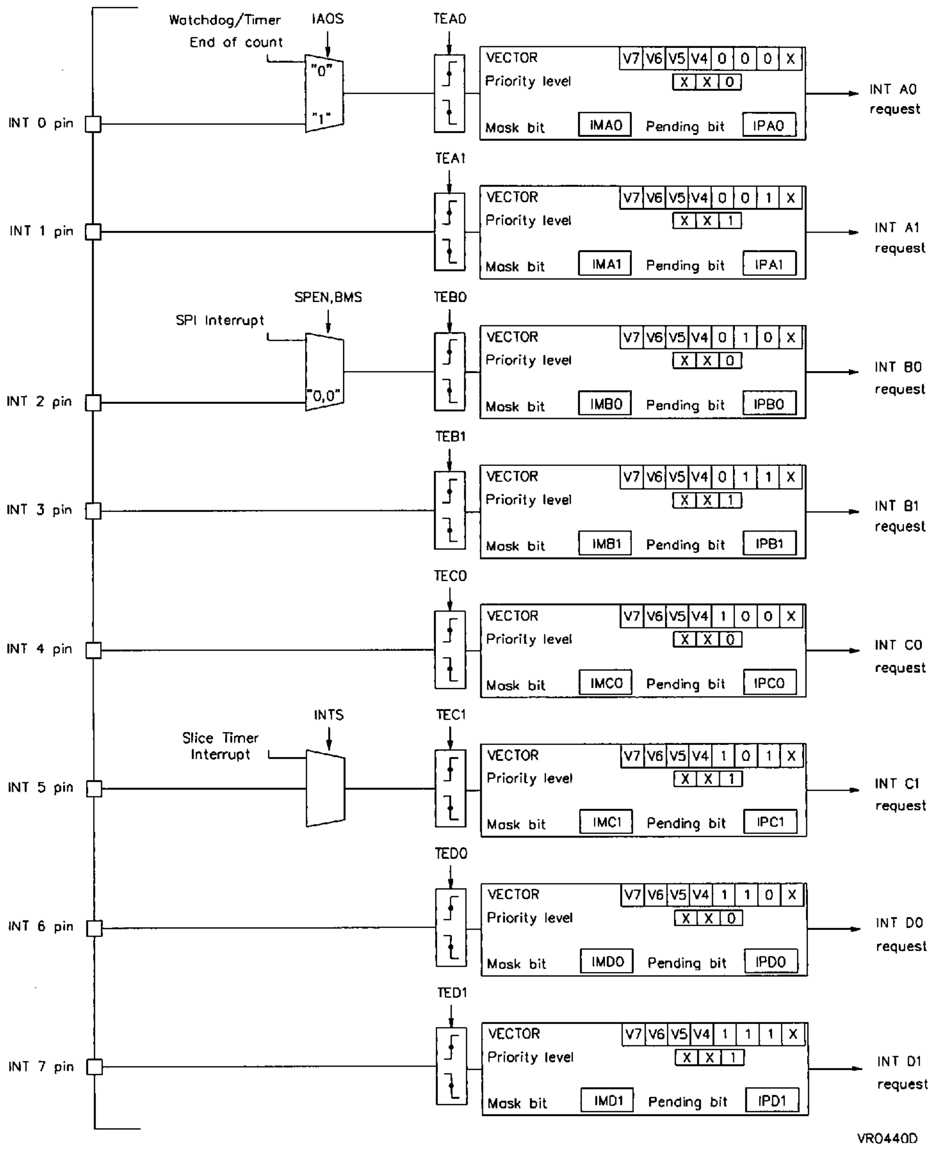
All other interrupt channels have an input pin as source, however, the input line may be multiplexed with an on-chip peripheral I/O or connected to an input pin that performs also other function (as in the case of the handshake feature).

Table 4-3. Internal/External Interrupt Source

Channel	Internal Interrupt Source	External Interrupt Source
INTA0	Timer/Watchdog	INT0
INTB0	SPI Interrupt	INT2
INTC1	Slice Timer	INT5

EXTERNAL INTERRUPTS (Continued)

Figure 4-12. External Interrupts Control Bits and Vectors


Watchdog/Timer End of count
IAOS
"0"
"1"
TEA0
INT 0 pin
VECTOR V7 V6 V5 V4 0 0 0 X
Priority level X X 0
Mask bit IMA0 Pending bit IPA0
INT A0 request
TEA1
INT 1 pin
VECTOR V7 V6 V5 V4 0 0 1 X
Priority level X X 1
Mask bit IMA1 Pending bit IPA1
INT A1 request
SPEN,BMS
SPI Interrupt
"0,0"
TEB0
INT 2 pin
VECTOR V7 V6 V5 V4 0 1 0 X
Priority level X X 0
Mask bit IMB0 Pending bit IPB0
INT B0 request
TEB1
INT 3 pin
VECTOR V7 V6 V5 V4 0 1 1 X
Priority level X X 1
Mask bit IMB1 Pending bit IPB1
INT B1 request
TEC0
INT 4 pin
VECTOR V7 V6 V5 V4 1 0 0 X
Priority level X X 0
Mask bit IMC0 Pending bit IPC0
INT C0 request
INTS
Slice Timer Interrupt
TEC1
INT 5 pin
VECTOR V7 V6 V5 V4 1 0 1 X
Priority level X X 1
Mask bit IMC1 Pending bit IPC1
INT C1 request
TED0
INT 6 pin
VECTOR V7 V6 V5 V4 1 1 0 X
Priority level X X 0
Mask bit IMD0 Pending bit IPD0
INT D0 request
TED1
INT 7 pin
VECTOR V7 V6 V5 V4 1 1 1 X
Priority level X X 1
Mask bit IMD1 Pending bit IPD1
INT D1 request
VR04400

4.6 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 cleared) 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 bit CICR.TLI (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 program. Only the processor RESET cycle can clear this bit.

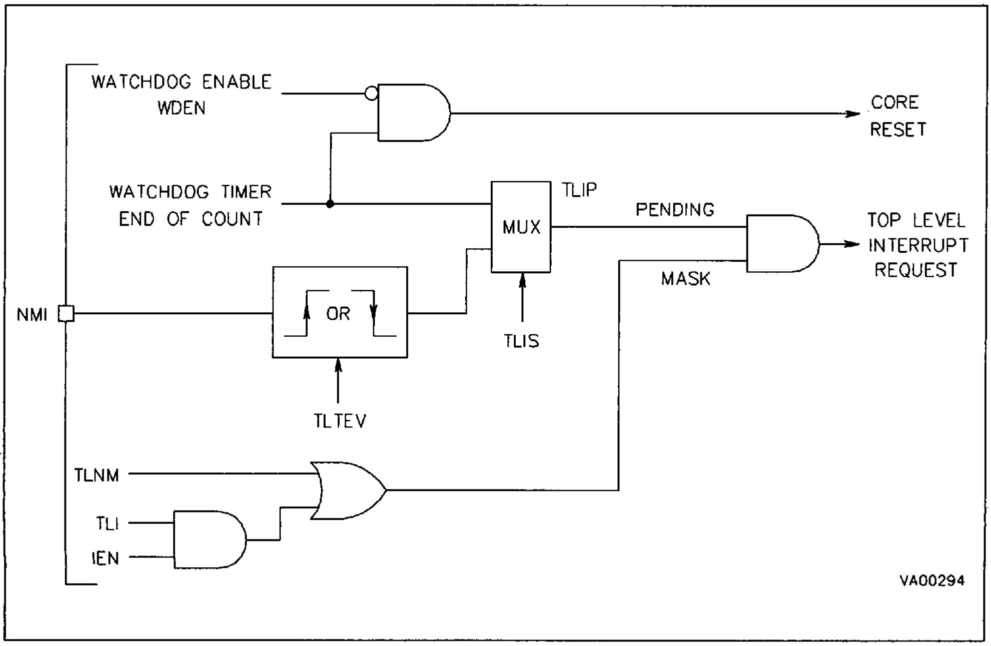
The Top Level Interrupt Service Routine cannot be interrupted by any other interrupt request, in any arbitration mode, even by another Top Level Interrupt request.

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.*

4.7 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.

Figure 4-13. Top Level Interrupt Structure

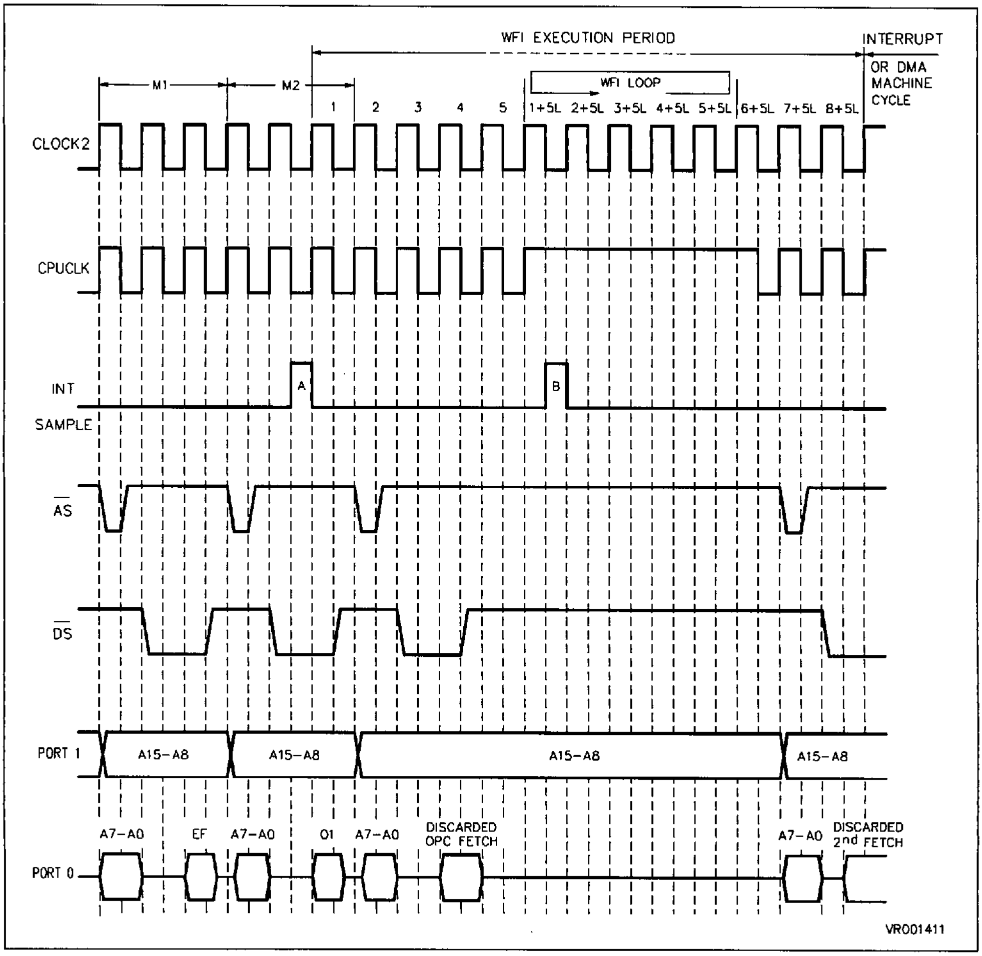


ON-CHIP PERIPHERAL INTERRUPTS (Continued)

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 source 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 4-14. Wait For Interrupt Timing



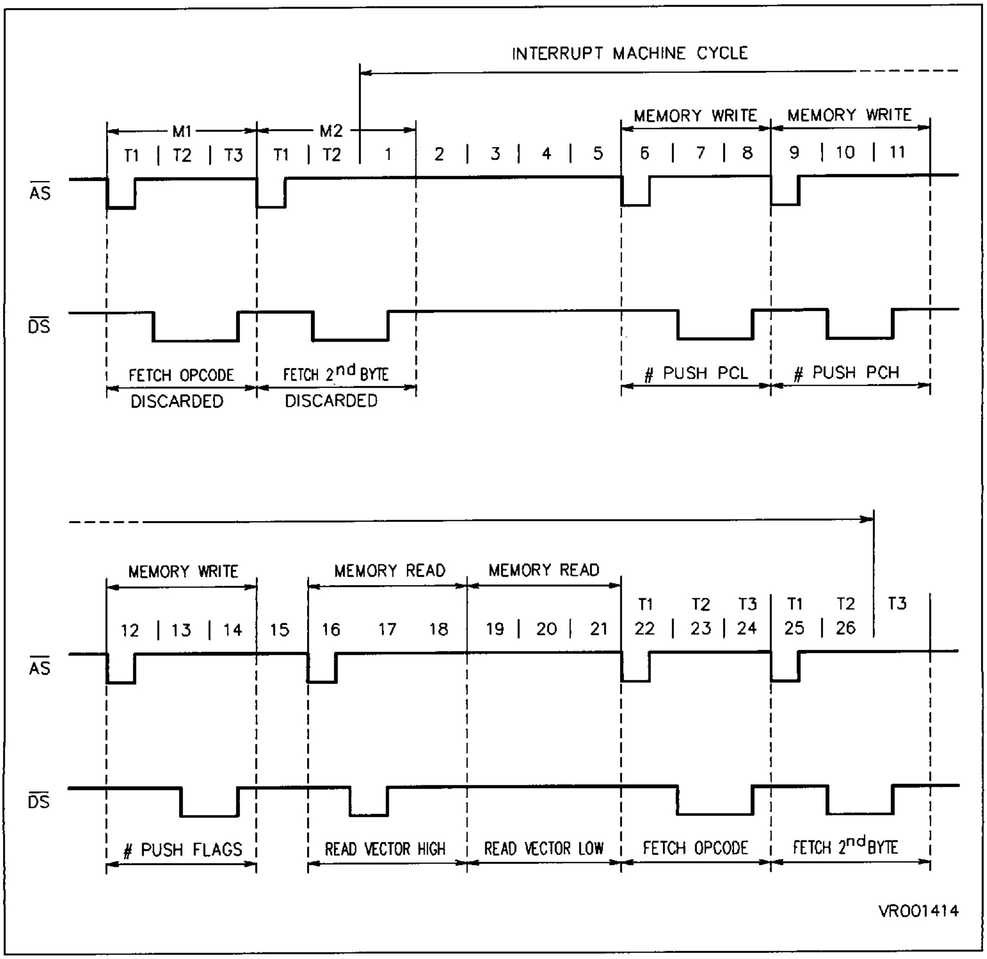
4.8 WAIT FOR INTERRUPT INSTRUCTION

The Wait For Interrupt instruction suspends program execution until an interrupt request is acknowledged. During the WFI instruction, the CPUCLK is halted while INTCLK keeps running. Under this state, the power consumption of the processor is lowered by the CORE power consumption value.

4.9 INTERRUPT RESPONSE TIME

Interrupt requests are sampled 6 CPUCLK cycles before the end of the instruction. If Wait For Interrupt is in progress, requests are sampled every 5 CPUCLK cycles. If the interrupt request comes from an external pin, the programmed event has to be set a minimum of one CPUCLK cycle before the sampling time.

Figure 4-15. Interrupt Acknowledge Timing



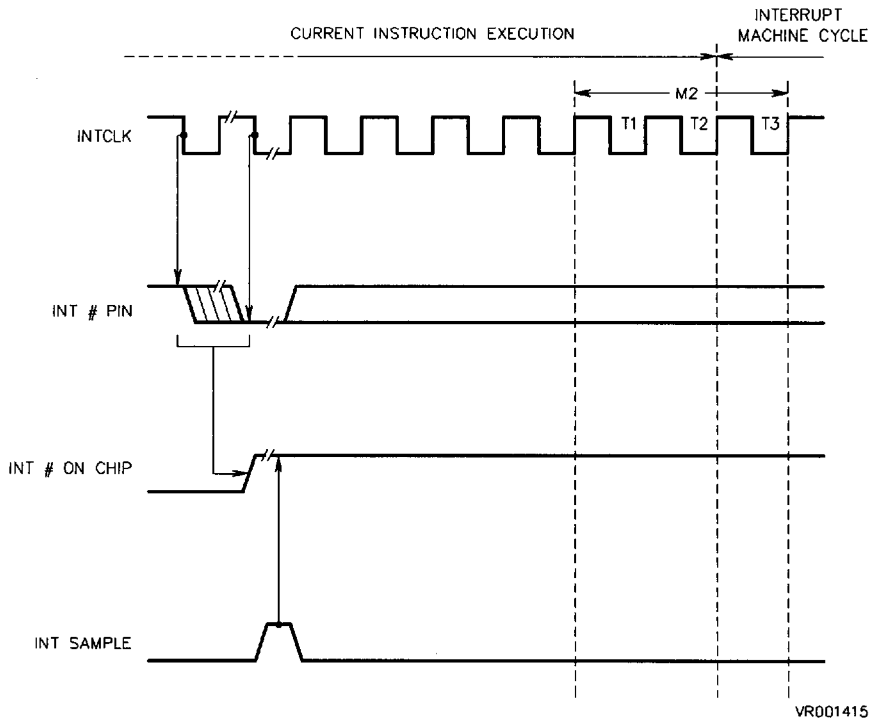
INTERRUPT RESPONSE TIME (Continued)

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

An interrupt machine cycle takes 26 internal clock cycles (CPUCLK), with some exceptions as follows:

- 28 internal clock cycles (CPUCLK), if a Wait For Interrupt is in progress
- 32 internal clock cycles (CPUCLK), if the acknowledge cycle follows a DMA transfer with Register File

Figure 4-16. External Interrupt Response Time



4.10 INTERRUPT REGISTERS

CICR R230 (E6h) System Read/Write
Central Interrupt Control Register

Reset value: 1000 0111 (87h)

7							0
GCEN	TLIP	TLI	IEN	IAM	CPL2	CPL1	CPL0

b7 = **GCEN:** *Global Counter Enable bit.* When set the 16 bit MultiFunction Timers are enabled (see Timer Control Register in MULTI FUNCTION TIMER chapter)

b6 = **TLIP:** *Top Level Interrupt Pending bit.* Set by hardware when the Trigger Event occurs. Cleared by hardware when the Top Level Interrupt is acknowledged.

b5 = **TLI:** *Top Level Interrupt bit.* If TLI ="1", and IEN is set, a Top Level Interrupt request is generated as TLIP is set. If TLI = "0", a request is generated only if TLNM is set.

b4 = **IEN:** *Interrupt Enable.* If IEN = "0", no maskable Interrupt requests are generated. This bit is cleared by the interrupt machine cycle and it is set by the IRET instruction of maskable routines.

b3 = **IAM:** *Interrupt Arbitration Mode.* If IAM = "0", Concurrent Arbitration Mode is selected; If IAM = "1" Nested Mode is selected.

b2-b0 = **CPL2, CPL1, CPL0:** *Current Priority Level.* Defines the Current Priority Level under service. CPL=0 is the highest priority. CPL=7 is the lowest priority. This bits may be modified directly by the interrupt hardware when the Nested Interrupt Mode is used.

EITR R242 (F2h) Page 0 Read/Write
External Interrupt Trigger Event Register

Reset value: 0000 0000 (00h)

7							0
TED1	TED0	TEC1	TEC0	TEB1	TEB0	TEA1	TEA0

If TExy bit is set, the pending bit will be set upon the rising edge of the input signal.

If TExy is cleared, the pending bit will be set upon the falling edge of the input signal.

All bits are set/reset only by software.

b7 = **TED1:** *Trigger Event of Interrupt Channel D1*

b6 = **TED0:** *Trigger Event of Interrupt Channel D0*

b5 = **TEC1:** *Trigger Event of Interrupt Channel C1*

b4 = **TEC0:** *Trigger Event of Interrupt Channel C0*

b3 = **TEB1:** *Trigger Event of Interrupt Channel B1*

b2 = **TEB0:** *Trigger Event of Interrupt Channel B0*

b1 = **TEA1:** *Trigger Event of Interrupt Channel A1*

b0 = **TEA0:** *Trigger Event of Interrupt Channel A0*

EIPR R243 (F3h) Page 0 Read/Write
External Interrupt Pending Register

Reset value: 0000 0000 (00h)

7							0
IPD1	IPD0	IPC1	IPC0	IPB1	IPB0	IPA1	IPA0

b7 = **IPD1:** *Interrupt Pending bit Channel D1*

b6 = **IPD0:** *Interrupt Pending bit Channel D0*

b5 = **IPC1:** *Interrupt Pending bit Channel C1*

b4 = **IPC0:** *Interrupt Pending bit Channel C0*

b3 = **IPB1:** *Interrupt Pending bit Channel B1*

b2 = **IPB0:** *Interrupt Pending bit Channel B0*

b1 = **IPA1:** *Interrupt Pending bit Channel A1*

b0 = **IPA0:** *Interrupt Pending bit Channel A0*

IP bits are hardware set upon the occurence of the trigger event and are reset by the interrupt acknowledge machine cycle.

Note. IP bits may be set by the programmer to implement a software interrupt.

INTERRUPT REGISTERS (Continued)

EIMR R244 (F4h) Page 0 Read/Write
External Interrupt Mask-bit Register

Reset value: 0000 0000 (00h)

7							0
IMD1	IMD0	IMC1	IMC0	IMB1	IMB0	IMA1	IMA0

EIMR bits are set/reset by software

When the IM bit is set (and the global IEN is enabled), an interrupt request is generated if the corresponding IP bit is set. When IM = "0", no request will be generated.

- IMxy = "1": an interrupt request can be acknowledged (depending on IEN)
- IMxy = "0": an interrupt request is masked.

b7 = **IMD1:** *Interrupt Mask of Interrupt Channel D1*

b6 = **IMD0:** *Interrupt Mask of Interrupt Channel D0*

b5 = **IMC1:** *Interrupt Mask of Interrupt Channel C1*

b4 = **IMC0:** *Interrupt Mask of Interrupt Channel C0*

b3 = **IMB1:** *Interrupt Mask of Interrupt Channel B1*

b2 = **IMB0:** *Interrupt Mask of Interrupt Channel B0*

b1 = **IMA1:** *Interrupt Mask of Interrupt Channel A1*

b0 = **IMA0:** *Interrupt Mask of Interrupt Channel A0*

EIPLR R245 (F5h) Page 0 Read/Write
External Interrupt Priority Level Register

Reset value: 1111 1111 (FFh)

7							0
PL2D	PL1D	PL2C	PL1C	PL2B	PL1B	PL2A	PL1A

EIPLR bits are set/reset by software

b7-b6 = **PL1D, PL2D:** *Priority level for the Group INTD0, INTD1*

b5-b4 = **PL1C, PL2C:** *Priority level for the Group INTC0, INTC1*

b3-b2 = **PL1B, PL2B:** *Priority level for the Group INTB0, INTB1*

b1-b0 = **PL1A, PL2A:** *Priority level for the Group INTA0, INTA1*

EIVR R246 (F6h) Page 0 Read/Write
External Interrupt Vector Register

Reset value: xxxx 0110 (X6h)

7							0
V7	V6	V5	V4	TLTEV	TLIS	IAOS	EWEN

b7-b4 = **V7 to V4:** *Most significant nibble of External Interrupt Vector.* Not initialized by reset.

b3 = **TLTEV:** *Top Level Trigger Event bit* When set, the Top Level event is triggered on rising edge of NMI input pin. Triggering on the falling edge of the NMI input pin is activated when this bit is "0" (reset value)

b2 = **TLIS:** *Top Level Input Selection bit*This bit selects the source of the Top Level Interrupt between the external NMI pin (when "1", the reset value) and the Timer/Watchdog End of Count (when "0").

b1 = **IA0S:** *Interrupt A0 Selection bit*When set, the External Interrupt pin is selected as the External Interrupt Channel A0 source. When reset the source is the Timer/Watchdog End of Count interrupt.

b0 = **EWEN:** *External Wait Enable bit* When set, this bit enables the WAIT input pin to stretch the external memory access cycle. For more details of the WAIT mode, the reader should refer to the Clock and Wait chapter or External memory Interface chapter.

NICR R247 (F7h) Page 0 Read/Write
Nested Interrupt Control Register

Reset value: 0000 0000 (00h)

7							0
TLNM	HL6	HL5	HL4	HL3	HL2	HL1	HL0

b7 = **TLNM:** *Top Level Not Maskable.*

If TLNM = "1", a top level request is generated as TLIP is set. Once TLNM is set, it can be cleared only with an hardware reset

bx = **HLx:** *Hold Level x* These bits are set to "1" 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). It is cleared by the `iret` execution when the routine at level x is recovered.

5 ON-CHIP DMA

5.1 INTRODUCTION

The ST9 includes on chip Direct Memory Access (DMA) channels to provide high-speed data transactions between peripherals and memory or the Register File. Multi-channel DMA is fully supported by peripherals with their own controller and DMA channel(s). Each DMA channel transfers data to or from contiguous locations of the Register File, Program Memory or Data Memory. The maximum number of transactions that each DMA channel can perform is 222 if the Register File is selected, or 65536 if Program or Data Memory is selected.

The DMA controller in the Peripheral uses an indirect addressing mechanism to DMA Pointers and Counter Registers stored in the Register File. This is the reason that the maximum number of transactions for Register File is 222, as two Registers are allocated for the Pointer and Counter. Register pairs are used for memory pointers and counters to offer the full 65536 byte and count capability.

The ST9 supports a fully programmable DMA priority structure included within the interrupt structure.

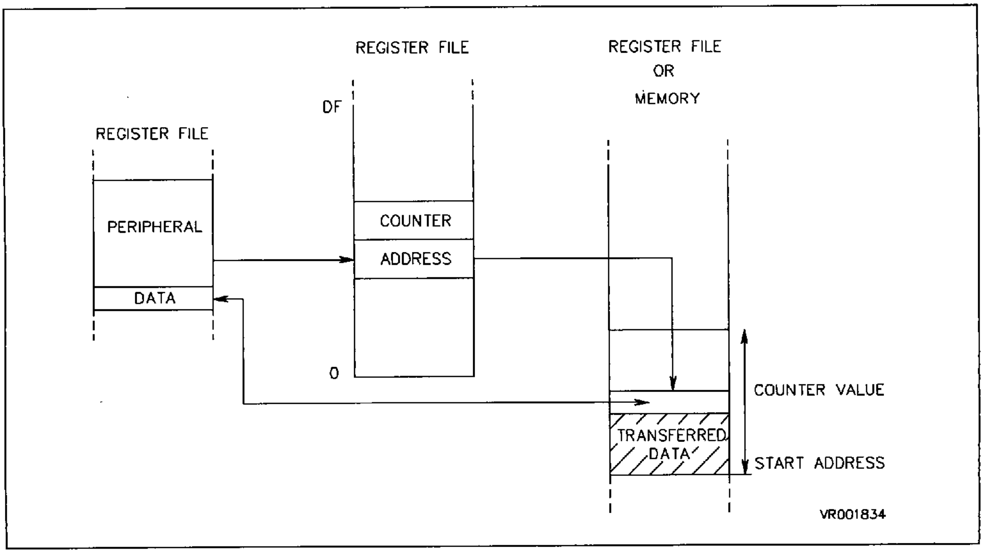
5.2 DMA PRIORITY LEVEL ARCHITECTURE

The 8 priority levels used for interrupts are also used to prioritize the DMA requests, which are arbitrated in the same arbitration phase as interrupt requests. If the event occurrence requires a DMA transaction, this will take place at the end of the current instruction execution. When an interrupt and a DMA request occur simultaneously, on the same priority level, the DMA request is serviced before the interrupt.

Note however that an interrupt priority request must be *higher* than the CPL value in order to be acknowledged. For a DMA transaction request, it must be *equal to* or *higher than* the CPL value in order to be executed.
Thus only DMA transaction requests can be acknowledged when CPL = 7.

DMA requests do not modify the CPL value as the DMA transaction is non-interruptable.

Figure 5-1. DMA Overview



DMA TRANSACTIONS (Continued)

Figure 5-2. DMA Between Registers and peripherals

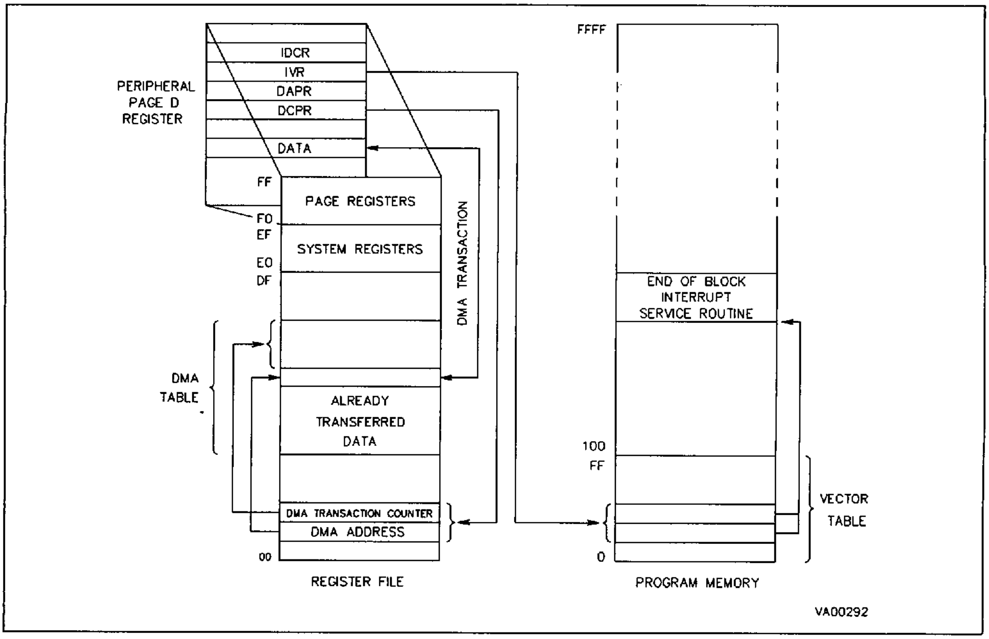
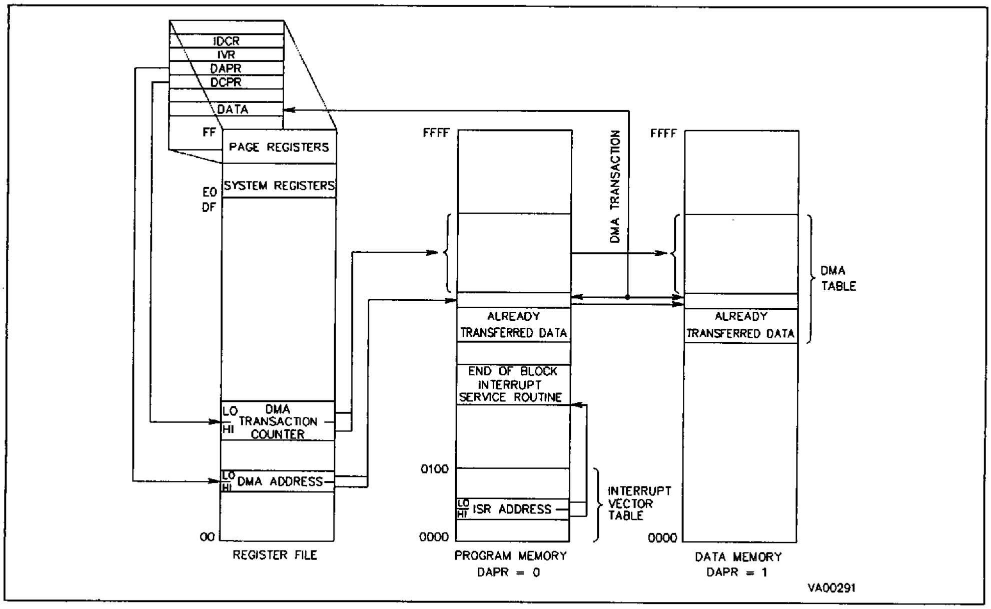


Figure 5-3. DMA Between Memory and peripherals



5.3 DMA TRANSACTIONS

The purpose of on-chip DMA channel is to transfer a block of data from/to the peripheral to/from Register File or Memory. Each DMA transfer consists of three operations:

- A load from/to the peripheral data register to a location of Register File (or Memory) addressed through the DMA Address Register (or Register pair)
- A post-increment of the DMA Address Register (or Register pair)
- A post-decrement of the DMA transaction counter, which contains the number of transactions that have still to be performed.

If the DMA transaction is made between **the peripheral and the Register File** (Figure 5-2), one register is required to hold the DMA Address and one to hold the DMA transaction counter. These two registers must be located in the Register File: the DMA Address Register in the even addressed register, the DMA transaction Counter in the following register (odd address). They are pointed to by the DMA Transaction Counter Pointer Register (DCPR) located in the page registers of the peripheral. In order to select the DMA transaction with the Register File, the control bit DCPR.RM (bit 0 of DCPR) must be set.

The Transaction Counter Register must be initialized with the number of DMA transfers to perform and will be decremented after each transaction. The DMA Address Register must be initialized with the starting address of the DMA table in the Register File, and is increased after each transaction. These two registers must be located between addresses 00h and DFh of the Register File.

If the transaction is made between **the peripheral and Memory Space (Program or Data Memory)**, a register pair (16 bits) is required for the DMA Address and for the DMA transaction Counter (Figure 5-3. Thus, two register pairs must be located in the Register File. The DMA Transaction Counter is pointed to by the DMA Transaction Counter Pointer Register (DCPR), the DMA Address is pointed to by the DMA Address Pointer Register (DAPR), both DCPR and DAPR are located in the page registers of the peripheral. When selecting the DMA transaction with memory, the control bit DCPR.RM (bit 0 of DCPR) must be cleared to "0".

To select between Program or Data Memory, the control bit DAPR.DP (bit 0 of DAPR) must be cleared or set respectively.

Once the DMA table is completed (the transaction counter reaches 0 value), an Interrupt request to the CPU is generated.

The DMA transaction Counter must be initialized with the number of transactions to perform and will be decremented after each transaction. The DMA Address must be initialized with the starting address of the DMA table and is increased after each transaction. These two register pairs must be located betweeen addresses 00h and DFh of the Register File.

Once a DMA channel is initialized, a transfer can start. The direction of the transfer (data from/to peripheral to/from memory or Register File) is automatically defined by the type of peripheral and programming mode.

When the Request Pending bit is set by a hardware event (or by software), and the DMA Mask bit is set, a DMA request is generated. If the Priority Level of the DMA source is higher than or equal to the Priority Level under service (CPL) the DMA transfer is executed at the end of the current instruction. DMA transfer reads/writes data from/to the location pointed by the DMA Address Register, increments the DMA Address register and decrements the Transaction Counter Register. When the content of the Transaction Counter is decremented to zero, the DMA Mask bit (DM) is cleared and an interrupt request is generated according to the Interrupt Mask bit (End of Block interrupt). This End-of-Block interrupt request is taken into account depending on the PRL value.

WARNING*. DMA requests are not acknowledged if the top level interrupt service is in progress.*

5.4 DMA CYCLE TIME

DMA and Interrupt requests are sampled 6 INTCLK cycles before the end of the instruction. If Wait For Interrupt is in progress, requests are sampled every 5 INTCLK cycles. DMA transactions are executed if their priority allows it.

A DMA transfer with the Register file takes 8 CPUCLK cycles, except when the Wait For Interrupt is in progress (10 CPUCLK cycles).

A DMA transfer with the memory takes 16 CPUCLK cycles except when the Wait For Interrupt is in progress (18 CPUCLK cycles).

Figure 5-4. DMA Transaction to Memory

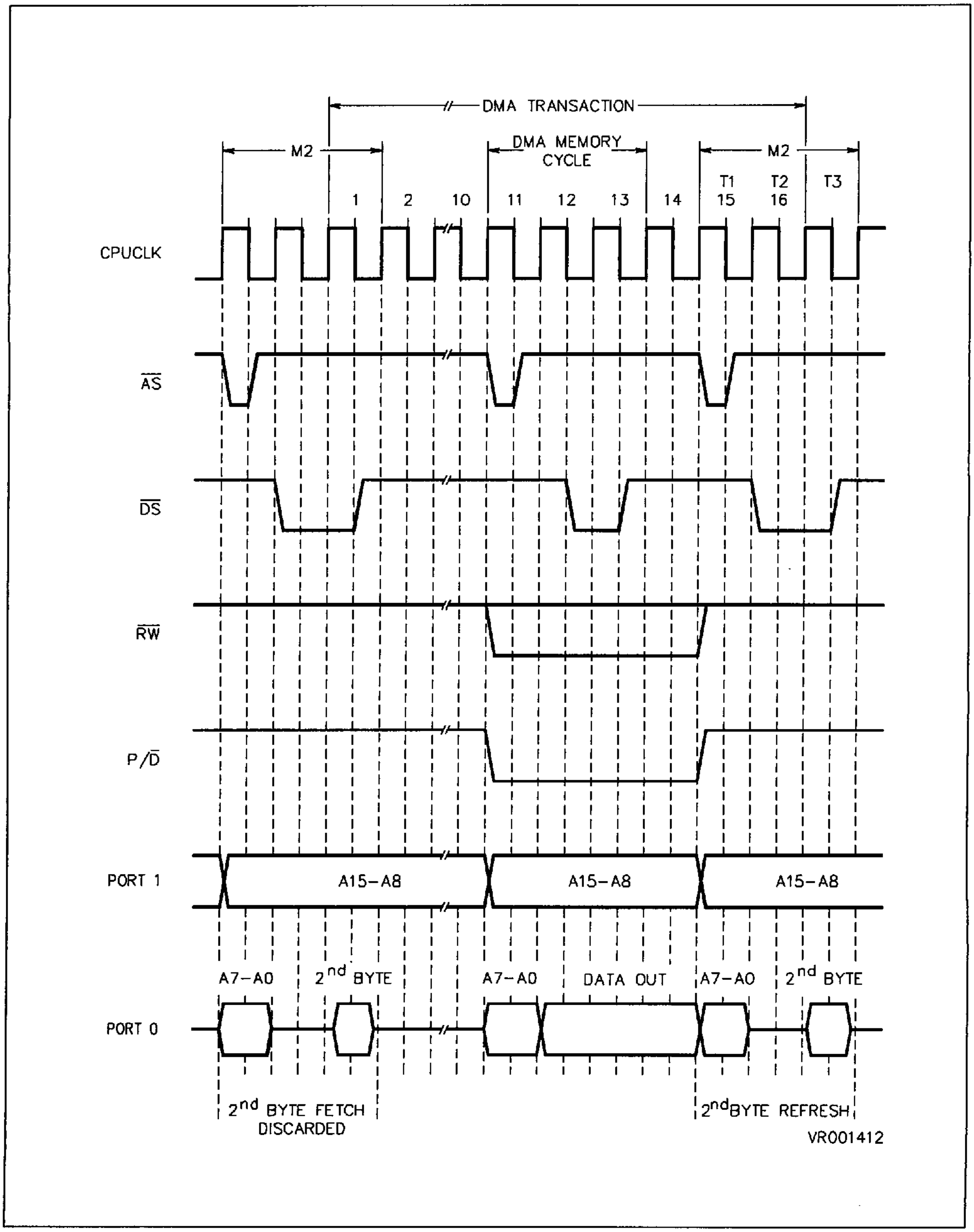
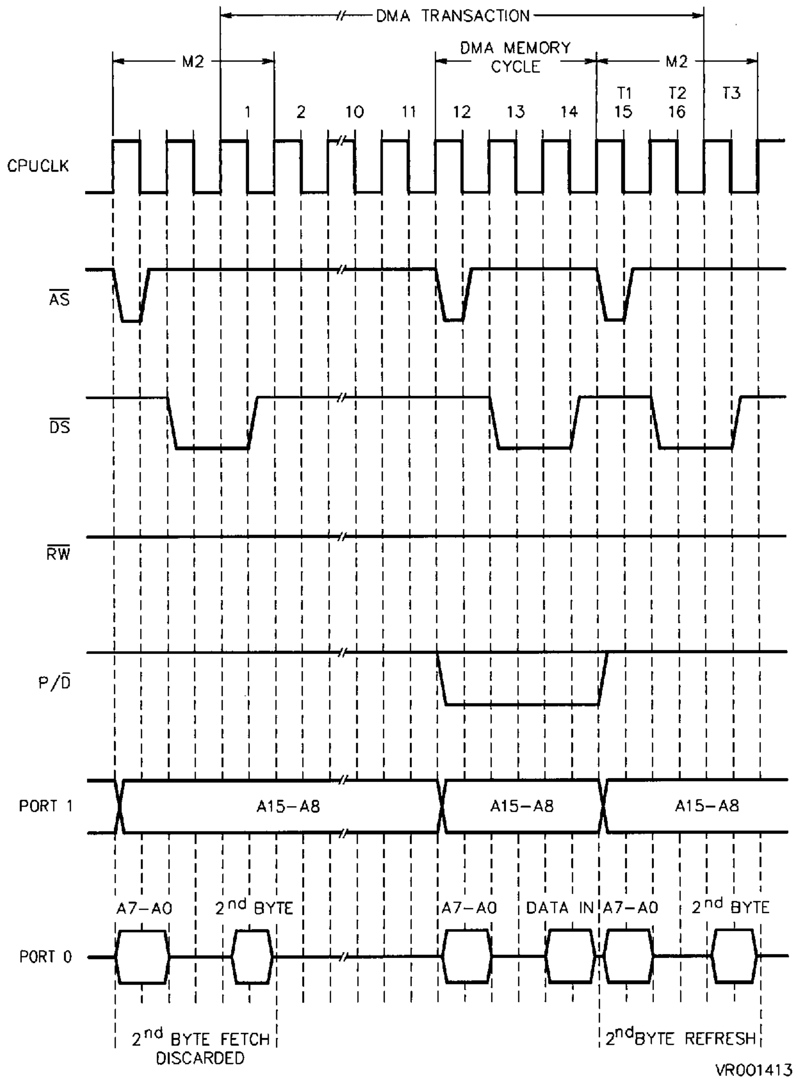


Figure 5-5. DMA Transaction from Memory



5.5 THE SWAP-MODE

An extra feature of ST9 DMA channels of some peripherals (i.e the MultiFunction TIMER) is the SWAP mode. This feature allows transaction from two DMA tables alternatively. All the DMA descriptors in the Register File are thus doubled. Two DMA transaction counters and two DMA address pointers allow the definition of two fully independent tables (they only have to belong the same space, Register file or Data memory or Program memory). The DMA transaction is programmed to start on one of the two tables (say table 0) and, at the end of block, the DMA controller automatically swaps to the other table (table 1) by pointing to the other DMA descriptors. In this case, the DMA mask (DM bit) control bit is not cleared, but the End Of Block interrupt request is generated to allow the optional updating of the first data table (table 0).

Until the swap mode is not disabled, the DMA controller will continue to swap between DMA Table 0 and DMA Table 1.

5.6 DMA REGISTERS

As each peripheral DMA channel has its own control registers, the following register list should be considered as a general example. The names and register bit allocation shown here may be different from those found in the peripheral chapters.

DCPR Address set by Peripheral Read/Write
DMA Counter Pointer Register

Reset value: undefined

7							0
C7	C6	C5	C4	C3	C2	C1	RM

b7-b1 = **C7-C1:** *DMA Transfer Counter Register(s) Address*

b0 = **RM:** *Register File/Memory Selector* If set, the DMA transactions are done with the Register File; if cleared, the DMA transactions are done with the Program or Data memory (see DAPR.DP)

IDCR Address set by Peripheral Read/Write
Generic Peripheral Interrupt and DMA Control

Reset value: undefined

7							0
		IP	DM	IM	PRL2	PRL1	PRL0

b5 = **IP:** *Interrupt Pending*. Set by hardware when the Trigger Event occurs. Cleared by hardware when the request is acknowledged for DMA cycles and external interrupts only. Can be set/cleared by software in order to generate/cancel a pending request. Identical in function to IP of ICR.

b4 = **DM:** *DMA Mask*. If DM = "0" no DMA request is generated when the trigger event occurs. This bit is cleared whenever the transaction counter reaches zero (unless SWAP mode is active).

b3 = **IM:** *Interrupt Mask*. If IM = "0" no interrupt request is generated. If IM = "1" DMA requests depend on DM bit value as shown above.

b2-b0 = **PRL2, PRL1, PRL0:** *Priority Level* Definition of the source priority level. PRL = 0 is highest priority. If PRL = 7, no interrupt can be acknowledged, DMA requests will be.

DAPR Address set by Peripheral Read/Write
DMA Address Pointer Register

Reset value: undefined

7							0
A7	A6	A5	A4	A3	A2	A1	DP

b7-b1 = **A7-A1:** *DMA Address Register(s) Address*

b0 = **DP:** *Data/Program Memory Selector:* (DAPR.RM is "0") if set the DMA transactions are made with the Data Memory; if cleared the DMA transactions are made with the Program Memory.

6 CLOCK

6.1 INTRODUCTION

The ST9 Clock generator module generates the internal clock for the ST9 core and the on-chip peripherals. The Clock generator can be driven by an external crystal circuit, connected to the OSCIN and OSCOUT pins, or by an external pulse generator, connected to OSCIN.

6.2 CLOCK MANAGEMENT

The oscillator circuit generates an internal clock signal with the same period and phase as at the OSCIN input pin. The maximum frequency allowed is 24MHz.

As shown in Figure 6-1, the CLOCK1 signal drives a programmable divider by two. If the control bit MODER.DIV2 (R235.5) is set, the internal clock CLOCK2 is CLOCK1 divided by two; otherwise, if DIV2 bit is cleared, the clock signal CLOCK2 has the same period and phase as CLOCK1. CLOCK2 drives the internal clock INTCLK delivered to all ST9 on-chip peripherals and acts as the central timebase for all timing functions (e.g. Multifunction Timer or Serial Communications Interface Baud Rate generator).
The maximum frequency allowed for INTCLK is 12MHz. For internal operating frequencies above 8MHz, it is recommended to work with the Clock Divider active in order to provide a duty cycle of 50% for INTCLK.

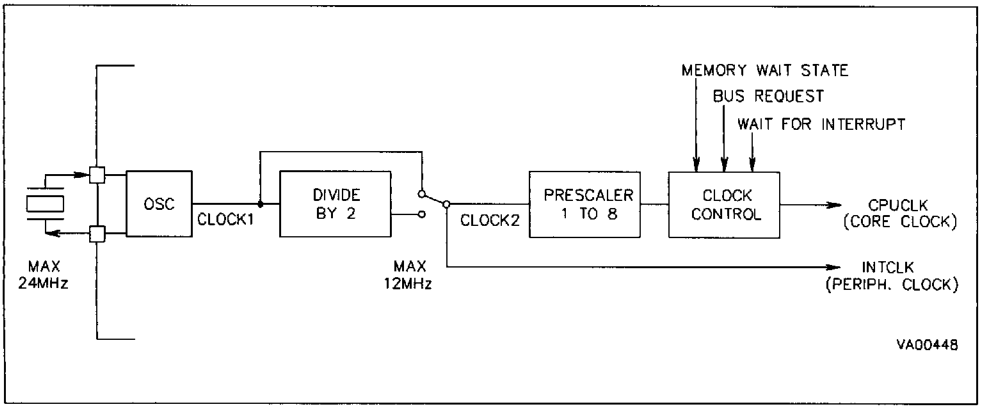
CLOCK2 also drives a programmable prescaler which generates the basic time base, CPUCLK, for the instruction executer of the ST9 core. This allows the user to slow the program execution time to reduce power dissipation, and to locally speed up certain code segments for time critical routines. The internal peripherals are not affected by the CPUCLK prescaler. The prescaler value divides the input clock by the value programmed in the control bits MODER.PRS2,1,0 (R235.4,3,2). If the prescaler value is zero, no prescale is made, thus CPUCLK has the same period and phase as CLOCK2 and 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 6-2. It must be noticed that the prescaling of the clock does not keep the duty cycle to 50%, but stretches the high level of the clock until completion.

When External Memory Wait (or Bus Request or Wait for Interrupt) events occur, CPUCLK is stretched on the high level for the whole period required by the function.

Note. The added wait cycles refer to the INTCLK frequency and not the original CPUCLK.

Figure 6-3 shows an example of a memory access cycle with the CPUCLK prescaled by 2 and with 5 added Wait states.

Figure 6-1. Peripheral and Core Clocks



6.3 CLOCK CONTROL REGISTER

The ST9 clock division by 2 and the clock prescaling are controlled by the MODER register.

Note. This register contains bits with other functions. Only the bits relating to control of the clock are shown here.

MODER R235 (EBh) System Read/Write Mode Register

Reset Value : 1110 0000 (E0h)

7							0
X	X	DIV2	PRS2	PRS1	PRS0	X	X

b5 = **DIV2:** *OSCIN Divided by 2.* This bit controls the divide by 2 circuit which operates on the OSCIN Clock. A logical "1" value means that the OSCIN clock is internally divided by 2, and a logical "0" value means that no division of the OSCIN Clock occurs.

b4-b2 = **PRS2, PRS1, PRS0:** *Prescaling of ST9 Clock.* These bits define the prescaler value used to prescale the 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 (PRS2,1,0 + 1).

Figure 6-2. Core Clock Prescaling

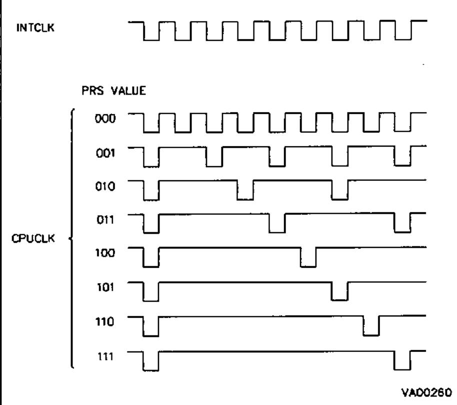
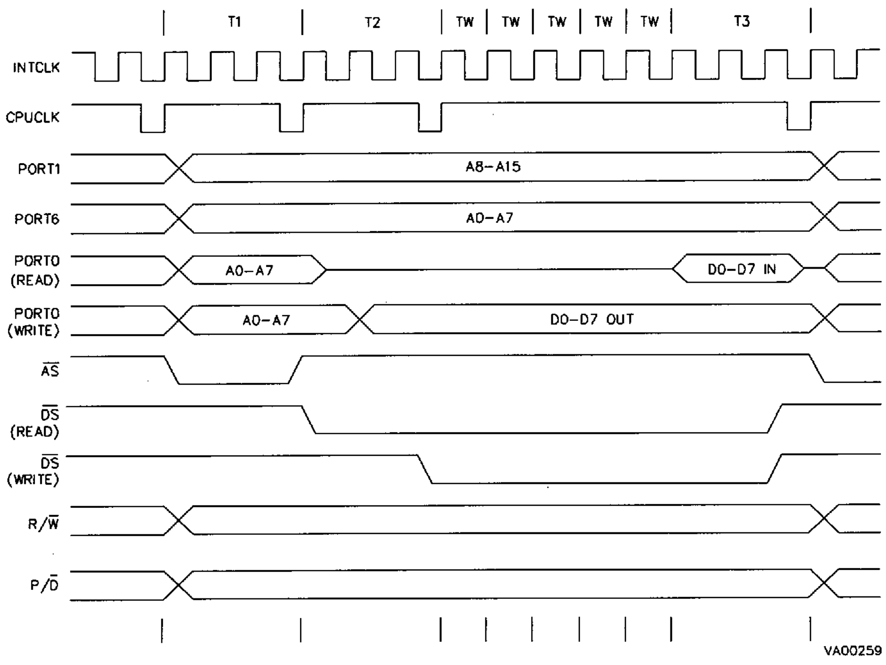


Figure 6-3. Memory Access with a Clock Prescaler by 2 and 5 Wait Cycle



6.4 OSCILLATOR CHARACTERISTICS

The on-chip oscillator circuit (Figure 6-4) is an inverting gate circuit.

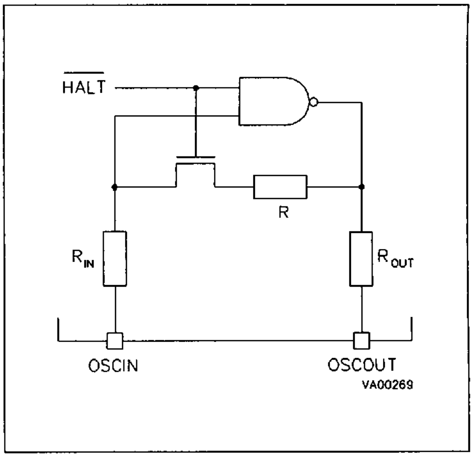
Note. Owing to the Q factor required, Ceramic Resonators may not provide a reliable oscillator source.

In Halt mode, set by means of the `HALT` instruction, 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 level.

To exit the HALT condition and restart the oscillator, an external RESET pulse is required of a minimum duration of 12ms (Figure 6-7).

It must be noted that if the Timer/Watchdog watchdog function is enabled, a `HALT` instruction will not disable the oscillator. This to avoid stopping the watchdog if, by an error, a `HALT` code is executed. When this occurs, the ST9 CPU falls into an endless loop ended by the watchdog (or external) reset.

Figure 6-4. Internal Oscillator Schematic



Note: ROUT< 50Ω R > 1MΩ 300Ω < RIN < 500Ω

Figure 6-5. Crystal Oscillator

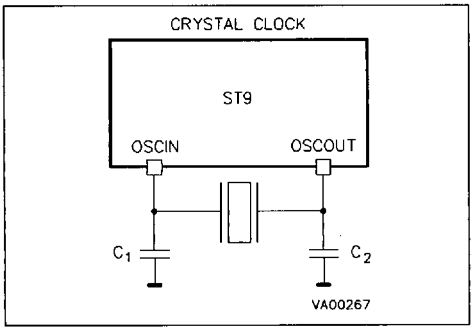


Figure 6-6. External Clock

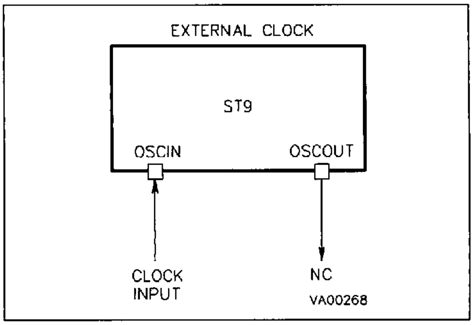


Table 6-1. Crystal Specification (C0 <= 7pF)

Frequency (MHz)	C1=C2=56pF Rs Max	C1=C2=47pF Rs Max	C1=C2=22pF Rs Max
24	20	25	70
16	40	60	150
12	80	100	250
8	180	240	600
4	700	800	600

Table 6-2. Oscillator Transductance

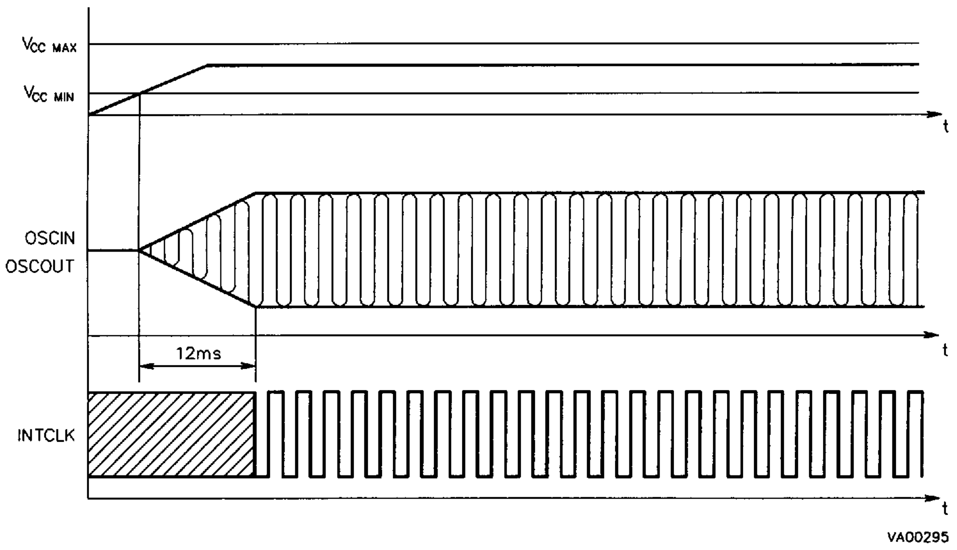
gm	Min	Typ	Max
mA/V	3	5.8	9.5

Legend:
Rs: Parasitic Series Resistance of the quartz crystal (upper limit)
C0: Parasitic capacitance of the quartz crystal (upper limit, < 7pF)
C1, C2: Maximum Total Capacitances on pins OSCIN and OSCOUT(the value includes the external capacitance tied to the pin plus the parasitic capacitance of the board and of the device)
gm: Transconductance of the oscillator

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

OSCILLATOR CHARACTERISTICS (Continued)

Figure 6-7. Oscillator Start-up Sequence



7 RESET

7.1 INTRODUCTION

The processor Reset overrides all the other conditions and forces the ST9 to the reset state. During Reset, the internal registers are set to the reset value, as shown in Table 7-1 for the system and Page 0 Registers and the I/O pins are set to the Bidirectional Weak Pull-up mode (see Warning). The programmer must then initialize the ST9 system and peripheral control registers to give the required functions.

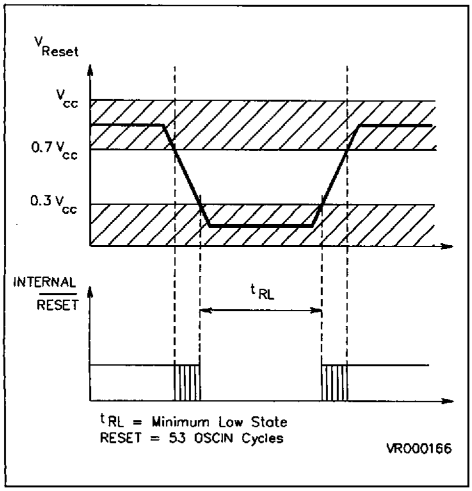
7.2 RESET GENERATION

The reset condition can be generated by the external pin $\overline{RESET}$ or by the on-chip Timer/Watchdog.

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.

During reset, the $\overline{DS}$ output signal is kept low and the $\overline{AS}$ output is toggled with the crystal frequency (input at OSCIN) divided by 32. This condition is recognized by off-chip Z-bus peripherals as a reset condition.

Figure 7-8. Signal to be applied on Reset Pin



7.3 RESET PIN TIMING

The $\overline{RESET}$ pin has a Schmitt trigger input circuit with hysteresis. The internal reset is generated by the external pin synchronized with the internal clock. The power up reset circuit must keep the $\overline{RESET}$ input low for a minimum of the crystal startup period plus 53 crystal periods.

Once the $\overline{RESET}$ pin reaches a logical "1", the processor exits from the reset status after 67 crystal periods from the rising edge ($\overline{DS}$ is set). The processor then fetches from Program Memory locations 0 and 1 (power-on reset vector) and begins program execution from the address contained in the vector. If the ST9 is a ROMLESS version, without on-chip program memory, ports Port0, Port1 (and Port6 for ST909x family) are set to external memory mode (i.e Alternate Function) and the memory accesses are made to external Program memory with wait cycles insertion.

WARNING: *I/O pins are set to the Weak Pull-up mode during the Reset cycle. This state is forced during the reset sequence, but the I/O pins can be in a random state for up to 64 crystal periods.*
The application circuit must take this into account if it can lead to critical situations in the external circuitry.

7.4 PROCESSOR SYNCHRONIZATION UNDER RESET

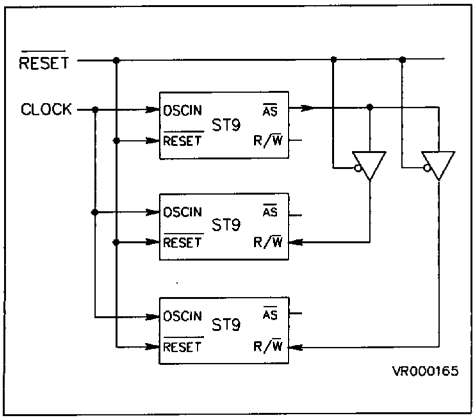
During reset, a specific procedure has been implemented to synchronize two or more oscillators in a multi-micro ST9 based system, for example a majority voting high reliability system. Figure 7-9 shows the principle schematic for the multi-microprocessor synchronization. The master processor delivers the synchronous signal, output at its $\overline{AS}$ pin, to the R/$\overline{W}$ pin of the slave processors. The R/$\overline{W}$ pin is, under reset status, set to input with a weak (10kΩ typical) pull up resistor. The slave processor(s) synchronizes its internal clock phase with the clock received at its R/$\overline{W}$ pin. To guarantee the phase synchronization, the reset status must be at least 32x31 = 992 crystal periods. All the processors must have the same input clock.

RESET (Continued)

Table 7-1. Internal Registers Reset Values

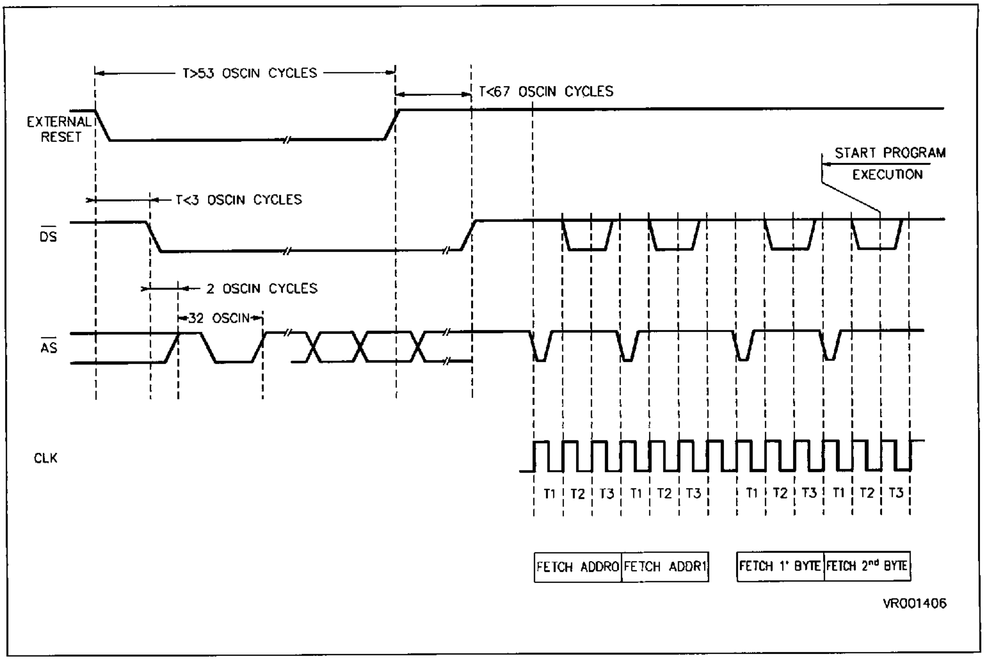
Register Number	System Register	Reset Value	Page 0 Register	Reset Value
F	(SSPLR)	undefined	Reserved	
E	(SSPHR)	undefined	(SPICR)	00h
D	(USPLR)	undefined	(SPIDR)	undefined
C	(USPHR)	undefined	(WCR)	7Fh
B	(MODER)	E0h	(WDTCR)	12h
A	(Page Ptr)	undefined	(WDTPR)	undefined
9	(Reg Ptr 1)	undefined	(WDTLR)	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	Reserved	undefined	(EIPR)	00h
2	(PORT2)	FFh	(EITR)	00h
1	(PORT1)	FFh	(MIRRG)	undefined
0	(PORT0)	FFh	Reserved	

Figure 7-9. Synchronization Under Reset



RESET (Continued)

Figure 7-10. Exit From Reset Timing



8 EXTERNAL MEMORY INTERFACE

8.1 INTRODUCTION

In the event of an application requiring more ROM space than available on-chip, or for easier program management and customization with external memory or peripherals, the ST9 microcontroller provides an external memory interface. The external memory interface provides the memory lines and timing and status control signals, plus enhanced features including programmable memory wait cycles, bus request/acknowledge cycles and shared memory bus access control.

The ST9 Memory Control Unit automatically recognizes if a memory location belongs to on-chip memory. When the memory location is on-chip, it performs a machine cycle without $\overline{\text{DS}}$ generation, and the access is performed on-chip. If the location does not belong to on-chip memory, an access to off-chip memory is performed (generating the $\overline{\text{DS}}$ low pulse) through the Ports 0, 1 (and 6 for ST90R9x family).

During Reset, $\overline{\text{AS}}$ and $\overline{\text{DS}}$ are driven to perform external peripherals reset and to implement, in conjunction with the R/$\overline{\text{W}}$ pin, a multi-microprocessor synchronization procedure (see Clock and Reset chapters).

8.2 CONTROL SIGNALS

$\overline{\text{AS}}$

Address Strobe (Output, Active low, Tristate). The rising edge of $\overline{\text{AS}}$ indicates that Memory Address, R/$\overline{\text{W}}$ and P/$\overline{\text{D}}$ Memory signals are valid.
$\overline{\text{AS}}$ is released in high-impedance during a Bus acknowledge cycle or under processor control by setting the HIMP bit (MODER.0).

$\overline{\text{DS}}$

Data Strobe (Output, Active low, Tristate). Data Strobe provides the memory data timing during external memory access cycle. When internal memory is accessed, $\overline{\text{DS}}$ is kept high during the whole memory cycle. During an External memory write cycle, the data output at Port 0 is valid when $\overline{\text{DS}}$ is active. During a read cycle, the data at Port 0 must be valid before the trailing edge of $\overline{\text{DS}}$.
$\overline{\text{DS}}$ is released in high-impedance during a Bus Acknowledge cycle or under processor control by setting the HIMP bit (MODER.0).

R/$\overline{\text{W}}$

Read/Write (Output, Active low, Tristate). The R/$\overline{\text{W}}$ output signal identifies the type of memory cycle. When R/$\overline{\text{W}}$ = "1", the memory cycle is a Memory Read cycle; when R/$\overline{\text{W}}$ = "0", it is a Memory Write Cycle. R/$\overline{\text{W}}$ output signal is defined at the beginning of the memory cycle and is stable until the next Memory cycle.
R/$\overline{\text{W}}$ is released in high-impedance during Bus acknowledge cycle or under processor control by setting the HIMP bit.

P/$\overline{\text{D}}$

Program/Data Memory (Alternate Function Output, Active low). The P/$\overline{\text{D}}$ output signal selects between Program and Data Memory. When P/$\overline{\text{D}}$ = "1", the memory referenced by the processor is the Program Memory; when P/$\overline{\text{D}}$ = "0", the memory referenced is the Data Memory. The P/$\overline{\text{D}}$ output signal is defined at the beginning of the memory cycle and is stable until the next Memory cycle.
P/$\overline{\text{D}}$ is enabled by software as the Alternate Function output of a parallel port bit (refer to the Pin Configuration and Alternate Function tables to identify the specific port and pin).

$\overline{\text{WAIT}}$

External Memory Wait (Input, Active low). The $\overline{\text{WAIT}}$ input signal indicates to the ST9 that the external memory requires more time to complete the memory access cycle. The memory cycle will then be stretched. $\overline{\text{WAIT}}$ is enabled by setting EWEN (EIVR.0 R246 Page 0).

$\overline{\text{BREQ}}$

Bus Request (Input, Active low). The $\overline{\text{BREQ}}$ input signal indicates to the ST9 that a bus request has tried or is trying to gain control of the memory bus. $\overline{\text{BREQ}}$ is enabled by setting BRQEN (MODER.1 R235).

$\overline{\text{BACK}}$

Bus Acknowledge (Alternate Function Output, Active low). The $\overline{\text{BACK}}$ output signal indicates that the ST9 has relinquished control of the memory bus in response to a bus request.

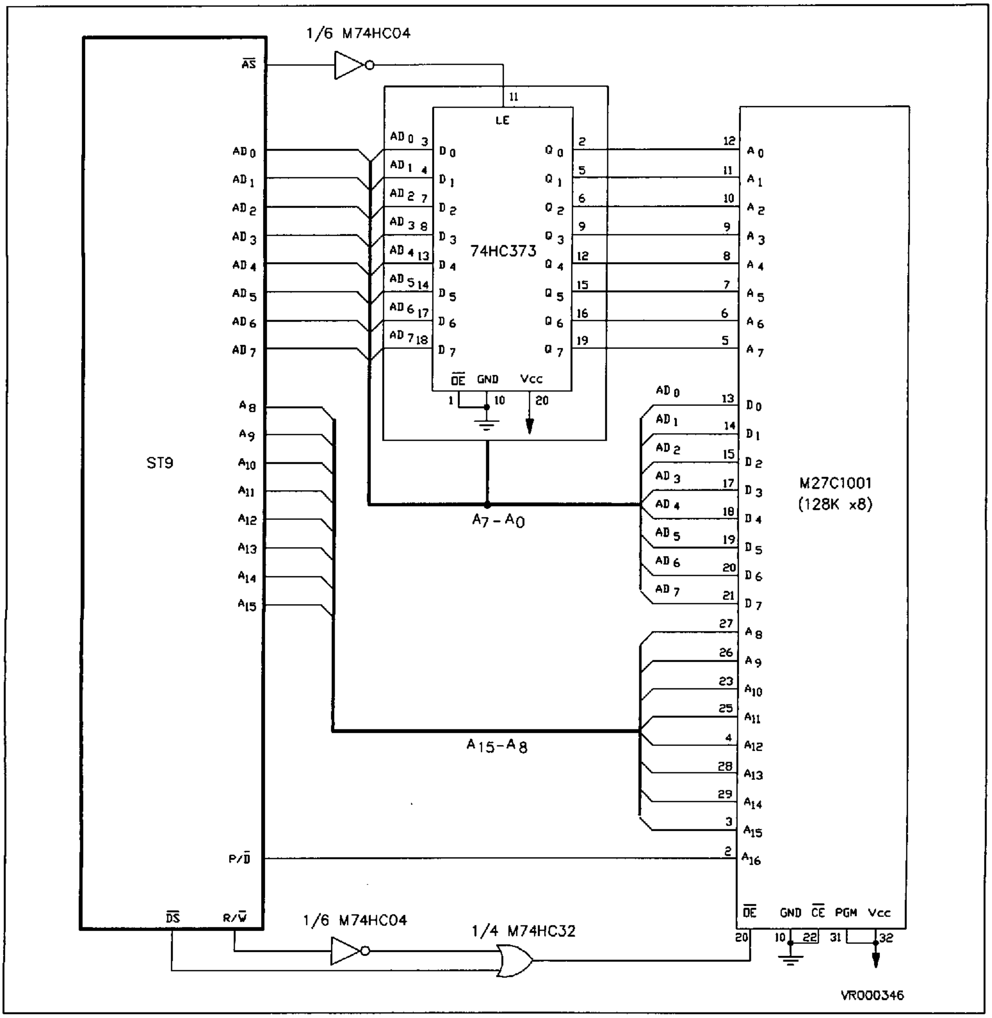
CONTROL SIGNALS (Continued)

P0
Port 0 (Input/Output, Push-Pull/Open-Drain/Weak Pull-up). Port0 can be configured as a bit programmable Parallel I/O port or as External Memory interface for multiplexed Low-Address/Data (A0-7/D0-7).

P1
Port 1 (Input/Output, Push-Pull/Open-Drain/Weak Pull-up). Port1 can be configured as a bit programmable Parallel I/O port or as External Memory interface for the High-Address (A8-A15).

P6 *(When available)*
Port 6 (Input/output, Push-Pull/Open-Drain/Weak Pull-up). This port, when available, can be configured as bit programmable Parallel I/O port or as External Memory interface for the Low-Address (A0-7), allowing a non-multiplexed memory bus capability.

Figure 8-1. ST9 Accessing External Program and Data Memory.

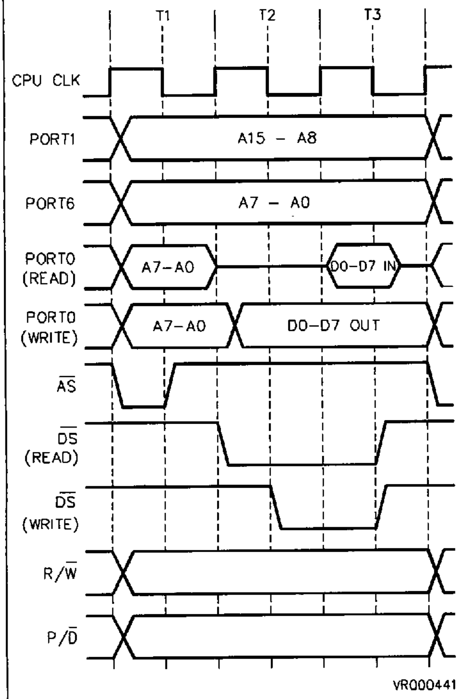


7929237 0058003 0T5

8.3 MEMORY ACCESS CYCLE

Each memory access cycle is composed of three CPUCLK phases: T1, T2, T3. During phase T1, the memory address is output upon $\overline{AS}$ falling edge and is valid upon the rising edge of $\overline{AS}$. Port1 and Port6 maintain the address stable until the next T1 phase.
If the Memory access cycle is a Read cycle, Port0 pins are released to high impedance with the falling edge of $\overline{DS}$ until the next $\overline{AS}$ falling edge.
If the Memory access is a Write cycle, Port0 is held active, the data is output during T2 and is maintained until the next address is output (upon the falling edge of $\overline{AS}$). DS is pulled low during T2 only if the Memory access is an External Memory access. If the memory cycle is a Memory Read, it is pulled low at the beginning of T2. If it is a Memory Write, $\overline{DS}$ is kept low from the middle of T2 until the middle of T3.

Figure 8-2. External Memory Read/Write.



8.4 STRETCHED ACCESS CYCLE

The ST9 can interface to memory with slow access times by automatically inserting additional Wait cycles during the External Memory cycle. On-chip memory accesses do not require $\overline{WAIT}$ cycles and run at the full speed of CPUCLK.

Three Wait cycle sources are available:

- The input pin WAIT from external sources
- The internal programmable Wait cycle generator
- Internal memories with stretched access cycle

The input pin $\overline{WAIT}$ (when enabled) is sampled on the CPUCLK falling edge of phase T2. If active (low), one INTCLK clock cycle will be added. During the added clock cycle, the $\overline{WAIT}$ pin is sampled again. CPUCLK is stretched for as long as the $\overline{WAIT}$ input is active.

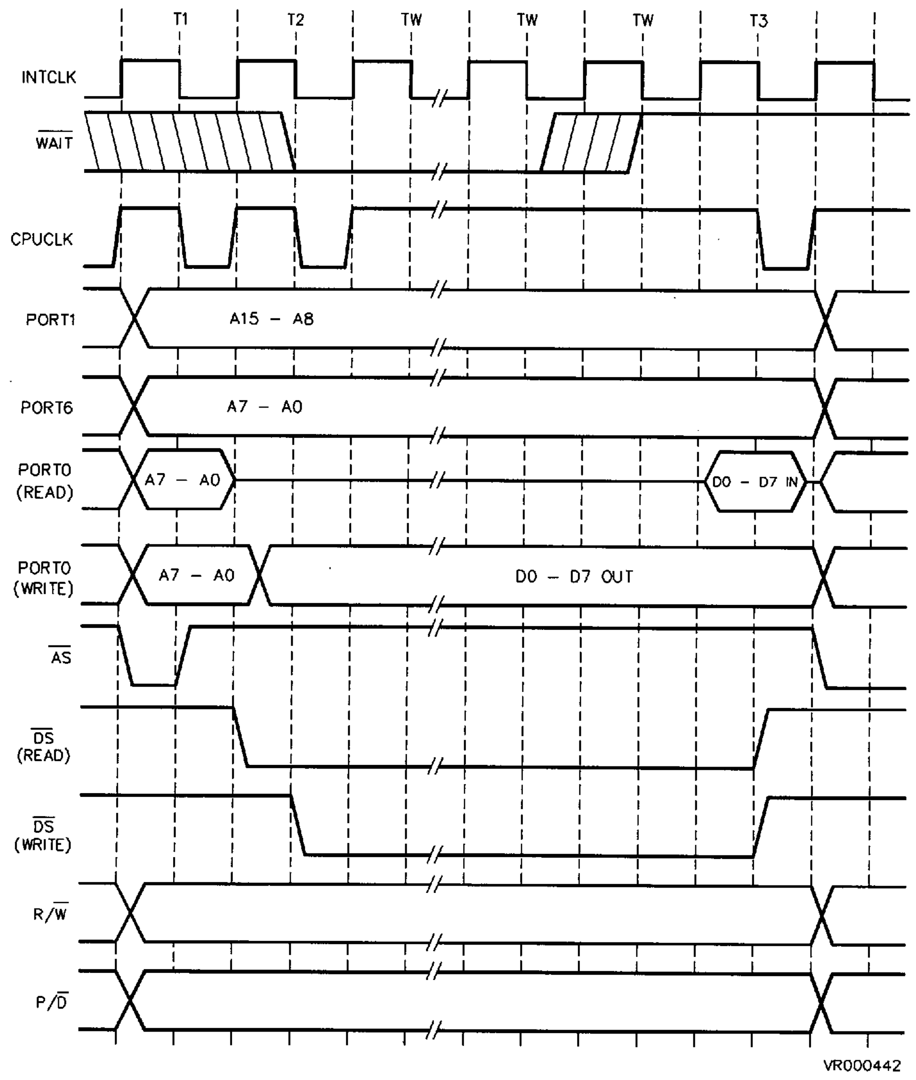
The internal programmable $\overline{WAIT}$ cycle generator allows the extension of the External Memory cycle automatically by the programmed number of $\overline{WAIT}$ cycles. Two three bit fields in the Wait Control Register WCR (R252 Page 0) allow the stretching of Program and Data Memory access cycles independently by 0 to 7 cycles. WPM2,1,0 (WCR.5,4,1) contain the number of Program memory wait cycles to be added, WDM2,1,0 (WCR.2,1,0) contain the number of Data memory wait cycles to be added.

Table 8-1. Number of wait cycles added

WDM2 WPM2	WDM1 WPM1	WDM0 WPM0	Nb of Clock cycle added	Note
0	0	0	0	No Wait cycle
0	0	1	1	
0	1	0	2	
0	1	1	3	
1	0	0	4	
1	0	1	5	
1	1	0	6	
1	1	1	7	Reset Value

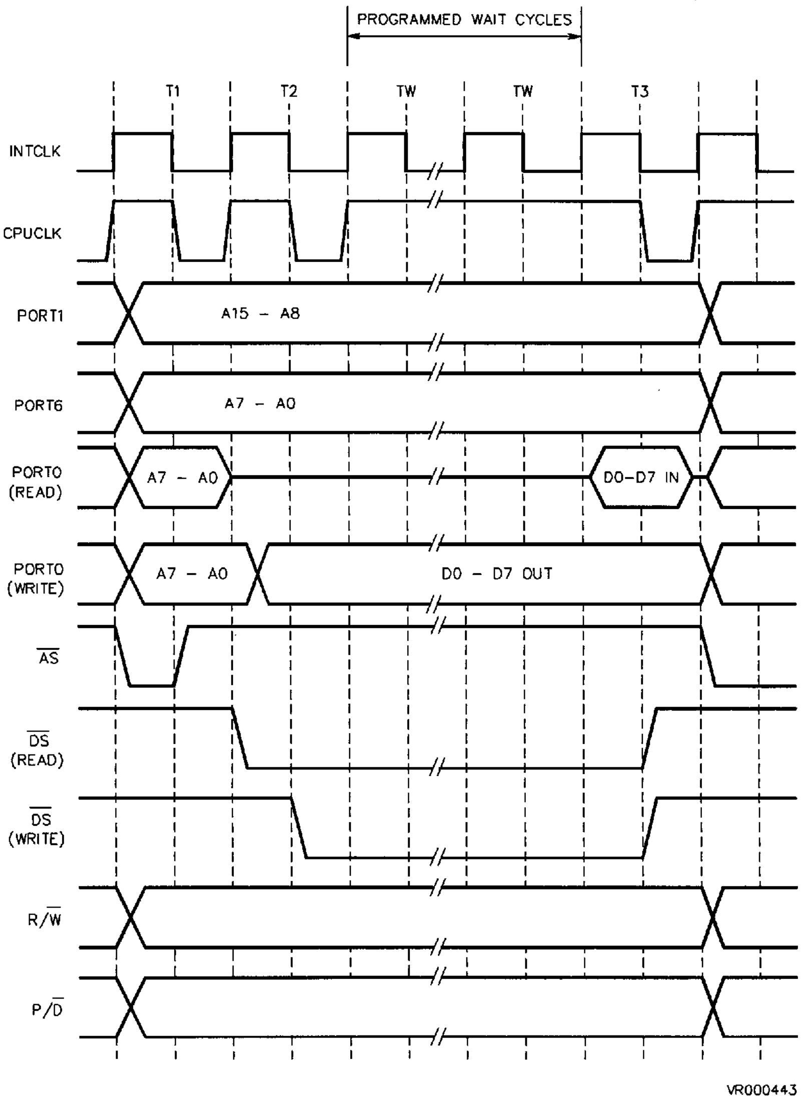
STRETCHED ACCESS CYCLE (Continued)

Figure 8-3. External Memory Read/Write Sequence with External Wait.



STRETCHED ACCESS CYCLE (Continued)

Figure 8-4. External Memory Read/Write Sequence with Programmable Wait.



8.5 SHARED BUS

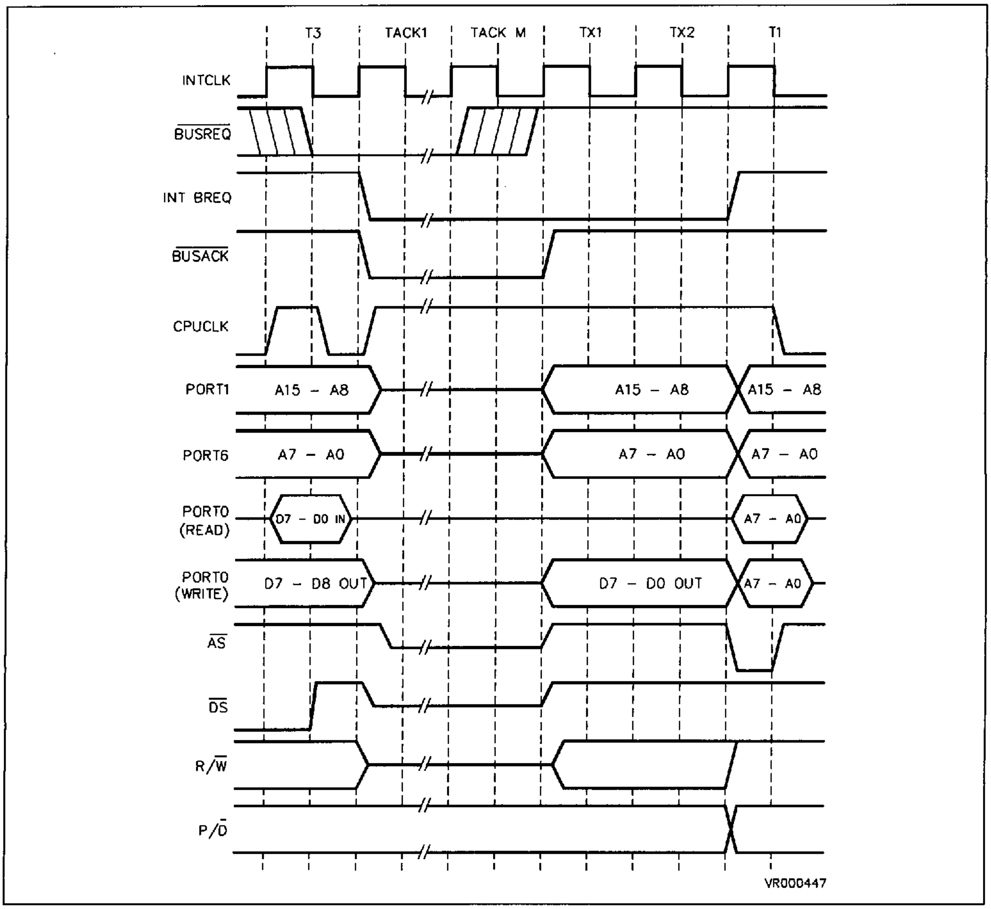
When the ST9 runs in a multi-master bus system, it is necessary to release the bus control to other bus master(s). This operation can be performed by the Bus Request/Acknowledge capability supported by the ST9.

Once enabled by setting BRQEN (MODER.1 R235), $\overline{\text{BREQ}}$ is sampled by the ST9 upon the falling edge of the internal clock during the phase T3. When the $\overline{\text{BREQ}}$ signal is sampled low, the CPUCLK clock is stretched and the External Memory signals ($\overline{\text{AS}}$, $\overline{\text{DS}}$, R/$\overline{\text{W}}$, P0, P1 (, P6 for ST909x)) are released to high-impedance. The input signal $\overline{\text{BREQ}}$ is then continuously monitored, and when it is sampled high the External Memory interface pins are driven again by the ST9 after two additional internal clock cycles. These cycles are used to fully drive and propagate the control and data signals through the external memory bus before CPUCLK is restarted.

The output signal $\overline{\text{BACK}}$ is driven low during the whole period when the External Memory interface is released to high impedance.

Under the Reset status, the bits of the I/O port(s) associated to $\overline{\text{BREQ}}$ and $\overline{\text{BACK}}$ are set to Bidirectional Weak pull-up mode and the enable bit BRQEN is cleared. To enable this function, the program must set the $\overline{\text{BACK}}$ port as an Alternate Function output and enable (set to "1") the bit BRQEN.

Figure 8-5. Bus Request/Acknowledge Timing.



SHARED BUS (Continued)

When it is required to disable the external bus, but to keep the processor running in the on-chip memory, the external memory bus can be disabled by software programming of the HIMP (MODER.0) control bit. By setting HIMP, the External Memory Interface ($\overline{AS}$, $\overline{DS}$, $R/\overline{W}$ and Port0, Port1 (and Port6), if not configured as I/O lines) is set into a high impedance state. In this way, the external memory bus can be used by another resource (e.g. diagnostic equipment or external programming of system memories) and the ST9 program can continue accessing the on-chip memory. This feature can also be useful for high security applications where the flow and addresses of the on-chip security algorithms must not be shown on the external address pins.

When running in internal memory, disabling the external bus will reduce the noise emitted by the micro.

The disabling of the External Memory Interface by setting HIMP = "1" can be interrupt driven by applying the "Bus Request" input signal to an External Interrupt pin. In this case the bus disable response time will be longer than the automatic system using the $\overline{BREQ}$ request, however the ST9 can continue to execute the program written in the on-chip memory.

8.6 PORTS P0, P1, P6 INITIALIZATION AFTER RESET

The Port 0, Port 1 and Port 6 (for ST909x family) initialization after reset depends on the configuration of the ST9 as shown in Table 8-2.

If the device has on-chip Program memory (ROM or EPROM), the ports (or the existing parts of them) are set to Bidirectional Weak Pull-up Mode.

Table 8-2. Port status after Reset

Device	Port 0, 1, 6 Initialization
ROM EPROM	Bidirectional Weak-Pull-Up (PxC0, PxC1, PxC2 = 0,0,0; Data = 1)
ROMLESS	Memory Address and Data Alternate Function Push-Pull (PxC0, PxC1, PxC2 = 1,1,0; Data = 1)

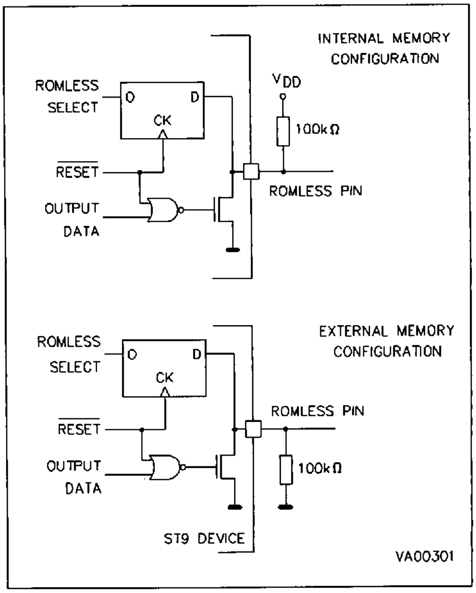
If the device is ROMLESS or a ROM device with the ROMless function enabled, the ports (or the existing part of them) are set to Alternate Function Push-Pull Mode, providing the Address and Data lines to interface to the external Program and Data Memory from Reset.

8.7 ROMLESS FUNCTION

In order to accomodate the use of ROM based ST9 devices in the event of a subsequent ROM code change, a ROMless function may be enabled on a specified Port I/O pin by Mask Option. This function is activated by pulling the ROMless select pin to ground with a 100kohm resistor. This status is latched on the rising edge of the $\overline{RESET}$ pin and, when low, the on-chip PROGRAM memory (ROM or EPROM) is disabled, causing all instruction fetch cycles to be external. On-chip Data memory (RAM or EEPROM) is not affected.

If the ROMless function is enabled by the mask option, and the internal program is to be used, then the ROMless pin must be held to a high level (via a 100kohm resistor to V_{DD}) during the Reset cycle. After the Reset cycle the ROMless pin may be programmed for any I/O or Alternate function.

Figure 8-6. ROMless Selection

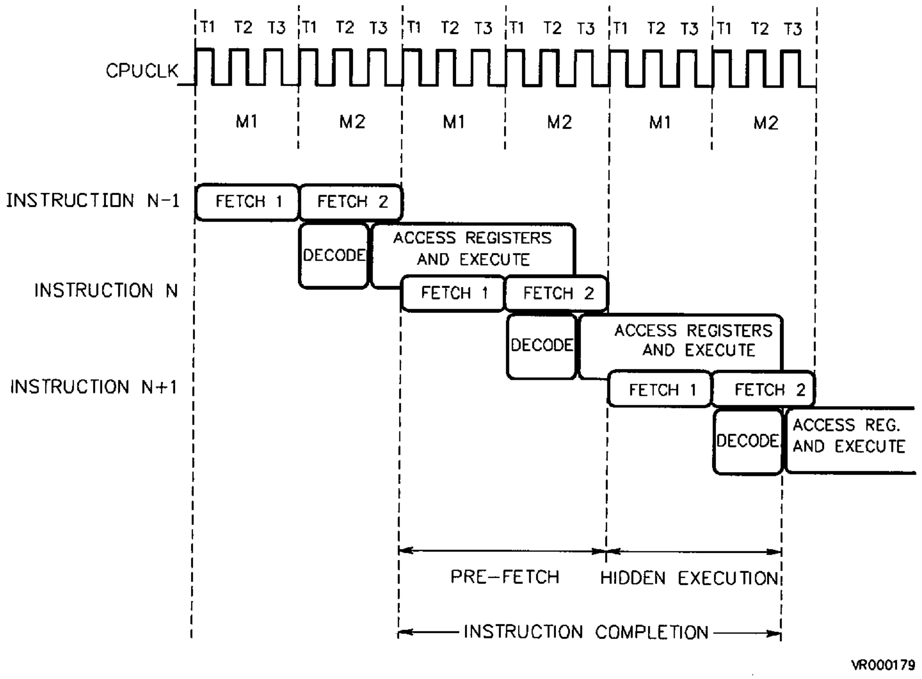


8.8 PIPELINE

The ST9 implements pipe-line stages on instruction fetch and execution in order to increase the execution speed. The instruction execution is in fact hidden by the Memory access cycles: the execution of one instruction is overlapped with the prefetch of the two successive bytes. The fetch of the first byte (opcode) is identified by the machine cycle M1, the fetch of the second byte by M2.

The 2 bytes instructions, whose execution time is 6 CPUCLK cycles, have the instruction execution hidden by the following instruction prefetch. For those instructions that require an execution time longer than the time to prefetch the following bytes, perform memory access during their execution or interrupt the sequential memory access, the pipe is flushed.

Figure 8-7. Instruction Pipe-line Stages



8.9 "SPURIOUS" MEMORY READ ACCESSES

The ST9 in certain cases produces external memory accesses which may be regarded as "Spurious" in their nature. While these do not affect the correct operation of the ST9, these accesses may cause misunderstandings when developing and debugging applications as the signals $\overline{AS}$ and $\overline{DS}$ are produced, and Ports 0, 1, (6 for ST909x) output updated addresses (if used to interface to external memory).

The spurious reading cycle is produced when executing specific instructions. This is one of 4 cases: double reading, reading before writing, reading when the stack is internal or prefetch reading.

- DOUBLE READING
  A memory location read by the ST9 is read two times consecutively (instead of one).

Involved instruction(s):

```
DIV           rr,r   ; divide (16/8)
```

The first byte of the code following `DIV` is fetched two times. The double reading does not occur if the Overflow flag was set by `DIV`, or if Divide by zero was trapped. The P/$\overline{D}$ line remains high during the cycle.

- READING BEFORE WRITING
  A memory location which is to be written to by the ST9 is previously read.

Involved instruction(s):

```
LD            (rr)+, (r)+   ;load (byte)
                            ;Memory, Register
```

The destination memory location is read before being written. The P/$\overline{D}$ line reflects the memory space of the destination memory location.

- READING WHEN THE STACK IS INTERNAL
  If the System and/or User Stack has been programmed to use the Register File, a memory location of Data Space is POPed in parallel.

Involved instruction(s):

```
POP       R  ; POP (byte) from System Stack
POP      (R) ;        "     "     "
POPU      R  ; POP (byte) from User Stack
POPU     (R) ;        "     "     "
```

While a byte is being `POP`ed from the Register File, a memory location in Data Space is read in parallel with its address given by SSPHR+SSPLR for `POP` instructions and by USPHR+USPLR for `POPU` instructions. The external data is ignored.

```
POPW    RR    ; POP (word) from System Stack
POPUW   RR    ; POP (word) from User Stack
```

While the high er address byte is being popped from the Register File, a memory location in Data Space is read in parallel with its address given by SSPHR+SSPLR for `POPW` instructions and by USPHR+USPLR for `POPUW` instructions. No spurious reading is made for the lower byte.

```
RET           ; Return from Subroutine
```

While the Program Counter Higher and Lower bytes are `POP`ed from the Register File, two memory locations are read at addresses given by SSPHR+SSPLR.

```
IRET          ; Return from Interrupt
```

While the Program Counter Higher and Lower bytes and the FLAGS are `POP`ed from the Register File, three memory locations are read at addresses given by SSPHR+SSPLR. When working with Internal Stacks, SSPHR and USPHR contents are don't care from the point of view of program execution, but they must be considered RESERVED by the User as the instructions listed in this section perform updating of SSPHR/USPHR, together with the spurious reading.

- PREFETCH READING
  Due to the ST9 Pipeline, instructions which stop the Core or which perform program branches can fetch bytes of the following program code while the pipeline is being flushed.

Involved instruction(s):

```
WFI           ; Wait For Interrupt
```

reads two bytes of the following code.

```
HALT          ; Halt
CALL(rr)      ; Unconditional Call subroutine
```

read one byte of the following code in Program space (P/$\overline{D}$ high).

8.10 REGISTERS

WCR R252 (FCh) Page 0 Read/Write
Wait Control Register

Reset Value: 0111 1111 (7Fh)

7							0
X	WDGEN	WDM2	WDM1	WDM0	WPM2	WPM1	WPM0

b7 = Reserved, reads as a “0”.

b6 = **WDGEN:** refer to Timer/Watchdog chapter.

WARNING. Resetting this bit to zero has the effect of setting the Timer/Watchdog to the Watchdod mode. Unless this is desired, this must be set to “1”.

b5-b3 = **WDM2-0:** *Data Space Wait Cycles.* These bits contain the number of INTCLK cycles to be added automatically to external Data memory accesses. WDM = 0 gives no additional wait cycles, WDM = 7 provides the maximum 7 INTCLK cycles (this is the reset condition in order to allow the use of slow access time external memory, if faster memory is used, then this value may be modified by the User).

b2-b0 =**WPM2-0:** *Program Space Wait Cycles.* These bits contain the number of INTCLK cycles to be added automatically to external Program memory accesses. WPM = 0 gives no additional wait cycles, WPM = 7 provides the maximum 7 INTCLK cycles (this is the reset condition in order to allow the use of slow access time external memory, if faster memory is used, then this value may be modified by the User).

Note. the number of clock cycles added refer to INTCLK and NOT to CPUCLK.

WARNING. *The Wait Control Register is reset to give the maximum number of Wait cycles for external memory. To give the optimum performance of the ST9 when used in single-chip mode (no external memory) the WDM2,1,0 and WPM2,1,0 bits should be reset to “0”.*

9 I/O PORTS

9.1 INTRODUCTION

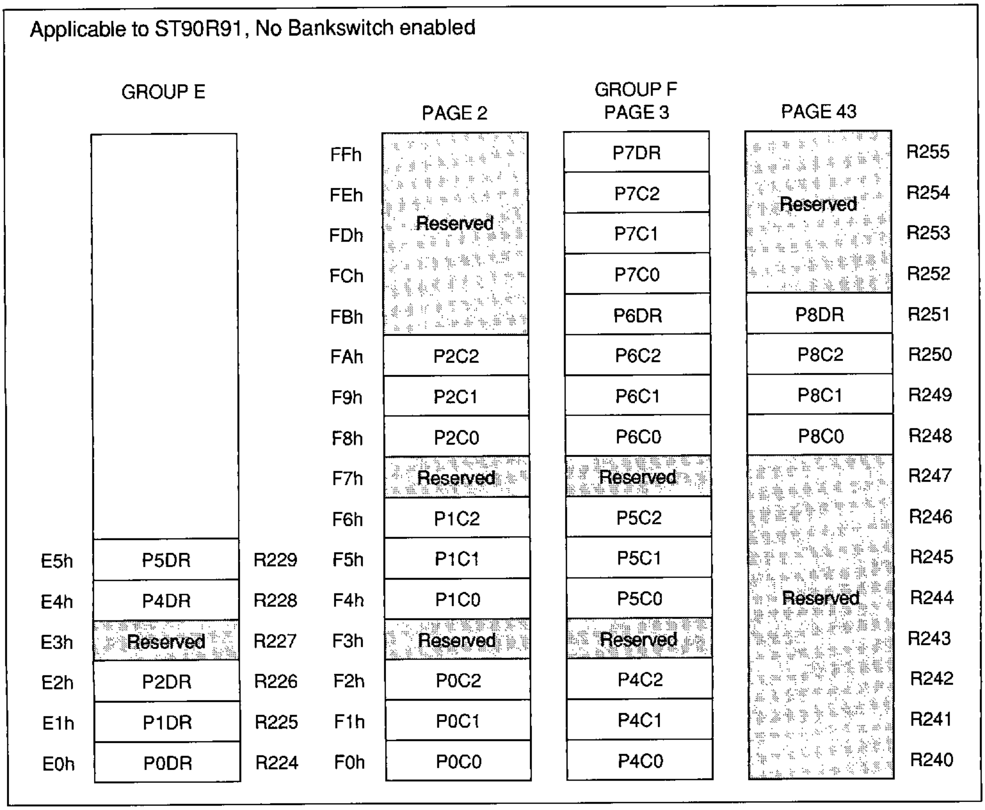
The ST9 is provided with dedicated lines for input/output. These lines, grouped into 8-bit ports, can be independently programmed to provide parallel input/output, or to carry input/output signals to/from the on-chip peripherals and Core (e.g. Timers and SPI). All ports have active pull-ups and pull-down resistors compatible with TTL loads. In addition, pull-ups can be turned off for open-drain operation and weak pull-ups can be turned on to save off-chip resistive pull-ups. Input buffers can be either TTL or CMOS compatible.

Certain pins have specific features, these are shown in the Pin Description section and are shown in section 9.9.6

9.2 CONTROL REGISTERS

Each port PX has a Data Register PX, and three associated control registers (PXC0, PXC1, PXC2) which define the port line configuration and allow dynamic change in port configuration during program execution. Ports and control registers are mapped into the Register File as shown in Figure 9-1 Ports and control registers are treated like any other general-purpose register. There are no special instruction for port manipulation, any instruction that addresses a register can address the ports. Data can be directly accessed in the port register, without passing through other memory or "accumulator" locations.

Figure 9-1. I/O Register Map



Applicable to ST90R91, No Bankswitch enabled

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

CONTROL REGISTERS (Continued)

During the reset state, all the Ports are set as bidirectional/weak pull-up mode, with the output data register set to FFh. This condition is also held after reset (except for Ports 0, 1 (, 6 for ST909x) in ROM-less devices, see Memory chapter) and can be redefined under software control at any time.

9.3 PORT BIT STRUCTURE AND PROGRAMMING

By programming the control bits PXC0.n and PXC1.n (see Figure 9-2) 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 by programming the control bit PXC2.n.

The output buffer can be programmed as Push-pull or Open-drain. A Weak Pull-up configuration can be used when the port bit is programmed as Bidirectional. This is an Open-drain configuration with a high pull-up resistor value (turned on by writing a "1"), to avoid the requirement for external resistances.

The basic structure of the bit PX.n of a general purpose port PX is shown in Figure 9-3.

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

When PX.n is programmed as Input: (Figure 9-4)

- The Output Buffer is forced tristate
- The data present on the I/O pin is sampled into the Input Latch at the beginning of the execution of the instruction which is accessing the port.
- The data stored in the Output Master Latch is copied into the Output Slave Latch at the end of the execution of each instruction. So if bit PX.n is reconfigured as Output or Bidirectional, the data stored in the Output Slave Latch is reflected on the I/O pin.

Figure 9-2. Control Bits

	Bit 7			Bit n				Bit 0
PXC2	PXC27			PXC2n				PXC20
PXC1	PXC17			PXC1n				PXC10
PXC0	PXC07			PXC0n				PXC00

Table 9-1. Port Bit Configuration Table

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	
PXn Configuration	BID	BID	OUT	OUT	IN	IN	AF	AF	
PXn Output	WP	OD	PP	OD	HI	HI	PP	OD	HI[(1)]
PXn Input	TTL	TTL	TTL	TTL	CMOS	TTL	TTL	TTL	V_{SS}[(1)]

Legend:

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

Note 1: For A/D Converter inputs

PORT BIT STRUCTURE AND PROGRAMMING (Continued)

Figure 9-3. Basic Structure of an I/O Port Pin

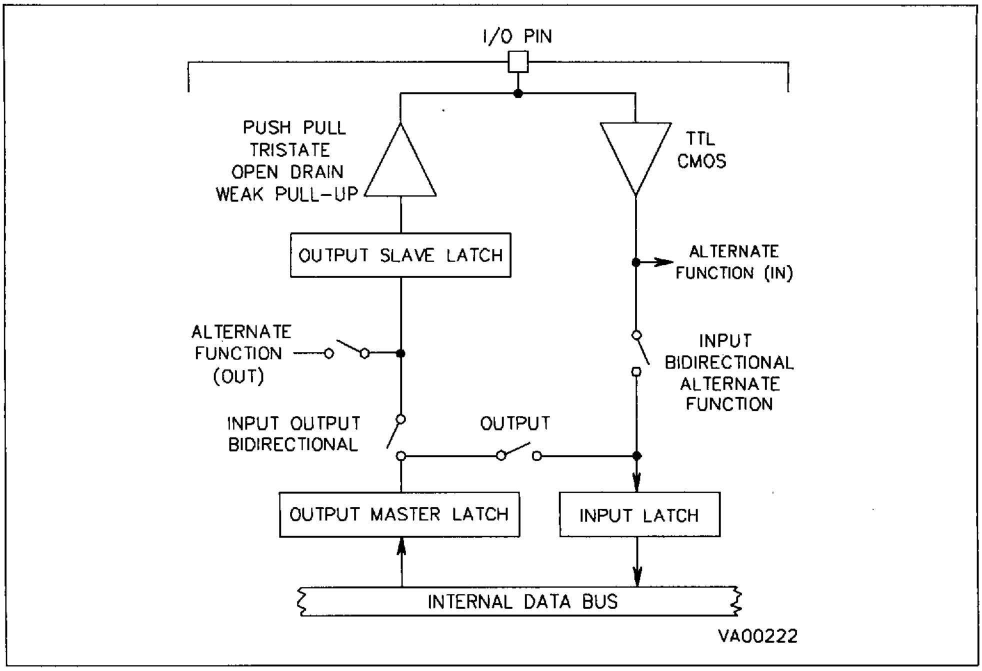


Figure 9-4. Input Configuration

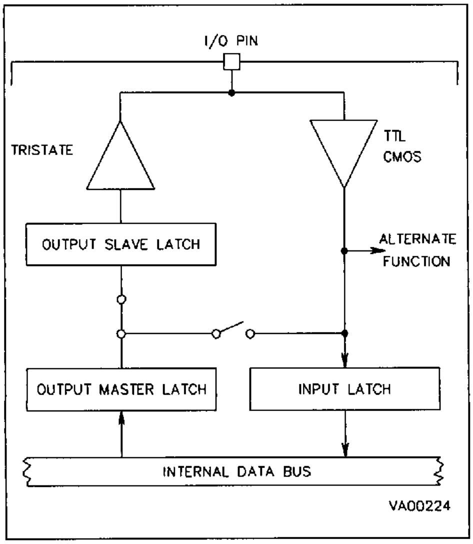
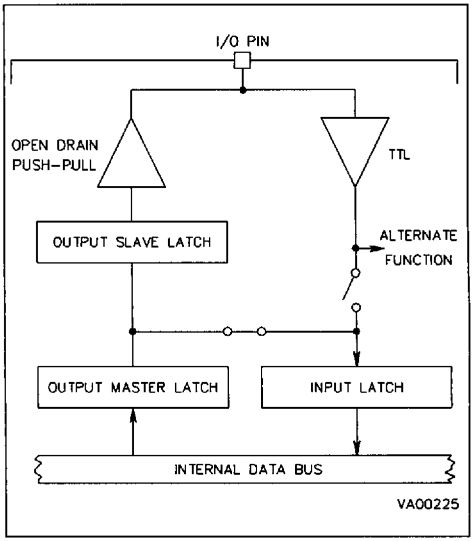


Figure 9-5. Output Configuration



PORT BIT STRUCTURE AND PROGRAMMING (Continued)

When PX.n is programmed as Output: (Figure 9.5)

- 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 each instruction.

When PX.n is programmed as Bidirectional: (Figure 9-6)

- The Output Buffer is turned on in an Open-drain or Weak Pull-up configuration
- The data present on the I/O pin is sampled into the Input Latch at the beginning of the execution of each 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 each instruction.

WARNING. Due to the unique feature of the bidirectional mode of reading the external pin 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	external port value
0Fh	03h

(Bits 3 and 2 are externally forced to 0)

Making a `bset` instruction on bit 7 will return:

Port register content	external port value
83h	83h

(Bits 3 and 2 have been cleared.)

To avoid this situation, it is suggested that all the 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 Alternate Function Output (Figure 9-7) except fo Analog Inputs :

- 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 each instruction
- A signal coming 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 Function. If no Function is connected to PX.n the I/O pin is driven to a high level in Push-pull configuration and is driven to high impedance in open drain configuration.

Figure 9-6. Bidirectional Configuration

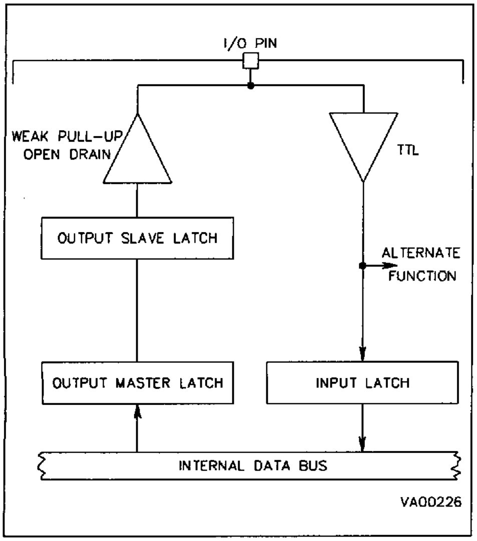
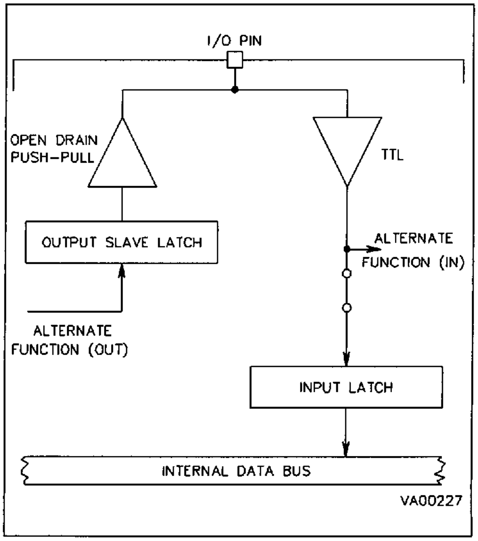


Figure 9-7. Alternate Function Configuration



9.4 ALTERNATE FUNCTION ARCHITECTURE

Each single I/O pin may access three different types of ST9 internal signals:

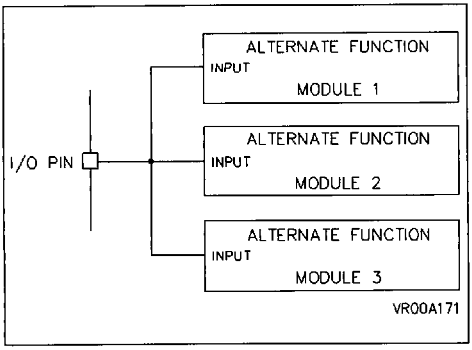
- Data bus line (I/O)
- 'Alternate Function' Input
- Alternate Function Output

Each pin configuration is made by software, thus allowing the User to choose the type of signal to access a pin. The choice of type of signal is made with the registers PXC2, PXC1, PXC0 of the I/O Port X (Please refer to the previous section for more details)

Pins Declared as an I/O

A pin declared as an I/O is a pin connected to the I/O buffer. In such a case, this pin may either be an Input or an Output or an I/O depending on the value stored in (PXC2, PXC1, PXC0)

Figure 9-8. Example of 3 Alternate Function Inputs



Pin Declared As An 'Alternate Function' Input

A single pin may be directly connected to several Alternate Function inputs. In such a case, the User has to select the required input mode (TTL or CMOS levels) and to enable, by software, the selected Alternate Function module (by enabling it) and unselect all other Alternate Functions (by disabling them).

No specific configuration of the port is required to enable the input Alternate Function, as the input buffer is directly connected to each module using it. As more than one module can use the same input Alternate Function line, it is under User software control to enable a module to use the input Alternate Function.

The digital I/O remains operational even when using the Alternate Function input. The exception to this is for an I/O port bit connected to analog voltages (for the Analog to Digital Converter).

Pin Declared As An Alternate Function Output

A pin declared as an Alternate function output corresponds to (PXC2,PXC1,PXC0) = 1,1,1 or 0,1,1. Several Alternate Function outputs may drive a common pin. In such a case, the Alternate Function output signals are ANDed before driving the common pin. The User has therefore to select, by software, the Alternate Function Output required by enabling it and disabling all other Alternate Function Outputs on the same pin (a disabled Alternate Function Output outputs a "1").

The inputs to on-chip Functions and Alternate Function Outputs are predefined for each I/O pin of an ST9. Please refer to the Alternate Function Table at the beginning of this datasheet for the exact configuration.

Figure 9-9. Example of 3 Alternate Function Inputs

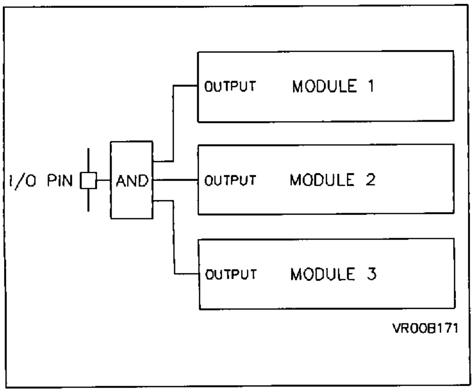
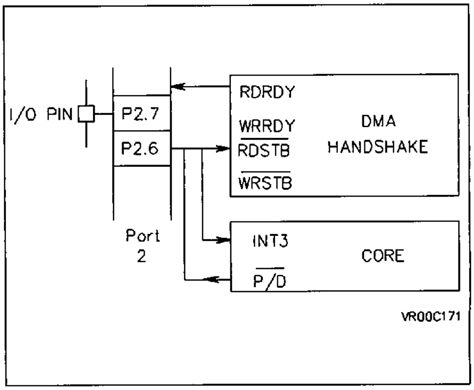


Figure 9-10. Example of One I/O Pin Configuration



ALTERNATE FUNCTION (Continued)

General Configuration
A single pin may be used, according to different phases of the software, as an I/O or connected to an input or an Alternate Function output. An example is given in Figure 9-10.

WARNING: *When a pin is connected to an Input Function and to an Alternate Function output, the User must be aware of the fact that the Alternate Function output signal always input to the Alternate Function module(s) declared as input(s).*
Figure 9-10 shows an example where the signal P/$\overline{D}$ also enters $\overline{RDSTB}$ and INT3.

9.5 I/O STATUS AFTER WFI, HALT AND RESET

The status of the ST9 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, however, 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.

WARNING: *I/O pins are set to the Weak Pull-up mode during the Reset cycle. This state is forced during the reset sequence, but the I/O pins can be in a random state for up to 64 crystal periods.*
The application circuit must take this into account if it can lead to critical situations in the external circuitry.

Mode	P0	P1, P6	I/O
WFI	High Impedance	Next Address	No Affect (clocks output from ST9 running)
HALT	High Impedance	Next Address	No Affect (clocks output from ST9 stopped)
RESET			Bidirectional Weak Pull-up

9.6 DEDICATED PIN FUNCTIONALITY

The following pins differ from the standard I/O functionality, in the mode specified in the table only. All other pins are as described in the previous sections

Dedicated Pin Functionality

Port/pin	Mode	Difference
P8.5	Input	Schmitt trigger input only
P8.3 P8.4 P5.0-P5.7	Output Open-drain	High Voltage output (to 12V)
P5.0-P5.7 P8.3 P8.4 P7.4 P7.6	Reset	High Impedance state
P2.0-P2.7	Reset (Bankswitch enabled)	Push-pull output

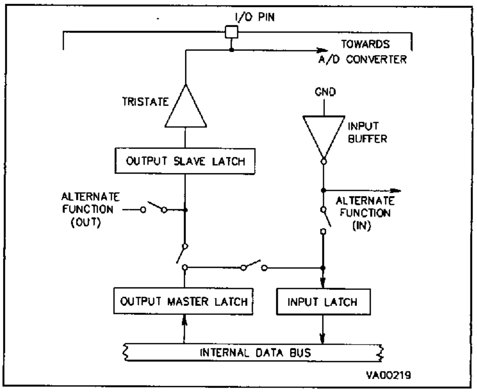
9.7 SPECIAL PORTS

9.7.1 Bit Structure For A/D Converter Inputs

When a port bit is used as input for an on-chip A/D Converter, its structure is modified as shown in Figure 9-11.

The behaviour of this bit is identical to the general purpose bit described in paragraph 9.9.3 except when it is programmed as Alternate Function. In this case, the Output Buffer is forced Tristate and the input of the Input Buffer is disconnected from the I/O pin and forced low. In this way the I/O pin is free to assume any analog value without causing power consumption in the Input Buffer. The bit **MUST** be programmed to **(PXC2, PXC1, PXC0) = 1,1,1)** to assume this special configuration.

Figure 9-11. A/D Input Port Bit Structure



7929237 0058017 69T

10 SERIAL PERIPHERAL INTERFACE

10.1 INTRODUCTION

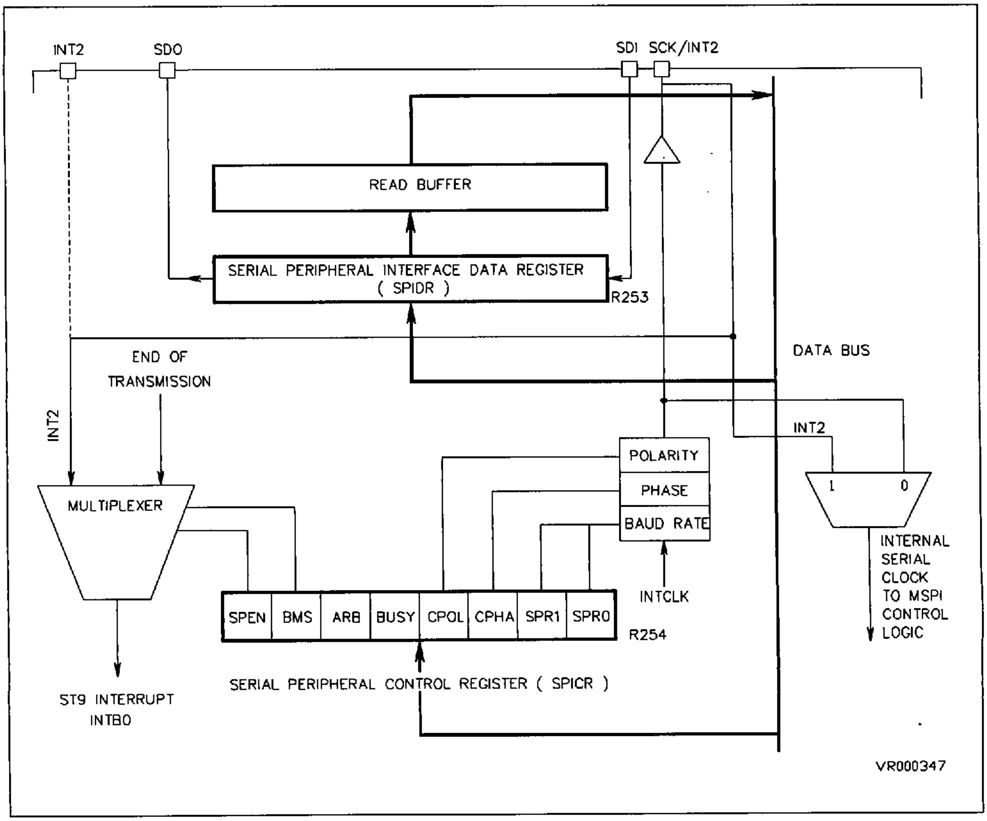
The Serial Peripheral Interface (SPI) is integrated into the Core module of the ST9 and provides a general purpose shift register based peripheral allowing several external peripherals to be linked through an SPI protocol bus. In addition, special modes allow reduced software overhead with I²C-bus and IM-bus Communication standards.

The SPI uses 3 lines comprising Serial Data In (SDI) and Alternate Function outputs Serial Data Out (SDO) and Synchronous Serial Clock (SCK). Additional I/O pins may act as device selects or IM-bus address ident signals.

Its Main Features are:

- Full duplex 3-wire synchronous transfer
- Master operation only
- 1.5MHz max bit transfer frequency (INTCLK = 12MHz)
- 4 Programmable bit rates
- Programmable clock polarity and phase
- Busy Flag
- End of transmission interrupt
- Additional hardware to facilitate more complex protocols

Figure 10-1. Block Diagram



10.2 FUNCTIONAL DESCRIPTION

The SPI, when enabled, receives input data from the ST9 Core internal data bus into SPIDR, and originates the Serial Clock (SCK) based upon dividing of the internal processor clock (INTCLK). The data is parallel loaded into the 8 bit shift register (from the internal bus) during a write cycle and then shifted out serially through the SDO pin (Most Significant bit first) to the slave device, which responds by sending its data to the master device via the SDI pin. This implies full duplex transmission with data-out and data-in both synchronized with the same clock signal. Thus the transmitted byte is replaced by the byte received, eliminating the need to have separate "Tx empty" and "Rx full" status bits.

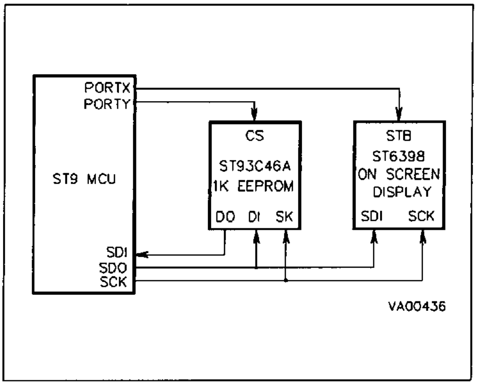
When the shift register is loaded, data is parallel transferred to the read buffer and data becomes available for the ST9 during a following read cycle.

The SPI requires three pins on an I/O port:

SCK	Serial Clock signal
SDO	Serial Data Out
SDI	Serial Data In

An additional output bit of an I/O port may be used to perform the slave chip select signal.

Figure 10-2. A Typical SPI Network



10.2.1 Input Signal Description

Serial Data In (SDI)

Data is transferred serially from a slave to a master on this line, most significant bit first. In an S-BUS/I^{2}C-bus configuration, SDI line senses the value forced on the data line (by SDO or by another peripheral connected to the S-bus/I^{2}C-bus environment).

10.2.2 Output Signal Description

Serial Data Out (SDO)

The SDO pin is configured as an output for the master device. This is obtained by programming the corresponding I/O pin as an output alternate function. Data is transferred serially from a master to a slave on SDO, most significant bit first. This pin is forced to the high impedance state when the SPI is disabled and is set to "1" when arbitration is lost (during an S-bus/I^{2}C-bus protocol transmission). The master device always allows data to be applied on the SDO line one half cycle before the clock edge in order to latch the data for the slave device.

Master Serial Clock (SCK)

The master device uses SCK to latch the incoming data on the SDI line. This pin is forced to a high impedance state when SPI is disabled (SPEN, SPICR.7 = "0"), in order to avoid clock contention from different masters in a multi-master system.
The master device generates SCK from INTCLK. SCK is used to synchronize the transfer of data both in and out of the device through its SDI and SDO pins. The SCK type and its relationship to data are controlled by the CPOL and CPHA bits in the Serial Peripheral Control Register.
This input is provided with a digital filter which cleans spikes lasting less than one INTCLK period.

Two bits (SPR1 and SPR0) in the Serial Peripheral Control Register, SPICR (R254) select the clock rate. Four frequencies can be selected, two in a high frequency range (mostly used with the SPI protocol) and two in a medium frequency range (mostly used for more complex protocols).

10.3 INTERRUPT STRUCTURE

SPI peripheral is associated with external interrupt channel B0 (pin INT2). Multiplexing between the external pin and SPI internal source is controlled by the SPEN and BMS bits according to the following table.

The two possible SPI interrupt sources are: End of transmission (after each byte) and S-bus/I²C-bus start condition. Care should be taken when toggling SPEN or/and BMS bits from (0,0) status, this should be done by masking the interrupt channel B0 (reset of EIMR.IMB0, bit 2 of External Interrupt Mask Register). Furthermore it is necessary to clear possible spurious requests on the corresponding channel by resetting the interrupt pending bit EIPR.IPB0 (bit 2 of External Interrupts Pending Register).

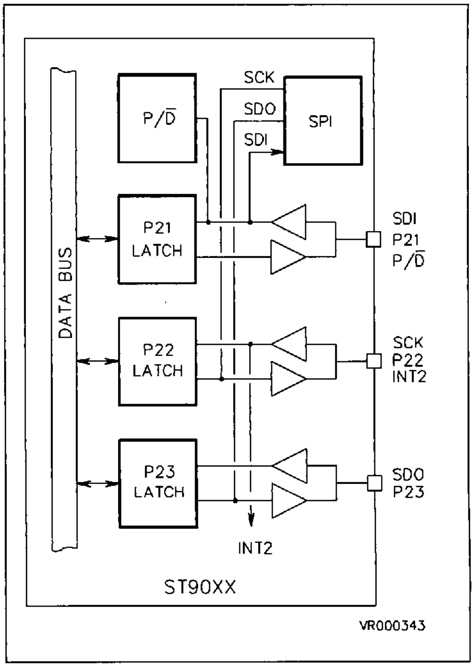
The INT2 input Function is always mapped together with the SCK input Function to allow start/stop bit detection when using S-bus/I²C-bus protocols.

A delay instruction (e.g. a NOP instruction) should be inserted between the SPEN toggle instruction and the interrupt pending bit reset instruction.

Table 10-1. Interrupt Configuration

SPEN	BMS	Interrupt Source
0	0	External channel INT2
0	1	S-bus/I^{2}C bus start or stop condition
1	X	End of one byte transmission

Figure 10-3. SPI I/O Pins



10.4 SPI REGISTERS

SPI uses two registers mapped on page 0 of the register file:

SPIDR R253 (FDh) Page 0 Read/Write
SPI Data Register

Reset Value: 0000 0000b (00h)

7							0
D7	D6	D5	D4	D3	D2	D1	D0

b7-b0 = **D0-D7:** *SPI Data Bits.* This register contains the data transmitted and received by the SPI. Data is transmitted b7 first, and receives incoming data into b0. Transmission is started by writing to this register.

SPICR R254 (FEh) Page 0 Read/Write
SPI Control Register

Reset Value: 0000 0000b (00h)

7							0
SPEN	BMS	ARB	BUSY	CPOL	CPHA	SPR1	SPR0

b7 = **SPEN:** *Serial Peripheral Enable.* When set, the two alternate functions SCK and SDO are enabled. When disabled, SCK and SDO are kept in high impedance. Furthermore, SPEN affects the selection of the source for interrupt channel B0. Transmission will start by simply writing the data into the SPIDR Register.

b6 = **BMS:** *S-bus/I²C-bus Mode Selector.* This bit should be set to "1" when the SPI is used in an S-bus/I²C-bus protocol. It enables S-bus/I²C-bus arbitration, clock synchronization and Start/ Stop detection.
When this bit is reset to "0", a reinitialisation of the SPI logic is performed allowing recovery procedures after a Rx/Tx failure. BMS (and SPEN) affects the selection of the source for interrupt channel B0.

b5 = **ARB:** *Arbitration flag bit* This bit is set when the SPI, in S-bus/I²C-bus mode, loses arbitration, and is reset when an S-bus/I²C-bus stop condition is detected. ARB can be reset by software. When ARB is set automatically, the SDO pin is set to high value until a write instruction on SPIDR is performed.

b4 = **BUSY:** *SPI Busy Flag.* BUSY flag is set when a transmission is in process. This bit allows the user to monitor the SPI status by polling its value.

b3 = **CPOL:***Transmission Clock Polarity.* CPOL controls the normal or steady state value of the clock when data is not being transferred.

As the SCK line is held in a high impedance state when the SPI is disabled (SPEN = "0"), the SCK pin must be connected to V_{SS} or V_{CC} through a resistor according to the CPOL state. Polarity should be selected during the reset routine according to the value set into all peripherals and must not be changed during program execution.

b2 = **CPHA:** *Transmission Clock Phase.* CPHA controls the relationship between the data on the SDI and SDO pins and the clock produced at the SCK pin. CPHA bit selects the clock edge which captures data and allows it to change state. It has its greatest impact on the first bit transmitted (MSB) because it does (or does not) allow a clock transition before the first data capture edge.
Figure 10-5 shows the relationship between CPHA, CPOL and SCK, and indicates active clock edges and strobe times.

CPOL	**CPHA**	**SCK** on Figure 10-5
0	0	(a)
0	1	(b)
1	0	(c)
1	1	(d)

b1-b0 = **SPR1,SPR0:** *SPI Rate.* These two bits select one (out of four) baud rates to be used as SCK.

SPR1	**SPR0**	**Clock Divider**	**SCK Frequency (INTCLK = 12MHz)**	
0	0	8	1500kHz	(T = 0.67µs)
0	1	16	750kHz	(T = 1.33µs)
1	0	128	93.75kHz	(T = 10.66µs)
1	1	256	46.87kHz	(T = 21.33µs)

The data on the SDA line is sampled with the low to high transition on the SCL line.

10.5 WORKING with DIFFERENT PROTOCOLS

The SPI peripheral offers the following facilities to work with S-bus/I^{2}C-bus and IM-bus protocols:

- Interrupt request on start/stop detection
- Hardware clock synchronisation
- Arbitration lost flag with an automatic set of data line

Note that the I/O bit associated to the SPI should be returned to a defined state as a normal I/O pin before changing the SPI protocol.

The following paragraphs provide information to manage these protocols.

10.5.1 I^{2}C-bus Interface

I^{2}C-bus is a two-wire bidirectional data-bus, the two lines being SDA (Serial DAta) and SCL (Serial CLock). Both are open drain lines to allow arbitration. As shown in figure 10-6, data is toggled with clock low and Start and Stop conditions are detected when a high to low (start) or a low to high (stop) transition on the SDA line occurs with the SCL line high.

Each transmission consists of nine clock pulses (SCL line). The first 8 pulses transmit the byte (msb first), the ninth is used by the receiver to acknowledge.

Figure 10-4. S-Bus/I^{2}C-bus Peripheral Compatibility without S-Bus Chip Select

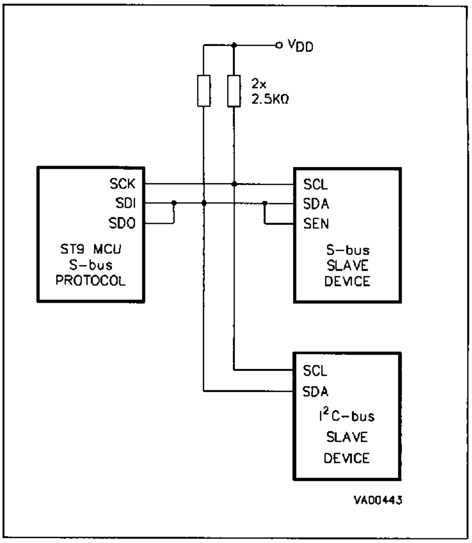
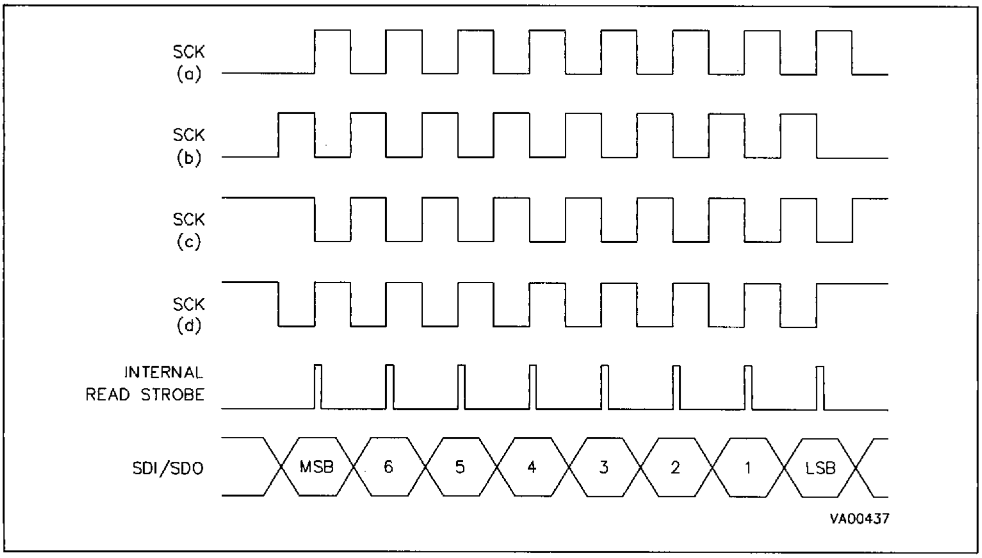


Figure 10-5. SPI Data and Clock Timing



DIFFERENT PROTOCOLS (Continued)

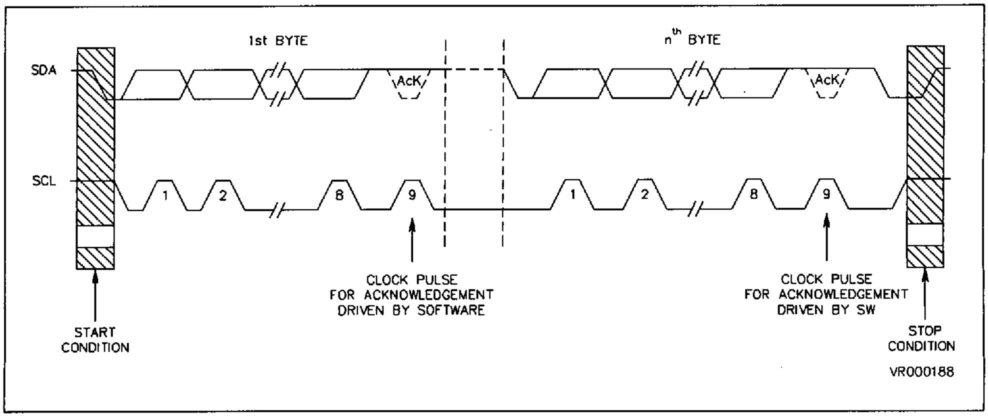
Table 10-2. Typical I²C-bus Sequences

Phase	Software	Hardware	Notes
INITIALIZE	SPICR.CPOL, CPHA = 0, 0 SPICR.SPEN = 0 SPICR.BMS = 1 SCK pin set as AF output SDI pin set as input Set SDO port bit to 1	SCK, SDO IN HI-Z SCL, SDA = 1, 1	Set polarity and phase SPI disable START/STOP interrupt Enable
START	SDO pin set as output Open Drain Set SDO port bit to 0	SDA = 0, SCL = 1 interrupt request	START condition receiver START detection
TRANSMISSION	SPICR.SPEN = 1 SDO pin as Alternate Function output load data into SPIDR	SCL = 0 Start transmission interrupt request	Managed by interrupt routine load FFh when receiving end of transmission detection

ACKNOWLEDGE	SPICR.SPEN = 0
	Poll SDA line
	Set SDA line
	SPICR.SPEN = 1
		SCK, SDO in HI-Z
		SCL, SDA = 1
		
		SCL = 0
			SPI disable
			only if transmitting
			only if receiving
			only if transmitting

STOP	SDO pin set as output Open Drain SPICR.SPEN = 0 Set SDO port bit to 1	SDA = 1 interrupt request	STOP condition

Figure 10-6. SPI Data and Clock Timing



1st BYTE
n th BYTE
SDA
SCL
AcK
AcK
1
2
8
9
1
2
8
9
CLOCK PULSE FOR ACKNOWLEDGEMENT DRIVEN BY SOFTWARE
CLOCK PULSE FOR ACKNOWLEDGEMENT DRIVEN BY SW
START CONDITION
STOP CONDITION
VR000188

DIFFERENT PROTOCOLS (Continued)

SPI Working With I²C-bus

To use the SPI with the I²C-bus protocol, the SCK line is used as SCL, the SDI and SDO lines, externally wired-OR'd, are used as SDA. All the output pins must be configured as open drain (see Figure 10-6).

When programmed to the I²C-bus protocol, the SPI functions up to a data baud rate of 750kHz (SPR1=0, SPR0=1) for f_{OSC}=12MHz

Table 10-2 shows the typical I²C-bus sequence divided in 5 phases: initialize, start, transmission, acknowledge and stop. Software and hardware will take care of each phase. According to this example, a master to slave transmission can be managed.

During the transmission phase, the following I²C-bus features are also supported by hardware.

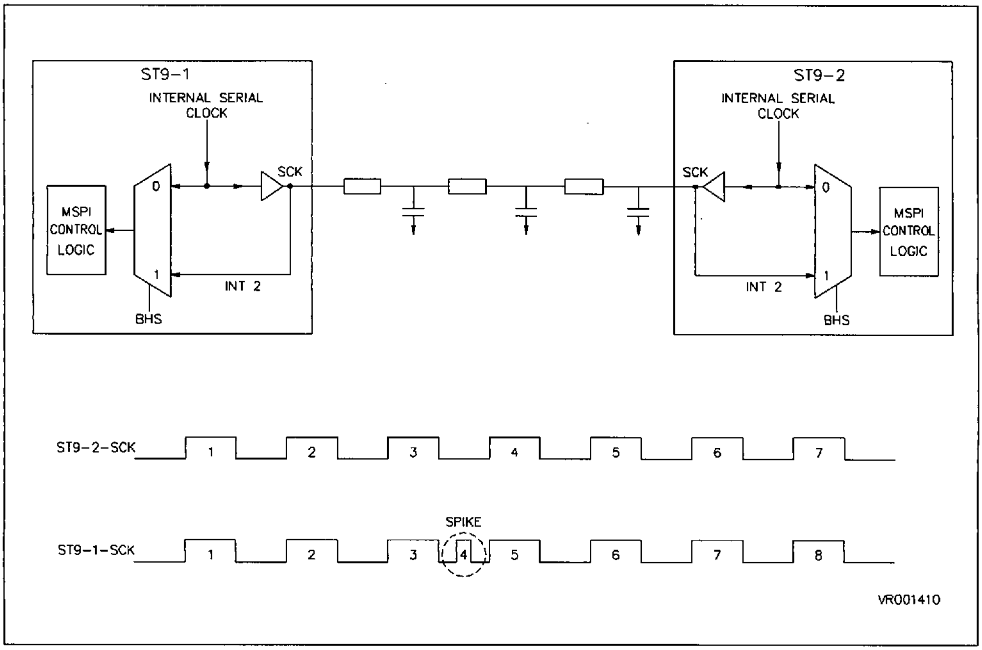
Clock Synchronization

In a multimaster I²C-bus system, when more masters generate their own clock, synchronization is needed. The first master which releases the SCL line stops internal counting, restarting only when the SCL line goes high (released by all the other masters). In this way, devices using different clock sources and different frequencies can be interfaced.

Arbitration Lost

When more masters are sending data on SDA line, the following mechanism is performed: if the transmitter sends a "1" and SDA line is forced low by another device the ARB flag (SPICR.5) is set and the SDO buffer is "switched off. (ARB is reset and SDO buffer is "switched on" when SPIDR is written to again). When BMS is set to "1" the peripheral clock is supplied through the INT2 line by the external clock line (SCL). Due to potential noise spikes (which must last longer than one INTCLK period to be detected), RX or TX may gain a clock pulse. Referring to Figure 10-7, if ST9-1 detects a noise spike and gains a clock pulse, it will stop its transmission in advance and hold the clock line low causing ST9-2 to be frozen at the 7th bit. To exit and recover from this condition the BMS bit must be reset to "0", this will cause the reset of the SPI logic, aborting the current transmission. An End of Transmission interrupt is generated after this reset sequence.

Figure 10-7. SPI Arbitration



DIFFERENT PROTOCOLS (Continued)

10.5.2 S-Bus Interface

S-bus is a three-wire bidirectional data-bus, with functional features similar to I²C-bus. Differently from I²C-bus, the START/STOP conditions are given by encoding the information on 3 wires instead of 2, as shown in Figure 10-8. The additional line is referred as SEN.

SPI Working With S-bus

The S-bus protocol uses the same pin configuration as I²C-bus for generating the SCL and SDA lines. The additional SEN line is managed through a standard ST9 I/O port under software control (see Figure 10-9).

Figure 10-8. Mixed S-bus and I²C-bus system

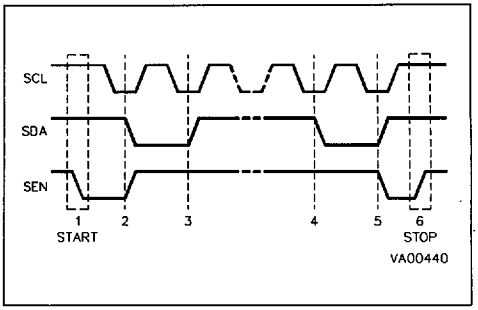


Figure 10-9. S-bus Configuration

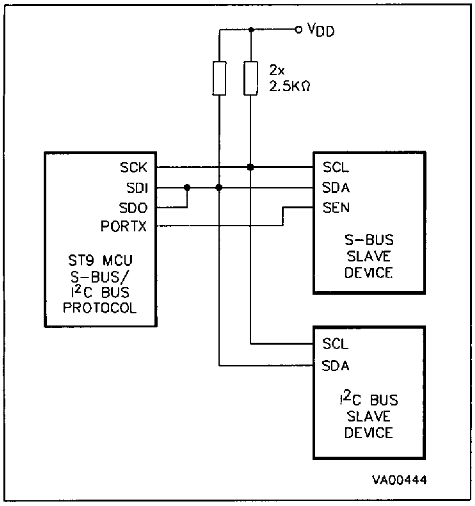
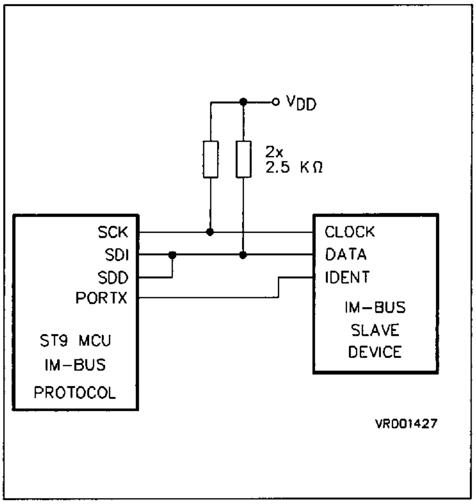


Figure 10-10. ST9 and InterMetal Peripheral



DIFFERENT PROTOCOLS (Continued)

10.5.3 IM-Bus Interface

The IM-bus has a bidirectional data line and a clock line, and in addition it requires an IDENT line that distinguishes an address from a data byte (Figure 10-11). Unlike the I²C-bus protocol, the IM-bus protocol sends the least significant bit first, this requires a software routine which reverses the bit order before sending, and after receiving a data byte. Figure 10-10 shows the connections for an IM-bus peripheral to an ST9 SPI. The SDO and SDI pins are connected to the bidirectional data pin of the peripheral device. The SDO alternate function is set in Open Drain (external 2.5KΩ pull-up resistors are required).

With this type of configuration, data is sent to the peripheral by writing the data byte to SPIDR. To receive data from the peripheral, the User should write FFh into SPIDR in order to generate the shift clock pulses. As the SDO line is set to the Open Drain configuration, the incoming data bits that are set to one do not affect the SDO/SDI line status (which defaults to a high level due to the FFh in the transmit register), while incoming bits that are set to "0" pull the input line low.

In software it is necessary to initialise the ST9 SPI with CPOL and CPHA set to "1", "1". By using a general purpose I/O as the IDENT line and forcing it to a logical "0" when writing to SPIDR, an address is sent (or read). Then, by setting this bit to a logical "1" and writing to SPIDR, data is sent to the peripheral. When all the address and data pairs are sent it is necessary to drive the IDENT line low and high to create a short pulse. In this way the stop condition is generated.

10.5.4 Mirror Register

This register provides bit reversal facilities for interfaces requiring the least significant bit of data to be the first presented.

MIRRG R241 (F1h) Page 0 Read/Write
Mirror Register

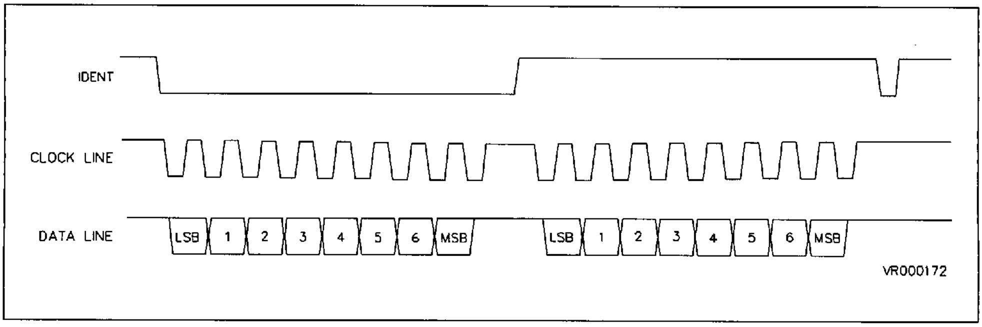
Reset Value: undefined

7							0
MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0

b7-b0 = **MR0-MR7:** *Mirrored Data Bits.* This register returns the mirroredimage of the data written to it.

Thus, writing 80h into MIRRG will return 01h when reading on next cycle.

Figure 10-11. IM bus Timing



11 TIMER/WATCHDOG

11.1 INTRODUCTION

A programmable 16-bit down counter with an 8-bit prescaler is included in the ST9 Core. This Timer can be programmed to be used as a general purpose 16-bit Timer, with associated input and output pins for timing functions, or as a Watchdog Timer offering security against possible processor malfunctions due to hardware or software failures.

The Timer/Watchdog functions can use inputs from an external pin and an Alternate Function output of an I/O Port. The Input pin can be used in one of the four programmable input modes:

- event counter,
- gated external input mode,
- triggerable input mode,
- retriggerable input mode.

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

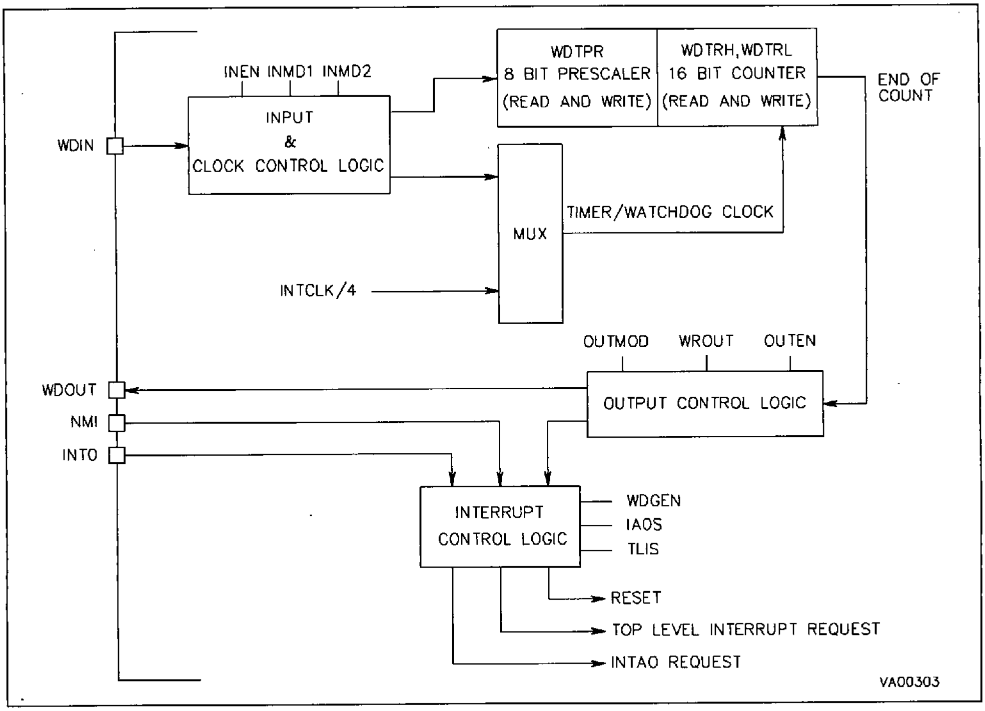
An interrupt generated by the unit (when running as a 16-bit Timer/counter and not as Watchdog) can be used as a Top Level Interrupt or as an interrupt source connected to channel A0 of the external interrupt structure (replacing the INT1 interrupt input).

The clock for the counter can be driven either by an external clock or an internal clock equal to INTCLK divided by 4.
When using an external 24MHz crystal (INTCLK = 12MHz), the End Of Count rate is:

5.59 sec. for Max. Count (Timer Const. = FFFFh, Prescaler Const. = FFh)

333 nsec. for Min. Count (Timer Const. = 0000h, Prescaler Const. = 00h)

Figure 11-1. Block Diagram



11.2 FUNCTIONAL DESCRIPTION

11.2.1 Timer/Counter Input Modes

Setting the Input Enable (INEN) bit enables the input mode which is selected via the INMD1 and INMD2 bits. When INEN is reset to zero, the input section is disabled and the values of INMD1 and INMD2 are don't-care.

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

The Timer is driven by the signal applied to the input pin which acts as an external clock. The unit works therefore as an event counter. The event is a high to low transition of the input signal.
Spacing between trailing edges should be at least 350ns (i.e. the maximum Watchdog Timer input frequency is 2.9MHz with INTCLK = 12MHz).

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

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

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

A Timer/Watchdog start is caused by:
a) a set of the Start-Stop bit, or
b) a High to Low (low trigger) transition on the input pin.

In order to stop the Timer, it is only necessary to reset the Start-Stop bit to zero.

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

In this mode when the Timer is running (TIMER/WATCHDOG internal clock), a High to Low transition of the input pin causes the counting to start from the initial value. When the Timer is stopped (ST_SP bit equal to zero), a High to Low transition of the input pin has no effect.

11.2.2 Timer/Watchdog Output Modes

OUTPUT modes are selected using 2 bits of WDTCR (R251): OUTEN (Output Enable) and OUTMD (Output Mode).

When OUTMD = "0", the Timer outputs a signal with a frequency equal to half the End Of Count repetition rate. With INTCLK = 12MHz, this allows generation of a square wave wth a period ranging from 666ns to 11.18 seconds.

The value of the WROUT bit is transferred to the output pin at the End Of Count and the value is held until the next End of Count when OUTMD = "1". This allows the user to generate PWM signals, by modifying the status of WROUT between End of Count events, based on software counters decremented on the Timer/Watchdog interrupt.

OUTEN = "1" enables the output function selected via OUTMD

When OUTEN = "0", the output is disabled and the output pin is held at a "1" level to allow several alternate functions on the same pin.

11.2.3 Timer/Counter Control

Start/Stop

ST_SP (WDTCR.7) enables down-counting. An instruction which sets this bit will cause the Timer to start at the beginning of the following instruction. Resetting this bit will stop 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. A new constant can be written with the counter running. The new value will be loaded at the following End Of Count (EOC).

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

Single/Continuous Mode

SINGLE MODE: At End Of Count the Timer stops, reloads the constant, and resets the Start/Stop bit (WDTCR.6) (user may check the current status by reading this bit). Restarting is done by setting the Start/Stop bit. Note that the Timer constant is reloaded only if it has been modified during the stop period.

CONTINUOUS MODE: At End Of Count the counter automatically reloads the constant and restarts. It is stopped only if the Start/Stop bit is reset. 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.

FUNCTIONAL DESCRIPTION (Continued)

11.2.4 Timer/Watchdog Mode

In this mode (WDGEN = "0") the counter generates a fixed time basis. When End Of Count is reached the Timer generates a system Reset.

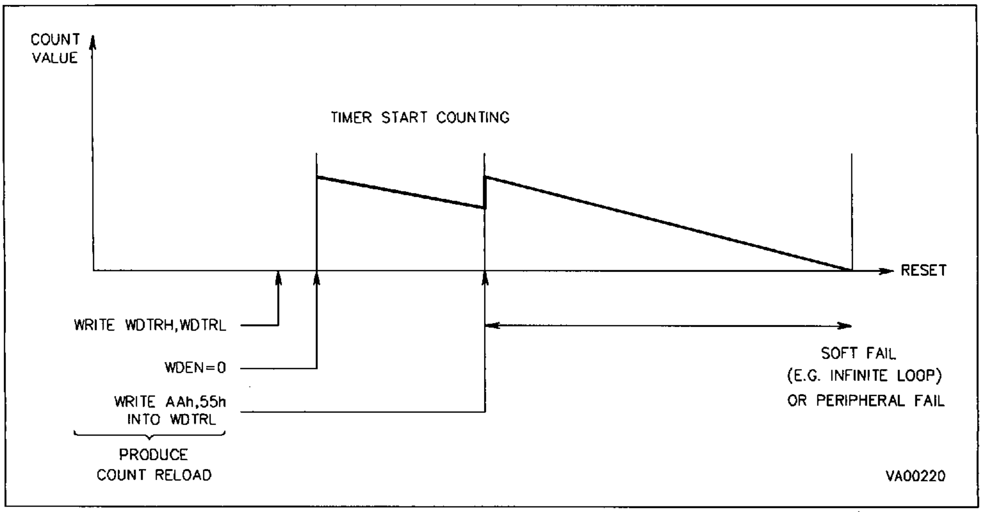
The time base is user-defined and must be written in the Timer registers before entering Watchdog mode. In Watchdog mode it is possible to modify only the Prescaler Constant. This new value will be loaded when the counter restarts.
Resetting WDGEN (bit 6 of the Wait Control Register) causes the counter to start regardless of the value of the Start-Stop. In order to prevent a system reset the sequence AAh, 55h should be entered in WDTLR (Watchdog Timer register low). Once the writing of 55h has been performed the Timer reloads the constant and counting restarts from the preset value.

The minimum time between the writing of the AAh and 55h codes is zero, i.e. the writing is sequential, and the maximum time is given by the Watchdog timeout period.

In Watchdog-mode a `halt` instruction is regarded as illegal. Execution of the `halt` instruction stops further core execution by the CPU and interrupt acknowledgment, but does not stop INTCLK or CPUCLK or the Watchdog Timer, which will cause a System Reset when reaching the End of Count. Furthermore ST_SP, S_C and input mode selection bits are "don't-care". 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 that particular mode of operation is meaningless.

Figure 11-2. Timer /Watchdog in Watchdog Mode



11.3 TIMER/WATCHDOG INTERRUPT

When enabled, the Timer/Watchdog will issue an interrupt request at every End Of Count.

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 (the Timer/Watchdog End of Count or an external pin) 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.

In the Watchdog mode the End Of Count always causes a system reset.

A block diagram of the interrupt logic is given in Figure 11-3 (Note: software traps can begenerated by setting the appropriate interruptpending bit):

The following table shows all the possible configurations of the interrupt/reset sources which involve the Timer/Watchdog:

Figure 11-3. Interrupt Sources

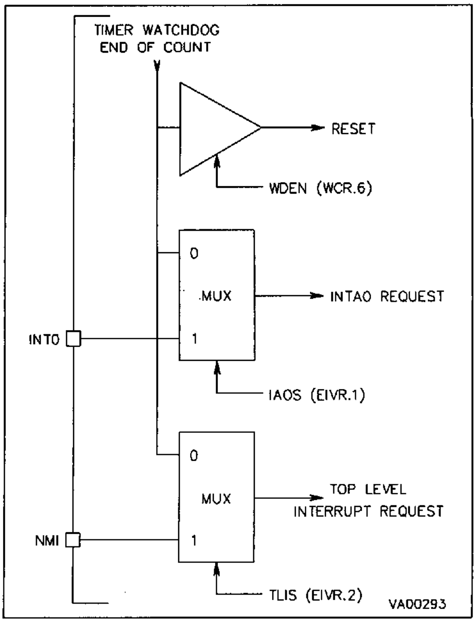


Table 11-1. Interrupt Configuration

Control Bits			Enabled Sources			Watchdog Timer Status
WDGEN	IAOS	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

Note.
WDG = Watchdog function
SW TRAP = Software Trap

11.4 TIMER/WATCHDOG REGISTERS

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

WDTHR (R248): Timer/Watchdog Counter High Register
WDTLR (R249): Timer/Watchdog Counter Low Register
WDTPR (R250): Timer/Watchdog Prescaler Register
WDTCR (R251): Timer/Watchdog Control Register

Three additional control bits are mapped in the following registers of 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 Registers

This 16 bit register is used to load the 16 bit counter value. The registers can be read or written "on the fly".

WDTHR R248 (F8h) Page 0 Read/Write
Timer/Watchdog Counter Register, High byte

Reset value: undefined

7							0
R15	R14	R13	R12	R11	R10	R9	R8

WDTLR R249 (F9h) Page 0 Read/Write
Timer/Watchdog Counter Register, Low byte.

Reset value: undefined

7							0
R7	R6	R5	R4	R3	R2	R1	R0

WDTPR R250 (FAh) Page 0 Read/Write
Timer/Watchdog Prescaler Register

Reset value: undefined

7							0
PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0

b7-b0 = **PR7-PR0:** *Timer/Watchdog Prescaler.* The value stored in this Register is used to select the prescaling factor from 1 (loading 00h) to 256 (loading FFh).

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

WDTCR R251 (FBh) Page 0 Read/Write
Timer/Watchdog Control Register

Reset value: 0001 0010 (12h)

7							0
ST_SP	S_C	INMD1	INMD2	INEN	OUTMD	WROUT	OUTEN

b7 = **ST_SP:** *Start/Stop Bit.* Setting this bit to a "1" starts the counting operation (see Warning above). When this bit is "0", the counter is stopped (reset status)

b6 = **S_C:** *Single/Continuous.* When this bit is set, the counter operates in Single Count Mode. Continuous Mode is set when this bit is "0"

b5-b4 = **INMD1, INMD2:** *Input mode selection bits.*

b3 = **INEN:** *Input Enable.* This bit enables ("1") and disables ("0") the input section

b2 = **OUTMD:** *Output Mode.* When this bit is "1", and the output is enabled, the value of WROUT is transferred to the output pin on every End Of Count. When "0", the output is toggled on every End of Count

b1 = **WROUT:** *WROUT bit.* The status of this bit is transferred to the Output pin when OUTMD = "1", it is user definable to allow PWM output (at reset WROUT = "1")

b0 = **OUTEN:** *Output Enable bit.* The output is enabled by setting this bit to "1", and disabled by resetting to "0"

TIMER/WATCHDOG REGISTERS (Continued)

WCR R252 (FCh) Page 0 Read/Write

Wait Control Register

Reset value: 0111 1111 (7Fh)

7							0
X	WDGEN	X	X	X	X	X	X

b6 = **WDGEN:** *Watchdog Enable Bit (active low).* Resetting this bit to zero via software enters the Watchdog mode. Once reset, it cannot be set to "1" by the user program. At system reset, the Watchdog mode is disabled

EIVR R246 (F6h) Page 0 Read/Write

External Interrupt Vector Register

Reset value: xxxx 0110 (X6h)

7							0
X	X	X	X	X	TLIS	IAOS	X

b2 = **TLIS:** *Top Level Input Selection bit.* This bit selects the Top Level interrupt source. When "0", the Top Level interrupt source is the Watchdog/Timer end of count, when = "1", it is the external pin NMI.

b1 = **IAOS:** *Interrupt channel A0 Selection Bit.* This bit allows the Timer/Watchdog interrupt to channel through the external Interrupt A0 source, allowing the setting of user-defined priority levels.

WARNING. *To avoid spurious interrupt requests, an access to the IAOS bit must be made only when the interrupt logic is disabled (i.e. after the DI instruction). It is also necessary to clear a possible interrupt pending request 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 IAOS write instruction.*

12 MULTIFUNCTION TIMER

12.1 INTRODUCTION

The Multifunction Timer is a 16-bit Up/Down counter, driven by the output of an 8-bit prescaler which may be driven by INTCLK/3 (giving a minimum timing resolution of 250ns at INTCLK = 12 MHz) or by an external source.
This timer is supported by two 16-bit Comparison Registers (CMP0R, CMP1R) for generating timed functions and two 16-bit Capture/(re)Load Registers (REG0R, REG1R) for timing and variable timebase functions. These features coupled with 2 input pins (TxINA and TxINB) and 2 Alternate Function output pins (TxOUTA and TxOUTB), where x = the number of the Timer, give the Timer 12 operating modes including automatic PWM generation and frequency measurement.

Several functional configurations are possible, e.g.:

- 2 input captures on two different external lines and 2 independent output compare functions (counter in free running mode), or 1 output compare on a fixed repetition rate.
- 1 input capture, 1 counter reload and 2 independent output compares.
- 2 alternate autoreloads and 2 independent output compares.
- 2 alternate captures on the same external line and 2 independent output compares on a fixed repetition rate.

When two timers are present on ST9 chip, a combined mode is available.

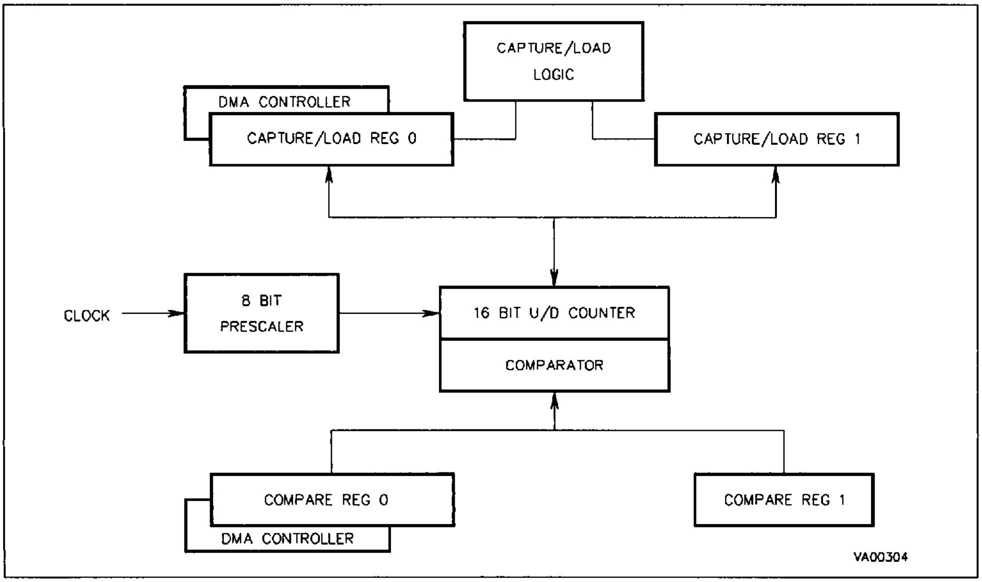
Four internal signals are also available for timing of on-chip functions: the On Chip Event signal can be used to control other peripherals on the chip itself, and 3 other signals which can be internally connected to I/O port(s) in order to allow automatic, timed, DMA transfers.

The two external inputs (TxINA/TxINB) of the timer can be individually programmed to catch a particular external configuration, i.e.:

- rising edge
- falling edge
- rising and falling edges

The configuration of each input is fixed by the Input Control Register (ICR).

Figure 12-1. MFT Simplified Block Diagram



INTRODUCTION (Continued)

Each of the two output pins (TxOUTA/TxOUTB) can be driven from any of three possible sources:

- Compare Register 0 logic
- Compare Register 1 logic
- Overflow/Underflow logic

Each of these three sources can cause one of the following four effects, independently, on each of the two outputs:

- Nop
- Set
- Reset
- Toggle

Furthermore an additional on-chip Event signal can be generated by two of the three sources mentioned above, i.e. Over/Underflow event and Compare 0 event. This signal can be used internally as synchronism for another on-chip peripheral or as strobe for an I/O port (see I/O port chapter).

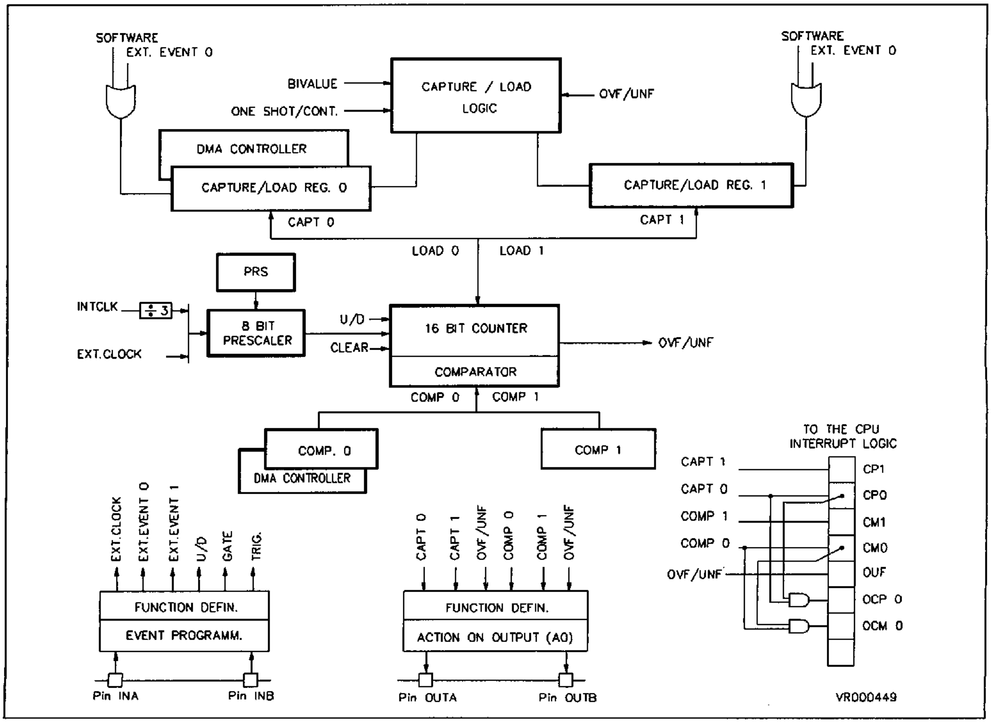
Five maskable interrupt sources referring to an End Of Count condition, 2 input captures and 2 output compares, can generate 3 different interrupt requests (with hardware fixed priority), pointing to 3 interrupt routine vectors.

Two independent DMA channels are available for a MFtimer and can be used for quick data flow operations. Each DMA request (associated to a capture on REG0R register, or a compare on CMP0R register) has priority on the INT request generated by the same source.

Each DMA channel can be employed in external transfers to/from memory from/to an I/O port using three internal lines (one for setting the data flow direction, and two for the transfer synchronization).

A SWAP mode is also available to allow high speed continuous transfers (see Interrupt and DMA chapter).

Figure 12-2. Detailed Block Diagram



12.2 FUNCTIONAL DESCRIPTION

The operating modes of the timer can be selected by programming the Timer Control Register (TCR) and the Timer Mode Register (TMR).

12.2.1 One Shot Mode

When the counter generates an overflow (in up-count mode) or an underflow (in down-count mode), i.e. an End Of Count is reached, the counter stops and no counter reload occurs. The counter can be restarted only by an external or software trigger. The One Shot Mode is entered by setting TMR bit CO.

12.2.2 Continuous Mode

Whenever the counter reaches an End Of Count, the counting sequence is automatically restarted and the counter is reloaded from REG0R (or REG1R when selected in Biload Mode). Continuous Mode is entered by resetting TMR bit CO.

12.2.3 Trigger And Retrigger Modes

A trigger event may be generated either by software action (setting either CP0 or CP1 bit in timer register FLAGR), or by an external source which may be programmed to be active on the rising edge, the falling edge or both, using the fields A0-A1 and B0-B1 in ICR.

In One Shot and Trigger Mode, every trigger event (used as a reload and start count) arriving before an End Of Count, is masked. In One Shot and Retrigger Mode, every trigger (used as a reload and start count) received while the counter is running automatically reloads the counter from REG0R (or REG1R when the register is selected in Biload Mode). Trigger/Retrigger Mode is set by the REN bit in TMR.

TxINA input refers to REG0R and TxINB input refers to REG1R.

WARNING. *If the Trigger Mode is selected when the counter is in Continuous Mode, then every trigger to reload the counter starting value is disabled, so it is not possible to synchronize the counting cycle by hardware or software.*

12.2.4 Gate Mode

In this mode the counting operation is performed only when the external gate input is active (logical state "0"). The selection of TxINA or TxINB input as gate input is made through IN0-IN3 bits in ICR.

12.2.5 Capture Mode

REG0R and REG1R registers may be independently set in Capture Mode by setting RM0 or RM1 in TMR, so that a capture of the current count value can be performed either on REG0R or REG1R, via software action (by setting CP0 or CP1 in the FLAGR register) or a programmable event on the external input pins.

WARNING. *Care should be taken when two software captures have to be performed on the same register. In this case, at least one extra instruction must be present between the first CP0/CP1 bit set and the subsequent CP0/CP1 bit reset.*

12.2.6 Up/Down Mode

The counter can count up or down depending on the state of the UDC bit (Software Up/Down) in TCR, or on the configuration of the external input pins, which have priority over UDC (see Input pin assignment in ICR). When read, the UDCS bit always returns the counter up/down current status (see also the Up/Down Autodiscrimination mode in the Input Pin Assignment Section).

12.2.7 Free Running Mode

The timer performs full range counting (in up or down mode) without reloading from REG0R at an End Of Count. This mode is automatically selected either in Bicapture Mode or by setting REG0R for capture function (Continuous Mode must also be set). In Autoclear Mode, free running with modulo less than 2^{16} may be obtained (see Autoclear Mode).

FUNCTIONAL DESCRIPTION (Continued)

12.2.8 Monitor Mode

When RM1 bit in TMR is reset and the timer is not in Bivalue Mode, then REG1R acts as monitor, reproducing the current U/D counter content enabling the ST9 to read the counter "on the fly".

12.2.9 Autoclear Mode

A clear command forces the counter to the value 0000h or 0FFFFh, when counting in up or down count mode respectively. The counter reset may be obtained either directly, through CCL bit in TCR, or by entering the Autoclear Mode, through CCP0 and CCMP0 fields in TCR.

Every capture performed on REG0R (if CCP0 = "1"), or every successful compare performed by CMP0R (if CCMP0 = "1"), clears the counter and reloads the prescaler.

The Clear On Capture mode allows the direct measurement of delta time between successive captures on REG0R, while the Clear On Compare mode allows free running with modulo less than 2^{16}.

12.2.10 Bivalue Mode

Depending on the value of RM0 bit in TMR, the Biload Mode (RM0 = "0") or the Bicapture Mode (RM0 = "1") can be selected as explained in the following table:

Table 12-1. Bivalues Modes

TMR bits			Timer Operating Modes
RM0	RM1	BM	
0	X	1	BiLoad mode
1	X	1	BiCapture Mode

A) Biload Mode

The Biload Mode is entered by selecting the Bivalue Mode (BM = "1" in TMR) and programming REG0R as a reload register (RM0 = "0" in TMR).

At any End Of Count, the counter reloading is performed alternately from REG0R and REG1R, (a low level for BM bit always sets REG0R as the current register, so that, after a Low to High transition of BM bit, the first reload is always from REG0R).

Every software or external trigger event on REG0R performs a reload from REG0R resetting the Biload cycle. In One Shot mode (reload made by a software or external trigger), the reload is always from REG0R.

B) Bicapture Mode

The Bicapture Mode is entered selecting the Bivalue Mode (BM = "1" in TMR) and programming REG0R as a capture register (RM0 = "1" in TMR).

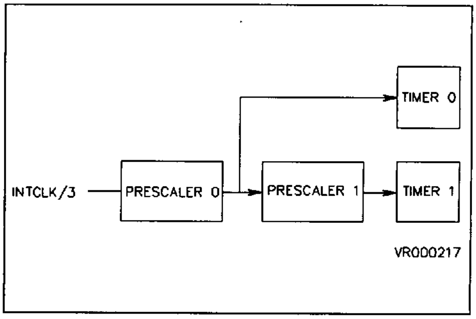
Every capture event, software simulated (by setting CP0 flag) or from the TxINA input line, captures the current counter value alternately into REG0R and REG1R. A low level for BM bit always sets REG0R as current register, so that the first capture, after setting BM bit, is always into REG0R.

12.2.11 Parallel Mode

When there are two timers on ST9 chip, the parallel mode is entered with ECK ="1" in TMR of Timer 1. Timer 1 prescaler input is internally connected to the Timer 0 prescaler output. Timer 0 prescaler input is connected to the system clock line.

By loading the Prescaler Register of Timer 1 with the value 00h the two timers (Timer 0 and Timer 1) are driven by the same frequency in parallel mode.

Figure 12-3. Parallel Mode Description



12.2.12 Autodiscriminator Mode

The phase difference sign of two overlapped pulses (respectively on TxINB and TxINA) generates a one step up(down) count, so that the up/down control and the counter clock are both external. The setting of the UDC bit in the TCR register has no effect in this configuration.

12.3 INPUT PIN ASSIGNMENT

The two external inputs (TxINA and TxINB) of the timer can be individually configured to catch a particular external event (i.e. rising edge, falling edge, rising and falling edges) by programming the two relevant bits (A0, A1 and B0, B1) for each input in the external Input Control Register (ICR).

The 16 different functional modes of the two external inputs can be selected by programming IN0 - IN3 bits of the ICR as explained in the following table.

Table 12-2. Input Pin Function

I C Reg. IN3-IN0 bits	TxINA Input Function	TxINB Input Function
0000	not used	not used
0001	not used	Trigger
0010	Gate	not used
0011	Gate	Trigger
0100	not used	Ext. Clock
0101	Trigger	not used
0110	Gate	Ext. Clock
0111	Trigger	Trigger
1000	Clock Up	Clock Down
1001	Up/Down	Ext. Clock
1010	Trigger Up	Trigger Down
1011	Up/Down	not used
1100	Autodiscr.	Autodiscr.
1101	Trigger	Ext. Clock
1110	Ext. Clock	Trigger
1111	Trigger	Gate

Some choices in the external input pin assignment are defined in conjunction with RM0 and RM1 bits in TMR.

For input pin assignment codes using the input pins as Trigger Inputs (except for code 1010, Trigger Up:Trigger Down):

- a trigger signal on TxINA input pin performs an U/D counter load if RM0 ="0", or an external capture if RM0 = "1".
- a trigger signal on TxINB input pin always performs an external capture on REG1R. The TxINB input pin is disabled when the Bivalue Mode is set.

Note. For proper operation of the External Input pins, the following must be observed:

– the minimum external clock/trigger pulse width cannot be less than the system clock (INTCLK) period if the input pin is programmed as rising or falling edge sensitive.

– the minimum external clock/trigger pulse width cannot be less than the prescaler clock period (INTCLK/3) if the input pin is programmed as rising and falling edges sensitive (valid also in Autodiscrimination mode). - the minimum delay between two clock/trigger pulse active edges must be greater than the prescaler clock period (INTCLK/3), while the minimum delay between two consecutive clock/trigger pulses must be greater than the system clock (INTCLK) period.

– the minimum gate pulse width must be at least twice the prescaler clock period (INTCLK/3).

– in Autodiscrimination mode, the minimum delay between the input pin A pulse edge (inside the input pin B pulse) and the edges of the input pin B pulse, must be at least the system clock (INTCLK) period.

– if a number N of external pulses must be counted using a Compare Register of a Timer in External Clock mode, then the Compare Register used must be loaded with the value [X +/- (N-1)], where X is the starting counter value and the sign is chosen depending if in Up or Down count mode respectively.

The sixteen external input functional modes available (referring to Table 12-2) are:

12.3.1 TxINA = I/O - TxINB = I/O

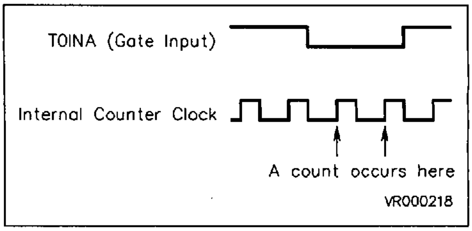
Input pins A and B are not used by the Timer. The counter clock is internally generated and the up/down control may be made only by software action through the UDC (Software Up/Down) bit in the TCR register.

12.3.2 TxINA = I/O - TxINB = Trigger

The signal applied to input pin B acts as a trigger signal on REG1R register. The prescaler clock is internally generated and the up/down control may be made only by software action through the UDC bit in the TCR register.

12.3.3 TxINA = Gate - TxINB = I/O

The signal applied to input pin A acts as a gate signal for the internal clock (i.e. the counter runs only when the gate signal is at a low level). The counter clock is internally generated and the up/down control may be made only by software action through the UDC bit in the TCR register.



INPUT PIN ASSIGNMENT (Continued)

12.3.4 TxINA = Gate - TxINB = Trigger

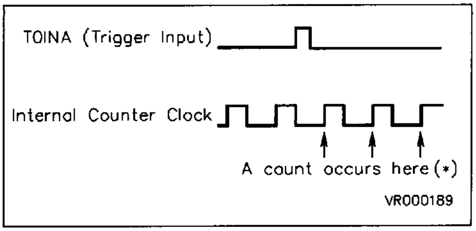
Both input pins A and B are connected to the timer, with the resulting effect of combining the actions due to the above explained configurations.

12.3.5 TxINA = I/O - TxINB = Ext. Clock

The signal applied to input pin B is used as the external clock for the prescaler. The up/down control may be made only by software action through the UDC bit in the TCR register.

12.3.6 TxINA = Trigger - TxINB = I/O

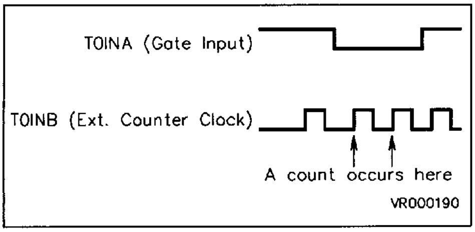
The signal applied to input pin A acts as a trigger signal on REG0R register performing the action for which the register was programmed (i.e. a reload or capture). The prescaler clock is internally generated and the up/down control may be made only by software action through the UDC bit in the TCR register.



(*) The timer is in One shot mode and REG0R in Reload mode

12.3.7 TxINA = Gate - TxINB = Ext. Clock

The signal applied to input pin B, gated by the signal applied to input pin A, acts as external clock for the prescaler. The up/down control may be made only by software action through the UDC bit in the TCR register.

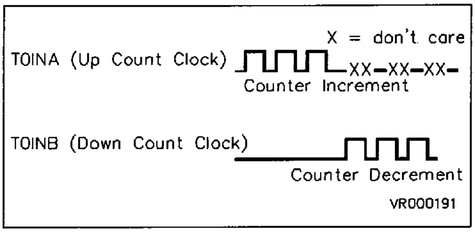


12.3.8 TxINA = Trigger - TxINB = Trigger

The signal applied to input pin A (or B) acts as trigger signal for the REG0R (or REG1R) register performing the action for which the register has been programmed. The counter clock is internally generated and the up/down control may be made only by software action through the UDC bit in the TCR register.

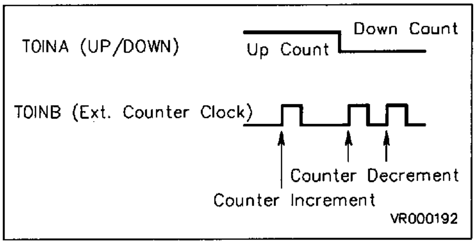
12.3.9 TxINA = Clock Up - TxINB = Clock Down

The pulse received on input pin A (or B) performs a one step up (or down) count, so that the counter clock and the up/down control are external. Setting the UDC bit in the TCR register has no effect in this configuration while input pin B has priority on input pin A.



12.3.10 TxINA = Up/Down - TxINB = Ext Clock

An High (or Low) level of the signal applied on input pin A sets the counter in the up (or down) count mode, while the signal applied to input pin B is used as clock for the prescaler. Setting the UDC bit in the TCR register has no effect in this configuration.

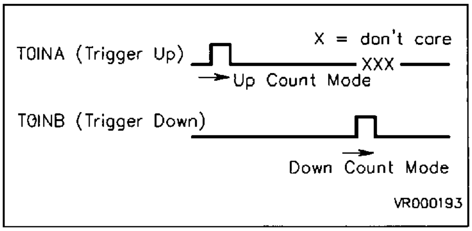


12.3.11 TxINA = Trigger Up - TxINB = Trigger Down

Up/down control is performed through both input pins A and B. A pulse on input pin A sets the up count mode, while a pulse on input pin B (which has priority on input pin A) sets the down count

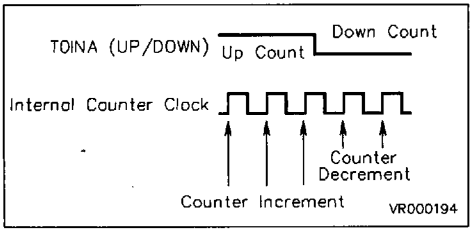
INPUT PIN ASSIGNMENT (Continued)

mode. The counter clock is internally generated while setting the UDC bit in the TCR register has no effect in this configuration.



12.3.12 TxINA = Up/Down - TxINB = I/O

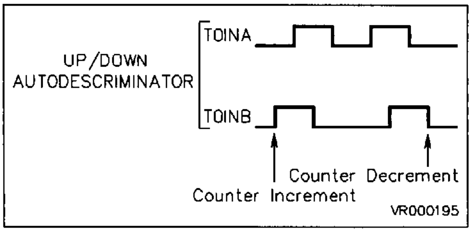
An High (or Low) level of the signal applied on input pin A sets the counter in the up (or down) count mode. The counter clock is internally generated. Setting the UDC bit in the TCR register has no effect in this configuration.



12.3.13 Autodiscrimination Mode

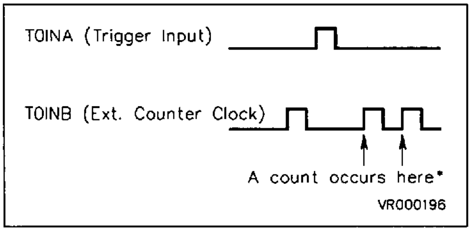
The phase between two pulses (respectively on input pin B and input pin A) generates a one step up (or down) count, so that the up/down control and the counter clock are both external. Thus, if the rising edge of TxINB arrives when TxINA is at level "0" the timer is incremented (no action if the rising edge of TxINB arrives when TxINA is at level "1"). If the falling edge of TxINB arrives when TxINA is at level "0" the timer is decremented (no action if the falling edge of TxINB arrives when TxINA is at level "1").

Setting the UDC bit in the TCR register has no effect in this configuration.



12.3.14 TxINA = Trigger - TxINB = Ext. Clock

The signal applied to input pin A acts as a trigger signal on REG0R register performing the action for which the register was programmed (i.e. a reload or capture), while the signal applied to input pin B is used as clock for the prescaler.



(*) The timer is in One shot mode and REG0R in reload mode

12.3.15 TxINA = Ext. Clock - TxINB = Trigger

The signal applied to input pin B acts as a trigger, performing a capture on REG1R register, while the signal applied to the input pin A is used as clock for the prescaler.

12.3.16 TxINA = Trigger - TxINB = Gate

The signal applied to input pin A acts as a trigger signal on REG0R register performing the action for which the register was programmed (i.e. a reload or capture), while the signal applied to input pin B acts as a gate signal for the internal clock (i.e. the counter runs only when the gate signal is at a low level).

12.4 OUTPUT PIN ASSIGNMENT

Two external outputs are available for each timer when programmed as Alternate Function Outputs of the I/O pins.

Two registers for every timer, Output A Control Register (OACR) and Output B Control Register (OBCR) define the driver for the outputs and the actions to be performed.

Each of the two output pins can be driven from any of the three possible sources:

- Compare Register 0 event logic
- Compare Register 1 event logic
- Overflow/Underflow event logic.

Each of these three sources can cause one of the following four effects on any of the two outputs:

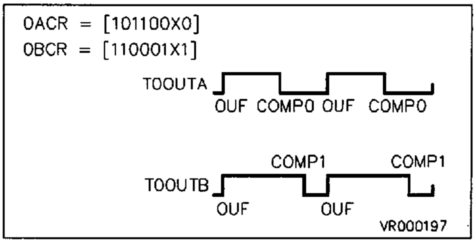
- Nop
- Set
- Reset
- Toggle.

Furthermore an On Chip Event signal can be driven by two of the three sources: the Over/Underflow event and Compare 0 event by programming the CEV bit of the OACR register and the OEV bit of OBCR register respectively. This signal can be used for another on-chip peripheral or as strobe for an I/O port.

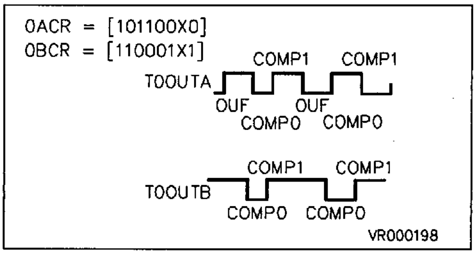
Output Waveforms

Depending on the different programmed values of OACR and OBCR the following example waveforms can be generated on TxOUTA and TxOUTB pins.

Configuration where TxOUTA is driven by Over/Underflow (OUF) and Compare 0 event (CM0), while TxOUTB is driven by the Over/Underflow and Compare 1 event (CM1).
OACR is programmed with TxOUTA preset to “0”, OUF sets TxOUTA, CM0 resets TxOUTA and CM1 does not affect the output.
OBCR is programmed with TxOUTB preset to “0”, OUF sets TxOUTB, CM1 resets TxOUTB while CM0 does not affect the output.

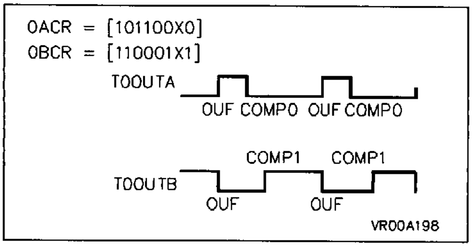


Configuration where TxOUTA is driven by Over/Underflow, Compare 0 and Compare 1, while TxOUTB is driven by both Compare 0 and Compare 1.
OACR is programmed with TxOUTA preset to “0”. OUF toggles the Output 0 as do CM0 and CM1.
OBCR is programmed with TxOUTB preset to “1”. OUF does not affect the output while CM0 resets TxOUTB and CM1 sets it.

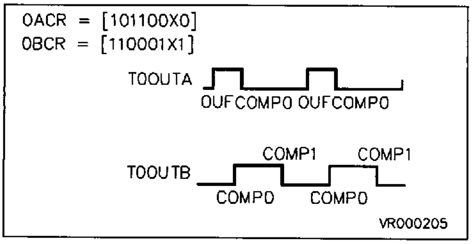


OUTPUT PIN ASSIGNMENT (Continued)

Configuration where TxOUTA is driven by Over/Underflow and Compare 0, while TxOUTB is driven by Over/Underflow and Compare 1.
OACR is programmed with TxOUTA preset to "0". OUF sets TxOUTA while CM0 resets it and CM1 has no affect.
OBCR is programmed with TxOUTB preset to "1". OUF toggles TxOUTB, CM1 sets it and CM0 has no affect.



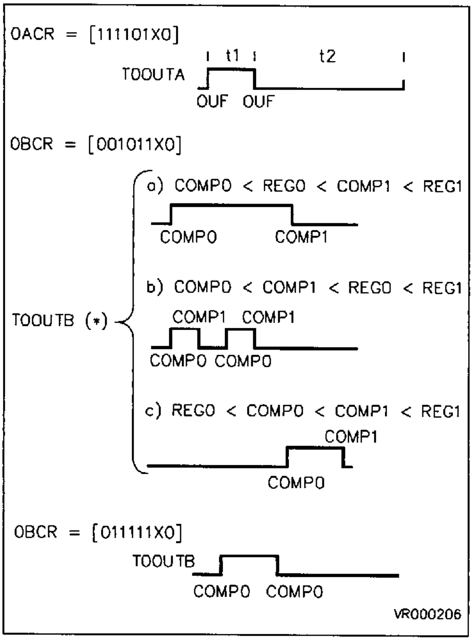
Configuration where TxOUTA is driven by Over/Underflow and Compare 0, while TxOUTB is driven by Compare 0 and 1.
OACR is programmed with TxOUTA preset to "1". OUF sets TxOUTA, CM0 resets it and CM1 has no affect.
OBCR is programmed with TxOUTB preset to "0". OUF has no affect, CM0 sets TxOUTB and CM1 toggles it.



Output Waveform Samples In Biload Mode

TxOUTA is programmed to monitor the two time intervals (t1 and t2) of the Biload Mode while TxOUTB is independent from the Over/Underflow and is driven by the different values of Compare 0 and Compare 1.
OACR is programmed with TxOUTA preset to "0". OUF toggles the output and CM0 and CM1 do not affect TxOUTA.
OBCR is programmed with TxOUTB preset to "0". OUF has no effect, while CM1 resets TxOUTB and CM0 sets it.

Depending on the CM1/CM0 values, three different example waveforms have been drawn starting from the above mentioned configuration of OBCR. In the last case, with a different programmed value of OBCR, only Compare 0 drives TxOUTB, toggling the output.



Note (*) Depending on the CMP1R/CMP0R values

12.5 INTERRUPT AND DMA

12.5.1 Timer Interrupt

The timer has 5 different Interrupt sources, grouped into 3 independent groups, assigned to the following Interrupt vectors:

Table 12-3. Timer Interrupt Structure

Interrupt Source	Vector Address
COMP 0 COMP 1	xxxx x110
CAPT 0 CAPT 1	xxxx x100
Overflow/Underflow	xxxx x000

The three least significant bits of the vector pointer address represent the relative priority assigned to each group, (000 value is the highest priority level) and are fixed by hardware depending on the source which generates the interrupt request. The 5 most significant bits are programmed by the user in the Interrupt Vector Register (IVR) of each Timer.

Each source can be masked by a dedicated bit in the Interrupt/DMA Mask Register (IDMR) of each timer, as well as a global mask enable bit (IDMR.7), masking all interrupts.

If an interrupt request (CM0 or CP0) happens before the corresponding pending bit is reset, an overrun condition occurs. This condition is flagged in two dedicated overrun bits, concerning the Comp0 and Capt0 sources, and placed in the Timer Flag Register (FLAGR).

12.5.2 Timer DMA

Two Independent DMA channels, associated to Compare 0 and Capture 0 sources, respectively allow DMA transfers from Register File/Memory to Comp0 Register and vice versa from Capt0 Register to Register File/Memory (also transfers in/from Memory from/into an I/O port are available). Their priority is hardware set as follows:

- Compare 0 Destination Lower Priority
- Capture 0 Source Higher Priority

The two DMA request sources are independently maskable by two DMA Mask bits, mapped in the Timer Interrupt/DMA Mask register (IDMR).

The two End of Block procedures, associated to each Interrupt mask and DMA mask combination, follow the standard architecture as shown in the Interrupt and DMA chapters.

12.5.3 DMA Pointers

The 6 programmable most significant bits of the Timer Address and Counter Pointer registers (DAPR-DCPR) are common to both channels (Comp0 and Capt0 sources). As a consequence, the Comp0 and Capt0 Address pointers are mapped by pair in the Register File, as well as the Comp0 and Capt0 DMA Counter pair.

The different address specification, in order to point either Capt0 or Comp0 pointers, is provided by the Timer according to the channel under service (replacing the address bit 1 with "0" for CAPT0 or with "1" for COMP0), when D0 bit on DCPR register is equal to zero (Word address in Register File). In this condition (register with program/data memory transfer), the pointers will be split in two groups of adjacent Address pointer and Counter pairs respectively.

In the case of register to register transfers (selected by programming the value "1" into bit 0 of the DCPR register), only one pair of pointers are required and the pointers are mapped into one group of adjacent positions.

DAPR (the DMA/Address Pointer Register) in this case in not used, but must be considered reserved.

INTERRUPT AND DMA (Continued)

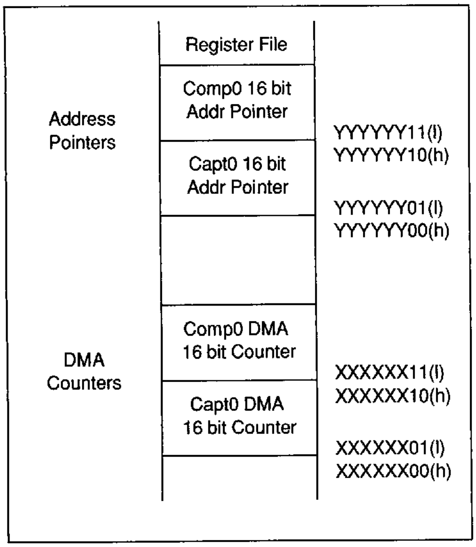


Figure 12-4. Map Pointer for Register to Prog/Data Memory Transfer

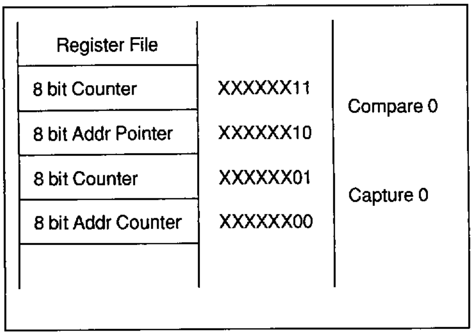


Figure 12-5. Map Pointer for Register to Register Transfer

12.5.4 Priority During The DMA Transactions

Each Timer DMA transaction is a 16-bit operation, therefore two different bytes must be transferred subsequently. This is accomplished by two DMA transfers. In order to speed up each word transfer, the second byte transfer is executed by forcing automatically the peripheral priority to the highest level (000) regardless to the previous set level. It will be then restored to the original value after executing this transfer. Furthermore, once one request is being served, its hardware priority is kept at the highest level regardless to the other Timer internal sources, i.e. once a Comp0 request is being served, it keeps a higher priority on the Capt0 channel, even if a Capt0 request occurs between the two byte transfers.

12.5.5 The DMA Swap Mode

After a complete data table transfer, the transaction counter is reset and an End Of Block condition occurs, the block transfer is completed.

The End Of Block Interrupt routine has at this point to reload both address and counter pointers of the channel referred by the End Of Block interrupt source if the application requires a continuous high speed data flow. This procedure causes speed limitations because of the time consumed by the reload routine.

The SWAP feature overcomes this drawback, allowing high speed continuous transfers. Bit 2 of the Timer Address and Counter Pointer registers (DAPR-DCPR), toggles after any End Of Block condition, alternately providing odd and even address (D2-D7) for the pair of pointers, thus pointing to an updated pair, after a block has been completely transferred. This allows the User to be updating or reading the first block, and to update the pointer values while the second is being transferred. These two toggle bits are software writable and readable, mapped in DCPR bit 2 for the CM0 channel, and in DAPR bit 2 for the CP0 channel (though a DMA event on a channel, in Swap mode, modifies a field in DAPR and DCPR common to both channels, the DAPR/DCPR content used in the transfer is always the bit related to the correct channel).

The SWAP mode can be enabled by a control bit placed in the Interrupt Control Register.

WARNING: *this mode is always set for both channel (CM0 and CP0).*

INTERRUPT AND DMA (Continued)

12.5.6 The DMA End Of Block Interrupt Routine

This Interrupt request is generated after each block transfer (EOB) and its priority is the same as assigned in the usual Interrupt request, for the two channels. As a consequence, they will be served only when no DMA request occurs, and will be submitted to a possible OUF Interrupt request, which has higher priority.

Here is a typical EOB procedure (with swap mode enabled):

- Toggle bit test and Jump.
- Pointers (odd or even depending on toggle bit status) reload.
- Reset EOB bit: this bit must be reset only after the old couple of pointers has been restored, so that, if a new EOB condition occurs, the next pointers are ready to be swapped.
- Verify the software protection condition.
- Read the corresponding Overrun bit: this makes the user sure that NO DMA request has been lost in the meantime.
- Return.

WARNING: *The EOB bits are read/write bits only for testing reasons. Writing a logical "1" by software (when SWEN bit is set) will cause a spurious interrupt request. During normal operation, these bits must only be reset by software.*

12.5.7 DMA Software Protection

A second EOB condition may occur before the first EOB routine is completed, this would cause a not yet updated pointer couple to be addressed, with consequent overwriting of memory. To prevent these errors, a protection mechanism is provided, such that the attempted setting of the EOB bit before it has been reset by software will cause the DMA mask on that channel to be reset (DMA disabled), locking any further DMA operation. As shown above, this mask bit should always be checked in each EOB routine, to ensure all DMA transfers are properly served.

12.6 REGISTER DESCRIPTION

Twenty control and data registers are associated to each Multifunction timer, and are located in the Group F I/O pages of the ST9 Register File.

The registers of the Multifunction Timers are located in the I/O pages as follows:

Note that unused registers must be regarded as reserved registers.

In the following pages there is a detailed description of every register with the meaning and the function of every bit. The register is referred to without the absolute address which is dependent on the number of the timer used (the configuration and the functions of the internal bits of i.e. TCR - TIM0 are the same as for TCR - TIM1).

Table 12-4. Multifunction Timer Register Map (Group F)

Applicable to ST90R91

	Page 9	Page 10 Page 0Ah	
R255		IMDR - TIM0	FFh
R254		FLAGR - TIM0	FEh
R253		OBCR - TIM0	FDh
R252		OACR - TIM0	FCh
R251		PRSR - TIM0	FBh
R250		ICR - TIM0	FAh
R249		TMR - TIM0	F9h
R248		TCR - TIM0	F8h
R247		CMP1LR - TIM0	F7h
R246		CMP1HR - TIM0	F6h
R245		CMP0LR - TIM0	F5h
R244		CMP0HR - TIM0	F4h
R243	IDCR - TIM0	REG1LR - TIM0	F3h
R242	IVR - TIM0	REG1HR - TIM0	F2h
R241	DAPR - TIM0	REG0LR - TIM0	F1h
R240	DCPR - TIM0	REG0HR - TIM0	F0h

REGISTER DESCRIPTION (Continued)

12.6.1 Register 0 (REG0R) Registers

This pair of registers (REG0LR and REG0HR) is used to capture values from the U/D counter or to load preset values into the U/D counter.

REG0HR R240 (F0h) Read/Write
Capture Load Register 0 (High)

Reset value: undefined

7							0
R15	R14	R13	R12	R11	R10	R9	R8

REG0LR R241 (F1h) Read/Write
Capture Load Register 0 (Low)

Reset value: undefined

7							0
R7	R6	R5	R4	R3	R2	R1	R0

12.6.2 Register 1 (REG1R) Registers

This pair of registers (REG1LR and REG1HR) is used (as REG0R) to capture values from the U/D counter or to load preset values into the U/D counter.

REG1HR R242 (F2h) Read/Write
Capture Load Register 1 (High)

Reset value: undefined

7							0
R15	R14	R13	R12	R11	R10	R9	R8

REG1LR R243 (F3h) Read/Write
Capture Load Register 1 (Low)

Reset value: undefined

7							0
R7	R6	R5	R4	R3	R2	R1	R0

12.6.3 Compare 0 (CMP0R) Registers

This pair of Registers (CMP0L and CMP0H) is used to store 16-bit values to be compared to the U/D counter content.

CMP0HR R244 (F4h) Read/Write
Compare 0 Register (High)

Reset value: undefined

7							0
R15	R14	R13	R12	R11	R10	R9	R8

CMP0LR R245 (F5h) Read/Write
Compare 0 Register (Low)

Reset value: undefined

7							0
R7	R6	R5	R4	R3	R2	R1	R0

12.6.4 Compare 1 (CMP1R) Registers

This pair of Registers (CMP1L and CMP1H) is used (as CMP0R) to store 16-bit values to be compared to the U/D counter content.

CMP1HR R246 (F6h) Read/Write
Compare 1 Register (High)

Reset value: undefined

7							0
R15	R14	R13	R12	R11	R10	R9	R8

CMP1LR R247 (F7h) Read/Write
Compare 1 Register (Low)

Reset value: undefined

7							0
R7	R6	R5	R4	R3	R2	R1	R0

REGISTER DESCRIPTION (Continued)

12.6.5 Timer Control Register (TCR)

This register is used to control the status of the timer.

TCR R248 (F8h) Read/Write
Timer Control Register
Reset value: 0000 0xxxb

7							0
CEN	CCP0	CCMP0	CCL	UDC	UDCS	OF0	CS

b7 = **CEN:** *Counter Enable.* This bit is ANDed with the Global Counter Enable bit (GCEN bit on R230 - Central Interrupt Control Register; the GCEN bit is set after the Reset cycle). Setting the CEN bit starts the counter and prescaler (without reload). When this bit is reset, the counter and prescaler stop.

b6 = **CCP0:** *Clear on Capture.* When this bit is set, a clear of the counter and a reload of the prescaler are performed on REG0R or REG1R capture. No effect when this bit is reset.

b5 = **CCMP0:** *Clear on Compare.* When this bit is set, a clear of the counter and a reload of the prescaler are performed on CMP0R compare. No effect when this bit is reset.

b4 = **CCL:** *Counter clear.* When this bit is set, the counter is cleared without generation of interrupt request. No effect when this bit is reset.

b3 = **UDC:** *Software Up/Down.* When the direction of the counter is not fixed by TxINA and/or TxINB (see par. 10.3) it can be software controlled by the UDC bit. Setting the UDC bit selects the Up mode counting. Resetting this bit the Down counting is performed.

b2 = **UDCS:** *Up/Down Count status.* This bit is read only and monitors the direction of the counter. Reading "1" means that the counter is using the Up mode counting. Reading "0" means that the Down mode counting is in use.

b1 = **OF0:** *OVF/UNF state.* This bit is read only and is set if an Overflow or an Underflow occurs during a Capture on Register 0.

b0 = **CS:** *Counter Status.* This bit is read only and monitors the status of the counter. Reading "1" means that the counter is running. Reading "0" indicates that the counter is halted.

12.6.6 Timer Mode Register (TMR)

This register is used to select the operating mode of the timer.

TMR R249 (F9h) Read/Write
Timer Mode Register
Reset value: 0000 0000b (00h)

7							0
OE1	OE0	BM	RM1	RM0	ECK	REN	CO

b7 = **OE1:** *Output 1 Enable.* Setting this bit enables the Output 1 (TxOUTB) of the relevant timer. When this bit is reset, the TxOUTB is disabled and forced to the logic state "1". The relevant I/O bit must also be set to Alternate Function.

b6 = **OE0:** *Output 0 Enable.* Setting this bit enables the Output 0 (TxOUTA) of the relevant timer. When this bit is reset, the TxOUTA is disabled and forced to the logic state "1". The relevant I/O bit must also be set to Alternate Function.

b5 = **BM:** *Bivalue Mode.* This bit enables the Bivalue mode when is set. When the bit is reset, the Bivalue mode is disabled. After that, depending on the value of RM0 bit (TMR - bit 3), the Biload or Bicapture mode is selected.

b4 = **RM1:** *REG1R mode.* When this bit is set, the REG1R can be used to capture the value of the counter. When the bit is reset, the REG1R monitors the value of the counter. The selection performed by this bit has no effect when the Bivalue Mode is enabled.

b3 = **RM0:** *REG0R mode.* When this bit is set, the REG0R can be used to capture the value of the counter (also the Bicapture mode can be selected if the BM bit is equal to 1). When the bit is reset, the REG0R can be used to load the new value of the counter (also the Biload mode can be selected if the BM bit is equal to "1").

b2 = **ECK:** *Timer clocking mode.* This bit selects the clock source which drives the prescaler. When the ECK bit is reset, either the Internal or External clock is used depending on IN0 - IN3 configuration in ICR. When ECK bit is set, different functions are performed depending on the number of the relevant timer. For odd timers (Timer 1, Timer 3 and so on) setting the ECK bit enables the Parallel mode where the prescaler of the odd timer is driven by the prescaler output of the even timer.

REGISTER DESCRIPTION (Continued)

b1 = **REN:** *Retrigger mode.* When this bit is reset, the Retriggerable mode is enabled. When the bit is set, this operating mode is disabled.

b0 = **CO:** *Continous/One shot mode.* When this bit is reset, the Continuous mode is selected (with autoreload on condition). The bit must be set to select the one shot mode. The following table summarizes the different operating modes depending on the values of RM0, RM1 and BM bits.

Table 12-5. Timer Operating Modes

TMR Bits			Timer Operating Modes
BM	RM1	RM0	
1	X	0	Biload mode
1	X	1	Bicapture mode
0	0	0	Load from REG0R and Monitor on REG1R
0	1	0	Load from REG0R and Capture on REG1R
0	0	1	Capture on REG0R and Monitor on REG1R
0	1	1	Capture on REG0R and REG1R

12.6.7 External Input Control Register(ICR)

By this register it is possible to program the function and the operation to be performed on TxINA and TxINB inputs.

ICR R250 (FAh) Read/Write
External Input Control Register
Reset value: 0000 xxxxb (0Xh)

7							0
IN3	IN2	IN1	IN0	A0	A1	B0	B1

b7-b4 = **IN3 - IN2:** *Input pin assignment.* The different functions of TxINA and TxINB inputs of every timer can be selected by IN0 - IN3 bits as explained below.

b3-b2 = **A0, A1:** *TxINA event programming.* The following TxINA configurations can be selected according to the values of A0 and A1 bits:

b1-b0 = **B0, B1:** *TxINB event programming.* The following TxINB configurations can be selected according to the values of B0 and B1 bits:

A0/B0	A1/B1	TxINA/TxINB Configuration
0	0	No operation
0	1	Falling edge sensitive
1	0	Rising edge sensitive
1	1	Rising and falling edges

I C Reg. IN3-IN0 bits	TxINA Input Function	TxINB Input Function
0000	not used	not used
0001	not used	Trigger
0010	Gate	not used
0011	Gate	Trigger
0100	not used	Ext. Clock
0101	Trigger	not used
0110	Gate	Ext. Clock
0111	Trigger	Trigger
1000	Clock Up	Clock Down
1001	Up/Down	Ext. Clock
1010	Trigger Up	Trigger Down
1011	Up/Down	not used
1100	Autodiscr.	Autodiscr.
1101	Trigger	Ext. Clock
1110	Ext. Clock	Trigger
1111	Trigger	Gate

REGISTER DESCRIPTION (Continued)

12.6.8 Prescaler Register (PRSR)

This register holds the preset value for the 8-bit prescaler. The PRSR content may be modified at any time, but it will be loaded into the prescaler at the following prescaler underflow, or as a consequence of a counter reload (either by software or upon external request). On an external RESET condition, the prescaler is automatically loaded with the 00h value, so that the prescaler divides by 1 and the maximum counter clock is generated (OSCIN frequency divided by 6 when MODER.5 = DIV2 bit is set).

PRSR R251 (FBh) Read/Write
Prescaler Register
Reset value: 0000 0000b (00h)

7							0
P7	P6	P5	P4	P3	P2	P1	P0

The binary value stored (by programmer) in the PRSR register is equal to [divider value - 1]. For example, loading PRSR with 24 makes the prescaler divide by 25.

12.6.9 Output A Control Register (OACR)

This register selects the sources that can perform actions on a TxOUTA pin. TxOUTA can be driven from any of three possible sources:

- OVF/UNF being an Overflow or Underflow event on the U/D counter,
- COMP0 being a successful compare event on CMP0R register, and
- COMP1 being a successful compare event on CMP1R.

By programming bits B0 and B1 of the relevant source can cause one of the following four effects on TxOUTA (which can be preset previously):

B0	B1	Event
0	0	Set
0	1	Toggle
1	0	Reset
1	1	Nop

Note: In any case of contemporary events the action will be taken which results from 'ANDing' the B1-B0 fields. Through this register the action of COMP0 on the on-chip event can be also selected.

OACR R252 (FCh) Read/Write
Output A Control Register
Reset value: xxxx xx0xb

7							0
B0	B1	B0	B1	B0	B1	CEV	OP
< COMP0 >		< COMP1 >		< OVF/UNF >			

b7-b6 = **B0, B1:** *Control bits of COMP0.* Control bits for event driven by COMP0.

b5-b4 = **B0, B1:** *Control bits of COMP1.* Control bits for event driven by COMP1.

b3-b2 = **B0, B1:** *Control bits of OVF/UNF.* Control bits for event driven by OVF/UNF.

b1 = **CEV:** *On-Chip Event on CMP0R.* When this bit is set, a successful compare on CMP0R activates the on-chip event signal (a single pulse is generated). No action when this bit is reset.

b0 = **OP:** *Control bit of TxOUTA preset.* The value of this bit is the preset value of TxOUTA output pin. Reading this bit returns the current state of the TxOUTA output pin (i.e. useful when this output is selected in toggle mode).

REGISTER DESCRIPTION (Continued)

12.6.10 Output B Control Register (OBCR)

This register selects the sources that can perform actions on TxOUTB output pin. TxOUTB can be driven from any of three possible sources:

- OVF/UNF being an Overflow or Underflow event on the U/D counter,
- COMP0 being a successful compare event on CMP0R register, and
- COMP1 being a successful compare event on CMP1R.

By programming bits B0 and B1 of the relevant source can cause one of the following four effects on TxOUTB (which can be previously preset):

B0	B1	Event
0	0	Set
0	1	Toggle
1	0	Reset
1	1	Nop

Note: In any case of contemporary events the action will be taken which results from 'ANDing' the B1-B0 fields. Through this register the action of Overflow/Underflow on the on-chip event can be also selected.

OBCR R253 (FDh) Read/Write
Output B Control Register

Reset value: xxxx xx0xb

7							0
B0	B1	B0	B1	B0	B1	OEV	OP

< COMP0 > < COMP1 > < OVF/UNF >

b7-b6 = **B0, B1:** *control bits of COMP0.* Control bits for event driven by COMP0.

b5-b4 = **B0, B1:** *control bits of COMP1.* Control bits for event driven by COMP1.

b3-b2 = **B0, B1:** *control bits of OVF/UNF.* Control bits for event driven by OVF/UNF.

b1 = **OEV:** *On-Chip Event on OVF/UNF.* When this bit is set, a successful Overflow/Underflow activates the on-chip event signal (a single pulse is generated). No action when this bit is reset.

b0 = **OP:** *control bit of TxOUTB preset.* The value of this bit is the preset value of TxOUTB output pin. Reading this bit, it returns the current state of the TxOUTB output pin (i.e. useful when this output is selected in toggle mode).

12.6.11 Flag Register (FLAGR)

This register contains the flags of the successful captures or comparisons together with the Overflow/Underflow and overrunning indications. Also the mode of the Interrupt on capture can be selected. By writing into the capture flags it is possible to generate software captures. It is necessary to clear the capture flag before subsequent sofware captures can be generated. By reading this register, user can know which source has generated an interrupt (several sources may share the same interrupt vector).

FLAGR R254 (FEh) Read/Write
Flags Register

Reset value: 0000 0000b (00h)

7							0
CP0	CP1	CM0	CM1	OUF	OCP0	OCM0	A0

b7 = **CP0:** *Flag on Capture 0.* This bit is set after a capture on REG0R register. Writing "1" acts as a software load/capture from/on REG0R.

b6 = **CP1:** *Flag on Capture 1.* This bit is set after a capture on REG1R register. Writing "1" acts as a software capture on REG1R, except when in Bi-capture mode.

b5 = **CM0:** *Flag on Compare 0.* This bit is set after a successful compare on CMP0R register.

b4 = **CM1:** *Flag on Compare 1.* This bit is set after a successful compare on CMP1R register.

b3 = **OUF:** *Flag on Overflow/Underflow.* This bit is set after a counter Over/Underflow condition.

b2 = **OCP0:** *Flag of overrun on Capture 0.* This bit is set when more than one INT/DMA request occurs before having reset the event flag CP0 or whenever a capture is software simulated.

b1 = **OCM0:** *Flag of overrun on Compare 0.* This bit is set when more than one INT/DMA request occurs before having reset the event flag CM0.

b0 = **A0:** *Capture Interrupt Function.* When this bit is set the Interrupt is generated by an AND function of REG0R/REG1R captures while when the A0 bit is reset, the Interrupt is generated by an OR function of REG0R/REG1R captures.

REGISTER DESCRIPTION (Continued)

12.6.12 Interrupt/DMA Mask Register (IDMR)

This register contains the Global Timer Interrupt enable bit and the INT/DMA enable bits of the following events:

- Capture on REG0R (CP0 field),
- Capture on REG1R (CP1I bit - only Interrupt mask),
- Compare on CMP0R (CM0 field),
- Compare on CMP1R (CM1I bit- only Interrupt mask), and
- Overflow/Underflow (OUI bit - only Interrupt mask).

IDMR R255 (FFh) Read/Write
Interrupt/DMA Mask Register
Reset value: 0000 0000b (00h)

7							0
GTIEN	CP0D	CP0I	CP1I	CM0D	CM0I	CM1I	OUI
	< CP0	>	<CP1>	< CM0	>	<CM1>	

b7 = **GTIEN:** *Global Timer Interrupt Enable.* When this bit is set, all the Interrupts (of the enabled sources) of the timer are enabled. When the bit is reset, all the Interrupts of timer are disabled.

b6 = **CP0D:** *Capture 0 DMA Mask.* Capture on REG0R DMA is enabled when CP0D = "1."

b5 = **CP0I:** *Capture 0 Interrupt Mask.* Capture on REG0R interrupt is enabled when CP0I = "1".

b4 = **CP1I:** *Capture 1 Interrupt Mask.* Capture on REG1R interrupt is enabled when CP1I = "1".

b3 = **CM0D:** *Compare 0 DMA Mask.* Compare on CMP0R DMA is enabled when CM0D = "1".

b3 = **CM0I:** *Compare 0 Interrupt Mask.* Compare on CMP0R interrupt is enabled when CM0I = "1".

b1 = **CM1I:** *Compare 1 Interrupt Mask.* Compare on CMP1R interrupt is enabled when CM1I = "1".

b0 = **OUI:** *Overflow/Underflow Interrupt Mask.* Overflow/Underflow condition interrupt is enabled when OUI = "1".

Note. The following Registers show in square brackets ([ ]) the Register address in the case of an odd numbered (1, 3, 5...) Multifunction Timer being available on-chip. If only one Timer is present these addresses may be ignored.

12.6.13 DMA Counter Pointer Register (DCPR)

This register is not used only as DMA Counter pointer but also to define the DMA area and the DMA source.

DCPR R240 (F0h) Read/Write
DMA Counter Pointer Register
Reset value: undefined

7							0
DCP7	DCP6	DCP5	DCP4	DCP3	DCP2	DMA SRCE	REG MEM

b7-b2 = **DCP7-DCP2:** *MSB of DMA counter register address.* Those bits contain the most significant bits of the DMA counter register address and are user programmable. Though user programmable, the D2 bit may be hardware toggled if the Swap mode is set for the Timer DMA section related to Compare 0 channel.

b1 = **DMA-SRCE:** *DMA source selection (hardware programmed).* This bit is hardware fixed by the Timer DMA logic and is set if the DMA destination is a Compare on CMP0R register and reset if the DMA source is a Capture on REG0R register.

b0 = **REG/MEM:** *DMA area selection.* When this bit is set, it selects the Source/Destination of the DMA area from/into Register File while when it is reset, the Source/Destination of the DMA area is from/to the External Program or Data Memory (according with the value of D0 bit in DAPR).

REGISTER DESCRIPTION (Continued)

12.6.14 DMA Address Pointer Register (DAPR)

This register is not used only as DMA Address pointer but also to define the DMA area and the DMA source.

DAPR R241 (F1h) Read/Write
DMA Address Pointer Register

Reset value: undefined

7							0
DAP7	DAP6	DAP5	DAP4	DAP3	DAP2	DMA SRCE	PRG/ DAT

b7-b2 = **DAP7-DAP2:** *MSB of DMA Address register location.* Those bits contain the most significant bits of the DMA Address register location and are user programmable. Through user programmable, the bit D2 may be hardware toggled if the Swap mode is set for the Timer DMA section related to Capture 0 channel.

b1 = **DMA-SRCE:** *DMA source selection (hardware programmed).* This bit is hardware fixed by the Timer DMA logic and is set if the DMA destination is a Compare on CMP0R register and reset if the DMA source is a Capture on REG0R register.

b0 = **PRG/DAT:** *DMA memory selection.* When this bit is set it selects the Source/Destination of the DMA area from/into Data Memory while when it is reset the Source/Destination of the DMA area is from/into the External Program Memory (according with the value of D0 bit in DCPR).

REG/MEM	PRG/DAT	DMA Source/Destination
0	0	Program memory
0	1	Data memory
1	0	Register file
1	1	Register file

12.6.15 Interrupt Vector Register (IVR)

This register is used as a vector pointing to the 16-bit interrupt vectors in the program memory which contain the starting addresses of the three interrupt subroutines managed by every timer.

Only one Interrupt Vector Register is available for every timer and is able to manage the three interrupt groups because the 3 least significant bits are fixed by hardware depending on the group which generated the interrupt request.

In order to understand which request generated the interrupt inside the same group, the FLAGR register can be used to check the relevant flag of the interrupt source.

IVR R242 (F2h) Read/Write
Interrupt Vector Register

Reset value: xxxx xxx0b

7							0
V4	V3	V2	V1	V0	W1	W0	D0

b7-b3 = **V4 - V0:** *MSB of the Vector address.* These bits are user programmable and contain the five most significant bits of the Timer interrupt vector addresses in the program memory. In any case, an 8-bit address can be used to indicate the Timer interrupt vector locations because they are within the first 256 locations of the program memory (see Interrupt and DMA chapters).

b2-b1 = **W1 - W0:** *Vector Address bits.* These bits are equivalent to bit 1 and bit 2 of the Timer interrupt vector addresses in the program memory. They are fixed by hardware depending on the group of sources which generated the interrupt request as follows:

b0 = **D0.** This bit is fixed by hardware. It always returns the value "0" if read.

W1	W0	Interrupt Source
0	0	Overflow/Underflow even interrupts
0	1	Not available
1	0	Capture event interrupts
1	1	Compare event interrupts

REGISTER DESCRIPTION (Continued)

12.6.16 Interrupt/DMA Control Register (IDCR)

This register is used to control the Interrupt and DMA priority level, the DMA transfer source and destination and the Swap mode. This register contains also the two End Of Block bits.

IDCR R243 (F3h) Read/Write
Interrupt/DMA Control Register
Reset value: 1100 0111b (C7h)

7							0
CPE	CME	DCTS	DCTD	SWEN	PL2	PL1	PL0

b7 = **CPE:** *Capture 0 EOB.* This bit is set by hardware when the End Of Block condition is reached during a Capture 0 DMA operation with the Swap mode enabled. When the Swap mode is disabled (SWEN bit = "0") the CPE bit is forced by hardware to "1".

b6 = **CME:** *Compare 0 EOB.* This bit is set by hardware when the End Of Block condition is reached during a Compare 0 DMA operation with the Swap mode enabled. When the Swap mode is disabled (SWEN bit = "0") the CME bit is forced by hardware to "1".

b5 = **DCTS:** *DMA Capture Transfer Source.* This bit selects the source of the DMA operation related to the channel associated to the Capture 0. When the DCTS bit is reset the selected source is the REG0R register. When the DCTS bit is set the ST9 port is selected as DMA transfer source (with this DMA channel the ST9 port can also be destination depending on the value of the DD bit in the HDCTL register of the port - see I/O port chapter 9).

b4 = **DCTD:** *DMA Compare Transfer Destination.* This bit selects the destination of the DMA operation related to the channel associated to the Compare 0. When this bit is reset, the selected destination is the CMP0R register. When the bit is set, the ST9 port is selected as DMA transfer destination.

b3 = **SWEN:** *Swap function Enable.* When this bit is set, the Swap function is enabled for the two DMA channels. Resetting the SWEN bit disables the Swap mode.

b2-b0 = **PL2 to PL0:** *Interrupt/DMA priority level.* With these three bits it is possible to select the Interrupt and DMA priority level of every single timer within eight different levels (see Interrupt/DMA chapter).

12.6.17 I/O Connection Register (IOCR)

This register allows user to select (or not) an on-chip connection between input A and output A of one same timer.

IOCR R248 (F8h) Read/Write
I/O Connection Register
Reset value: 1111 1100b (FCh)

7							0
						SC1	SC0

b7-b2 = not used.

b1 = **SC1:** *Select Connection Odd.* SC1 selects if connection between TxOUTA and TxINA for ODD timers is made on-chip or externally (physically on pins)

SC1 = "0": TxOUTA and TxINA unconnected

SC1 = "1": TxOUTA and TxINA connected internally

b0 = **SC0:** *Select Connection Even.* SC0 selects if connection between TxOUTA and TxINA for EVEN timers is made on-chip or externally (physically on pins)

SC0="0": TxOUTA and TxINA unconnected

SC0="1": TxOUTA and TxINA connected internally

13 SLICE TIMER

13.1 INTRODUCTION

The Slice Timer unit is similar in function to the Timer/Watchdog, but without the Watchdog function, providing a cost-effective solution to simple timing requirements. The Slice Timer includes a programmable 16-bit down counter and an associated 8-bit prescaler with Single and Continuous counting modes capability.

The Slice Timer uses an input from an external pin (SLIN) and an output (SLOUT) as an Alternate function of an I/O bit. SLIN can be used in one of four programmable input modes:

- event counter,
- gated external input mode,
- triggerable input mode,
- retriggerable input mode.

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

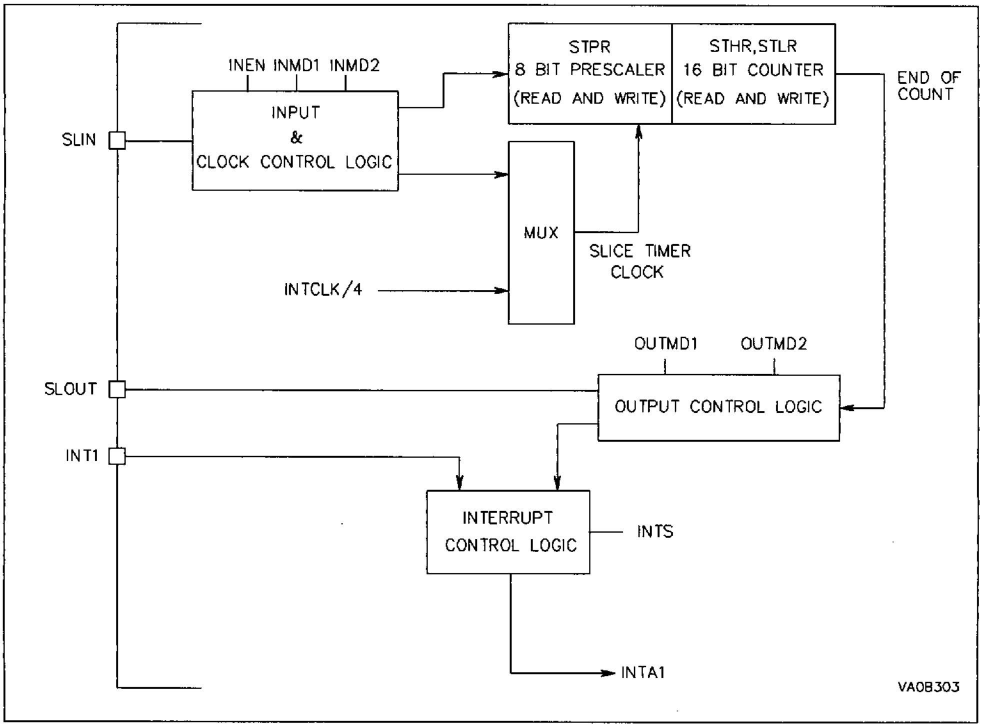
The Slice Timer is composed of a 16-bit down counter with an 8-bit prescaler. The input clock to the prescaler can be driven either by an external clock or an internal clock equal to INTCLK divided by 4.

Thus when a 24MHz crystal is used (INTCLK = 12MHz) a 3MHz maximum counting frequency can be reached or 2MHz with a 8MHz crystal and the divider by 2 disabled. The minimum and maximum counting period are:

- 5.59s for Maximum Count (Timer Const = FFFFh, Prescaler Const = FFh)

- 333ns for Minimum Count (Timer Const = 0000h, Prescaler Const = 00h)

Figure 13-1. Slice Timer Block Diagram



SLICE TIMER (Continued)

The Slice Timer End Of Count condition is able to generate an interrupt which is connected to INT1 of the external interrupt structure.

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

13.2 SLICE TIMER FUNCTIONS

13.2.1 Timer/Counter control

Start-stop Count. The ST-SP bit (STCR.7) is used in order to start and stop counting. An instruction which sets this bit will cause the Slice 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 Slice Timer registers. during the stop period. In this case, the new constant will be loaded as soon as counting is restarted.

WARNING: In order to prevent incorrect counting of the Slice Timer, the prescaler (STPR) and counter (STLR, STHR) registers must be initialised before the starting of the timer. If this is not done, counting will start with the reset (un-initialised) values.

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

SINGLE MODE: at the End of Count, the Slice 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 Slice Timer with the same instruction.

13.2.2 Slice Timer Input Modes

Bits INMD2, INMD1 and INEN are used to select the input modes. Input Enable (INEN) bit enables the input mode selected by the INMD2, 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 affect.

Event Counter Mode (INMD1 = “0”, INMD2 = “0”)

The Slice Timer is driven by the signal applied to the input pin (SLIN) which acts as an external clock. The unit works therefore as an event counter. The event is a high to low transition on SLIN.
Spacing between trailing edges should be at least the period of INTCLK divided by 4 (i.e. the maximum Slice Timer input frequency is 3MHz with INTCLK = 12MHz).

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 SLIN pin. When the status of the SLIN is High the Slice Timer count operation proceeds, and when Low, counting is stopped.

Triggerable Input Mode
(INMD1 = “1”, INMD2 = “0”)

The Slice Timer is started by:

a) a set of the Start-Stop bit, AND
b) a High to Low (low trigger) transition on SLIN.

In order to stop the Slice 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 Slice Timer is running (with internal clock), a High to Low transition on SLIN causes the counting to start from the last constant loaded into the STLR/STHR and STPR registers. When the Slice Timer is stopped (ST-SP bit equal to zero), a High to Low transition on SLIN has no effect.

13.2.3 Slice Timer Output Modes

OUTPUT modes are selected using 2 bits of STCR: OUTMD1 and OUTMD2.

No Output Mode (OUTMD1 = “0”, OUTMD2 = “0”)

With this setting the Slice Timer output is disabled and the output pin is held at a “1” level to allow several alternate functions on the same pin.

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

The Slice Timer toggles the state of the SLOUT pin on every End Of Count condition. With INTCLK = 12MHz, this allows generation of a square wave with a period ranging from 666ns to 11.18 seconds.

SLICE TIMER (Continued)

PWM Output Mode (OUTMD1 = “1”)

The value of the OUTMD2 bit is transferred to the SLOUT 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 Slice Timer interrupt.

13.2.4 Interrupt Selection

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

The STCR bit 2 (INTS) selects the interrupt source between the Slice Timer interrupt and the external interrupt INT1. Thus the Slice Timer Interrupt uses the INT1 interrupt channel and takes the priority and vector of the INTA1 external interrupt channel.

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

Care must be taken when disabling the Slice Timer Interrupt as a rising edge may be generated on the INTA1 channel, causing an unwanted interrupt.

13.2.5 Slice Timer Registers

This unit has 4 registers mapped into the page 0Bh in Group F of the Register File:

Register Address	Register	Function
F0	STH	Counter High-Byte Register
F1	STL	Counter Low-Byte Register
F2	STP	Slice Timer Prescaler Register
F3	STC	Slice Timer Control Register
F4-FF	—	Reserved

13.2.6 Register Description

STH R240 (F0h) Page 0B Read/Write
Counter High-Byte Register

Reset value: undefined

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

b7-b0 = **ST.15-ST.8:** *Counter High-Byte.*

STL R241 (F1h) Page 0B Read/Write
Counter Low-Byte Register

Reset value: undefined

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

b7-b0 = **ST.7-ST.0:** *Counter Low-Byte.* Writing to the STH and STL registers allows the user to enter the Slice Timer constant, while reading provides the counter current value. Thus it is possible to read the counter on-the-fly.

STP R242 (F2h) Page 0B Read/Write
Slice Timer Prescaler Register

Reset value: undefined

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

b7-b0 = **STP.7-STP.0:** *Prescaler.* The Prescaler value for the Slice Timer is programmed into this register. When reading the STPR register, the returned value corresponds to the programmed data instead of the current data.

STC R243 (F3h) Page 0B Read/Write
Slice Timer Control Register

Reset value: 000x x1xx

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

b7 = **ST-SP:** *Start-Stop Bit.* Setting ST-SP to “1” starts the counting operation. Writing “0” stops the Slice Timer.

b6 = **S-C:** *Single-Continuous Mode Select.* Setting S-C to “1” sets the Slice Timer to Single Mode. Writing “0” sets the Continuous Mode (Reset Status)

b5-b4 = **INMD1, INMD2:** *Input Mode Selection.* These bits select the Input functions as shown in the preceding text, when enabled by INEN.

b3 = **INEN:** *Input Enable.* INEN must be set to “0” (Input section disabled).

b2 = **INTS:** *Interrupt Selection.* Setting INTS to “1” disables the Slice Timer interrupt (the Reset status).
Writing “0” enables the Slice Timer interrupt on channel INT1.

b1-b0 = **OUTMD1, OUTMD2:** *Output Mode Selection.* These bits select the output functions as described in the preceding text.

14 A/D CONVERTER

14.1 INTRODUCTION

The Analog to Digital Converter (A/D) is comprised of an 8 channel multiplexed input selector and a Successive Approximation converter. The conversion time is thus a function of the INTCLK frequency; for the maximum 12MHz clock rate, conversion of the selected channel requires 11µs. This time also includes the 3µs of the integral Sample and Hold circuitry, which minimizes the need for external components and allows quick sampling of the signal for the minimum warping effect and Integral conversion error.

The resolution of the converted channel is 8 bits, with ±1/2 LSB maximum DNL error between the V_{SS} and V_{DD} references.

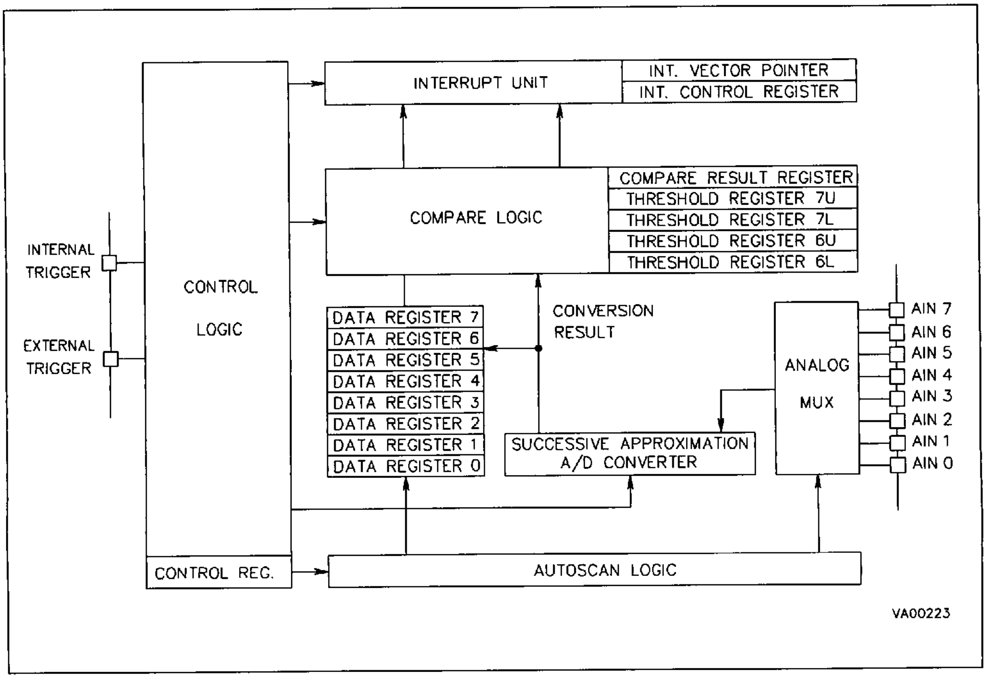
The converter uses a fully differential analog input configuration for the best noise immunity and precision performance.

Up to 8 multiplexed Analog Inputs are available. A group of signals can be converted sequentially by simply programming the starting address of the first analog channel to be converted and using the AUTOSCAN feature.

Two Analog Watchdogs are provided, on analog input channels 6 and 7, allowing a continuous hardware monitoring of these two inputs. An alarm Interrupt request will be generated whenever the converted value of either of these two analog inputs exceed one of the two programmed threshold values (Upper and Lower) for each channel. The comparison result is stored in a dedicated register.

Single, continuous, or externally triggered conversion modes are available, internal clock sample synchronization is also available through the "On chip Event" synchronization logic of a Multifunction Timer Unit,

Figure 14-1. Block Diagram



INTRODUCTION (Continued)

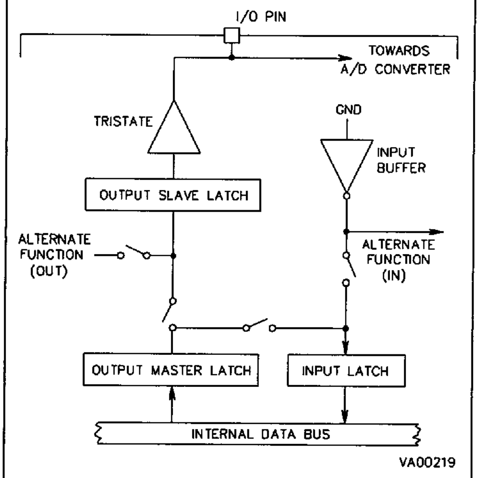
A Power-Down programmable bit allows to set the A/D converter to a minimum consumption idle status.

The ST9 A/D Interrupt Unit provides two maskable channels (Analog Watchdog and End of Conversion) with hardware fixed priority, and up to 7 programmable priority levels.

WARNING: A/D INPUT PIN DECLARATION

The input Analog channel is selected by using the Alternate Function setting (PXC2, PXC1, PXC0 = 1,1,1) as shown in the I/O ports section. The I/O bit structure of the port connected to the A/D converter is modified as shown in Figure 11-2 to prevent the Analog voltage present at the I/O pin from causing high power dissipation across the input buffer. Un-selected analog channels should also be maintained in the Alternate function mode for this reason.

Figure 14-2. A/D Input Configuration Status



14.2 FUNCTIONAL DESCRIPTION

14.2.1 Operational Modes

Two main operational 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 = "1", while CONT = "0" enables the Single Mode.

Both modes operate in the AUTOSCAN configuration, allowing a sequential conversion flow of the input channels. It is possible to choose by software the number of analog inputs to be converted by writing into the Control Register (SC2, SC1, SC0 bits) the number of the first channel to be converted. Subsequentially, after each conversion is completed, the channel number is automatically incremented, up to channel 7. For example, if (SC2, SC1, SC0) = 0,1,1 the conversion flow runs from channel 3 up to channel 7. If (SC2, SC1, SC0) = 1,1,1 only channel 7 is converted.

When the ST bit of the Control Logic Register is written to "1" by software or hardware, 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 at the end of conversion of channel 7, an End of Conversion (ECV) interrupt request is issued, and the A/D waits for a new start event.

In **Continuous Mode** (CONT = "1"), a continuous conversion flow is entered by the start event. After the conversion of channel 7 ends, the conversion of channel 's' starts (where 's' is specified in the (SC2, SC1, SC0) bits), this will continue until the ST bit is reset by software. In all cases, an ECV interrupt is issued each time the channel 7 conversion ends.

When channel 'i' is converted ('s' < 'i' < 7), the Data Register is reloaded with the new conversion result and the previous value is lost. The ECV interrupt routine can be used to save the current values before a new conversion sequence (so as to create signal sample tables within Register File or Memory).

14.2.2 Synchronisation

Conversion start synchronisation for all modes may be internal or external. The external (ADTRG, as an Alternate Function input of an I/O port) or the internal (INTRG, produced by a Multifunction Timer peripheral) can be used to synchronise the conversion start with a trigger pulse. Both external and internal events can be seperately masked by the programming of the EXTG/INTG bits of the Control Logic Register. The events are internally

FUNCTIONAL DESCRIPTION (Continued)

OR'ed, thus avoiding potential hardware conflicts, however the correct procedure is to always enable only one alternate synchronisation input at any time.

The effect of the alternate synchronisation is to set the ST bit by hardware. This bit is reset, only in Single Mode, at the end of each group of conversions. In Continuous Mode all trigger pulses, following the first, are ignored.

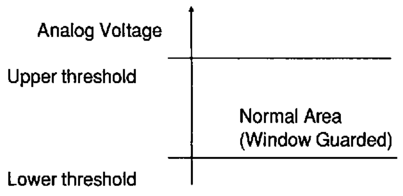
The two synchronisation sources must have a clock cycle minimum length of 83ns (at INTCLK = 12MHz), and a period greater (in Single Mode) than the total time of a group of conversions (11.5µs x the number of channels scanned (at INTCLK = 12MHz)). If a trigger occurs when the ST bit is still "1" (conversion is still in progress), it is ignored.

14.2.3 Analog Watchdog

Two internal Analog Watchdogs are available, allowing great flexibility in automatic threshold monitoring in those applications experiencing a maximum range of fluctuations. Analog channels 6 and 7 define a voltage window for the allowed values of the converted analog input. The range of values of the external voltage applied to input 6 and 7 are accepted as normal whenever *below* the Upper threshold and *above or equal* to the Lower threshold.

When the external voltage is greater or equal to the upper, or is less than the lower programmed voltage limits, a maskable interrupt request is generated and the Compare Results Register is updated to inform which threshold (Upper or Lower) of which channel (6 or 7) has been exceeded. The 4 threshold voltages are user programmable in 4 dedicated registers (8 up to B) of the A/D register page, storing their 8 bit binary code. Only the 4 MSB of the Compare Results Register are used (the 4 LSB always return "1" if read), each bit for each threshold possible overflow or underflow status.

After an hardware reset, these bit values are "0". During the normal A/D operation, the CRR bits are set to "1" to flag an over-range and areautomatically reset by hardware after a software reset of the analog Watchdog request flag in the ECR Register.



14.2.4 Power down Mode

Before enabling any A/D conversion, it is mandatory to set the POW bit of the Control Logic Register to "1" at least 60µs before the first conversion start. This is in order to correctly bias the analog section of the converter, if this is not done, then functionality of the converter will be locked.

Setting POW to "0" is useful when the A/D is not required in order to reduce the total power consumption. This is the reset configuration, and is also

Figure 14-3. A/D Trigger Source

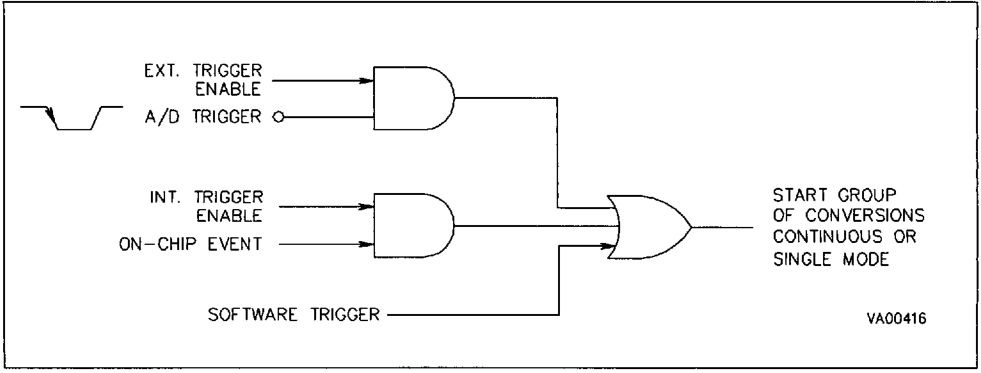
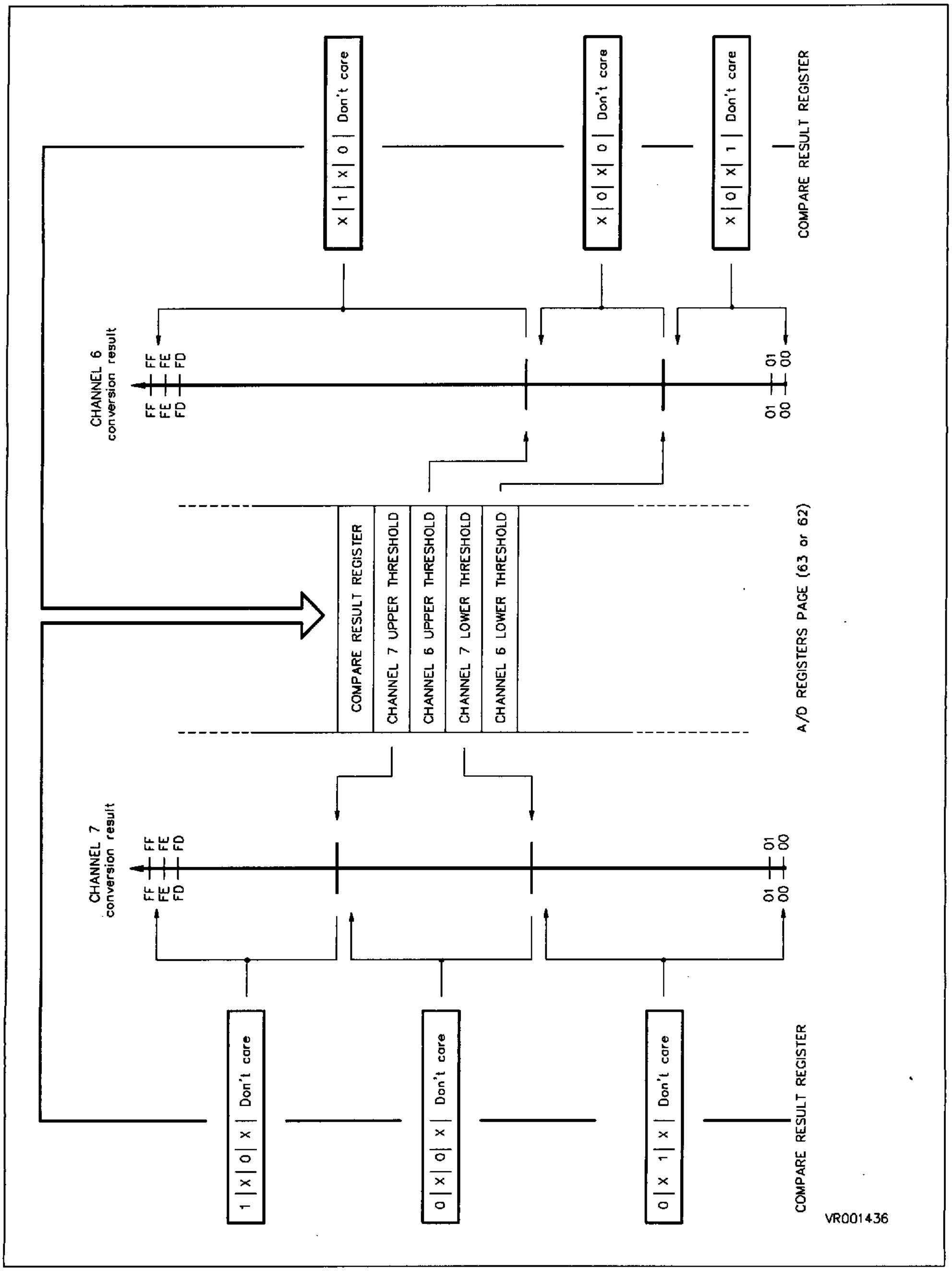
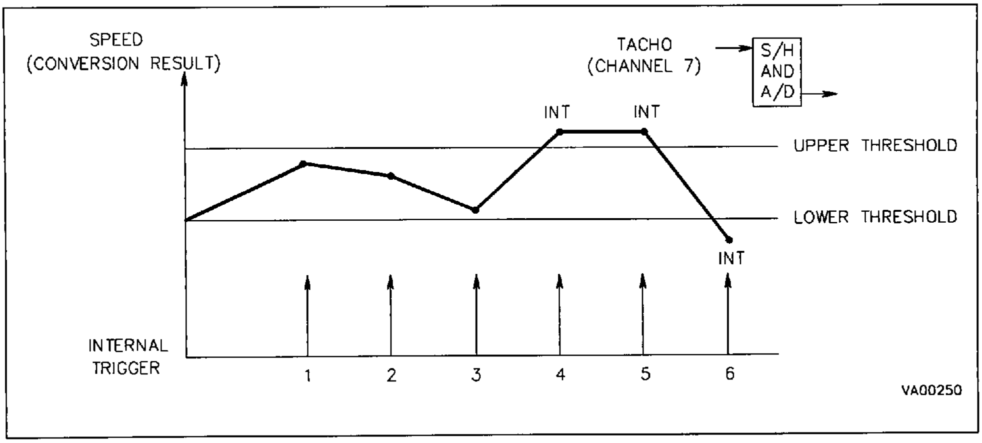


Figure 14-4. Functional Diagram


CHANNEL 6 conversion result
FF
FE
FD
01
00
X | 1 | X | 0 | Don't care
X | 0 | X | 0 | Don't care
X | 0 | X | 1 | Don't care
COMPARE RESULT REGISTER
COMPARE RESULT REGISTER
CHANNEL 7 UPPER THRESHOLD
CHANNEL 6 UPPER THRESHOLD
CHANNEL 7 LOWER THRESHOLD
CHANNEL 6 LOWER THRESHOLD
A/D REGISTERS PAGE (63 or 62)
CHANNEL 7 conversion result
1 | X | 0 | X | Don't care
0 | X | 0 | X | Don't care
0 | X | 1 | X | Don't care
COMPARE RESULT REGISTER
VR001436

FUNCTIONAL DESCRIPTION (Continued)

Figure 14-5. Analog Watchdog used in Motorspeed Control



entered automatically when the ST9 is in Halt Mode (following the execution of the `halt` instruction).

14.3 INTERRUPT

The A/D converter provides two interrupt sources, End of Conversion and an Analog Watchdog Request. The A/D Interrupt Vector Register (IVR) provides 1 bit generated in hardware to follow the interrupt source, allowing the automatic addressing of the relevant A/D Interrupt Service routine for the two sources.

	7							0	
ANALOG WATCHOG 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 to store the address of the A/D interrupt service routines.

The Analog Watchdog Interrupt Pending bit (AWD, ICR.6), is automatically hardware set whenever any of the two guarded analog inputs goes out of bounds. The Compare Result Register (CRR) keeps track of the analog inputs exceeding the thresholds.

When 2 requests occur simultaneously the Analog watchdog request has priority over the End of Conversion request which is held pending, to be served after the current routine.

The Analog Watchdog Request requires the user to poll within the Compare Result Register (CRR) to determine which of the four thresholds has been exceeded. The threshold status bits are set to "1" to flag an over-range and are automatically reset by hardware after a software reset of the analog Watchdog request flag in the ECR Register.

The interrupt pending flags, ECV (End of Conversion, ICR.7) and AWD should also be reset by the

User in the Interrupt service routine before the return. Setting either of these two bits by software will cause a software interrupt request to be generated.

14.4 REGISTERS

14.4.1 Register Mapping

A/D registers are mapped page 63.

14.4.2 Data Registers (DiR)

The result of the conversions of the 8 available channels are loaded in the 8 DiR (channel0→D0R ....channel7→D7R); every Data Register is reloaded with a new value at the end of the conversion of the correspondent analog input.

D0R R240 (F0h) Page 63 Read/Write
Channel 0 Data Register

Reset Value: Undefined

7							0
D0.7	D0.6	D0.5	D0.4	D0.3	D0.2	D0.1	D0.0

b7-b0 = **D0.7-D0.0**: Channel 0 Data

D1R R241 (F1h) Page 63 Read/Write
Channel 1 Data Register

Reset Value: Undefined

7							0
D1.7	D1.6	D1.5	D1.4	D1.3	D1.2	D1.1	D1.0

b7-b0 = **D1.7-D1.0**: Channel 1 Data

D2R R242 (F2h) Page 63 Read/Write
Channel 2 Data Register

Reset Value: Undefined

7							0
D2.7	D2.6	D2.5	D2.4	D2.3	D2.2	D2.1	D2.0

b7-b0 = **D2.7-D2.0**: Channel 2 Data

D3R R243 (F3h) Page 63 Read/Write
Channel 3 Data Register

Reset Value: Undefined

7							0
D3.7	D3.6	D3.5	D3.4	D3.3	D3.2	D3.1	D3.0

b7-b0 = **D3.7-D3.0**: Channel 3 Data

D4R R244 (F4h) Page 63 Read/Write
Channel 4 Data Register

Reset Value: Undefined

7							0
D4.7	D4.6	D4.5	D4.4	D4.3	D4.2	D4.1	D4.0

b7-b0 = **D4.7-D4.0**: Channel 4 Data

D5R R245 (F5h) Page 63 Read/Write
Channel 5 Data Register

Reset Value: Undefined

7							0
D5.7	D5.6	D5.5	D5.4	D5.3	D5.2	D5.1	D5.0

b7-b0 = **D5.7-D5.0**: Channel 5 Data

D6R R246 (F6h) Page 63 Read/Write
Channel 6 Data Register

Reset Value: Undefined

7							0
D6.7	D6.6	D6.5	D6.4	D6.3	D6.2	D6.1	D6.0

b7-b0 = **D6.7-D6.0**: Channel 6 Data

D7R R247 (F7h) Page 63 Read/Write
Channel 7 Data Register

Reset Value: Undefined

REGISTERS (Continued)

7							0
D7.7	D7.6	D7.5	D7.4	D7.3	D7.2	D7.1	D7.0

b7-b0 = **D7.7-D7.0**: Channel 7 Data

14.4.3 Lower Threshold Registers (LTiR)

The 2 lower threshold registers are used to store the 2 user programmable lower threshold voltages (i.e. their 8 bit binary code) to be compared with the present channel 6 or 7 conversion result. They fix the lower voltage window limit.

LT6R R248 (F8h) Page 63 Read/Write
Channel 6 Lower Threshold Register

Reset Value: Undefined

7							0
LT6.7	LT6.6	LT6.5	LT6.4	LT6.3	LT6.2	LT6.1	LT6.0

b7-b0 = **LT6.7-LT6.0**: *Channel 6 Lower Threshold*

LT7R R249 (F9h) Page 63 Read/Write
Channel 7 Lower Threshold Register

Reset Value: Undefined

7							0
LT7.7	LT7.6	LT7.5	LT7.4	LT7.3	LT7.2	LT7.1	LT7.0

b7-b0 = **LT7.7-LT7.0**: *Channel 7 Lower Threshold*

14.4.4 Upper Threshold Registers (UTiR)

The 2 upper threshold registers are used to store the 2 user programmable upper threshold voltages (i.e. their 8 bit binary code) to be compared with the present channel 6 or 7 conversion result. They fix the upper voltage window limit.

UT6R R250 (FAh) Page 63 Read/Write
Channel 6 Upper Threshold Register

Reset Value: Undefined

7							0
UT6.7	UT6.6	UT6.5	UT6.4	UT6.3	UT6.2	UT6.1	UT6.0

b7-b0 = **UT6.7-UT6.0**: *Channel 6 Upper Threshold*

UT7R R251 (FBh) Page 63 Read/Write
Channel 7 Upper Threshold Register

Reset Value: Undefined

7							0
UT7.7	UT7.6	UT7.5	UT7.4	UT7.3	UT7.2	UT7.1	UT7.0

b7-b0 = **UT7.7-UT7.0**: *Channel 7 Upper Threshold*

14.4.5 Compare Result Register (CRR)

The result of comparison between the current value of data registers 6 and 7 and the threshold registers is stored in this 4 bit register.

CRR R252 (FCh) Page 63 Read/Write
Compare Result Register

Reset Value: 0000 1111 (0Fh)

7							0
C7U	C6U	C7L	C6L	X	X	X	X

b7 = **C7U:** *Compare Reg 7 Upper threshold*Set to "1" when converted data is greater than or equal to the threshold value. Not affected otherwise

b6 = **C6U:** *Compare Reg 6 Upper threshold*Set to "1" when converted data is greater than or equal to the threshold value. Not affected otherwise

b5 = **C7L:** *Compare Reg 7 Lower threshold*Set to "1" when converted data is less than the threshold value. Not affected otherwise.

b4 = **C6L:** *Compare Reg 6 Lower threshold*Set to "1" when converted data is less than the threshold value. Not affected otherwise.

These bits should be Software reset at the end of the 'Out of Bounds' Interrupt routine.

b3-b0 = undefined, return "1" when read.

REGISTERS (Continued)

Note. any Software request reset in the ICR, will cause also all the Compare status bits to be hardware forced to zero, to prevent possible overwriting if an Interrupt request occurs between the Software reset and the Interrupt Request Software reset.

14.4.6 Control Logic Register (CLR)

This register manages the A/D logic operations. Writing into this register will cause the current conversion to be aborted and the autoscan logic to be re-initialized to the starting configuration. CLR is programmable as following:

CLR R253 (FDh) Page 63 Read/Write
Control Logic Register

Reset Value: 0000 0000 (00h)

7							0
SC2	SC1	SC0	EXTG	INTG	POW	CONT	ST

b7-b5 = **SC2, SC1, SC0:** *Start Conversion Address.* These 3 bits define the starting analog input in Autoscan mode. The first channel addressed by SC2-SC0 is converted, then the address is incremented for the successive conversion, until channel 7 (111) is converted. The (SC2, SC1, SC0) bits define the group of channels to be scanned. When setting SC2=1 SC1=1 SC0=1 only channel 7 is converted.

b4 = **EXTG:** *External Trigger.* When set to a logical "1", this bit allows to start a group of conversion synchronized on the following edge of the external signal applied on pin ADTRG (when enabled for Alternate Function)..

b3 = **INTG:** *Internal Trigger.* When set to a logical level "1", this bit enables the start of a group of conversion, synchronized with an internal signal (On chip Event signal) from a Multifunction Timer Unit.

Both External and Internal Trigger inputs are internally OR'ed, thus avoiding Hardware conflicts, however the correct procedure is to enable only one alternate synchronization input at time.

b2 = **POW:** *Power Up/Power Down* A logical "1" enables the A/D logic and analog circuitry.
A logical "0" disables all power consuming logic, thus allowing a low power idle status.

b1 = **CONT:** *Continuous/Single.* When this bit is set to "1" (Continuous Mode), the first group of conversions are started either by software (setting to "1" the ST bit) or by hardware (on an Internal or external trigger, depending on the INTG and EXTG bits status), and a continuous conversion flow is then processed.

When this bit is set to "0" (single mode), only a single group of conversions (1 up to 8) are started whenever any External (or Internal) trigger occurs, or the ST bit is set to "1" by software.

The effect of the alternate synchronization is to hardware set the START/STOP bit which is hardware reset when in SINGLE mode, at the end of each group of conversions.

Requirements:

The External Synchronisation Input must receive a pulse (low level) wider than an INTCLK period (83ns) minimum and for both External and On chip Event synchronisation, a period greater than the time required for a group of conversion (number of channels times x 11µs).

REGISTERS (Continued)

b0 = **ST:** *Start/Stop* A logical "1" level enables the starting of a group of conversions; a logical level "0" stops the conversion. When the A/D is running in the Single Mode, this bit is hardware reset at the end of a group of conversions.

14.4.7 Interrupt Control Register (ICR)

This register contains the three priority level bits, the two sources flags, and their bit mask:

ICR **R254** (FEh) Page 63 Read/Write
Interrupt Control Register

Reset Value: 0000 1111 (0Fh)

7							0
ECV	AWD	ECI	AWDI	X	PL2	PL1	PL0

b7 = **ECV:** *End of Conversion.* ECV is automatically set by hardware after a group of conversions is completed.

b6 = **AWD:** *Analog Watchdog.* AWD is automatically hardware set whenever any of the two guarded 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.

AWD and ECV must be reset by the user, before returning from the Interrupt service routine. Setting either of them by software will cause a software interrupt request to be generated.

b5 = **ECI:** *End of Conversion Interrupt Enable*This bit masks the End of Conversion interrupt request. A logical level "1" enables the request, a logical level "0" masks the request.

b4 = **AWDI:** *Analog Watchdog Interrupt Enable.* This bit masks the Analog Watchdog interrupt request. A logical level "1" enables the request, a logical level "0" masks the request.

b3 = **D3:** *Undefined*

b2-b0 = **PL2, PL1, PL0:** *A/D Interrupt Priority Level.* With these three bits it is possible to select the Interrupt priority level of the A/D Converter.

14.4.8 Interrupt Vector Register (IVR)

IVR **R255** (FFh) Page 63 Read/Write
Interrupt Vector Register

Reset Value: xxxx xx10 (x2h)

7							0
V7	V6	V5	V4	V3	V2	W1	D0

b7-b2 = **V7-V2:** *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.

b1 = **W1:** *Word Select.* This bit is set by hardware, according to the A/D interrupt source. It is set to "0" if the source is the Analog Watchdog, pointing to the lower word of the A/D interrupt service block (defined by V7-V2). It is set to "1" if the source is the End of Conversion interrupt, thus pointing to the upper word.

When 2 requests occur simultaneously the Analog watchdog request has priority over the End of Conversion request which is held pending, to be served after the current routine.

b0 = **D0:** This bit is fixed by hardware. It always returns the value "0" if read.

REGISTER MAP

15 ELECTRICAL CHARACTERISTICS

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_{DD}	Supply Voltage	– 0.3 to 7.0	V
V_I	Input Voltage	– 0.3 to V_{DD}+0.3	V
V_O	Output Voltage	– 0.3 to V_{DD}+0.3	V
T_{STG}	Storage Temperature	– 55 to + 150	°C
I_{INJ}	Pin Injection Current Digital Input	-5 to +5	mA
	Maximum Accumulated Pin injection Current in the device	-50 to +50	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}

RECOMMENDED OPERATING CONDITIONS

Symbol	Parameter	Value		Unit
		Min.	Max.	
T_A	Operating Temperature	0	70	°C
V_{DD}	Operating Supply Voltage	4.5	5.5	V
f_{OSCE}	External Oscillator Frequency		24	MHz
f_{OSCI}	Internal Clock Frequency (INTCLK)		12	MHz

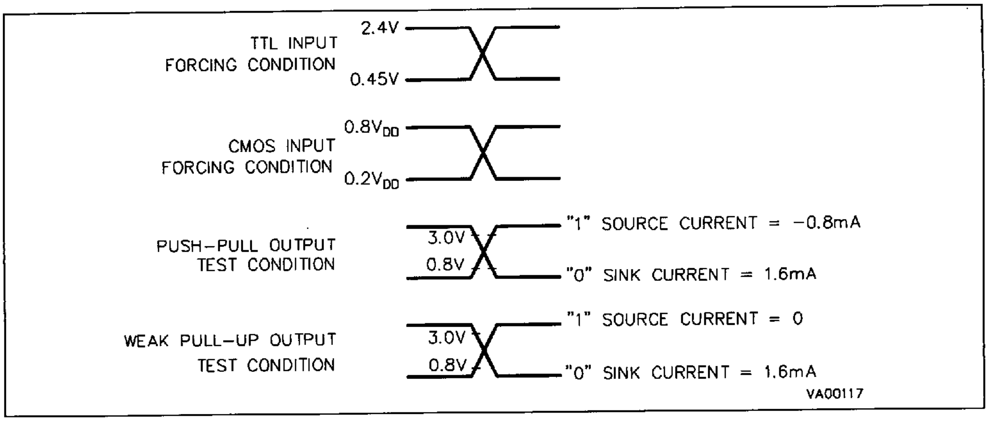
DC ELECTRICAL CHARACTERISTICS

V_{DD} = 5V ± 10% T_A = 0°C to + 70°C, unless otherwise specified)

Symbol	Parameter	Test Conditions	Value			Unit
			Min.	Typ.	Max.	
V_{IHCK}	Clock Input High Level	External Clock	0.7 V_{DD}		V_{DD} + 0.3	V
V_{ILCK}	Clock Input Low Level	External Clock	– 0.3		0.3 V_{DD}	V
V_{IH}	Input High Level	TTL	2.0		V_{DD} + 0.3	V
		CMOS	0.7 V_{DD}		V_{DD} + 0.3	V
V_{IL}	Input Low Level	TTL	– 0.3		0.8	V
		CMOS	– 0.3		0.3 V_{DD}	V
V_{IHRS}	$\overline{\text{RESET}}$ Input High Level		0.7 V_{DD}		V_{DD} + 0.3	V
V_{ILRS}	$\overline{\text{RESET}}$ Input Low Level		–0.3		0.3 V_{DD}	V
V_{HYRS}	$\overline{\text{RESET}}$ Input Hysteresis		0.3		1.5	V
V_{OH}	Output High Level	Push Pull, Iload = – 0.8mA	V_{DD} – 0.8			V
V_{OL}	Output Low Level	Push Pull or Open Drain, Iload = 1.6mA			0.4	V
I_{WPU}	Weak Pull-up Current	Bidirectional Weak Pull-up, V_{OL} = 0V	– 50	– 200	– 420	µA
I_{APU}	Active Pull-up Current, for INT0 and INT7 only	V_{IN} < 0.8V, under Reset	– 80	– 200	– 420	µA
I_{LKIO}	I/O Pin Input Leakage	Input/Tri-State, 0V < V_{IN} < V_{DD}	– 10		+ 10	µA
I_{LKRS}	$\overline{\text{RESET}}$ Pin Input Leakage	0V < V_{IN} < V_{DD}	– 30		+ 30	µA
I_{LKAP}	Active Pull-up Input Leakage	0V < V_{IN} < 0.8V	– 10		+ 10	µA
I_{LKOS}	OSCIN Pin Input Leakage	0V < V_{IN} < V_{DD}	– 10		+ 10	µA

Note: All I/O Ports are configured in Bidirectional Weak Pull-up Mode with no DC load, External Clock pin (OSCIN) is driven by square wave external clock. No peripheral working.

DC TEST CONDITIONS



TTL INPUT FORCING CONDITION
2.4V
0.45V
CMOS INPUT FORCING CONDITION
0.8VDD
0.2VDD
PUSH-PULL OUTPUT TEST CONDITION
3.0V
0.8V
"1" SOURCE CURRENT = –0.8mA
"0" SINK CURRENT = 1.6mA
WEAK PULL-UP OUTPUT TEST CONDITION
3.0V
0.8V
"1" SOURCE CURRENT = 0
"0" SINK CURRENT = 1.6mA
VA00117

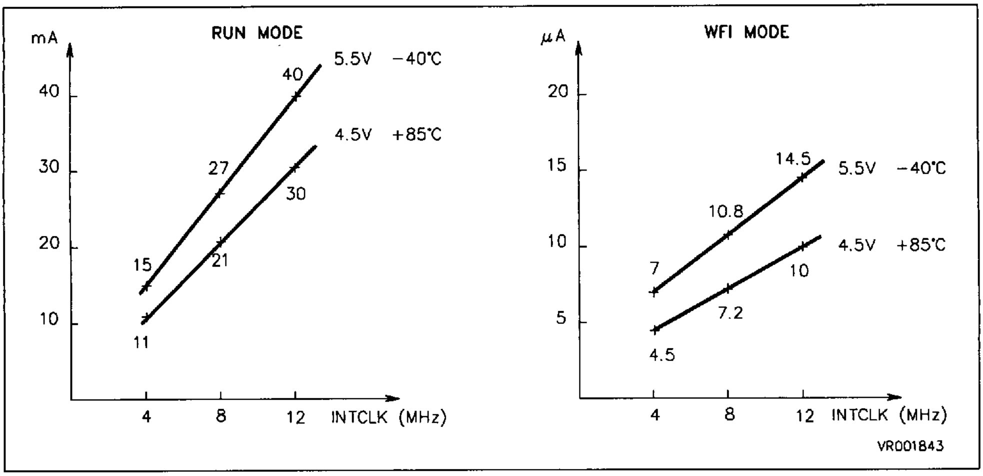
AC ELECTRICAL CHARACTERISTICS

V_{DD} = 5V ± 10% T_A = 0°C to + 70°C, unless otherwise specified)

Symbol	Parameter	Test Conditions	Value			Unit
			Min.	Typ.	Max.	
I_{DD}	Run Mode Current no CPUCLK prescale, Clock divide by 2	24MHz, Note 1		40	70	mA
I_{DP2}	Run Mode Current Prescale by 2, Clock divide by 2	24MHz, Note 1		19	40	mA
I_{WFI}	WFI Mode Current no CPUCLK prescale, Clock divide by 2	24MHz, Note 1		15	20	mA
I_{HALT}	HALT Mode Current	24MHz, Note 1		50	100	µA

Note 1 : All I/O Ports are configured in Bidirectional Weak Pull-up Mode with no DC load, External Clock pin (OSCIN) is driven by square wave external clock. No peripheral working.

Typical Current Versus Frequency of Operation (fosc)



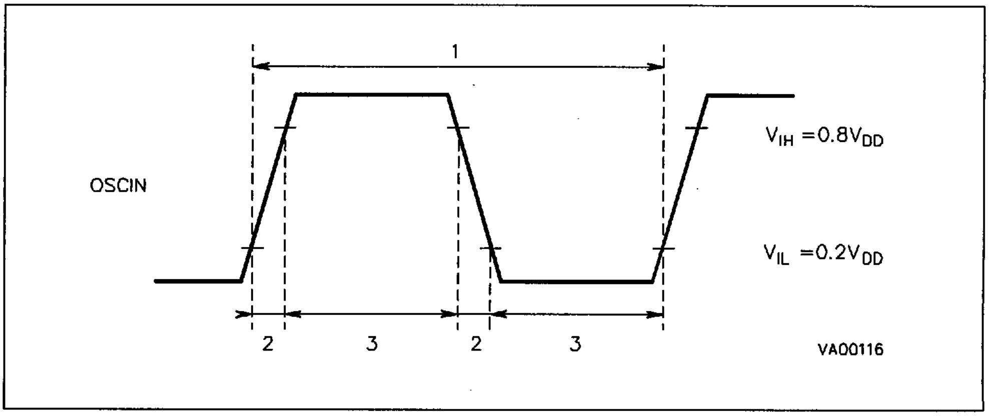
CLOCK TIMING TABLE

(V_{DD} = 5V ± 10%, T_A = 0°C to + 70°C, INTCLK = 12MHz, unless otherwise specified

N°	Symbol	Parameter	Value		Unit	Note
			Min.	Max.		
1	TpC	OSCIN Clock Period	41.5		ns	1
			83		ns	2
2	TrC, TfC	OSCIN Rise and Fall Time		12	ns	
3	TwCL, TwCH	OSCIN Low and High Width	17	25	ns	1
			38		ns	2

Notes:
1. Clock divided by 2 internally (MODER.DIV2=1)
2. Clock not divided by 2 internally (MODER.DIV2=0)

CLOCK TIMING



EXTERNAL BUS TIMING TABLE (V_{DD} = 5V ± 10%, T_A = 0°C to + 70°C, Cload = 50pF, CPUCLK = 12MHz, unless otherwise specified)

N°	Symbol	Parameter	Value (Note)				Unit
			OSCIN Divided By 2	**OSCIN Not Divided By 2**	**Min.**	**Max.**	
1	TsA (AS)	Address Set-up Time before $\overline{\text{AS}}$ ↑	TpC (2P+1) –22	TWCH+PTpC –18	20		ns
2	ThAS (A)	Address Hold Time after $\overline{\text{AS}}$ ↑	TpC –17	TwCL –13	25		ns
3	TdAS (DR)	$\overline{\text{AS}}$ ↑ to Data Available (read)	TpC (4P+2W+4) –52	TpC (2P+W+2) –51		115	ns
4	TwAS	$\overline{\text{AS}}$ Low Pulse Width	TpC (2P+1) –7	TwCH+PTpC –3	35		ns
5	TdAz (DS)	$\overline{\text{DS}}$ ↓ to Address Float			12		ns
6	TwDSR	$\overline{\text{DS}}$ Low Pulse Width (read)	TpC (4P+2W+3) –20	TwCH+TpC (2P+W+1) –16	105		ns
7	TwDSW	$\overline{\text{DS}}$ Low Pulse Width (write)	TpC (2P+2W+2) –13	TpC (P+W+1) –13	70		ns
8	TdDSR (DR)	$\overline{\text{DS}}$ ↓ to Data Valid Delay (read)	TpC (4P+2W-3) –50	TwCH+TpC(2P+W+1) –46		75	ns
9	ThDR (DS)	Data to $\overline{\text{DS}}$ ↑ Hold Time (read)	0	0	0		ns
10	TdDS (A)	$\overline{\text{DS}}$ ↑ to Address Active Delay	TpC –7	TwCL –3	35		ns
11	TdDS (AS)	$\overline{\text{DS}}$ ↑ to $\overline{\text{AS}}$ ↓ Delay	TpC –18	TwCL –14	24		ns
12	TsR/W (AS)	R/$\overline{\text{W}}$ Set-up Time before $\overline{\text{AS}}$ ↑	TpC (2P+1) –22	TwCH+PTpC –18	20		ns
13	TdDSR (R/W)	$\overline{\text{DS}}$ ↑ to R/$\overline{\text{W}}$ and Address Not Valid Delay	TpC –9	TwCL –5	33		ns
14	TdDW (DSW)	Write Data Valid to $\overline{\text{DS}}$ ↓ Delay (write)	TpC (2P+1) –32	TwCH+PTpC –28	10		ns
15	ThDS (DW)	Data Hold Time after $\overline{\text{DS}}$ ↑ (write)	TpC –9	TwCL –5	33		ns
16	TdA (DR)	Address Valid to Data Valid Delay (read)	TpC (6P+2W+5) –68	TwCH+TpC (3P+W+2) –64		140	ns
17	TdAs (DS)	$\overline{\text{AS}}$ ↑ to $\overline{\text{DS}}$ ↓ Delay	TpC –18	TwCL –14	24		ns

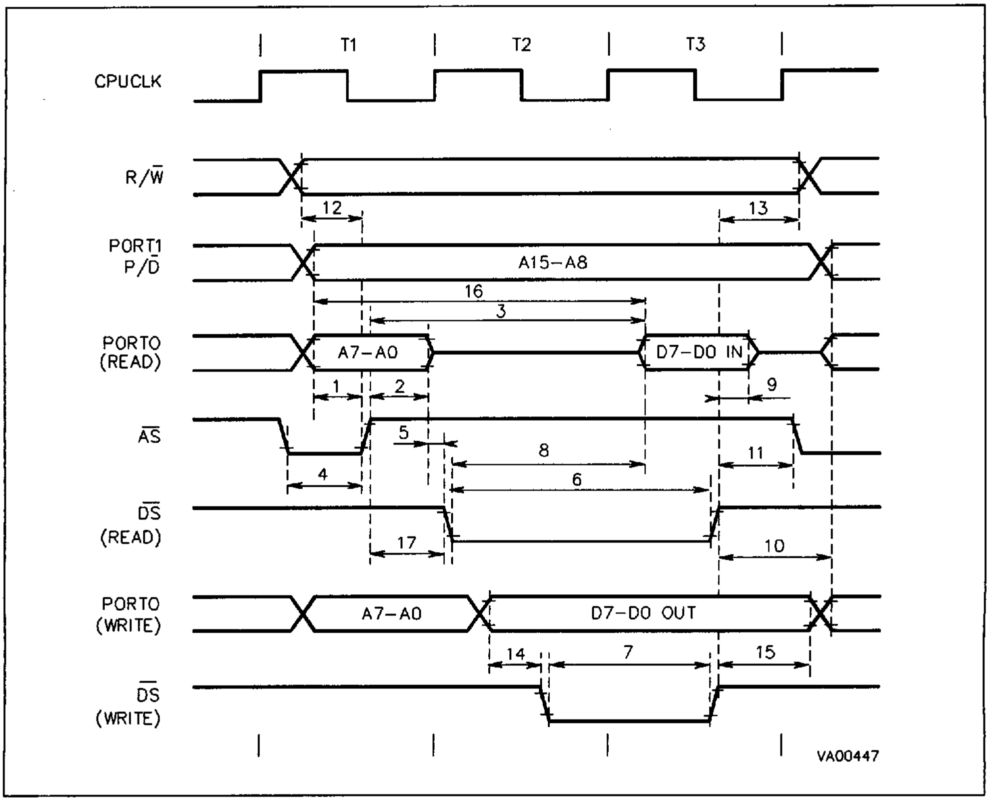
EXTERNAL WAIT TIMING TABLE (V_{DD} = 5V ± 10%, T_A = 0°C to +70°C, Cload = 50pF, INTCLK = 12MHz, Push-pull output configuration, unless otherwise specified)

N°	Symbol	Parameter	Value (Note)				Unit
			OSCIN Divided By 2	**OSCIN Not Divided By 2**	**Min.**	**Max.**	
1	TdAs (WAIT)	$\overline{\text{AS}}$ ↑ to $\overline{\text{WAIT}}$ ↓ Delay	2(P+1)TpC –29	2(P+1)TpC –29		40	ns
2	TdAs (WAIT)	$\overline{\text{AS}}$ ↑ to $\overline{\text{WAIT}}$ ↓ Min. Delay	2(P+W+1)TpC –4	2(P+W+1)TpC –4	80		ns
3	TdAs (WAIT)	$\overline{\text{AS}}$ ↑ to $\overline{\text{WAIT}}$ ↓ Max. Delay	2(P+W+1)TpC –29	2(P+W+1)TpC –29		83W+ 40	ns

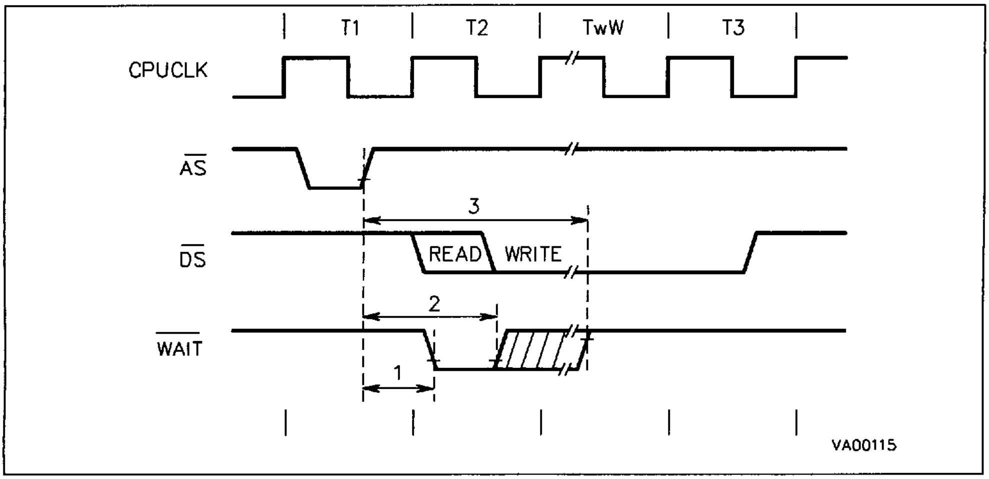
Note: (for both tables) The value in the left hand two columns show the formula used to calculate the timing minimum or maximum from the oscillator clock period, prescale value and number of wait cycles inserted.
The value in the right hand two columns show the timing minimum and maximum for an external clock at 24 MHz divided by 2, prescaler value of zero and zero wait status.

Legend:	TpC =OSCIN Period
P = Clock Prescaling Value	TwCH =High Level OSCIN half period
W = Wait Cycles	TwCL =Low Level OSCIN half period

EXTERNAL BUS TIMING



EXTERNAL WAIT TIMING

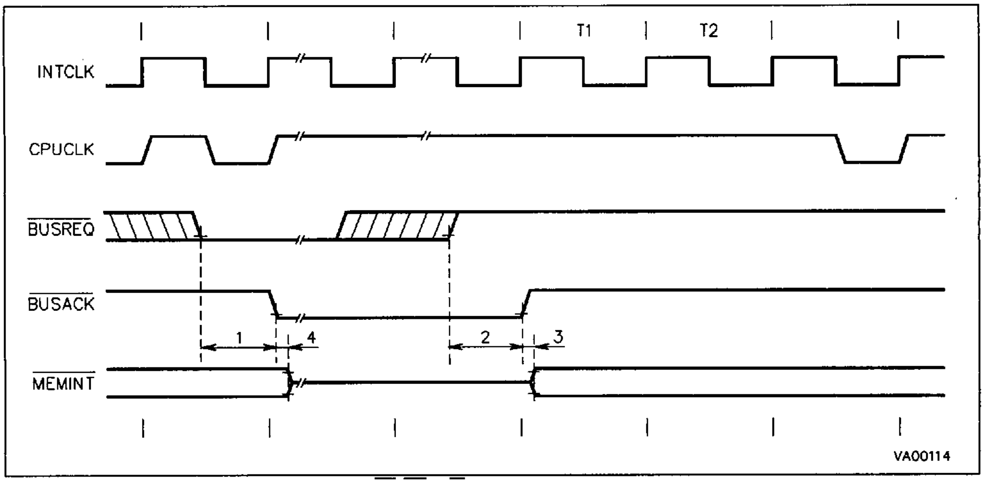


BUS REQUEST/ACKNOWLEDGE TIMING TABLE (V$_{DD}$ = 5V ± 10%, T$_A$ = 0°C to +70°C, Cload = 50pF, INTCLK = 12MHz, Push-pull output configuration, unless otherwise specified)

N°	Symbol	Parameter	Value (Note)				Unit
			OSCIN Divided By 2	OSCIN Not Divided By 2	Min.	Max.	
1	TdBR (BACK)	$\overline{\text{BREQ}}$↓ to $\overline{\text{BUSACK}}$↓	TpC+8	TwCL+12	50		ns
			TpC(6P+2W+7)+65	TpC(3P+W+3)+TwCL+65		360	ns
2	TdBR (BACK)	$\overline{\text{BREQ}}$↑ to $\overline{\text{BUSACK}}$↑	3TpC+60	TpC+TwCL+60		185	ns
3	TdBACK (BREL)	$\overline{\text{BUSACK}}$↓ to Bus Release	20	20		20	ns
4	TdBACK (BACT)	$\overline{\text{BUSACK}}$↑ to Bus Active	20	20		20	ns

Note: The value left hand two columns show the formula used to calculate the timing minimum or maximum from the oscillator clock period, prescale value and number of wait cycles inserted.
The value right hand two columns show the timing minimum and maximum for an external clock at 24MHz divided by 2, prescale value of zero and zero wait status.

BUS REQUEST/ACKNOWLEDGE TIMING


T1
T2
INTCLK
CPUCLK
$\overline{\text{BUSREQ}}$
$\overline{\text{BUSACK}}$
1
4
2
3
$\overline{\text{MEMINT}}$
VA00114

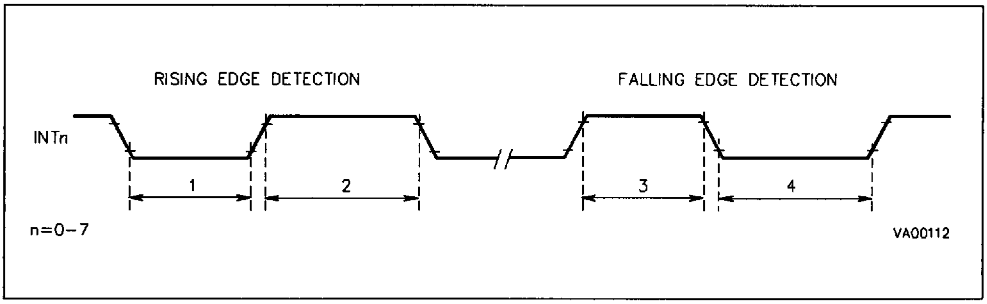
Note : MEMINT = Group of memory interface signals: $\overline{\text{AS}}$, $\overline{\text{DS}}$, R/$\overline{\text{W}}$, P00-P07, P10-P17

EXTERNAL INTERRUPT TIMING TABLE (V_{DD} = 5V ± 10%, T_A = 0°C to +70°C, Cload = 50pF, INTCLK = 12MHz, Push-pull output configuration, unless otherwise specified)

N°	Symbol	Parameter	Value (Note)				Unit
			OSCIN Divided By 2 Min.	OSCIN Not Divided By 2 Min.	Min.	Max.	
1	TwLR	Low Level Minimum Pulse Width in Rising Edge Mode	2TpC+12	TpC+12	95		ns
2	TwHR	High Level Minimum Pulse Width in Rising Edge Mode	2TpC+12	TpC+12	95		ns
3	TwHF	High Level Minimum Pulse Width in Falling Edge Mode	2TpC+12	TpC+12	95		ns
4	TwLF	Low Level Minimum Pulse Width in Falling Edge Mode	2TpC+12	TpC+12	95		ns

Note: The value left hand two columns show the formula used to calculate the timing minimum or maximum from the oscillator clock period, prescale value and number of wait cycles inserted.
The value right hand two columns show the timing minimum and maximum for an external clock at 24 MHz divided by 2, prescale value of zero and zero wait status.

EXTERNAL INTERRUPT TIMING

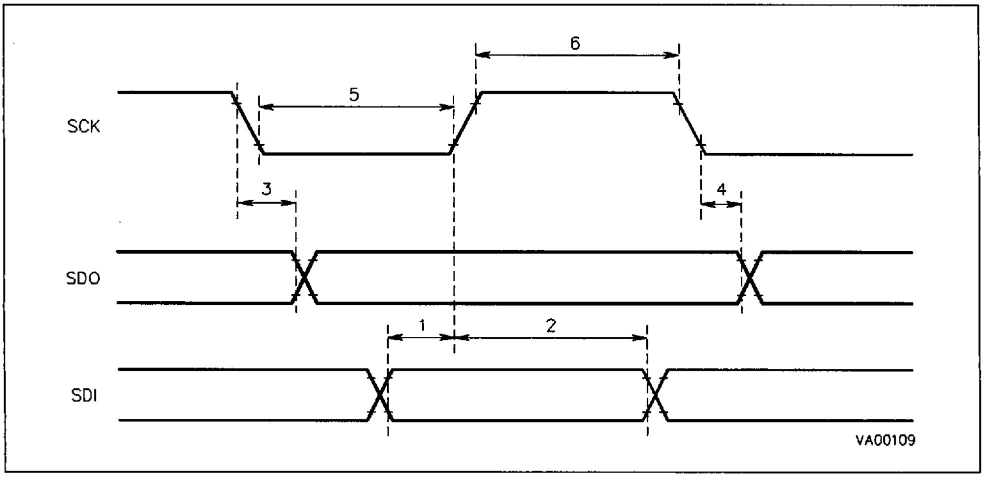


SPI TIMING TABLE (V_{DD} = 5V ± 10%, T_A = 0°C to +70°C, Cload = 50pF, INTCLK = 12MHz, Output Alternate Function set as Push-pull)

N°	Symbol	Parameter	Value		Unit
			Min.	Max.	
1	TsDI	Input Data Set-up Time	100		ns
2	ThDI (1)	Input Data Hold Time	1/2 TpC+100		ns
3	TdOV	SCK to Output Data Valid		100	ns
4	ThDO	Output Data Hold Time	-20		ns
5	TwSKL	SCK Low Pulse Width	300		ns
6	TwSKH	SCK High Pulse Width	300		ns

Note: TpC is the OSCIN Clock period.

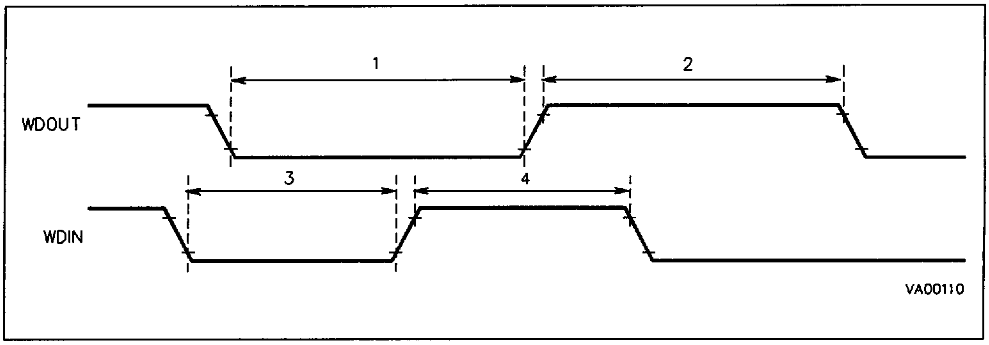
SPI TIMING



WATCHDOG TIMING TABLE (V_{DD} = 5V ± 10%, T_A = 0°C to +70°C, Cload = 50pF, INTCLK = 12MHz, Push-pull output configuration, unless otherwise specified)

N°	Symbol	Parameter	Values		Unit
			Min.	Max.	
1	TwWDOL	WDOUT Low Pulse Width	620		ns
2	TwWDOH	WDOUT High Pulse Width	620		ns
3	TwWDIL	WDIN High Pulse Width	350		ns
4	TwWDIH	WDIN Low Pulse Width	350		ns

WATCHDOG TIMING

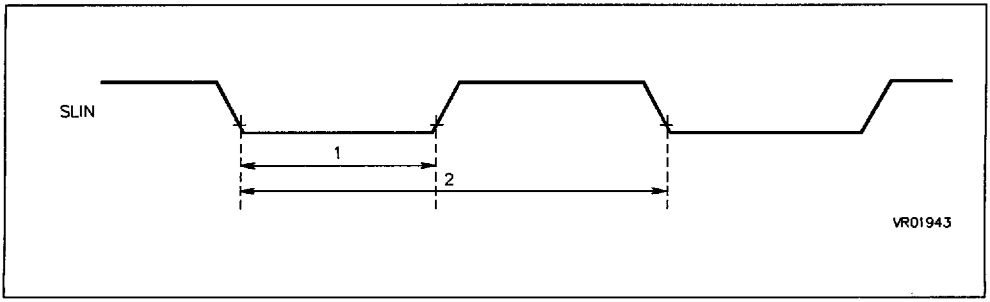


SLICE TIMER TIMING TABLE (V_{DD} = 5V ± 10%, T_A = 0°C to +70°C, Cload = 50pF, INTCLK = 12MHz, Push-pull output configuration, unless otherwise specified)

N°	Symbol	Parameter	Value (Note)				Unit
			OSCIN Divided By 2 Min.	OSCIN Not Divided By 2 Min.	Min.	Max.	
1	TwSLPW	Pulse Width	4TpC+15	2TpC+15	190		ns
2	TwSLDF	Separation between falling edges	8TpC+30	4TpC+30	380		ns

Note: TpC = OSCIN Clock period

SLICE TIMER TIMING



7929237 0058078 TTO

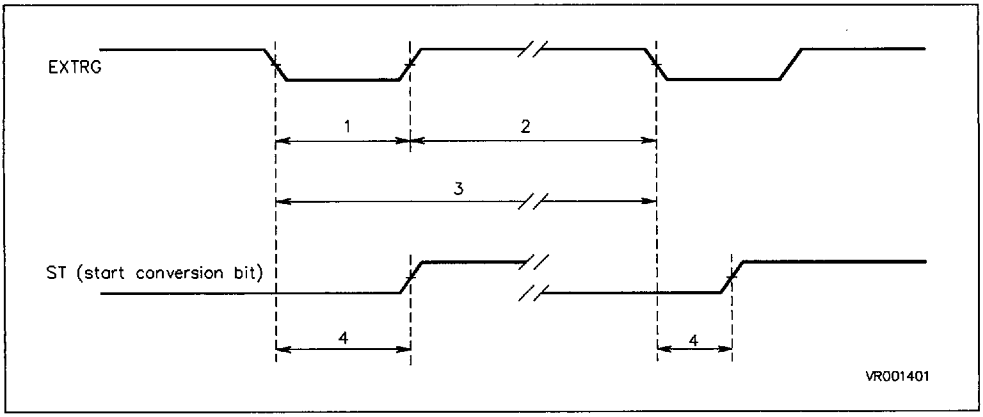
A/D CONVERTER

EXTERNAL TRIGGER TIMING (V_{DD} = 5V ± 10%, T_A = - 0°C to + 70°C, Cload = 50pF)

N°	Symbol	Parameter	Oscin divided by 2 [1]		Oscin not divided [1]		Value[2]		Unit
			Min.	Max.	Min.	Max.	Min.	Max.	
1	T_{LOW}	External Trigger pulse width	2xTpC		TpC		83		ns
2	T_{HIGH}	External Trigger pulse distance	2xTpC		TpC		83		ns
3	T_{EXT}	External trigger active edges distance	138xTpC		68xTpC		4.5		µs
4	T_{STR}	Internal delay between EXTRG falling edge and first conversion start	TpC	3xTpC	0.5xTpC	1.5xTpC	41.5	125	ns

Notes:
1. Variable clock (TpC=OSCIN clock period)
2. CPUCLK=12MHz

A/D External Trigger Timing



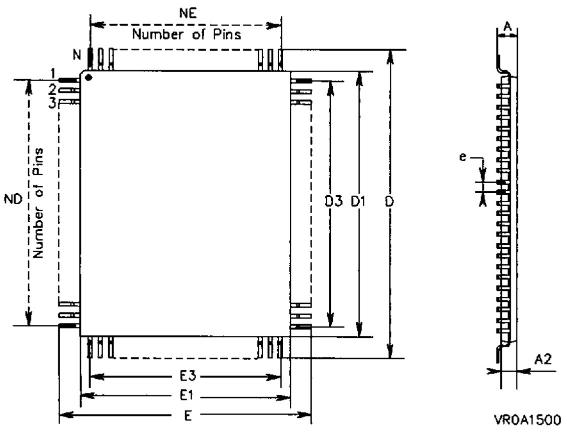
ANALOG SPECIFICATIONS (V_{DD} = 5V ± 10%, T_A = +25°C)

Parameter	Min.	Typical	Max.	Units [1]	Notes
An. Input Range			AV_{CC}	V	
Conversion Time	4.5-5.5			µs	(2,3)
Sample Time	1.5			µs	(2)
Power-up Time	60			µs	
Resolution	6-8	6-8		bit	
Monotonicity	GUARANTEED				
No missing Codes	GUARANTEED				
Zero Input reading	00				
Full scale reading			FC-FF	Hex	
Offset Error	TBD			LSBs	
Gain Error	-1		+1	LSBs	
Diff. Non lin.	-1		+1	LSBs	
Int Non Lin.	-1		+1	LSBs	
Absolute Accuracy	TBD			LSBs	
S/N	TBD			dB	
Input Resistance	8	12	15	KΩ	(4)
Hold capacitance Max.			4-15	pF	(5)
Input Leakage			3	mA	

Notes:
1. "LSBs", as used here, has a value of AV_{CC}/64 or /256
2. At 24MHz external clock
3. Including sample time
4. It must be intended as the internal series resistance before the sampling capacitor
5. For 6 or 8 bit converter respectively.

PACKAGE MECHANICAL DATA

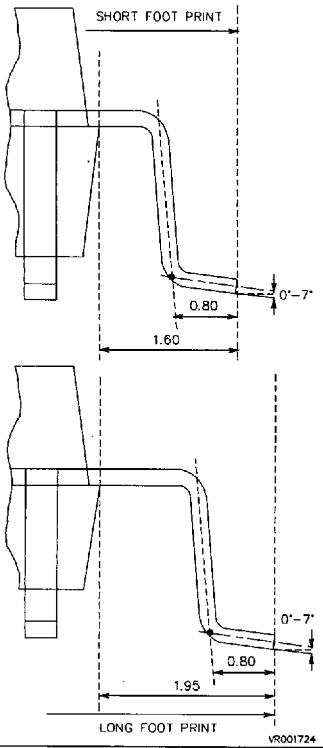
80-Pin Plastic Quad Flat Package



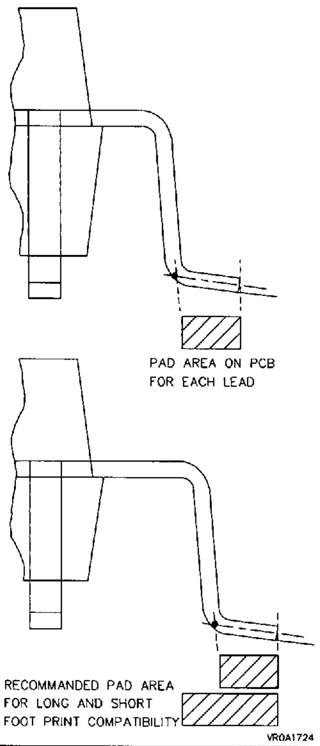
Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A			3.30			0.130
A2	2.55	2.80	3.05	0.100	0.110	0.120
D*	23.65	23.90	24.15	0.931	0.941	0.951
D1	19.90	20.00	20.10	0.783	0.787	0.791
D3		18.40			0.724	
E*	17.65	17.90	18.15	0.695	0.705	0.715
E1	13.90	14.00	14.10	0.547	0.551	0.555
E3		12.00			0.472	
e		0.80			0.032	
Number of Pins						
N	80					
ND	24					
NE	16					

* Subject to change,
Typ. D will change to 23.20mm
Typ. E will change to 17.20mm

Short/Long Footprint Measurement

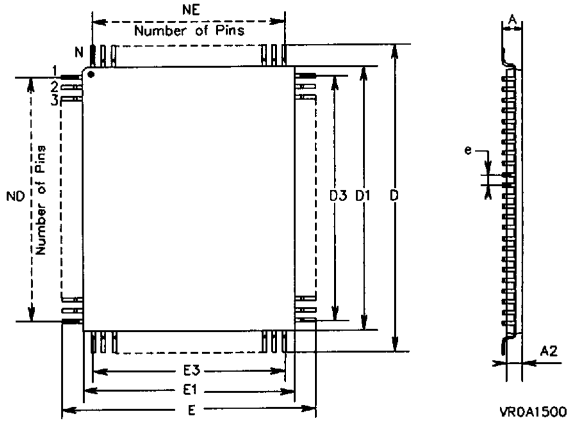


Short/Long Footprint Recommended Padding



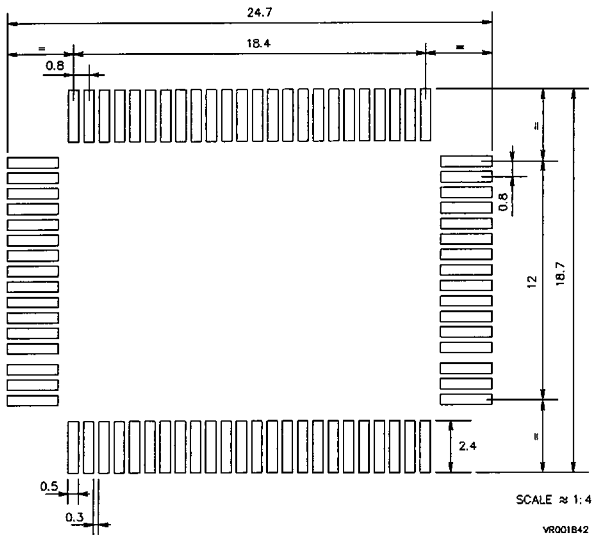
PACKAGE MECHANICAL DATA (Continued)

68-Pin Plastic Leadless Chip Carrier Package (PLCC68)



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A	4.20		5.08	0.165		0.200
A1	0.51			0.020		
A3	2.29		3.30	0.090		0.130
B	-	-	-	-	-	-
B1	-	-	-	-	-	-
D	25.02		25.27	0.985		0.995
D1	24.13		24.33	0.950		0.958
D3		20.32			0.800	
E	25.02		25.27	0.985		0.995
E1	24.13		24.33	0.950		0.958
E3		20.32			0.800	
K1	-	-	-	-	-	-
h						
e		1.27			0.050	
Number of Pins						
N	68					
ND	17					

Solder Pad Footprint Recomended For QFP80 Package



ORDERING INFORMATION

Sales Type	Frequency	Temperature Range	Package
ST90R91Q1	24MHz	0°C to + 70°C	PQFP80
ST90R91C1			PLCC68