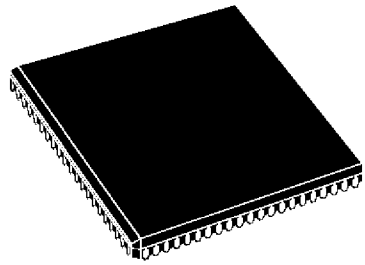
SGS-THOMSON MICROELECTRONICS

# ST90158 - ST90135

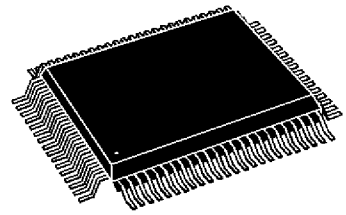
## 8/16-BIT MCU FAMILY WITH 16 to 64K ROM and 512 to 2K RAM

**PRELIMINARY DATA**

- Register File based 8/16 bit Core Architecture with RUN, WFI, SLOW and HALT modes
- 0 - 16 MHz Operation @ 5V±10%
  0 - 12 MHz Operation @ 3V±10%
- -40°C to +85°C and 0°C to +70°C Operating Temperature Ranges
- Fully Programmable PLL Clock Generator, with Frequency Multiplication and low frequency, low cost external crystal
- Minimum 8-bit Instruction Cycle time: 250ns - (@ 16MHz internal clock frequency)
- Minimum 16-bit Instruction Cycle time: 375ns - (@ 16MHz internal clock frequency)
- Internal Memory:
  – EPROM/OTP/ROM 16/24/32/48/64K bytes
  – RAM 512/768/1K/1.5K/2K bytes
- 224 general purpose registers available as RAM, accumulators or index pointers (register file)
- 84-pin Plastic Leaded Chip Carrier and 80-pin Plastic Quad Flat Package
- 72/67 fully programmable I/O bits
- 8 external and 1 Non-Maskable Interrupts
- DMA Controller and Programmable Interrupt Handler
- Single Master Serial Peripheral Interface
- Two 16-bit Timers with 8-bit Prescaler, one able to be used as a Watchdog Timer (software and hardware)
- Three (ST90158) or two (ST90135) 16-bit Multifunction Timers, each with an 8 bit prescaler, 12 operating modes and DMA capabilities
- 8 channel 8-bit Analog to Digital Converter, with Automatic voltage monitoring capabilities and external reference inputs
- Two (ST90158) or one (ST90135) Serial Communication Interfaces with asynchronous, synchronous and DMA capabilities
- Rich Instruction Set with 14 Addressing modes
- Division-by-Zero trap generation
- Versatile Development Tools, including Assembler, Linker, C-compiler, Archiver, Source Level Debugger and Hardware Emulators with Real-Time Operating System available from Third Parties



**PLCC84 (ST90158 only)**



**PQFP80 (all)**

See ordering Information in Table 4

### DEVICE SUMMARY

| DEVICE | Program Memory (Bytes) | RAM (Bytes) | MFT | SCI | PACKAGE |
|---|---|---|---|---|---|
| ST90135 | 16K ROM | 512 | 2 | 1 | PQFP80 |
| | 24K ROM | 768 | 2 | 1 | |
| | 32K ROM | 1K | 2 | 1 | |
| ST90158 | 48K ROM | 1.5K | 3 | 2 | |
| | 64k ROM | 2K | 3 | 2 | PLCC84/ PQFP80 |
| ST90E158 | 64K EPROM | 2K | 3 | 2 | CLCC84/ CQFP80 |
| ST90T158 | 64K OTP | 2K | 3 | 2 | PLCC84/ PQFP80 |

Rev. 2.2

# Table of Contents

# Table of Contents

# Table of Contents

# 1 GENERAL INFORMATION

## 1.1 THE ST9+ FAMILY OF MCUs

The ST9+ Family is one of the most powerful range of 8/16-bit MCUs available on the market. Its performance derives from the use of a sophisticated and flexible Register-File based Core optimized for sophisticated data manipulation and Real-Time event handling, together with a range of application-aware intelligent peripherals having the power to carry out most tasks with the minimum processor overhead.

The distributed intelligence between the ST9+ Core and its Peripherals, as well as the capability of exchanging data via Direct Memory Access (DMA) channels and the sophisticated programmable multiple-priority interrupt channels, requires a full and correct understanding of the device architecture.

The user will find that thorough familiarisation with the operational and architectural features of the ST9+ will be well rewarded in terms of application efficiency and integrity.

To use the ST9+ effectively, and to avoid the most common programming mistakes, users should pay particular attention to the following comments:

**The ST9+ is a Register Based machine:** The configuration of the CPU, of the peripherals and of the I/O ports, as well as the selection of the operating modes, all depend on the correct programming of the entire set of registers. Special care should be taken to ensure that, following execution of the initialization routine, no register, even those associated with unused peripherals or functions, is unprogrammed.

The ST9+ MCU Family devices offer probably the greatest possible flexibility on the market: it is therefore essential to carefully cross-check the initialisation setting of each and every register.

**The ST9+ is a paged machine:** A paging mechanism controls the access to the peripheral pages, to the current Working Register group.

At any location of the software, a user should have perfect control of the value of his pointers. This can be achieved by systematically safeguarding the complete context of the application in the stack prior to any sub-routine call that might affect the paged registers.

## 1.2 DESCRIPTION OF THE ST90158/ST90135

The ST90158 & ST90135 are members of the ST9+ family of microcontrollers, completely developed and produced by SGS-THOMSON Microelectronics using a proprietary n-well HCMOS process.

The nucleus of these MCUs is the advanced Core which includes the Central Processing Unit (CPU), the Register File, the interrupt and DAM controller, and the Memory Management Unit. 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 ST90158 & ST90135 with 67/72 I/O lines dedicated to digital Input/Output. These lines are grouped into up to 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. Two memory spaces are available to support this wide range of configurations: a combined Program/Data Memory Space and the internal Register File, which includes the control and status registers of the on-chip peripherals.

Three (ST90158) or two (ST90135) 16-bit Multi-Function Timers, each with an 8 bit Prescaler and 13 operating modes allow 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 8 channel Analog to Digital Converter with integral sample and hold, fast conversion time and 8-bit resolution. An automatic voltage monitoring feature is included for two input channels.

Completing the device are two (ST90158) or one (ST90135) full duplex Serial Communications Interfaces with an integral generator, asynchronous and synchronous capability (fully programmable format) and associated address/wake-up option, plus two DMA channels.

Finally, a programmable PLL Clock Generator allows the usage of standard 3 to 5 MHz crystals to obtain a large range of internal frequencies up to 16MHz. Low power Run and Wait for interrupt modes are also available.

**Figure 1. ST90158 Block Diagram**

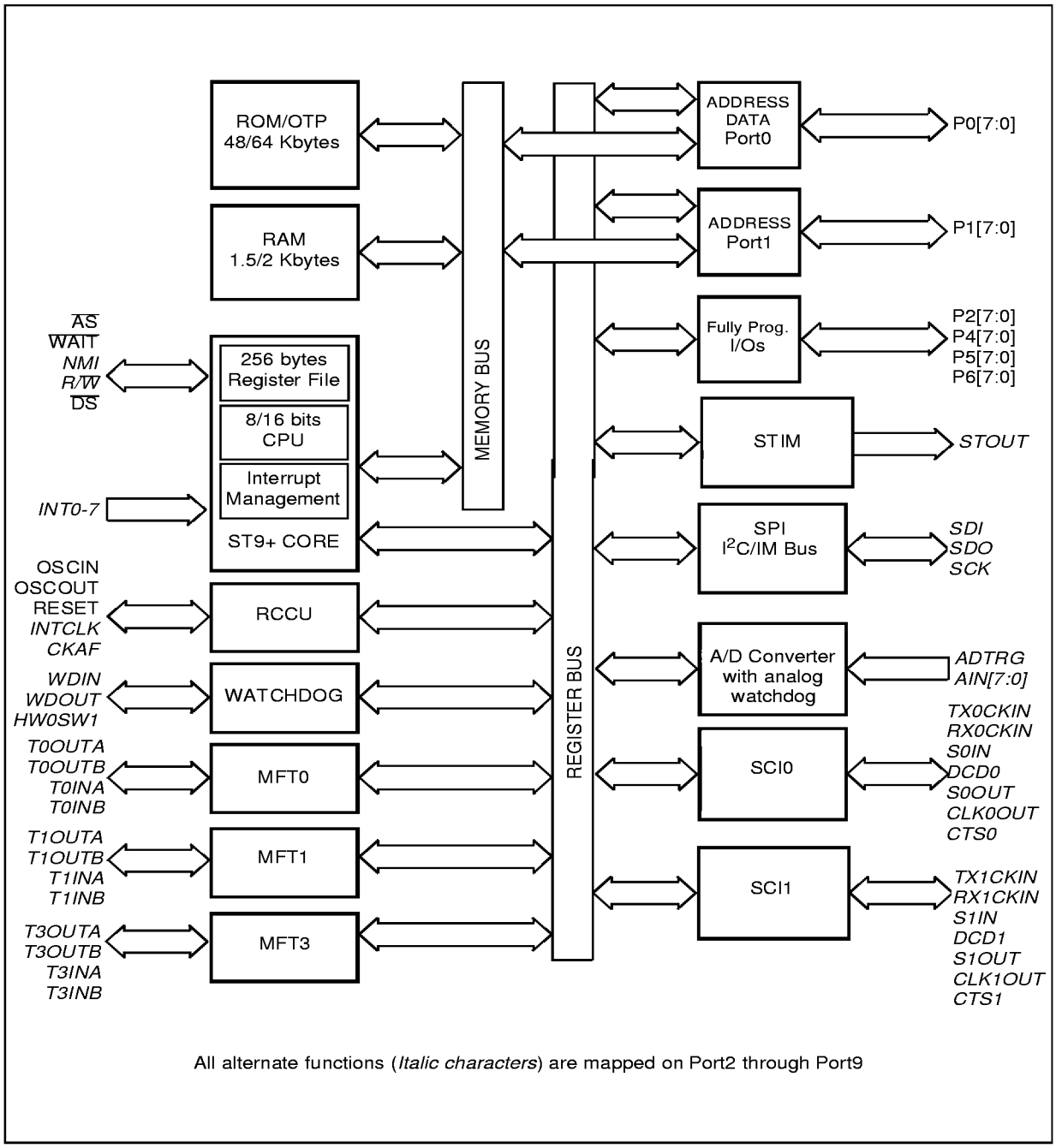
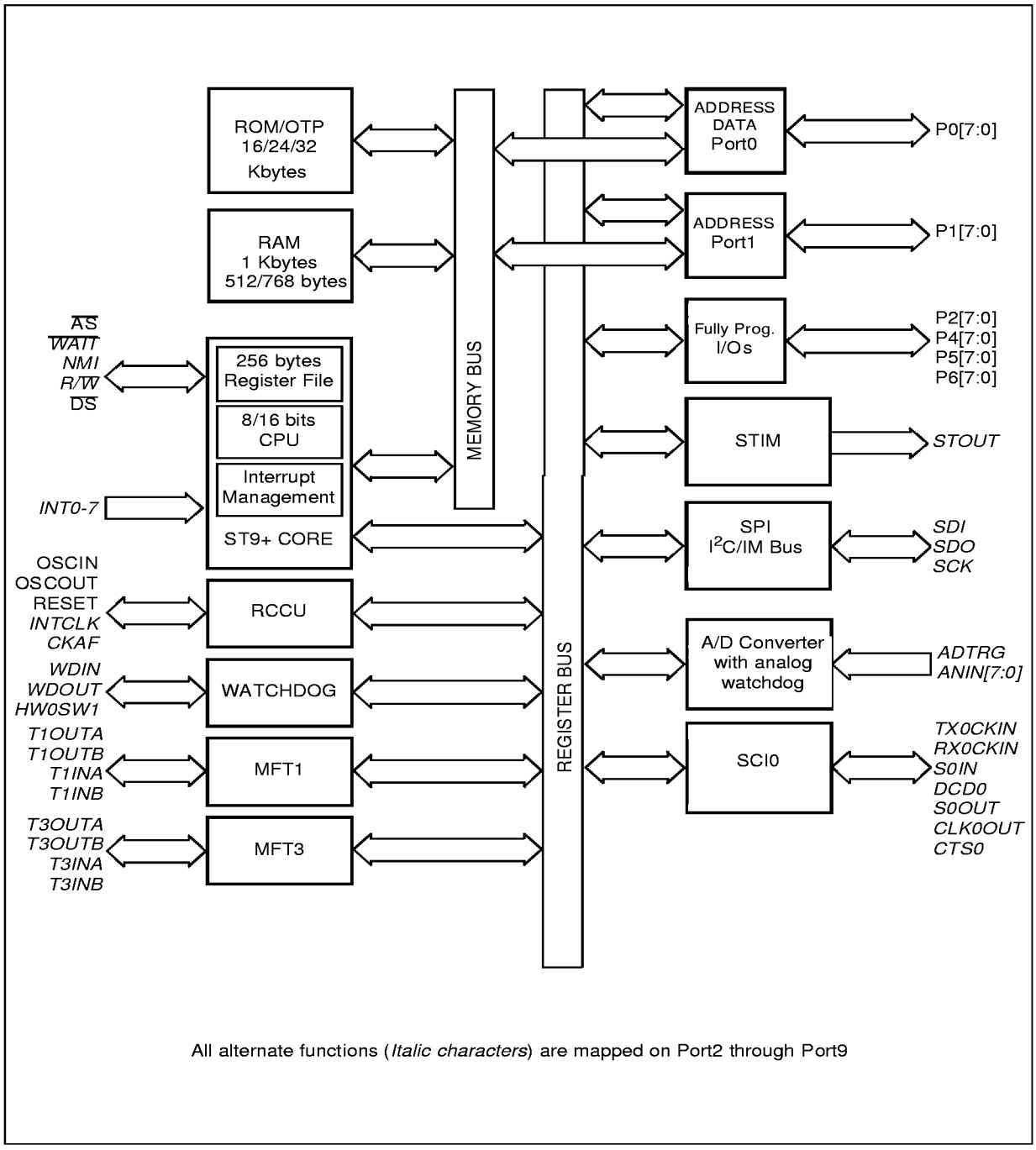


**Figure 2. ST90135 Block Diagram**



### 1.3 PIN DESCRIPTION

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

**$\overline{DS}$**. 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 $\overline{DS}$. During a read cycle, Data In must be valid prior to the trailing edge of $\overline{DS}$. When the ST90158 accesses on-chip memory, $\overline{DS}$ is held high during the whole memory cycle. It can be placed in a high impedance state along with Port 0, Port 1 and $\overline{AS}$.

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

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

**OSCIN, OSCOUT**. Oscillator (input and output). These pins connect a parallel-resonant crystal (3 to 5 MHz), 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.

**HW0_SW1.** When connected to $V_{DD}$ through a 1K pull-up resistance, the software watchdog option is selected. When connected to $V_{SS}$ through a 1K pull-down resistance, the hardware watchdog option is selected.

**$AV_{DD}$**. Analog $V_{DD}$ of the Analog to Digital Converter.

**$AV_{SS}$**. Analog $V_{SS}$ of the Analog to Digital Converter.

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

**$V_{SS}$**. Digital Circuit Ground.

**P0.0-P0.7, P1.0-P1.7,** (*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,P6.0-P6.7, P7.0-P7.7, P8.0-P8.7, P9.0-P9.7** *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.

#### 1.3.1 I/O Port Alternate Functions

Each pin of the I/O ports of the ST90158 may assume software programmable Alternate Functions as shown in the Pin Configuration Drawings.

**Table 1. PLCC Pin Description (ST90158 only)**

| Pin | Name |
|---|---|
| 75 | P8.6/INT7/T3OUTA |
| 76 | P8.5/P/$\overline{D}$/T3INB |
| 77 | P8.4/T1INA /$\overline{WAIT}$/WDOUT |
| 78 | P8.3/T1OUTB/INT3 |
| 79 | P8.2/T1OUTA/INT1 |
| 80 | P8.1/T1INB |
| 81 | P8.0/T3INA |
| 82 | $V_{DD}$ |
| 83 | P5.7/T3OUTB/CTS1 /CKAF |
| 84 | P5.6/T3OUTA/DCD1 |
| ●1 | P5.5/T1OUT1/CTS0 |
| 2 | P5.4/T1OUTA/DCD0 |
| 3 | P5.3/P/$\overline{D}$ |
| 4 | P5.2 |
| 5 | P5.1/SDI |
| 6 | P5.0 |
| 7 | OSCOUT |
| 8 | $V_{SS}$ |
| 9 | OSCIN |
| 10 | RESET |
| 11 | P1.0/A8 |

| Pin | Name |
|---|---|
| 12 | P1.1/A9 |
| 13 | P1.2/A10 |
| 14 | P1.3/A11 |
| 15 | P1.4/A12 |
| 16 | P1.5/A13 |
| 17 | P1.6/A14 |
| 18 | P1.7/A15 |
| 19 | P6.0 |
| 20 | P6.1 |
| 21 | P6.2 |
| 22 | P6.3 |
| 23 | P6.4 |
| 24 | P6.5/R$\overline{W}$ |
| 25 | P6.6 |
| 26 | P6.7 |
| 27 | P0.0/AD0 |
| 28 | P0.1/AD1 |
| 29 | P0.2/AD2 |
| 30 | P0.3/AD3 |
| 31 | P0.4/AD4 |
| 32 | P0.5/AD5 |

| Pin | Name |
|---|---|
| 53 | P2.5 |
| 52 | P2.4 |
| 51 | P2.3 |
| 50 | P2.2 |
| 49 | P2.1 |
| 48 | P2.0 |
| 47 | P4.7/T0OUTA |
| 46 | P4.6/INT5/T0OUTB |
| 45 | P4.5/INT4 |
| 44 | P4.4/INT0/WDOUT |
| 43 | P4.3/STOUT |
| 42 | P4.2/INTCLK |
| 41 | P4.1 |
| 40 | P4.0 |
| 39 | $V_{PP}$ |
| 38 | $\overline{DS}$ |
| 37 | $\overline{AS}$ |
| 36 | $V_{DD}$ |
| 35 | P07/AD7 |
| 34 | $V_{SS}$ |
| 33 | P0.6/AD6 |

| Pin | Name |
|---|---|
| 74 | P8.7/NMI/T3OUTB |
| 73 | $AV_{SS}$ |
| 72 | $AV_{DD}$ |
| 71 | P7.7/AIN7 |
| 70 | P7.6/AIN6 |
| 69 | P7.5/AIN5 |
| 68 | P7.4/AIN4 |
| 67 | P7.3/AIN3/T0INA/P/$\overline{D}$ |
| 66 | P7.2/AIN2/CLK0OUT /TX0CKIN |
| 65 | P7.1/AIN1/T0INB |
| 64 | P7.0/AIN0/RX0CKIN /WDIN/ADTRG |
| 63 | P9.7/INT6/SDO |
| 62 | P9.6/INT2/SCK |
| 61 | P9.5/S0IN |
| 60 | P9.4/S0OUT |
| 59 | P9.3/T0OUTA /RX1CKIN |
| 58 | P9.2/TX1CKIN /CLK1OUT |
| 57 | P9.1/T0OUTB/S1IN |
| 56 | P9.0/S1OUT |
| 55 | P2.7 |
| 54 | P2.6 |

**84 Pin PLCC Package**

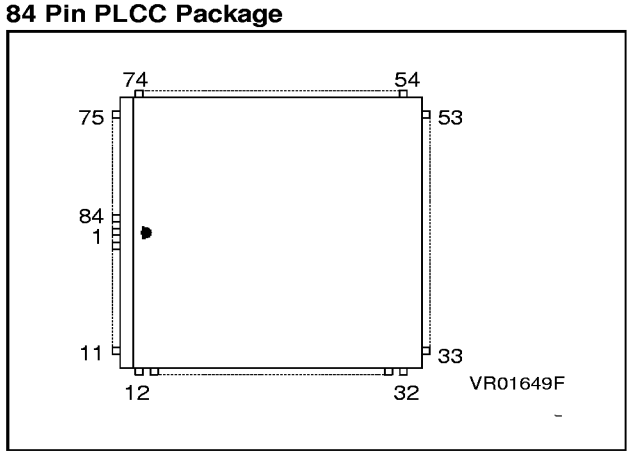


**Table 2. PQFP Pin Description**

| Pin | Name |
|---|---|
| ● 1 | P0.4/AD4 |
| 2 | P0.5/AD5 |
| 3 | P0.6/AD6 |
| 4 | $V_{SS}$ |
| 5 | P0.7/AD7 |
| 6 | VCC |
| 7 | $\overline{AS}$ |
| 8 | $\overline{DS}$ |
| 9 | $V_{PP}$ |
| 10 | P4.0 |
| 11 | P4.1 |
| 12 | P4.2/INTCLK |
| 13 | P4.3/STOUT |
| 14 | P4.4/INT0/WDOUT |
| 15 | P4.5/INT4 |
| 16 | P4.6/INT5/T0OUTB |
| 17 | P4.7/T0OUTA |
| 18 | P2.0 |
| 19 | P2.1 |
| 20 | P2.2 |
| 21 | P2.3 |
| 22 | P2.4 |
| 23 | P2.5 |
| 24 | P2.6 |

| Pin | Name |
|---|---|
| 25 | P9.0/S1OUT |
| 26 | P9.1/T0OUTB/S1IN |
| 27 | P9.2/TX1CKIN/CLK1OUT |
| 28 | P9.4/S0OUT |
| 29 | P9.5/S0IN |
| 30 | P9.6/INT2/SCK |
| 31 | P9.7/INT6/SDO |
| 32 | P7.0/AIN0/RX0CKIN/ WDIN/ADTRG |
| 33 | P7.1/AIN1/T0INB |
| 34 | P7.2/AIN2/CLK0OUT/ TX0CKIN |
| 35 | P7.3/AIN3/T0IN/$\overline{PD}$ |
| 36 | P7.4/AIN4 |
| 37 | P7.5/AIN5 |
| 38 | P7.6/AIN6 |
| 39 | P7.7/AIN7 |
| 40 | $AV_{DD}$ |

| Pin | Name |
|---|---|
| 64 | P1.2/A10 |
| 63 | P1.1/A9 |
| 62 | P1.0/A8 |
| 61 | $\overline{RESET}$ |
| 60 | OSCIN |
| 59 | $V_{SS}$ |
| 58 | OSCOUT |
| 57 | P5.1/SDI |
| 56 | HW0SW1 |
| 55 | P5.3/$\overline{PD}$ |
| 54 | P5.4/T1OUTA/DCD0 |
| 53 | P5.5/T1OUTB/CTS0 |
| 52 | P5.6/T3OUTA/DCD1 |
| 51 | P5.7/T3OUTB/CTS1/ CK_AF |
| 50 | $V_{CC}$ |
| 49 | P8.0/T3INA |
| 48 | P8.1/T1INB |
| 47 | P8.2/T1OUTA/INT1 |
| 46 | P8.3/T1OUTB/INT3 |
| 45 | P8.4/T1INA/$\overline{WAIT}$/ WDOUT |
| 44 | P8.5/$\overline{PD}$/T3INB |
| 43 | P8.6/INT7/T3OUTA |
| 42 | P8.7/NMI/T3OUTB |
| 41 | $AV_{SS}$ |

| Pin | Name |
|---|---|
| 80 | P0.3/AD3 |
| 79 | P0.2/AD2 |
| 78 | P0.1/AD1 |
| 77 | P0.0/AD0 |
| 76 | P6.6 |
| 75 | P6.5/$\overline{RW}$ |
| 74 | P6.4 |
| 73 | P6.3 |
| 72 | P6.2 |
| 71 | P6.1 |
| 70 | P6.0 |
| 69 | P1.7/A15 |
| 68 | P1.6/A14 |
| 67 | P1.5/A13 |
| 66 | P1.4/A12 |
| 65 | P1.3/A11 |

**80 Pin PQFP Package**

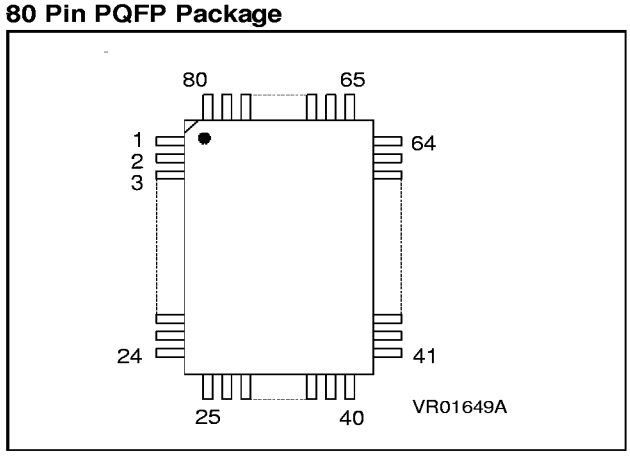


**Table 3. ST90158/ST90135 I/O Port Alternate Function Summary**

| I/O PORT Port.bit | Name | Function IN/OUT | Alternate Function | Pin Number | |
|---|---|---|---|---|---|
| | | | | PLCC | PQFP |
| P0.0 | A0/D0 | I/O | Address/Data bit 0 mux / I/O Port 0.0 | 27 | 77 |
| P0.1 | A1/D1 | I/O | Address/Data bit 1 mux / I/O Port 0.1 | 28 | 78 |
| P0.2 | A2/D2 | I/O | Address/Data bit 2 mux / I/O Port 0.2 | 29 | 79 |
| P0.3 | A3/D3 | I/O | Address/Data bit 3 mux / I/O Port 0.3 | 30 | 80 |
| P0.4 | A4/D4 | I/O | Address/Data bit 4 mux / I/O Port 0.4 | 31 | 1 |
| P0.5 | A5/D5 | I/O | Address/Data bit 5 mux / I/O Port 0.5 | 32 | 2 |
| P0.6 | A6/D6 | I/O | Address/Data bit 6 mux / I/O Port 0.6 | 33 | 3 |
| P0.7 | A7/D7 | I/O | Address/Data bit 7 mux / I/O Port 0.7 | 35 | 5 |
| P1.0 | A8 | I/O | Address bit 8 / I/O Port 1.0 | 11 | 62 |
| P1.1 | A9 | I/O | Address bit 9 / I/O Port 1.1 | 12 | 63 |
| P1.2 | A10 | I/O | Address bit 10 / I/O Port 1.2 | 13 | 64 |
| P1.3 | A11 | I/O | Address bit 11 / I/O Port 1.3 | 14 | 65 |
| P1.4 | A12 | I/O | Address bit 12 / I/O Port 1.4 | 15 | 66 |
| P1.5 | A13 | I/O | Address bit 13 / I/O Port 1.5 | 16 | 67 |
| P1.6 | A14 | I/O | Address bit 14 / I/O Port 1.6 | 17 | 68 |
| P1.7 | A15 | I/O | Address bit 15 / I/O Port 1.7 | 18 | 69 |
| P2.0 | | I/O | I/O Port 2.0 | 48 | 18 |
| P2.1 | | I/O | I/O Port 2.1 | 49 | 19 |
| P2.2 | | I/O | I/O Port 2.2 | 50 | 20 |
| P2.3 | | I/O | I/O Port 2.3 | 51 | 21 |
| P2.4 | | I/O | I/O Port 2.4 | 52 | 22 |
| P2.5 | | I/O | I/O Port 2.5 | 53 | 23 |
| P2.6 | | I/O | I/O Port 2.6 | 54 | 24 |
| P2.7 | | I/O | I/O Port 2.7 | 55 | - |
| P4.0 | | I/O | I/O Port 4.0 | 40 | 10 |
| P4.1 | | I/O | I/O Port 4.1 | 41 | 11 |
| P4.2 | | I/O | I/O Port 4.2 | 42 | 12 |
| P4.2 | INTCLK | 0 | Internal main Clock | 42 | 12 |
| P4.3 | | I/O | I/O Port 4.3 | 43 | 13 |
| P4.3 | STOUT | O | Standard Timer Output | 43 | 13 |
| P4.4 | INT0 | I | External Interrupt 0 | 44 | 14 |
| P4.4 | WDOUT | O | T/WD output | 44 | 14 |
| P4.4 | | I/O | I/O Port 4.4 | 44 | 14 |
| P4.5 | INT4 | I | External interrupt 4 | 45 | 15 |
| P4.5 | | I/O | I/O Port 4.5 | 45 | 15 |
| P4.6 | INT5 | I | External Interrupt 5 | 46 | 16 |
| P4.6 | T0OUTB | O | MF Timer 0 Output B [1)] | 46 | 16 |
| P4.6 | | I/O | I/O Port 4.6 | 46 | 16 |
| P4.7 | T0OUTA | O | MF Timer 0 Output A [1)] | 47 | 17 |
| P4.7 | | I/O | I/O Port 4.7 | 47 | 17 |
| P5.0 | | I/O | I/O Port 5.0 | 6 | - |

| I/O PORT Port.bit | Name | Function IN/OUT | Alternate Function | Pin Number | |
|---|---|---|---|---|---|
| | | | | PLCC | PQFP |
| P5.1 | | I/O | I/O Port 5.1 | 5 | 57 |
| P5.1 | SDI | I | SPI Serial Data In | 5 | 57 |
| P5.2 | | I/O | I/O Port 5.2 | 4 | - |
| P5.3 | | I/O | I/O Port 5.3 | 3 | 55 |
| P5.3 | P/$\overline{D}$ | O | Program/Data space select | 3 | 55 |
| P5.4 | T1OUTA | O | MF Timer 1 output A | 2 | 54 |
| P5.4 | | I/O | I/O Port 5.4 | 2 | 54 |
| P5.4 | DCD0 | I | SCI0 DataCarrier Detect | 2 | 54 |
| P5.5 | CTS0 | O | SCI0 Clear to Send | 1 | 53 |
| P5.5 | T1OUTB | O | MF Timer 1 output B | 1 | 53 |
| P5.5 | | I/O | I/O Port 5.5 | 1 | 53 |
| P5.6 | T3OUTA | O | MF Timer 3 output A | 84 | 52 |
| P5.6 | | I/O | I/O Port 5.6 | 84 | 52 |
| P5.6 | DCD1 | I | SCI1 Data Carrier Detect [1] | 84 | 52 |
| P5.7 | CTS1 | O | SCI1 Clear to Send [1] | 83 | 51 |
| P5.7 | T3OUTB | O | MF Timer 3 output B | 83 | 51 |
| P5.7 | | I/O | I/O Port 5.7 | 83 | 51 |
| P5.7 | CK_AF | I | External Clock Input | 83 | 51 |
| P6.0 | | I/O | I/O Port 6.0 | 19 | 70 |
| P6.1 | | I/O | I/O Port 6.1 | 20 | 71 |
| P6.2 | | I/O | I/O Port 6.2 | 21 | 72 |
| P6.3 | | I/O | I/O Port 6.3 | 22 | 73 |
| P6.4 | | I/O | I/O Port 6.4 | 23 | 74 |
| P6.5 | | I/O | I/O Port 6.5 | 24 | 75 |
| P6.5 | R/$\overline{W}$ | O | Read/Write | 24 | 75 |
| P6.6 | | I/O | I/O Port 6.6 | 25 | 76 |
| P6.7 | | I/O | I/O Port 6.7 | 26 | - |
| P7.0 | AIN0 | I | A/D Analog input 0 | 64 | 32 |
| P7.0 | RX0CKIN | I | SCI0 Receive Clock input | 64 | 32 |
| P7.0 | WDIN | I | T/WD input | 64 | 32 |
| P7.0 | ADTRG | I | A/D External Trigger | 64 | 32 |
| P7.1 | AIN1 | I | A/D Analog input 1 | 65 | 33 |
| P7.1 | T0INB | I | MF Timer 0 input B [1] | 65 | 33 |
| P7.2 | AIN2 | I | A/D Analog input 2 | 66 | 34 |
| P7.2 | CLK0OUT | O | SCI0 Byte Sync Clock output | 66 | 34 |
| P7.2 | TX0CKIN | I | SCI0 Transmit Clock input | 66 | 34 |
| P7.3 | AIN3 | I | A/D Analog input 3 | 67 | 35 |
| P7.3 | P/$\overline{D}$ | O | Program/data space select | 67 | 35 |
| P7.3 | T0INA | I | MF Timer 0 input A [1] | 67 | 35 |
| P7.4 | AIN4 | I | A/D Analog input 4 | 68 | 36 |
| P7.5 | AIN5 | I | A/D Analog input 5 | 69 | 37 |
| P7.6 | AIN6 | I | A/D Analog input 6 | 70 | 38 |

| I/O PORT Port.bit | Name | Function IN/OUT | Alternate Function | Pin Number | |
|---|---|---|---|---|---|
| | | | | PLCC | PQFP |
| P7.7 | AIN7 | I | A/D Analog input 7 | 71 | 39 |
| P8.0 | T3INA | I | MF Timer 3 input A | 81 | 49 |
| P8.1 | T1INB | I | MF Timer 1 input B | 80 | 48 |
| P8.2 | INT1 | I | External interrupt 1 | 79 | 47 |
| P8.2 | T1OUTA | O | MF Timer 1 output A | 79 | 47 |
| P8.3 | INT3 | I | External interrupt 3 | 78 | 46 |
| P8.3 | T1OUTB | O | MF Timer 1 output B | 78 | 46 |
| P8.4 | T1INA | I | MF Timer 1 input A | 77 | 45 |
| P8.4 | $\overline{WAIT}$ | I | External Wait input | 77 | 45 |
| P8.4 | WDOUT | O | T/WD output | 77 | 45 |
| P8.5 | P/$\overline{D}$ | O | Program/Data space select | 76 | 44 |
| P8.5 | T3INB | I | MF Timer 3 input B | 76 | 44 |
| P8.6 | INT7 | I | External interrupt 7 | 75 | 43 |
| P8.6 | T3OUTA | O | MF Timer 3 output A | 75 | 43 |
| P8.7 | NMI | I | Non-Maskable Interrupt | 74 | 42 |
| P8.7 | T3OUTB | O | MF Timer 3 output B | 74 | 42 |
| P9.0 | S1OUT | O | SCI1 Serial Output [1] | 56 | 25 |
| P9.1 | T0OUTB | O | MF Timer 0 output B [1] | 57 | 26 |
| P9.1 | S1IN | I | SCI1 Serial Input [1] | 57 | 26 |
| P9.2 | CLK1OUT | O | SCI1 Byte Sync Clock output [1] | 58 | 27 |
| P9.2 | TX1CKIN | I | SCI1 Transmit Clock input [1] | 58 | 27 |
| P9.3 | RX1CKIN | I | SCI1 Receive Clock input [1] | 59 | - |
| P9.3 | T0OUTA | O | MF Timer 0 output A [1] | 59 | - |
| P9.4 | S0OUT | O | SCI0 Serial Output | 60 | 28 |
| P9.5 | S0IN | I | SCI0 Serial Input | 61 | 29 |
| P9.6 | INT2 | I | External interrupt 2 | 62 | 30 |
| P9.6 | SCK | O | SPI Serial Clock | 62 | 30 |
| P9.7 | INT6 | I | External interrupt 6 | 63 | 31 |
| P9.7 | SDO | O | SPI Serial Data Out | 63 | 31 |

Note 1) Not present on ST90135

### 1.4 OPERATING MODES

To optimize the performance versus the power consumption of the device, the ST90158 supports different operating modes that can be dynamically selected depending on the performance and functionality requirements of the application at a given moment.

**RUN MODE**: This is the full speed execution mode with CPU and peripherals running at the maximum clock speed delivered by the Phase Locked Loop (PLL) of the Clock Control Unit (CCU).

**SLOW MODE**: Power consumption can be significantly reduced by running the CPU and the peripherals at reduced clock speed using the CPU Prescaler and CCU Clock Divider.

**WAIT FOR INTERRUPT MODE** The Wait For Interrupt (WFI) instruction suspends program execution until an interrupt request is acknowledged. During WFI, the CPU clock is halted while the peripheral and interrupt controller keep running at a frequency depending on the CCU programming. Under this mode, the power consumption of the device can be reduced by more than 95% (LP WFI).

**HALT MODE**: When executing the HALT instruction, and if the Watchdog Timer is not programmed as a watchdog, the CPU and its peripherals stop operation and the I/O ports enter high impedance mode. A reset is necessary to exit from Halt mode.

**Table 4. Ordering Information**

| Salestype | Program Memory (Bytes) | RAM (Bytes) | Watchdog Option | Temperature Range | Package | Emulation Device |
|---|---|---|---|---|---|---|
| ST90135M4Q6 | 16K ROM | 512 | HW/SW | -40°C +85°C<br>0°C +70°C | PQFP80 | ST90E158<br>ST90T158 |
| ST90135M5Q6 | 24K ROM | 768 | | | | |
| ST90135M6Q6 | 32K ROM | 1K | | | | |
| ST90158M7Q6 | 48K ROM | 1.5k | | | | |
| ST90158M9Q6 | 64k ROM | 2k | | | | |
| ST90158P9C6 | | | SW | -40°C +85°C | PLCC84 | |
| ST90E158M9G0 | 64k EPROM | | HW/SW | + 25°C | CQFP80-W | |
| ST90E158P9L0 | | | SW | | CLCC84-W | |
| ST90T158M9Q6 | 64k OTP | | HW/SW | -40°C +85°C | PQFP80 | |
| ST90T158P9C6 | | | SW | | PLCC84 | |

# 2 DEVICE ARCHITECTURE

## 2.1 CORE ARCHITECTURE

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

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 which connects the interrupt and DMA controllers in the on-chip peripherals with the Core.

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

## 2.2 MEMORY SPACES

There are two separate memory spaces:

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

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

### 2.2.1 Register File

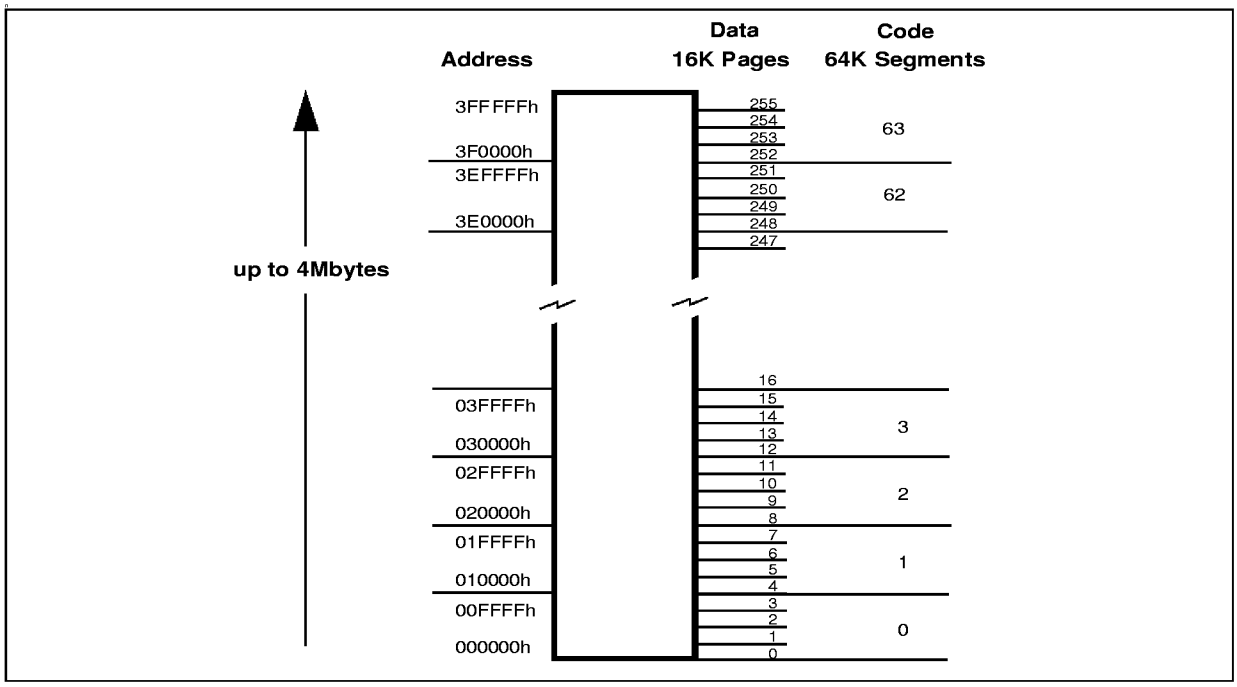
The Register File consists of:

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

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

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

**Figure 3. Single Program and Data Memory Address Space**



MEMORY SPACES (Cont'd)

**Figure 4. Register Groups**

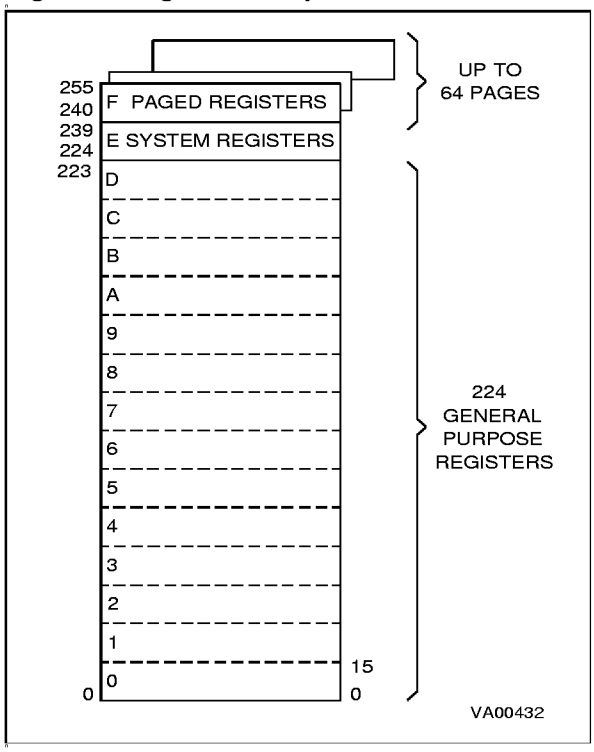



**Figure 5. Page Pointer for Group F mapping**

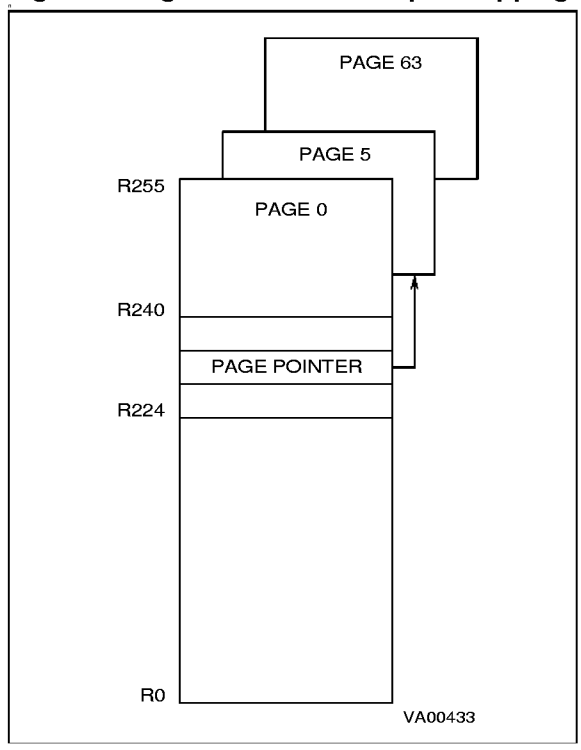
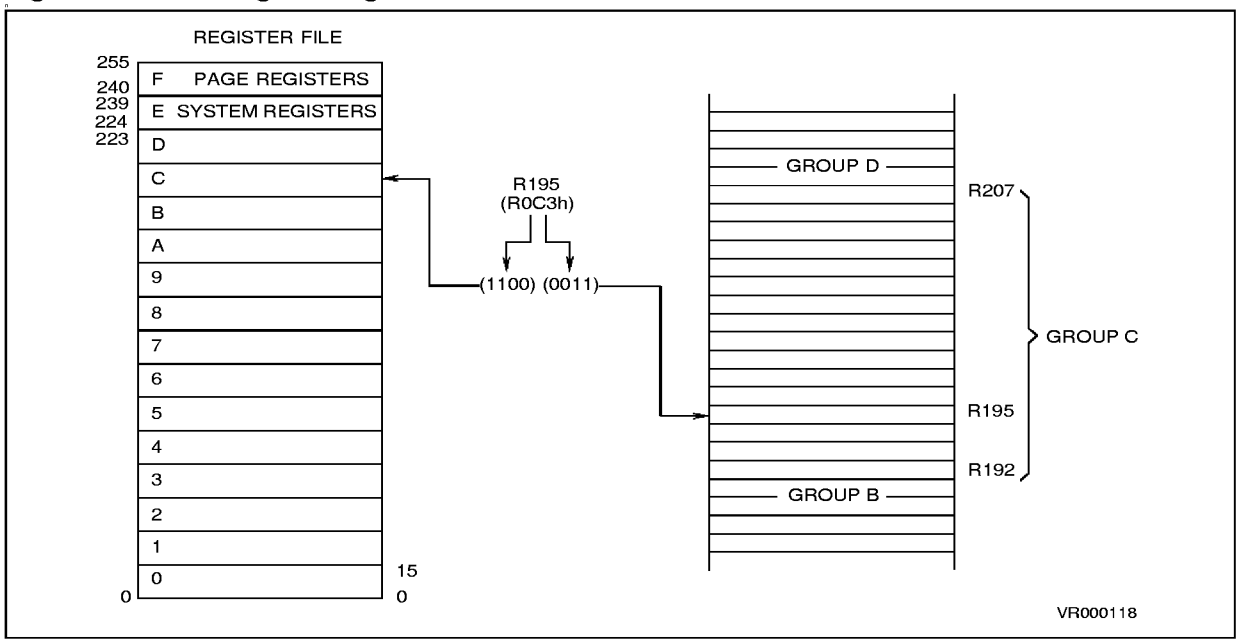



**Figure 6. Addressing the Register File**



**MEMORY SPACES (Cont'd)**

### 2.2.2 Register Addressing

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

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

**Working Registers**

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

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

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

**System Registers**

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

**Paged Registers**

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

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

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

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

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

A table of available Group F pages is given in the Register Map Chapter.

**Table 5. Register File Organization**

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

### 2.3 SYSTEM REGISTERS

The System registers are listed in Table 2. They are used to perform all the important system settings. Their purpose is described in the following pages. Refer to the chapter dealing with I/O for a description of the PORT0-5 Data registers.

**Note:** Some reset values indicated for the register-here may be modified by the internal boot ROM. Refer to the table of boot ROM reset values in Chapter 3.

**Table 6. System Registers (Group E)**

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

#### 2.3.1 Central Interrupt Control Register

Please refer to the "INTERRUPT" and "DMA" chapters for a detailed description of the ST9 interrupt philosophy.

**CICR R230** (E6h) System Read/Write

Central Interrupt Control Register

Reset Value: 1000 0111b (87h)

| 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.

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

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

b4 = **IEN**: *Enable Interrupt.* This bit, when set, allows interrupts to be accepted. When reset, no interrupts, except for the NMI, will be acknowledged. The bit is cleared by interrupt acknowledgement, and set by interrupt return (`iret`). It may be managed both by hardware and software (`ei` and `di` instructions). IEN must be written only when the interrupt is disabled.

b3 = **IAM**: *Interrupt Arbitration Mode* This bit allows selection of the arbitration mode: Concurrent Mode is selected when the bit is reset and Fully Automatic Nested Mode when the bit is set. This bit is under software control.

b2-b0 = **CPL2-CPL0**: *Current Priority Level* These three bits record the priority level of the interrupt currently being serviced (i.e. the Current Priority Level, CPL). The highest priority level is represented by 000, and the lowest by 111. The CPL bits can be set by hardware or software and provide the reference according to which subsequent interrupts are either left pending or are allowed to interrupt the current interrupt service routine. When the current interrupt is replaced by one of a higher priority, the current priority value is automatically stored until required.

**SYSTEM REGISTERS** (Cont'd)

### 2.3.2 Flag Register

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

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

**FLAGR R231** (E7h) System Read/Write

Flag Register

Reset value: 0000 0000b (00h)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| C | Z | S | V | DA | H | - | DP |

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

Addition (`add, addw, adc, adcw`),

Subtraction (`sub, subw, sbc, sbcw`),

Compare (`cp, cpw`),

Shift Right Arithmetic (`sra, sraw`),

Rotate (`rrc, rrcw, rlc, rlcw, ror, rol`),

Decimal Adjust (`da`),

Multiply and Divide (`mul, div, divws`).

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

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

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

Addition (`add, addw, adc, adcw`),

Subtraction (`sub, subw, sbc, sbcw`),

Compare (`cp, cpw`),

Shift Right Arithmetic (`sra, sraw`),

Rotate (rrc, `rrcw, rlc, rlcw, ror, rol`),

Decimal Adjust (`da`),

Multiply and Divide (`mul, div, divws`),

Logical (`and, andw, or, orw, xor, xorw, cpl`),

Increment and Decrement (`inc, incw, dec, decw`),

Test (`tm, tmw, tcm, tcmw, btset`).

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

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 two's-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 = Reserved bit (must be 0).

b0 = **DP**: *Data/Program Memory Flag* This bit indicates the memory area addressed. Its value is affected by the Set Data Memory (`sdm`) and Set Program Memory (`spm`) instructions. Refer to the Memory Management Unit for further details.

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

It should be noted that, in the new ST9+ product generation (ST901xx and ST921xx), this bit is retained only for compatibility with software developed for the first generation of ST9+ devices (ST90xx and ST92xx). Thanks to the single memory addressing space, its use is now redundant.

**SYSTEM REGISTERS** (Cont'd)

### 2.3.3 Register Pointing Techniques

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

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

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

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

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

Thus:

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

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

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

**RPO R232** (E8h) System Read/Write

Pointer 0 Register

Reset Value: xxxx xx00b (xxh)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| RG4 | RG3 | RG2 | RG1 | RG0 | RPS | 0 | 0 |

b7-b3 = **RG4-RG0**: *Register Group number.* These bits contain the number (in the range 0 to 31) of the register block specified in the `srp0` or `srp` instructions. In single 16-register mode the number indicates the lower of the two 8-register blocks to which the 16 working registers are to be mapped, while in twin 8-register mode it indicates the 8-register block to which `r0` to `r7` are to be mapped.

b2 = **RPS**: *Register Pointer Selector.* This bit is set by the instructions `srp0` and `srp1` to indicate that the twin register pointing mode is selected. The bit is reset by the `srp` instruction to indicate that the single register pointing mode is selected.

b1-b0: These bits are fixed by hardware to zero and cannot be modified.

**RP1 R233** (E9h) System Read/Write

Pointer 1 Register

Reset Value: xxxx xx00b (xxh)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| RG4 | RG3 | RG2 | RG1 | RG0 | RPS | 0 | 0 |

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

b7-b3 = **RG4-RG0:** *Register Group number.* These bits contain the number (in the range 0 to 31) of the 8-register block specified in the `srp1` instruction, to which `r7` to `r15` are to be mapped

b2 = **RPS**: *Register Pointer Selector.* This bit is set by the `srp0` and `srp1` instructions to indicate that the twin register pointing mode is selected. The bit is reset by the `srp` instruction to indicate that the single register pointing mode is selected.

b1-b0: These bits are fixed by hardware to zero and cannot be modified.

**SYSTEM REGISTERS** (Cont'd)

**Figure 7. Pointing to a single group of 16 registers**

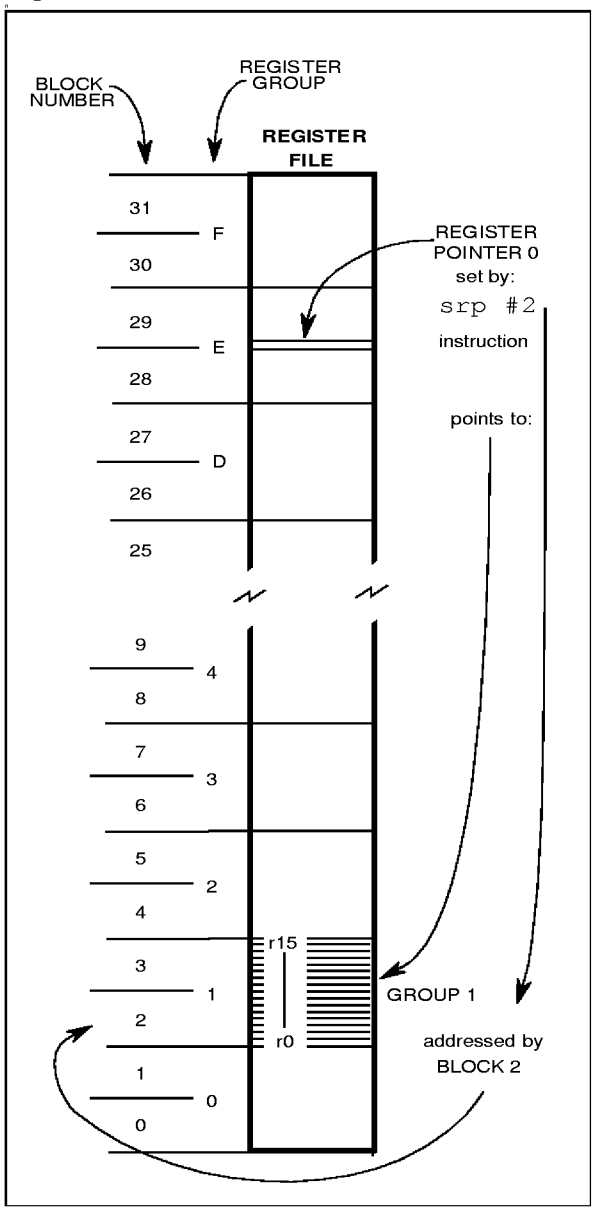
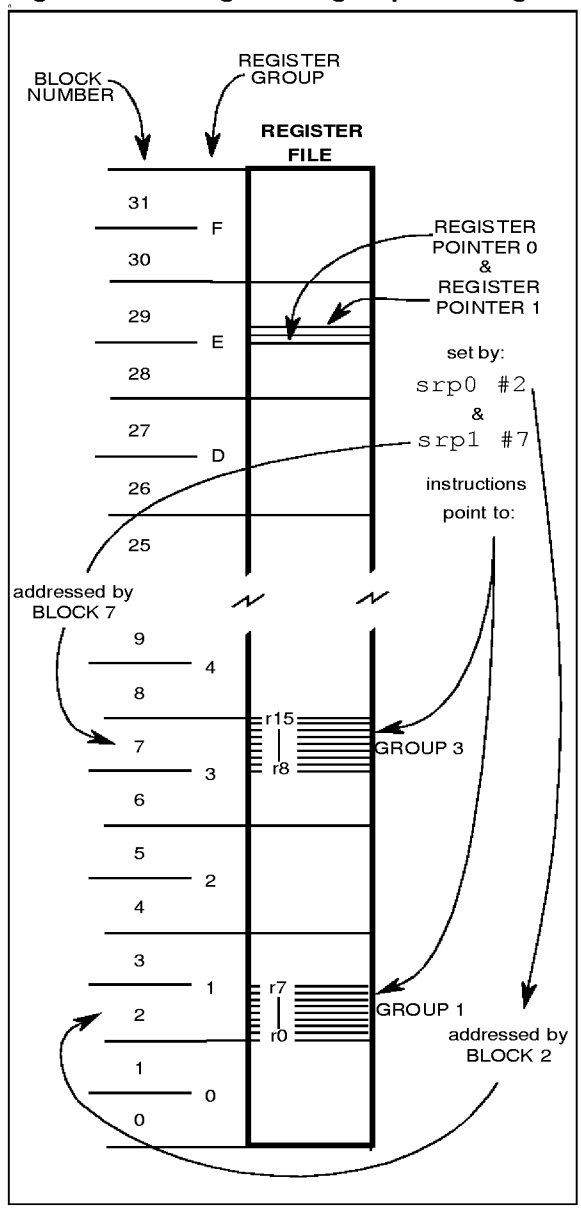



**Figure 8. Pointing to two groups of 8 registers**



**SYSTEM REGISTERS** (Cont'd)

### 2.3.4 Paged Registers

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

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

Thus the instructions:

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

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

**PPR R234** (EAh) System Read/Write

Page Pointer Register

Reset value: xxxx xx00b (xxh)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| PP5 | PP4 | PP3 | PP2 | PP1 | PP0 | 0 | 0 |

b7-b2 = **PP5-PP0**: *Page Pointer*. These bits contain the number (in the range 0 to 63) of the page specified in the `spp` instruction. Once the page pointer has been set, there is no need to refresh it unless a different page is required.

b1-b0: These bits are fixed by hardware to 0.

### 2.3.5 Mode Register

The Mode Register allows control of the following operating parameters:

– selection of internal or external System and User Stack areas,

– management of the clock frequency,

– enabling of Bus request and Wait signals when interfacing to external memory.

**MODER R235** (EBh) System Read/Write

Mode Register

Reset value: 1110 0000b (E0h)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| SSP | USP | DIV2 | PRS2 | PRS1 | PRS0 | BR-QEN | HIMP |

b7 = **SSP**: *System Stack Pointer*. This bit selects an internal or external System Stack area.
0 = External stack area, in data memory
1 = Internal stack area, in the Register File (reset state).

b6 = **USP**: *User Stack Pointer*. Performs the same function as bit 7 (SSP) for the User Stack.

b5 = **DIV2**: *OSCIN Clock Divided by 2* Controls the divide-by-2 circuit operating on OSCIN.
0 = Clock divided by 1
1 = Clock divided by 2

b4-b2 = **PRS2-PRS0**: *CPUCLK Prescaler*. These bits load the prescaler division factor for the internal clock (INTCLK). The prescaler factor selects the internal clock frequency, which can be divided by a factor from 1 to 8. Refer to the Clock chapter for further information.

b1 = **BRQEN**: *Bus Request Enable*.
0 = Bus Request disabled
1 = Bus Request enabled on the BUSREQ pin.

b0 = **HIMP**: *High Impedance Enable*. When any of Ports 0, 1, 2 or 6 depending on device configuration, are programmed as Address and Data lines to interface external Memory, these lines and the Memory interface control lines (AS, DS, R/W) can be forced into the High Impedance state by setting the HIMP bit. When this bit is reset, it has no effect.

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

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

**SYSTEM REGISTERS** (Cont'd)

### 2.3.6 Stack Pointers

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

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

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

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

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

**System Stack**

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

The following automatically push data onto the System Stack:

– **Interrupts**

When entering an interrupt, the PC and the Flag Register are pushed onto the System Stack. If the device has an external memory interface and the ENCSR bit in the EMR2 register is set to 1, then the Code Segment Register is also pushed onto the System Stack.

– **Subroutine Calls**

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

– **Link Instruction**

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

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

**User Stack**

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

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

**Stack Pointers**

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

The stack pointer registers are located in the System Group of the Register File, this is illustrated in Table 6 System Registers (Group E)

**Stack location**

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

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

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

**SYSTEM REGISTERS** (Cont'd)

**USPHR R236** (ECh) System Read/Write
User Stack Pointer High Register
Reset value: **undefined**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| USP15 | USP14 | USP13 | USP12 | USP11 | USP10 | USP9 | USP8 |

**USPLR R237** (EDh) System Read/Write
User Stack Pointer Low Register
Reset value: **undefined**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| USP7 | USP6 | USP5 | USP4 | USP3 | USP2 | USP1 | USP0 |

**SSPHR R238** (EEh) System Read/Write
System Stack Pointer High Register
Reset value: **undefined**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| SSP15 | SSP14 | SSP13 | SSP12 | SSP11 | SSP10 | SSP9 | SSP8 |

**SSPLR R239** (EFh) System Read/Write
System Stack Pointer Low Register
Reset value: **undefined**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| SSP7 | SSP6 | SSP5 | SSP4 | SSP3 | SSP2 | SSP1 | SSP0 |

**Figure 9. Internal Stack Mode**

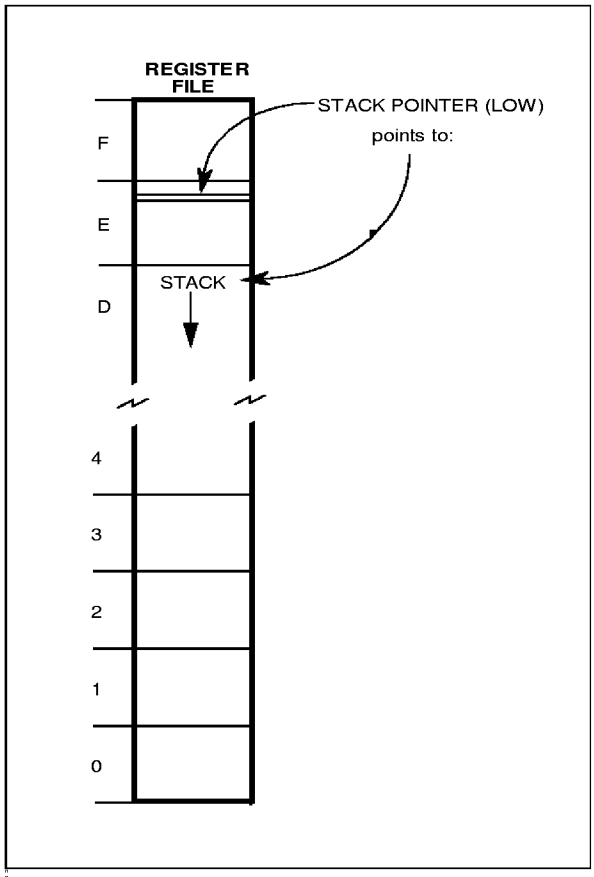
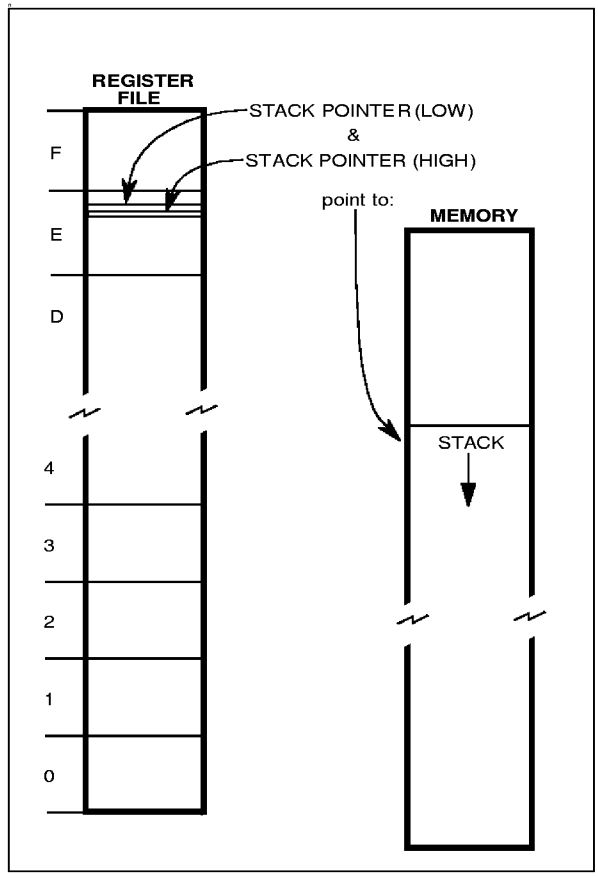



**Figure 10. External Stack Mode**



### 2.4 MEMORY ORGANIZATION

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

The ST9+ provides a total addressable memory space of 4 Mbytes. This address space is arranged as 64 segments of 64 Kbytes; each segment is again subdivided into four16 Kbyte pages.

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

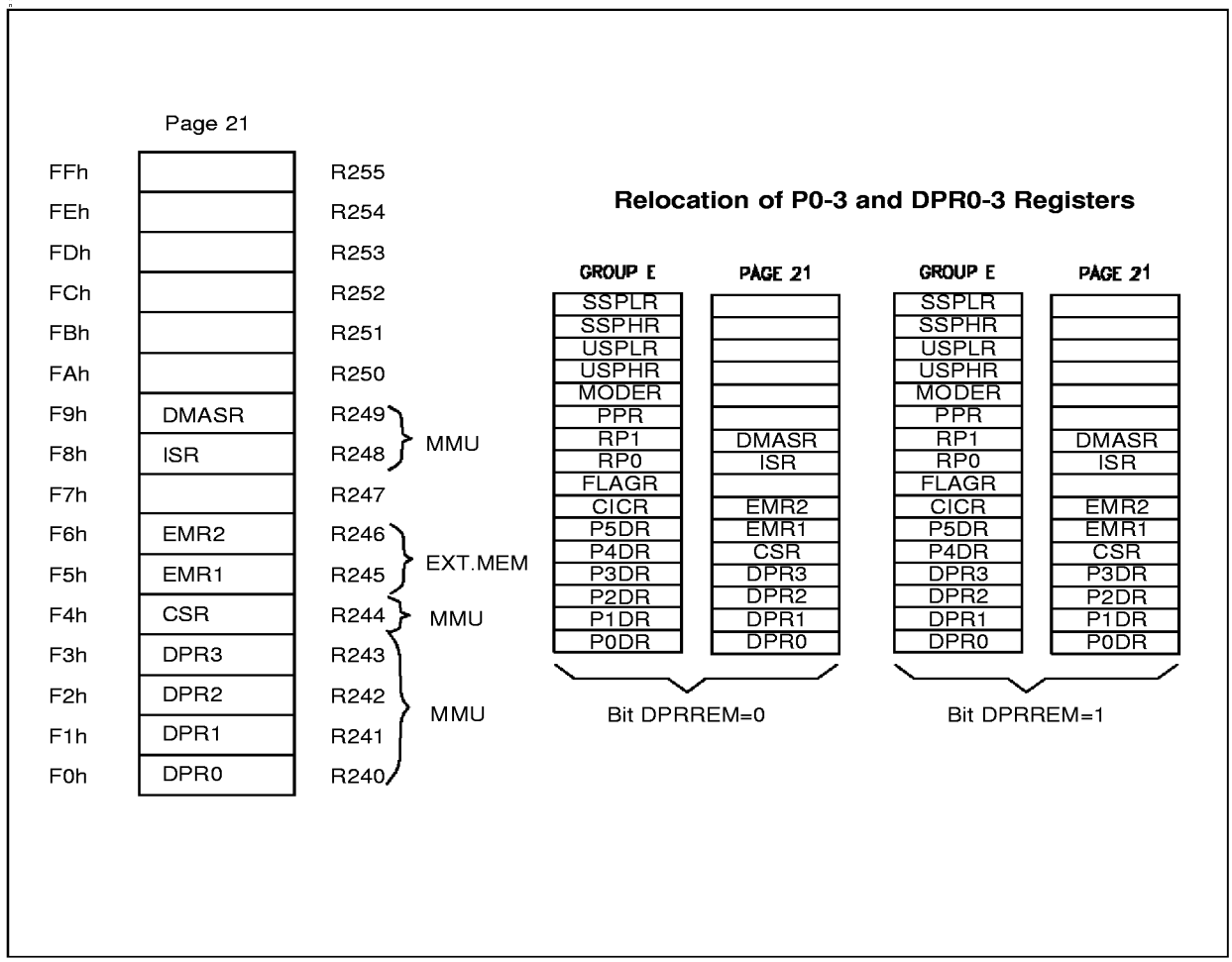
Refer to Chapter 3 for more details on the the memory map.

### 2.5 MEMORY MANAGEMENT UNIT

The CPU Core includes a Memory Management Unit (MMU) which allows the addressing space to be extended to 4 Mbytes.

The MMU is controlled by 7 registers and 3 bits (ENCSR, DPRREM, MEMSEL) present in EMR1 and EMR2, which may be written and read by the user program. These registers are mapped within group F, Page 21 of the Register File. The 7 registers may be sub-divided into 2 main groups: a first group of four 8-bit registers (DPR0-3), and a second group of three 6-bit registers (CSR, ISR, and DMASR). The first group is used to extend the address during Data Memory access (DPR0-3). The second is used to manage Program and Data Memory accesses during Code execution (CSR), Interrupts (ISR), and DMA service routines (DMASR).

**Figure 11. Page 21 Registers**



## 2.6 ADDRESS SPACE EXTENSION

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

### 2.6.1 Addressing 16-Kbyte Pages

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

The Data memory space is divided into 4 pages of 16 Kbytes. Each one of the four 8-bit registers (DPR0-3, Data Page Registers) selects a different 16-Kbyte page. The DPR registers allow access to the entire memory space which contains 256 pages of 16 Kbytes.

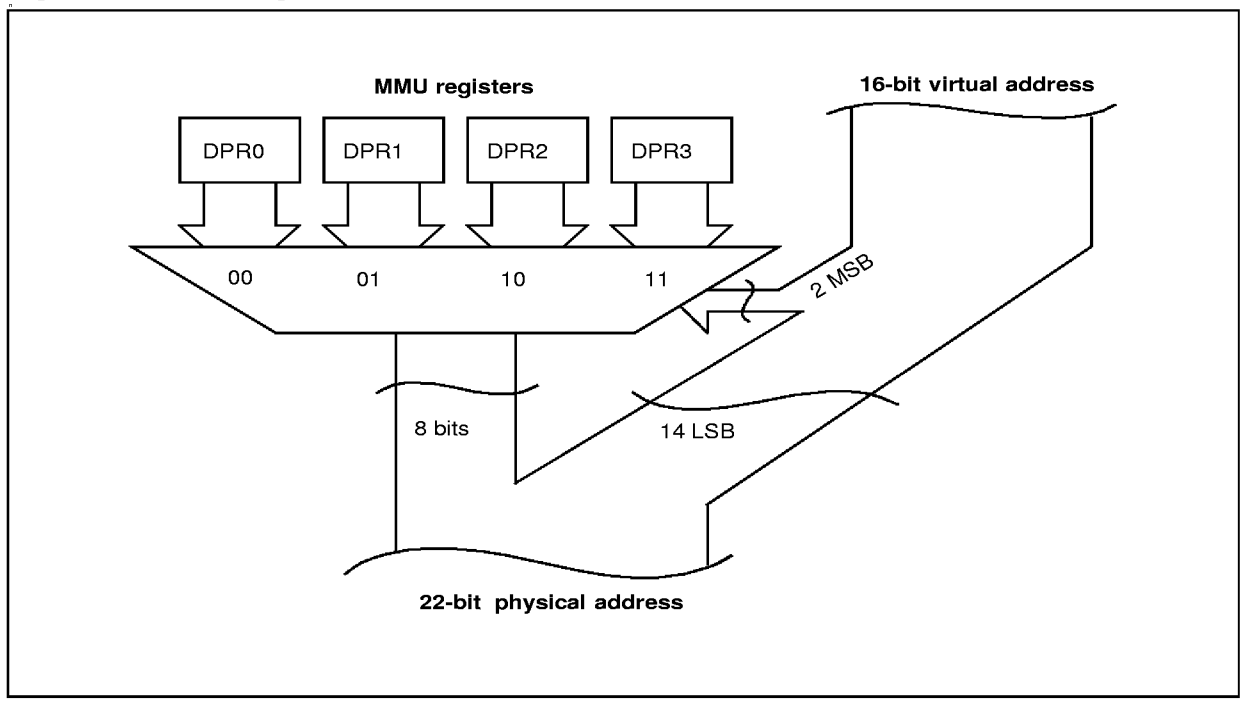
Data paging is performed by extending the 14 LSB of the 16-bit address with the contents of a DPR register. The two MSBs of the 16-bit address are interpreted as the identification number of the DPR register to be used. Therefore, the DPR registers are involved in the following virtual address ranges:

DPR0: from 0000h to 3FFFh;
DPR1: from 4000h to 7FFFh;
DPR2: from 8000h to BFFFh;
DPR3: from C000h to FFFFh.

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

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

**Figure 12. Addressing via DPR0-3**



**ADDRESS SPACE EXTENSION** (Cont'd)

### 2.6.2 Addressing 64-Kbyte Segments

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

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

## 2.7 MMU REGISTERS

The MMU uses 7 registers mapped into Group F, Page 21 of the Register File and 3 bits of the EMR2 register (This register is described in the External Memory Interface chapter).

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

### 2.7.1 DPR0, DPR1, DPR2, DPR3: Data Page Registers

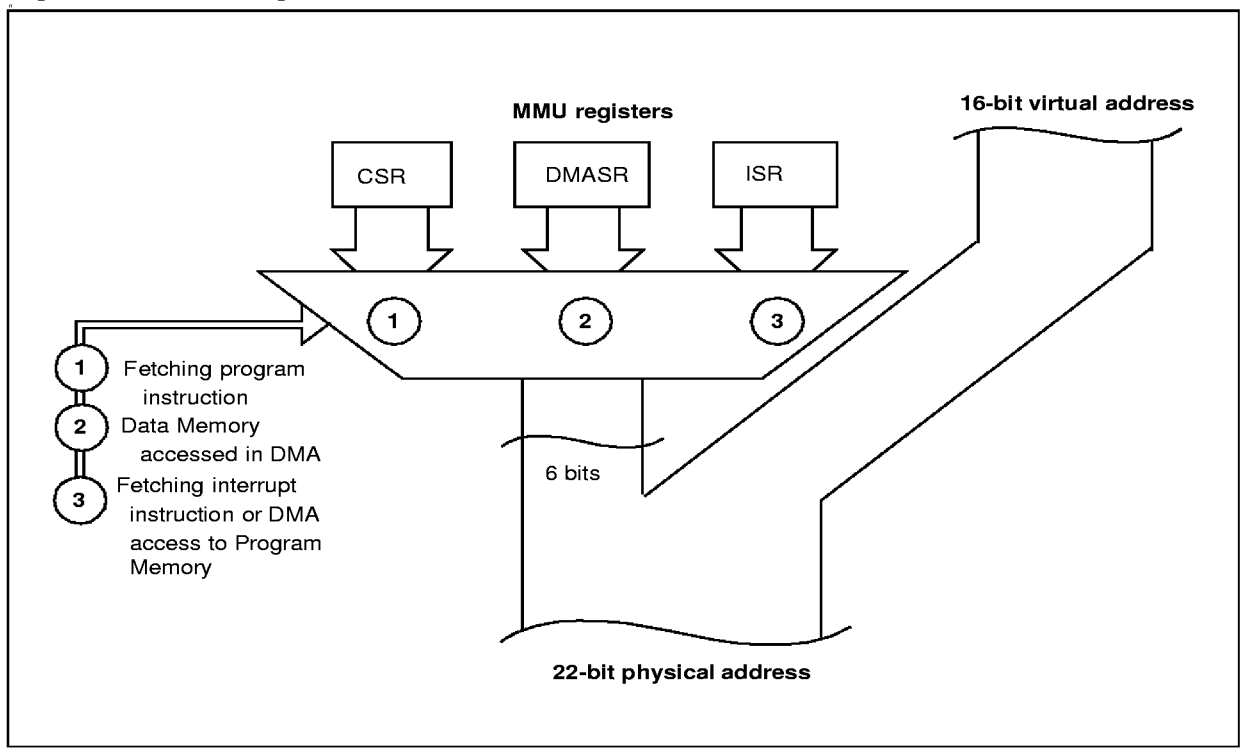
The DPR registers allow access to the entire 4-Mbyte memory space composed of 256 pages of 16 Kbytes.

#### 2.7.1.1 Data Page Register Relocation

If these registers are to be used frequently, they may be relocated in register group E, by programming bit 5 of the EMR2-R246 register in page 21. If this bit is set, the DPR0-3 registers are located at R224-227 in place of the Port 0-3 Data Registers, which are re-mapped to the default DPR's locations: R240-243 page 21.

Data Page Register relocation is illustrated inFigure 11.

**Figure 13. Addressing via CSR, ISR, and DMASR**



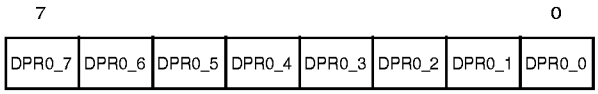
**MMU REGISTERS** (Cont'd)

**DPR0 R240** (F0h) Page 21 Read/Write

Data Page Register 0

Reset value: 0000 0000b (00h)

This register is relocated to R224 (E0h) if EMR2.5 is set.



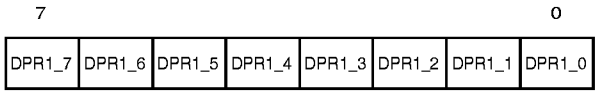

b7-b0 = **DPR0_7-0**: these bits define the 16-Kbyte Data Memory page number. They are used as the most significant address bits (A21-14) to extend the address during a Data Memory access. The DPR0 register is used when addressing the virtual address range 0000h-3FFFh.

**DPR1 R241** (F1h) Page 21 Read/Write

Data Page Register 1

Reset value: 0000 0001b (01h)

This register is relocated to R225 (E1h) if EMR2.5 is set.



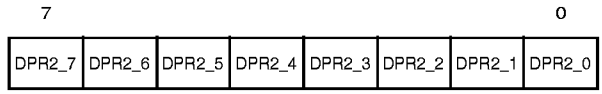

b7-b0 = **DPR1_7-0**: these bits define the 16-Kbyte Data Memory page number. They are used as the most significant address bits (A21-14) to extend the address during a Data Memory access. The DPR1 register is used when addressing the virtual address range 4000h-7FFFh.

**DPR2 R242** (F2h) Page 21 Read/Write

Data Page Register 2

Reset value: 0000 0010b (02h)

This register is relocated to R226 (E2h) if EMR2.5 is set.



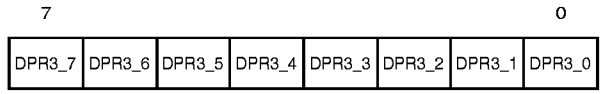

b7-b0 = **DPR2_7-0**: these bits define the 16-Kbyte Data memory page. They are used as the most significant address bits (A21-14) to extend the address during a Data memory access. The DPR2 register is involved when the virtual address is in the range 8000h-BFFFh.

**DPR3 R243** (F3h) Page 21 Read/Write

Data Page Register 3

Reset value: 1000 0011b (83h)

This register is relocated to R227 (E3h) if EMR2.5 is set.




b7-b0 = **DPR3_7-0**: these bits define the 16-Kbyte Data memory page. They are used as the most significant address bits (A21-14) to extend the address during a Data memory access. The DPR3 register is involved when the virtual address is in the range C000h-FFFFh.

**MMU REGISTERS** (Cont'd)

### 2.7.2 CSR: Code Segment Register

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

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

Note that the CSR register can only be read and not written for data operations (There are some exceptions which are documented in the following paragraph). It is, however, modified either directly by means of the JPS and CALLS instructions, or indirectly via the stack, by means of the RETS instruction.

**CSR R244 (**F4h) Page 21 Read/Write

Code Segment Register

Reset value: 0000 0000b (00h)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | CSR_5 | CSR_4 | CSR_3 | CSR_2 | CSR_1 | CSR_0 |

b7-b6 = Reserved, keep in reset state.

b5-b0 = **CSR_5-0**: these bits define the 64-Kbyte memory segment (among 64) which contains the code being executed. These bits are used as the most significant address bits (A21-16).

### 2.7.3 ISR: Interrupt Segment Register

**ISR R248** (F8h) Page 21 Read/Write

Interrupt Segment Register

Reset value: 0000 0000b (00h)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | ISR_5 | ISR_4 | ISR_3 | ISR_2 | ISR_1 | ISR_0 |

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

b7-b6 = Reserved, keep in reset state.

b5-b0 = **ISR_5-0**: These bits define the 64-Kbyte memory segment (among 64) which contains the interrupt vector table and the code for interrupt and DMA service routines (when Program Memory is used). These bits are used as the most significant address bits (A21-16). The ISR is used to extend the address space in two cases:

– Whenever an interrupt occurs: ISR points to the 64-Kbyte memory segment containing the interrupt vector table and the interrupt service routine code. See also Section 4 Interrupts.

– During DMA transactions between the peripheral and Program memory: ISR points to the 64 Kbyte Program Memory segment that will be involved in the DMA transaction.

### 2.7.4 DMASR: DMA Segment Register

**DMASR R249** (F9h) Page 21 Read/Write

DMA Segment Register

Reset value: 0010 0000b (20h)

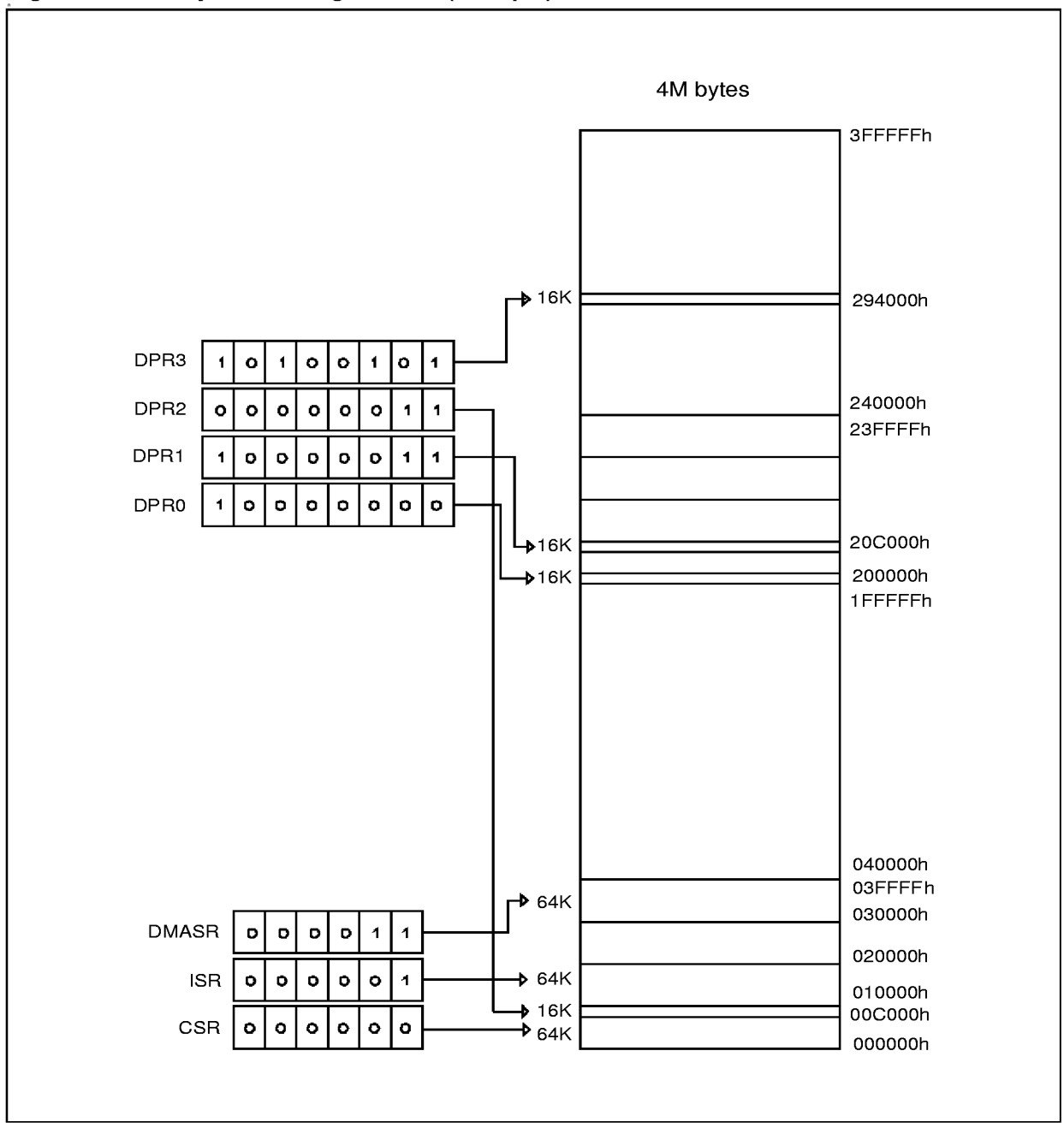
| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | DMA SR_5 | DMA SR_4 | DMA SR_3 | DMA SR_2 | DMA SR_1 | DMA SR_0 |

b7-b6 = Reserved, keep in reset state.

b5-b0 = **DMASR_5-0**: These bits define the 64-Kbyte Data Memory segment (among 64) used when a DMA transaction is performed between the peripheral's data register and Data Memory. These bits are used as the most significant address bits (A21-16). If Program Memory is used, the ISR register is used to extend the address.

**MMU REGISTERS** (Cont'd)

**Figure 14. Memory Addressing Scheme (example)**



## 2.8 MMU USAGE

### 2.8.1 Normal Program Execution

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

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

In typical microcontroller applications, less than 16 Kbytes of RAM are used, so just one of the four Data space pages is normally sufficient, and no change of DPR0-3 is needed during Program execution. It may be useful however to map part of the ROM into the data space if it contains strings, tables, bit maps, etc.

DPR registers must not be modified while they are referencing memory. For instance, the instruction “POPW DPR0” is legal only if the stack is kept either in the Register File or in a memory location above 8000h, where DPR2 and DPR3 are used. Otherwise, since DPR0 and DPR1 are modified by the instruction, unpredictable behaviour could result.

If there is to be frequent use of paging, the user can set bit 5 (DPRREM) in register R246 (EMR2) of Page 21. This swaps the location of registers DPR0-3 with that of the data registers of Ports 0-3. In this way, DPR registers can be accessed without the need to save/set/restore the Page Pointer Register. Port registers are therefore moved to page 21. Applications that require a lot of paging typically use more than 64 Kbytes of external memory, and as ports 0, 1 and 2 are required to address it, their data registers are unused.

**MMU USAGE** (Cont'd)

### 2.8.2 Interrupts

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

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

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

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

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

### 2.8.3 DMA

DMA uses the ISR for Program memory accesses and DMASR for Data memory accesses: this guarantees that a DMA will always find its memory segment(s), no matter what segment changes the application has performed. Unlike interrupts, DMA transactions cannot save/restore paging registers, so a dedicated segment register (DMASR) has been created for Data space accesses. Having only one register of this kind means that all DMA accesses should be programmed in one of the two following segments: the one pointed to by the ISR for the ROM, and the one referenced by the DMASR for the RAM.

# 3 REGISTER MAP, MEMORY AND PERIPHERAL CONFIGURATION

## 3.1 REGISTER MAP

The ST90158/135 register map, memory map and peripheral options are documented in this section. Use this reference information to supplement the functional descriptions given elsewhere in this document.

## 3.2 ST90158/135 REGISTER MAP

The following pages contain a list of ST90158/135 registers, grouped by peripheral or function.

Be very careful to correctly program both:

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

– Registers common to other functions.

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

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

**Table 7. Common Registers**

| Function or Peripheral | Common Registers |
|---|---|
| SCI, MFT | CICR + NICR + DMA REGISTERS + I/O PORT REGISTERS |
| ADC | CICR + NICR + I/O PORT REGISTERS |
| SPI, WDT, STIM | CICR + NICR + EXTERNAL INTERRUPT REGISTERS + I/O PORT REGISTERS |
| I/O PORTS | I/O PORT REGISTERS + MODER |
| EXTERNAL INTERRUPT | INTERRUPT REGISTERS + I/O PORT REGISTERS |
| RCCU | INTERRUPT REGISTERS + MODER |

| NAME | RXXX | HEXA | PAGE | TYPE | DESCRIPTION<br>RESET VALUE | Page n° |
|---|---|---|---|---|---|---|
| **DEVICE ARCHITECTURE** | | | | | | |
| **CICR** | **R230** | (E6h) | System | Read/Write | Central Interrupt Control Register<br>Reset Value: 1000 0111b (87h) | 18 |
| **FLAGR** | **R231** | (E7h) | System | Read/Write | Flag Register<br>Reset value: 0000 0000b (00h) | 19 |
| **RPO** | **R232** | (E8h) | System | Read/Write | Pointer 0 Register<br>Reset Value: xxxx xx00b (xxh) | 20 |
| **RP1** | **R233** | (E9h) | System | Read/Write | Pointer 1 Register<br>Reset Value: xxxx xx00b (xxh) | 20 |
| **PPR** | **R234** | (EAh) | System | Read/Write | Page Pointer Register<br>Reset value: xxxx xx00b (xxh) | 22 |
| **MODER** | **R235** | (EBh) | System | Read/Write | Mode Register<br>Reset value: 1110 0000b (E0h) | 22 |
| **USPHR** | **R236** | (ECh) | System | Read/Write | User Stack Pointer High Register<br>Reset value: undefined | 24 |
| **USPLR** | **R237** | (EDh) | System | Read/Write | User Stack Pointer Low Register<br>Reset value: undefined | 24 |
| **SSPHR** | **R238** | (EEh) | System | Read/Write | System Stack Pointer High Register<br>Reset value: undefined | 24 |
| **SSPLR** | **R239** | (EFh) | System | Read/Write | System Stack Pointer Low Register<br>Reset value: undefined | 24 |
| **DPR0** | **R240** | (F0h) | Page 21 | Read/Write | Data Page Register 0<br>Reset value: 0000 0000b (00h) | 29 |
| **DPR1** | **R241** | (F1h) | Page 21 | Read/Write | Data Page Register 1<br>Reset value: 0000 0001b (01h) | 29 |
| **DPR2** | **R242** | (F2h) | Page 21 | Read/Write | Data Page Register 2<br>Reset value: 0000 0010b (02h) | 29 |
| **DPR3** | **R243** | (F3h) | Page 21 | Read/Write | Data Page Register 3<br>Reset value: 1000 0011b (83h) | 29 |
| **CSR** | **R244** | (F4h) | Page 21 | Read/Write | Code Segment Register<br>Reset value: 0000 0000b (00h) | 30 |
| **ISR** | **R248** | (F8h) | Page 21 | Read/Write | Interrupt Segment Register<br>Reset value: 0000 0000b (00h) | 30 |
| **DMASR** | **R249** | (F9h) | Page 21 | Read/Write | DMA Segment Register<br>Reset value: 0010 0000b (20h) | 30 |

| NAME | RXXX | HEXA | PAGE | TYPE | DESCRIPTION<br>RESET VALUE | Page n° |
|---|---|---|---|---|---|---|
| **INTERRUPTS** | | | | | | |
| **CICR** | **R230** | (E6h) | System | Read/Write | Central Interrupt Control Register<br>Reset value: 1000 0111 (87h) | 57 |
| **EITR** | **R242** | (F2h) | Page 0 | Read/Write | External Interrupt Trigger Register<br>Reset value: 0000 0000 (00h) | 57 |
| **EIPR** | **R243** | (F3h) | Page 0 | Read/Write | External Interrupt Pending<br>Reset value: 0000 0000 (00h) | 57 |
| **EIMR** | **R244** | (F4h) | Page 0 | Read/Write | External Interrupt Mask-bit Register<br>Reset value: 0000 0000 (00h) | 58 |
| **EIPLR** | **R245** | (F5h) | Page 0 | Read/Write | External Interrupt Priority Level Register<br>Reset value: 1111 1111 (FFh) | 58 |
| **EIVR** | **R246** | (F6h) | Page 0 | Read/Write | External Interrupt Vector Register<br>Reset value: xxxx 0110 (X6h) | 58 |
| **NICR** | **R247** | (F7h) | Page 0 | Read/Write | Nested Interrupt Control<br>Reset value: 0000 0000 (00h) | 58 |
| **ON-CHIP DIRECT MEMORY ACCESS (DMA)** | | | | | | |
| **DCPR** | Address set by Peripheral | | | Read/Write | DMA Counter Pointer Register<br>Reset value: undefined | 64 |
| **IDCR** | Address set by Peripheral | | | Read/Write | Generic Peripheral Interrupt and DMA Control<br>Reset value: undefined | 64 |
| **DAPR** | Address set by Peripheral | | | Read/Write | DMA Address Pointer Register<br>Reset value: undefined | 64 |
| **RESET AND CLOCK CONTROL UNIT (RCCU)** | | | | | | |
| **MODER** | **R235** | (EBh) | System | Read/Write | Mode Register<br>Reset Value: 1110 0000 (E0h) | 71 |
| **CLKCTL** | **R240** | (0F0h) | Page 55 | Read Write | Clock Control Register<br>Reset Value: 00000000b (00h) | 71 |
| **CLK_FLAG** | **R242** | (0F2h) | Page 55 | Read/Write | Clock Flag Register<br>Reset Value: 01001000b after a Watchdog Reset<br>Reset Value: 00101000b after a Software Reset<br>Reset Value: 00001000b after a Power-On Reset | 72 |
| **PLLCONF** | **R246** | (0F6h) | Page 55 | Read/Write | PLL Configuration Register<br>Reset Value: xx00x111b | 72 |
| **EXTERNAL MEMORY INTERFACE (EXTMI)** | | | | | | |
| **EMR1** | **R245** | (F5h) | Page 21 | R/W | External Memory Register 1<br>Reset value: 1000 0000b (80h) | 85 |
| **EMR2** | **R246** | **(F6h)** | **Page 21** | R/W | External Memory Register 2<br>Reset value: 0000 1111b (0Fh) | 86 |
| **WCR** | **R252** | (FCh) | Page 0 | Read/Write | Wait Control Register<br>Reset Value: 0111 1111 (7Fh) | 87 |

| NAME | RXXX | HEXA | PAGE | TYPE | DESCRIPTION<br>RESET VALUE | Page n° |
|---|---|---|---|---|---|---|
| **TIMER/WATCHDOG (WDT)** | | | | | | |
| **WDTHR** | **R248** | (F8h) | Page 0 | Read/Write | Timer/Watchdog High Register<br>Reset value: 1111 1111b (FFh) | 99 |
| **WDTLR** | **R249** | (F9h) | Page 0 | Read/Write | Timer/Watchdog Low Register<br>Reset value: 1111 1111b (FFh) | 99 |
| **WDTPR** | **R250** | (FAh) | Page 0 | Read/Write | Timer/Watchdog Prescaler Register<br>Reset value: 1111 1111b (FFh) | 99 |
| **WDTCR** | **R251** | (FBh) | Page 0 | Read/Write | Watchdog Timer Control Register<br>Reset value: 0001 0010b (12h) | 100 |
| **WCR** | **R252** | (FCh) | Page 0 | Read/Write | Wait Control Register<br>Reset value: 0111 1111b (7Fh) | 100 |
| **EIVR** | **R246** | (F6h) | Page 0 | Read/Write | External Interrupt Vector Register<br>Reset value: xxxx 0110b (x6h) | 100 |
| **MULTIFUNCTION TIMER (MFT)** | | | | | | |
| **REG0HR** | **R240** | (F0h) | | Read/Write | Capture Load Register 0 High<br>Reset value: undefined | 117 |
| **REG1HR** | **R242** | (F2h) | | Read/Write | Capture Load High Register 1<br>Reset value: undefined | 117 |
| **REG1LR** | **R243** | (F3h) | | Read/Write | Capture Load Low Register 1<br>Reset value: undefined | 117 |
| **CMP0HR** | **R244** | (F4h) | | Read/Write | Compare 0 High Register<br>Reset value: 00 | 117 |
| **CMP0LR** | **R245** | (F5h) | | Read/Write | Compare 0 Register Low<br>Reset value: 00 | 117 |
| **CMP1HR** | **R246** | (F6h) | | Read/Write | Compare 1 High Register<br>Reset value: 00 | 117 |
| **CMP1LR** | **R247** | (F7h) | | Read/Write | Compare 1 Low Register<br>Reset value: 00 | 117 |
| **TCR** | **R248** | (F8h) | | Read/Write | Timer Control Register<br>Reset value: 0000 0xxxb | 118 |
| **TMR** | **R249** | (F9h) | | Read/Write | Timer Mode Register<br>Reset value: 0000 0000b (00h) | 118 |
| **ICR** | **R250** | (FAh) | | Read/Write | External Input Control Register<br>Reset value: 0000 xxxxb (0Xh) | 119 |
| **PRSR** | **R251** | (FBh) | | Read/Write | Prescaler Register<br>Reset value: 0000 0000b (00h) | 120 |
| **OACR** | **R252** | (FCh) | | Read/Write | Output A Control Register<br>Reset value: xxxx xx0xb | 120 |

| NAME | RXXX | HEXA | PAGE | TYPE | DESCRIPTION<br>RESET VALUE | Page n° |
|---|---|---|---|---|---|---|
| **OBCR** | **R253** | (FDh) | | Read/Write | Output B Control Register<br>Reset value: xxxx xx0xb | 121 |
| **FLAGR** | **R254** | (FEh) | | Read/Write | Flags Register<br>Reset value: 0000 0000b (00h) | 121 |
| **IDMR** | **R255** | (FFh) | | Read/Write | Interrupt/DMA Mask Register<br>Reset value: 0000 0000b (00h) | 122 |
| **DCPR** | **R240** | (F0h) | | Read/Write | DMA Counter Pointer Register<br>Reset value: undefined | 122 |
| **DAPR** | **R241** | (F1h ) [R245 (F5h)] | | Read/Write | DMA Address Pointer Register<br>Reset value: undefined | 123 |
| **IVR** | **R242** | (F2h) [R246 (F6h)] | | Read/Write | Interrupt Vector Register<br>Reset value: xxxx xxx0b | 123 |
| **IDCR** | **R243** | (F3h) [R247 (F7h)] | | Read/Write | Interrupt/DMA Control Register<br>Reset value: 1100 0111b (C7h) | 124 |
| **IOCR** | **R248** | (F8h) | | Read/Write | I/O Connection Register<br>Reset value: 1111 1100b (FCh) | 124 |
| **STANDARD TIMER (STIM)** | | | | | | |
| **STH** | **R240** | (F0h) | Page 0B | Read/Write | Counter High Byte Register | 128 |
| **STL** | **R241** | (F1h) | Page 0B | Read/Write | Counter Low-Byte Register<br>Reset value: 1111 1111b (FFh) | 128 |
| **STP** | **R242** | (F2h) | Page 0B | Read/Write | Standard Timer Prescaler Register<br>Reset value: 1111 1111b (FFh) | 128 |
| **STC** | **R243** | (F3h) | Page 0B | Read/Write | Standard Timer Control Register<br>Reset value: 0001 0100b (14h) | 128 |
| **SERIAL PERIPHERAL INTERFACE (SPI)** | | | | | | |
| **SPIDR** | **R253** | (FDh) | Page 0 | Read/Write | SPI Data Register<br>Reset Value: undefined | 132 |
| **SPICR** | **R254** | (FEh) | Page 0 | Read/Write | SPI Control Register<br>Reset Value: 0000 0000b (00h) | 132 |

| NAME | RXXX | HEXA | PAGE | TYPE | DESCRIPTION<br>RESET VALUE | Page n° |
|---|---|---|---|---|---|---|
| **SERIAL COMMUNICATIONS INTERFACE (SCI)** | | | | | | |
| **RDCPR** | **R240** | (F0h) | | Read/Write | Receiver DMA Transaction Counter Pointer<br>Reset value: undefined | 150 |
| **RDAPR** | **R241** | (F1h) | | Read/Write | Receiver DMA Source Address Pointer<br>Reset value: undefined | 150 |
| **TDCPR** | **R242** | (F2h) | | Read/Write | Transmitter DMA Transaction Counter Pointer<br>Reset value: undefined | 150 |
| **TDAPR** | **R243** | (F3h) | | Read/Write | Transmitter DMA Destination Address Pointer<br>Reset value: undefined | 150 |
| **IVR** | **R244** | (F4h) | | Read/Write | Interrupt Vector Register<br>Reset value: undefined | 151 |
| **ACR** | **R245** | (F5h) | | Read/Write | Address/Data Compare Register<br>Reset value: undefined | 151 |
| **IMR** | **R246** | (F6h) | | Read/Write | Interrupt Mask Register<br>Reset value: 0xx0 0000b | 151 |
| **ISR** | **R247** | (F7h) | | Read/Write | Interrupt Status Register<br>Reset value: undefined | 152 |
| **RXBR** | **R248** | (F8h) | | Read only | Receive Buffer Register<br>Reset value: undefined | 152 |
| **TXBR** | **R248** | (F8h) | | Write only | Transmitter Buffer Register<br>Reset value: undefined | |
| **IDPR** | **R249** | (F9h) | | Read/Write | Interrupt/DMA Priority Register<br>Reset value: undefined | 153 |
| **CHCR** | **R250** | (FAh) | | Read/Write | Character Configuration Register<br>Reset value: undefined | 154 |
| **CCR** | **R251** | (FBh) | Read/Write | | Clock Configuration Register<br>Reset value: 0000 0000 (00h) | 155 |
| **SICR** | **R254** | (FEh) | Read/Write | | Synchronous Input Control<br>Reset value : 0000 0011b (03h) | 156 |
| **SOCR** | **R255** | (FFh) | Read/Write | | Synchronous Output Control<br>Reset value: 0000 0001 (01h) | 156 |
| **BRGHR** | **R252** | (FCh) | Read/Write | | Baud Rate Generator High Register ()<br>Reset value: undefined | 157 |
| **BRGLR** | **R253** | (FDh) | Read/Write | | Baud Rate Generator Low Register<br>Reset value: undefined | 157 |

| NAME | RXXX | HEXA | PAGE | TYPE | DESCRIPTION<br>RESET VALUE | Page n° |
|---|---|---|---|---|---|---|
| **EIGHT-CHANNEL ANALOG TO DIGITAL CONVERTER (ADC8)** | | | | | | |
| **D0R** | **R240** | (F0h) | Page 63 | Read/Write | Channel 0 Data Register<br>Reset Value: undefined | 162 |
| **D1R** | **R241** | (F1h) | Page 63 | Read/Write | Channel 1 Data Register<br>Reset Value: undefined | 162 |
| **D2R** | **R242** | (F2h) | Page 63 | Read/Write | Channel 2 Data Register<br>Reset Value: undefined | 162 |
| **D3R** | **R243** | (F3h) | Page 63 | Read/Write | Channel 3 Data Register<br>Reset Value: undefined | 162 |
| **D4R** | **R244** | (F4h) | Page 63 | Read/Write | Channel 4 Data Register<br>Reset Value: undefined | 162 |
| **D5R** | **R245** | (F5h) | Page 63 | Read/Write | Channel 5 Data Register<br>Reset Value: undefined | 162 |
| **D6R** | **R246** | (F6h) | Page 63 | Read/Write | Channel 6 Data Register<br>Reset Value: undefined | 162 |
| **D7R** | **R247** | (F7h) | Page 63 | Read/Write | Channel 7 Data Register<br>Reset Value: undefined | 162 |
| **LT6R** | **R248** | (F8h) | Page 63 | Read/Write | Channel 6 Lower Threshold Register<br>Reset Value: undefined | 163 |
| **LT7R** | **R249** | (F9h) | Page 63 | Read/Write | Channel 7 Lower Threshold Register<br>Reset Value: undefined | 163 |
| **UT6R** | **R250** | (FAh) | Page 63 | Read/Write | Channel 6 Upper Threshold Register<br>Reset Value: undefined | 163 |
| **UT7R** | **R251** | (FBh) | Page 63 | Read/Write | Channel 7 Upper Threshold Register<br>Reset Value: undefined | 163 |
| **CRR** | **R252** | (FCh) | Page 63 | Read/Write | Compare Result Register<br>Reset Value: 0000 1111 (0Fh) | 163 |
| **CLR** | **R253** | (FDh) | Page 63 | Read/Write | Control Logic Register<br>Reset Value: 0000 0000 (00h) | 164 |
| **ICR** | **R254** | (FEh) | Page 63 | Read/Write | Interrupt Control Register<br>Reset Value: 0000 1111 (0Fh) | 165 |
| **IVR** | **R255** | (FFh) | Page 63 | Read/Write | Interrupt Vector Register<br>Reset Value: xxxx xx10 (x2h) | 165 |

### 3.3 MEMORY CONFIGURATION

The Program memory space of the ST90135/158, 32/64/K bytes of directly addressable on-chip memory, is fully available to the user.

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

### 3.4 EPROM PROGRAMMING

The 65536 bytes of EPROM memory of the ST90E158 may be programmed by using the EPROM Programming Boards (EPB) available from SGS-THOMSON.

#### EPROM Erasing

The EPROM of the windowed package of the ST90E158 may be erased by exposure to Ultra-Violet light.

The erasure characteristic of the ST90E158 is such that erasure begins when the memory is exposed to light with a wave lengths shorter than approximately 4000Å. It should be noted that sunlight and some types of fluorescent lamps have wavelengths in the range 3000-4000Å. It is thus recommended that the window of the ST90E158 packages be covered by an opaque label to prevent unintentional erasure problems when testing the application in such an environment.

The recommended erasure procedure of the EPROM is the exposure to short wave ultraviolet light which have a wave-length 2537Å. The integrated dose (i.e. U.V. intensity x exposure time) for erasure should be a minimum of 15W-sec/cm2. The erasure time with this dosage is approximately 30 minutes using an ultraviolet lamp with 12000mW/cm2 power rating. The ST92E158 should be placed within 2.5cm (1Inch) of the lamp tubes during erasure.

**Table 8. First 6 Bytes of Program Space**

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

**Figure 15. ST9 User Memory Map**

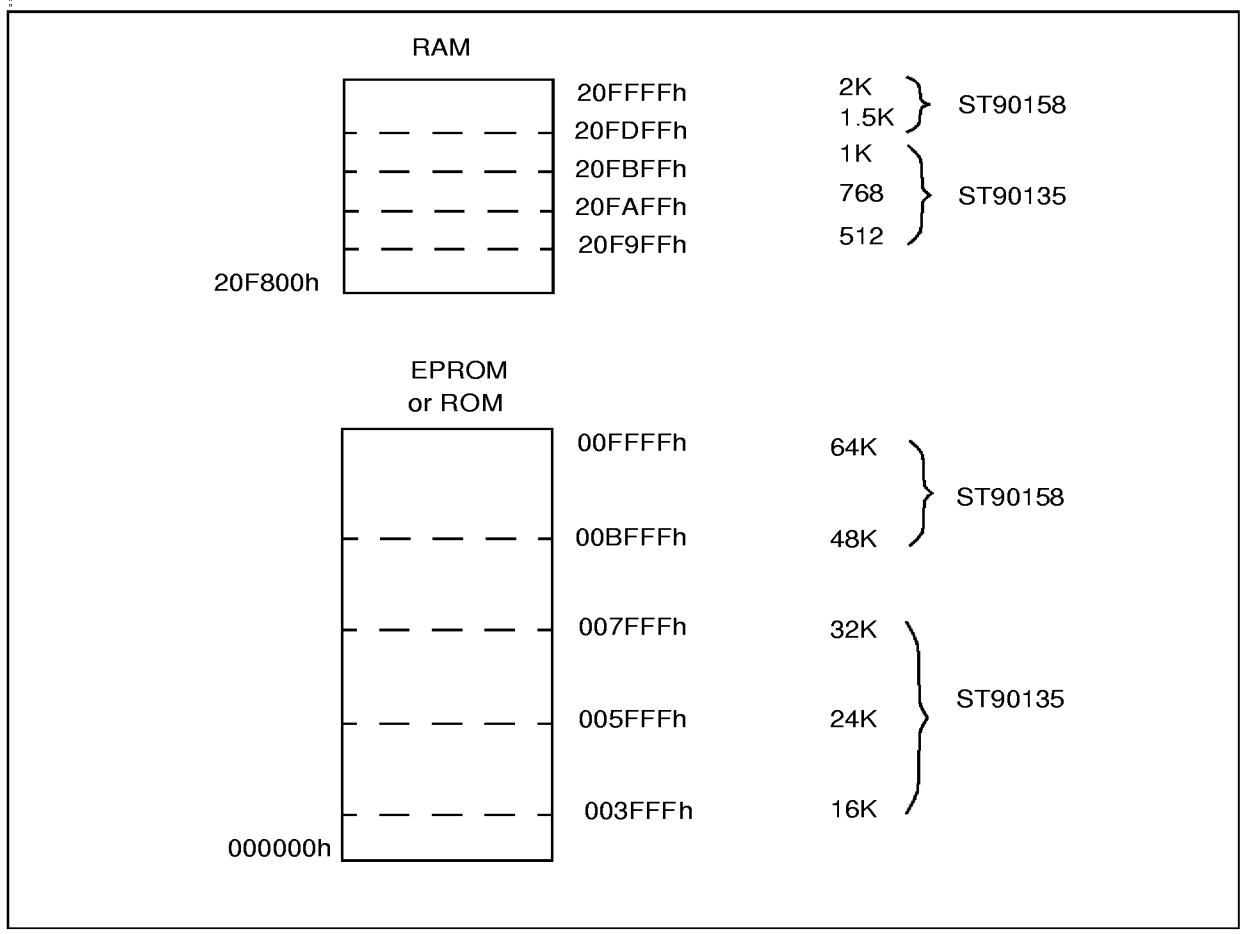


**Table 9. Group F Pages**

Resources available on the ST90158/ST90135 devices:

| DEC | HEX | 00 | 02 | 03 | 08 | 09 | 10 | 11 | 12 | 13 | 21 | 24 | 25 | 43 | 55 | 63 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R255 | RFF | Res. | | | | | | | | | | | | | | |
| R254 | RFE | MSPI | Res. | PORT 7 | | Res. | | | | Res. | | | | PORT 9 | | |
| R253 | RFD | | | | | | | | | | | | | | Res. | |
| R252 | RFC | WCR | | | | | | | | | | | | | | |
| R251 | RFB | WDT | | | | MFT | | Res. | | | | | | PORT 8 | | |
| R250 | RFA | | PORT 2 | PORT 6 | | | | | | | | | | | | |
| R249 | RF9 | | | | | | | | | | | | | | | |
| R248 | RF8 | | | | MFT1 | | MFT0 (*) | | MFT3 | | MMU | SCI0 | SCI1 (*) | | | A/D |
| R247 | RF7 | EXT INT | Res. | | | | | | | | | | | Res. | | |
| R246 | RF6 | | PORT 1 | PORT 5 | | MFT1 | | | | MFT3 | | | | | RCCU | |
| R245 | RF5 | | | | | | | | | | | | | | | |
| R244 | RF4 | | | | | | | | | | | | | | | |
| R243 | RF3 | | Res. | | | | | | | | | | | | | |
| R242 | RF2 | | PORT 0 | PORT 4 | | MFT0 (*) | | STIM | | Res. | | | | | | |
| R241 | RF1 | Res. | | | | | | | | | | | | | | |
| R240 | RF0 | | | | | | | | | | | | | | | |

(*) ST90158/ST90E158 only. Not present on ST90135.

## 3.5 PERIPHERAL CONFIGURATION

### 3.5.1 Serial Peripheral Interface (SPI)

| External Signal | Availability |
|---|---|
| SDI | Y |
| SDO | Y |
| SCK | Y |

### 3.5.2 Standard Timer (STIM)

| External Signal | Availability |
|---|---|
| STIN | N |
| STOUT | Y |

### 3.5.3 Timer/Watchdog (WDT)

| Signal | Availability |
|---|---|
| WDIN | Y |
| WDOUT | Y |
| NMI | Y |
| HW0_SW1 | Y |

### 3.5.4 I/O Port Styles

Refer to the I/O Ports section for a general description of the I/O ports. The ST90158/ST90135 specific configuration is given in the following tables.

| Ports | Physical Pull-up | Port Style |
|---|---|---|
| P0 | Yes | Standard |
| P1 | Yes | Standard |
| P2 | No | Standard |
| P4 | Yes | Schmitt Trigger |
| P5 | Yes | Schmitt Trigger |
| P6 | No | Standard |
| P7 | Yes | Schmitt Trigger |
| P8 | Yes | Schmitt Trigger |
| P9 | Yes | Schmitt Trigger |

# 4 INTERRUPTS

## 4.1 INTRODUCTION

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

The ST9 CPU can receive requests from the following sources:

– On-chip peripherals

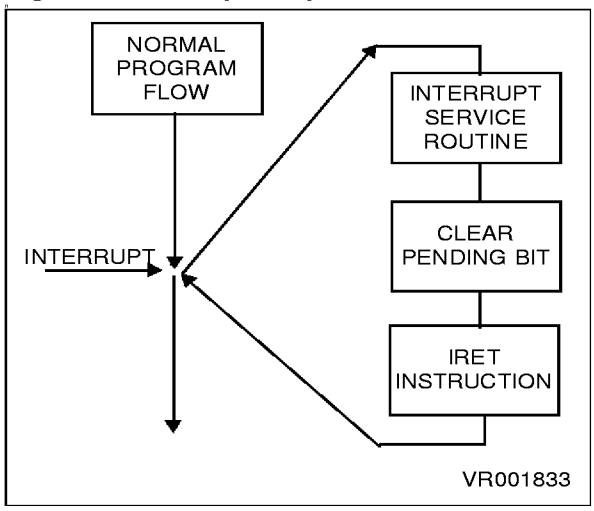
– External pins

– Top-Level Pseudo-non-maskable interrupt

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

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

**Figure 16. Interrupt Response**




## 4.2 INTERRUPT VECTORING

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

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

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

The Interrupt Vector table, containing the addresses of the Interrupt Service Routines, is located in the first 256 locations of Memory pointed to by the ISR, thus allowing 8-bit vector addressing.

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

***Important***:

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

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

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

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

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

### 4.3 INTERRUPT PRIORITY LEVELS

The ST9 supports a fully programmable interrupt priority structure. Nine priority levels are available to define the channel priority relationships. Each channel has a 3 bit field, PRL (Priority Level), that defines its priority level in the range from 0 to 7. 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 having the lowest priority, and level 0 the highest.

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/Watchdog. 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).

**Table 10. Daisy Chain Priority**

| | | |
|---|---|---|
| Highest Position | INTA0 | INT0/WDT |
| | INTA1 | INT1 |
| | INTB0 | INT2/SPI |
| | INTB1 | INT3 |
| | INTC0 | INT4/STIM |
| | INTC1 | INT5 |
| | INTD0 | INT6/RCCU |
| | INTD1 | INT7 |
| | TIMER0 | |
| | SCI0 | |
| | SCI1 | |
| | A/D | |
| | TIMER3 | |
| Lowest Position | TIMER1 | |

### 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 present, during which, for 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 Top Level Interrupt overrides every other priority.

If two or more requests occur at the same instant 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 10 Daisy Chain Priority.

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

The IAM control bit (CICR.3) selects Concurrent Arbitration mode when reset, or Nested Arbitration Mode when set.

#### 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.

The interrupt cycle performs the following steps:

– Disables all maskable interrupt requests by clearing CICR.IEN.

– Pushes the PC low byte onto the system stack.

– Pushes the PC high byte onto the system stack.

**PRIORITY LEVEL ARBITRATION**(Continued)

– Pushes the Flag register onto the system stack.

– Loads the PC with the 16-bit vector stored in the Vector Table, pointed to by the Interrupt Vector Register (IVR).

For the duration of the interrupt service routine, ISR is used instead of CSR, and the interrupt stack frame is identical to that of the original ST9: only the PC and Flags are pushed. This avoids saving the CSR on the stack in the event of an interrupt, thus ensuring a faster interrupt response time. It is not possible for an interrupt service routine to perform inter-segment calls or jumps: these instructions would update the CSR, which, in this case, is not used (ISR is used instead). The code segment size for all interrupt service routines is thus limited to 64K bytes.

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

– Pops the Flag register from the system stack.

– Pops CSR from the system stack.

– Pops the PC high byte from the system stack.

– Pops the PC low byte from the system stack.

– Enables all un-masked Interrupts, by setting the CICR.IEN bit.

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

**Note**: In Concurrent mode, the source priority level is only meaningful during the arbitration phase, where it is compared with all the other priority levels and with the CPL. However, 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's priority, giving rise to unforeseen and undesirable interrupt response sequences.

In a typical case, three pending requests with different priority levels (eg. 2, 3, 4) generate requests at the same time. The three interrupt service routines set Interrupt Enable by means of the `ei` instruction placed at the beginning of the routine, in order to minimise response time for requests having a higher priority than the one being serviced (usually, the higher the priority, the more urgently the routine needs to be executed).

Unfortunately, the three interrupt service routines will be effectively executed in exactly the opposite order to their priorities. The level 2 interrupt routine will be acknowledged first, then, when the `ei` instruction is executed, it will be interrupted by the level 3 interrupt routine, which itself will be interrupted by the level 4 interrupt routine. When the level 4 interrupt routine is completed, the level 3 interrupt routine will be recovered and finally, the level 2 interrupt routine, which in fact had the highest priority.

**It is therefore recommended to avoid inserting the `ei` instruction in the interrupt service routine, when in Concurrent mode**

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

REMARKS

**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 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.

**PRIORITY LEVEL ARBITRATION**(Continued)

**Figure 17. Example of a Sequence of Interrupt Requests with :**
**- Concurrent mode selected**
**- IEN set to 1 during interrupt service routine execution**

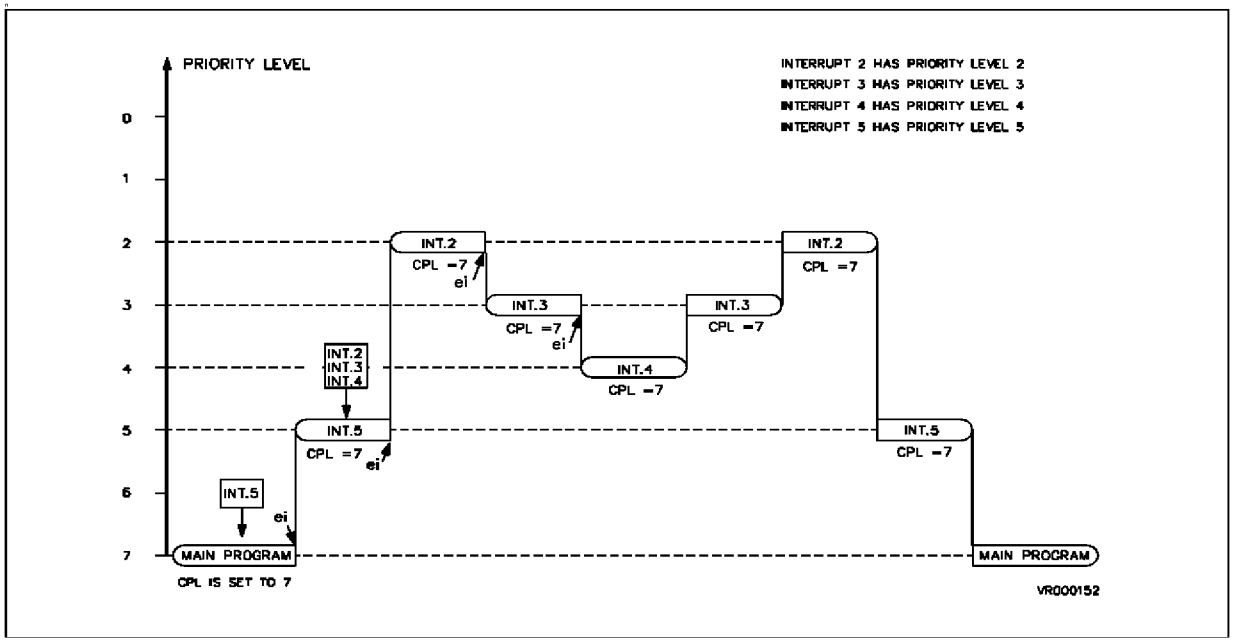
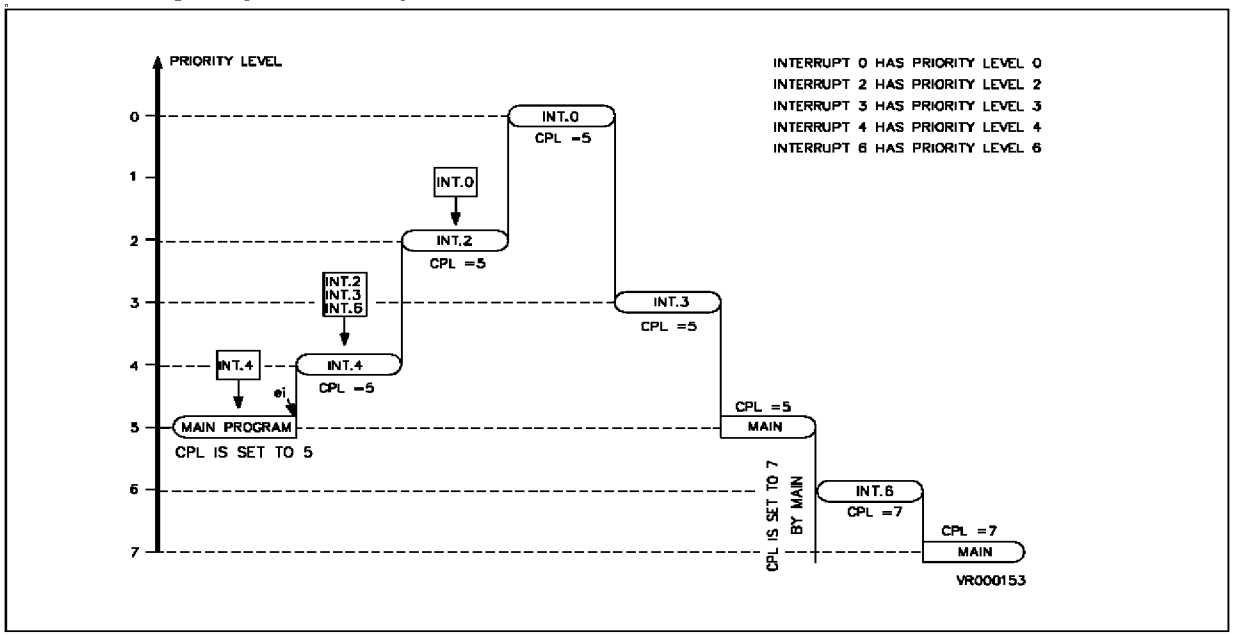



**Figure 18. Example of a Sequence of Interrupt Requests with :**
**- Concurrent mode selected and**
**- IEN unchanged by the interrupt routines**



**PRIORITY LEVEL ARBITRATION**(Continued)

### 4.4.2 Nested Mode

The difference between Nested mode and Concurrent mode, lies in the modification of the Current Priority Level (CPL) during interrupt processing. The arbitration phase is basically identical to Concurrent mode, however, once the request is acknowledged, the CPL is saved in the Nested Interrupt Control Register (NICR) by setting the NICR bit corresponding to the CPL value (i.e. if the CPL is 3, the bit 3 will be set). The CPL is then loaded with the priority of the request just acknowledged; the next arbitration cycle is thus performed with reference to the priority of the interrupt service routine currently being executed.

The interrupt cycle performs the following steps:

– Disables all maskable interrupts by clearing CICR.IEN.

– Saves the CPL in the special NICR stack 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 currently being serviced.

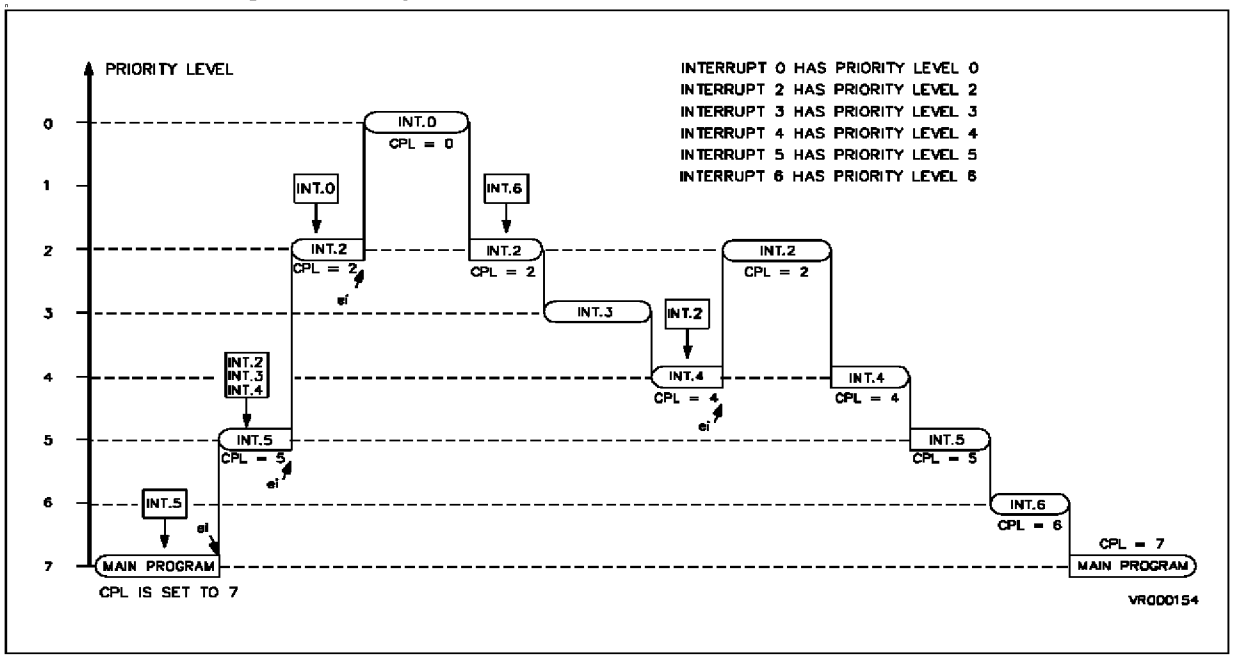
– Pushes the PC-low byte onto the System Stack.

– Pushes the PC-high byte onto the System Stack.

– Pushes the Flag Register onto the System Stack.

– Loads the PC with the vector pointed to by IVR.

**Figure 19. Example of a Sequence of Interrupt Requests with :**
**- Nested mode**
**- IEN set to 1 during the interrupt routine execution**



**PRIORITY LEVEL ARBITRATION**(Continued)

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

– Pops off the Flag Register from the System Stack;

– Pops off the PC-high byte from the System Stack;

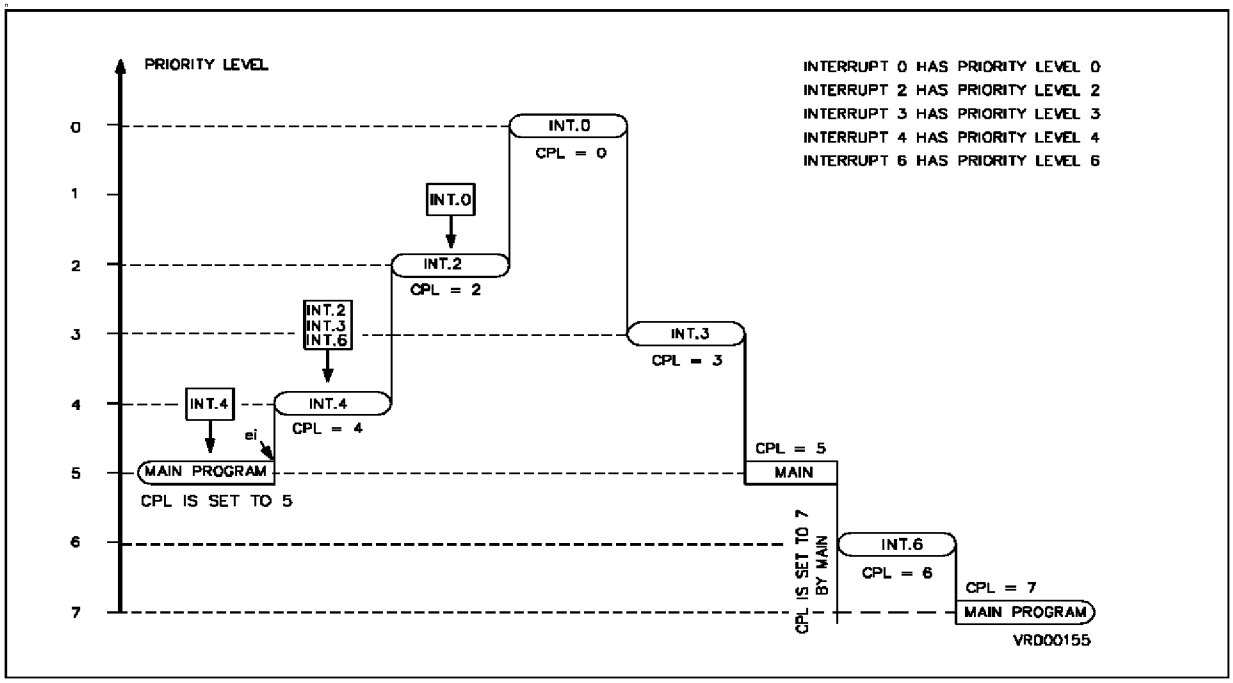
– Pops off the PC-low byte from the System Stack;

– Enables all unmasked interrupts by setting the CICR.IEN bit;

– Recovers 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.

**Figure 20. Example of a Sequence of Interrupt Requests with :**
**- Nested mode**
**- IEN unchanged by the interrupt routines**



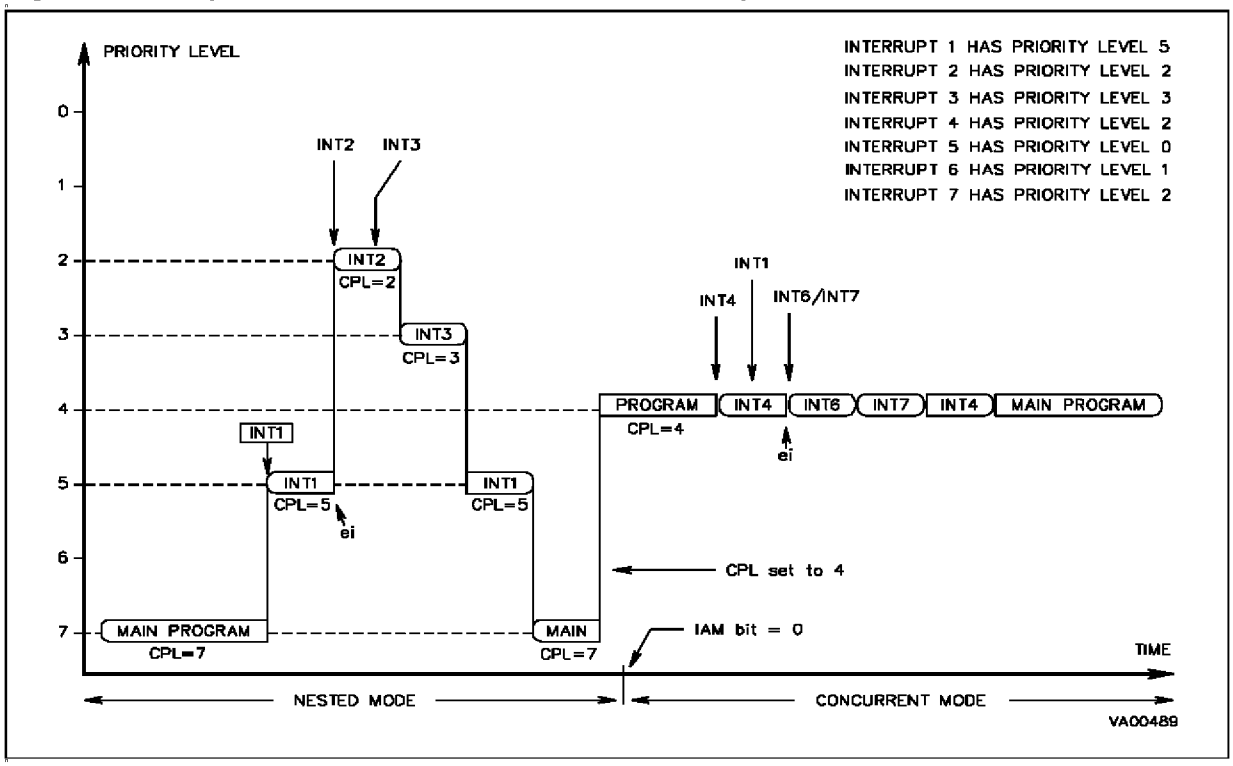
**PRIORITY LEVEL ARBITRATION**(Continued)

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

**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 illustrated in Figure 21. 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 remains pending, since it has a priority level lower than CPL.

**Figure 21. Example of a Nested and Concurrent Mode Sequence**



### 4.5 EXTERNAL INTERRUPTS

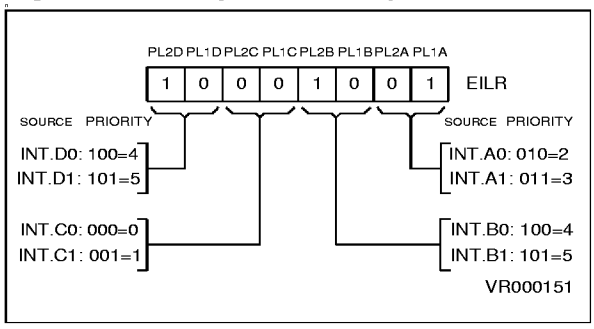
The standard ST9 core contains 8 external interrupts sources grouped into four pairs.

**Table 11. External Interrupt Channel Grouping**

| External Interrupt | Channel |
|---|---|
| INT7<br>INT6 | INTD1<br>INTD0 |
| INT5<br>INT4 | INTC1<br>INTC0 |
| INT3<br>INT2 | INTB1<br>INTB0 |
| INT1<br>INT0 | INTA1<br>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.

**Figure 22. Priority Level Examples**




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 22 shows an example of priority levels.

– 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.

– The source of the interrupt channel C0 can be selected between the external pin INT4 (when INTS = "1") or the on-chip Standard Timer.

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).

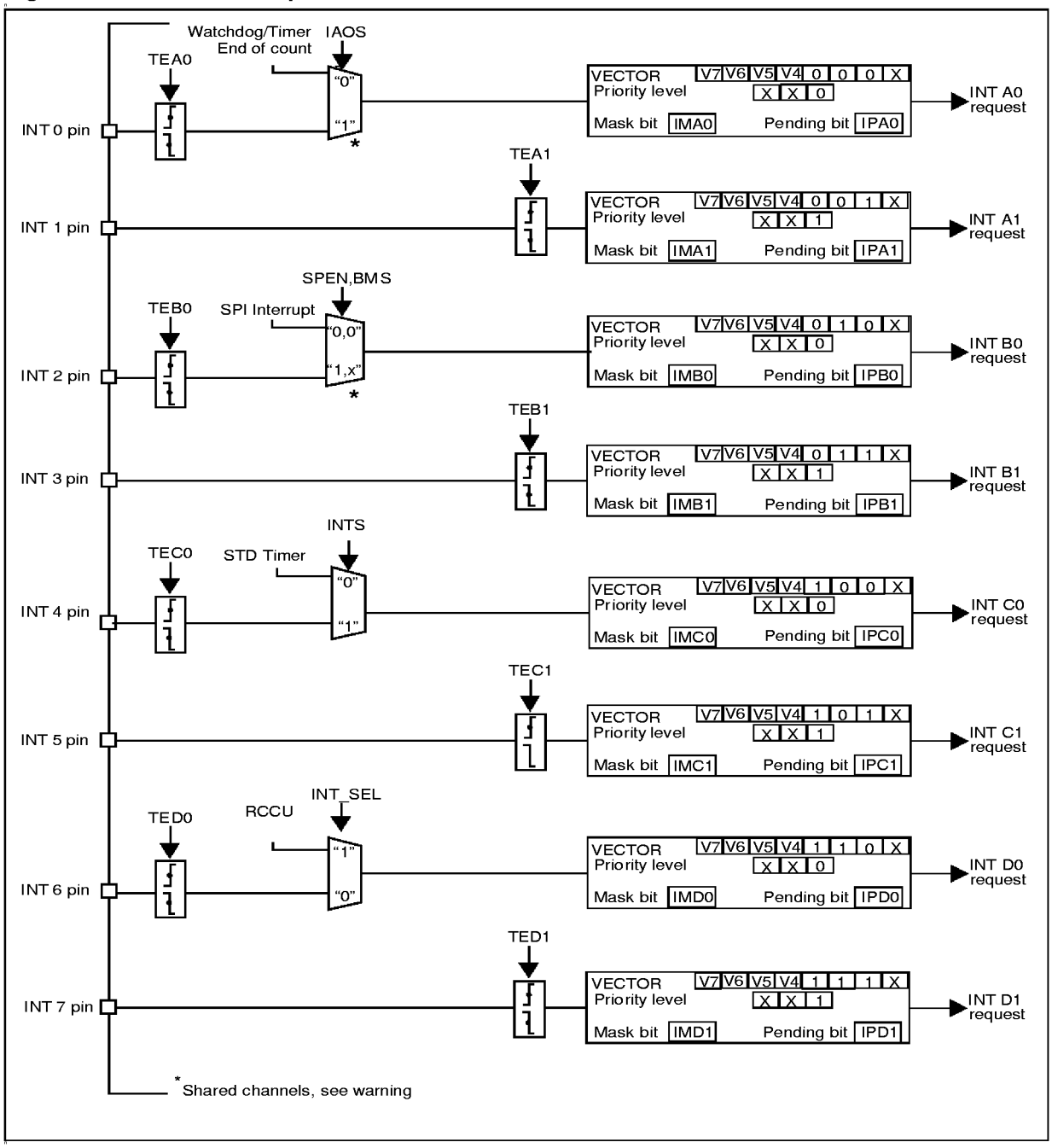
**Warning**: When using channels INTA0 and INTB0, shared by both external interrupts and SPI or Watchdog Timer, special care must be taken to configure control registers for peripheral and interrupts.

**Table 12. Multiplexed Interrupt Sources**

| Channel | Internal Interrupt Source | External Interrupt Source |
|---|---|---|
| INTA0 | Timer/Watchdog | INT0 |
| INTB0 | SPI Interrupt | INT2 |
| INTC0 | STD Timer | INT4 |
| INTD0 | RCCU | INT6 |

**EXTERNAL INTERRUPTS** (Cont'd)

**Figure 23. External Interrupts Control Bits and Vectors**



### 4.6 TOP LEVEL INTERRUPT

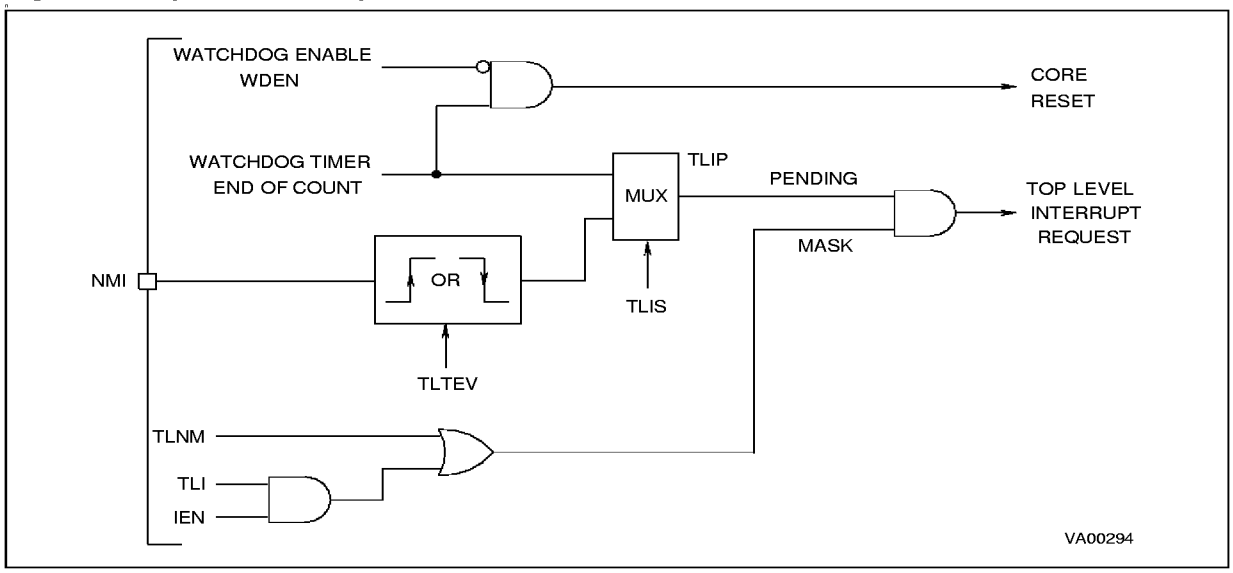
The Top Level Interrupt channel can be assigned either to the external pin NMI or to the Timer/ Watchdog according to the status of the control bit EIVR.TLIS (R246.2, Page 0). If this bit is high (the reset condition) the source is the external pin NMI, if it is low, the source is the Timer/ Watchdog 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 CICR.TLI bit (R230.5): it can be set or cleared to enable or disable respectively the Top Level Interrupt request. If it is enabled, the global Enable Interrupt bit, CICR.IEN (R230.4) must also be enabled in order to allow a Top Level Request.

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

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

**Warning**. The interrupt machine cycle of the Top Level Interrupt does not clear the CICR.IEN bit, and the corresponding `iret` does not set it.

**Figure 24. Top Level Interrupt Structure**



### 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.

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

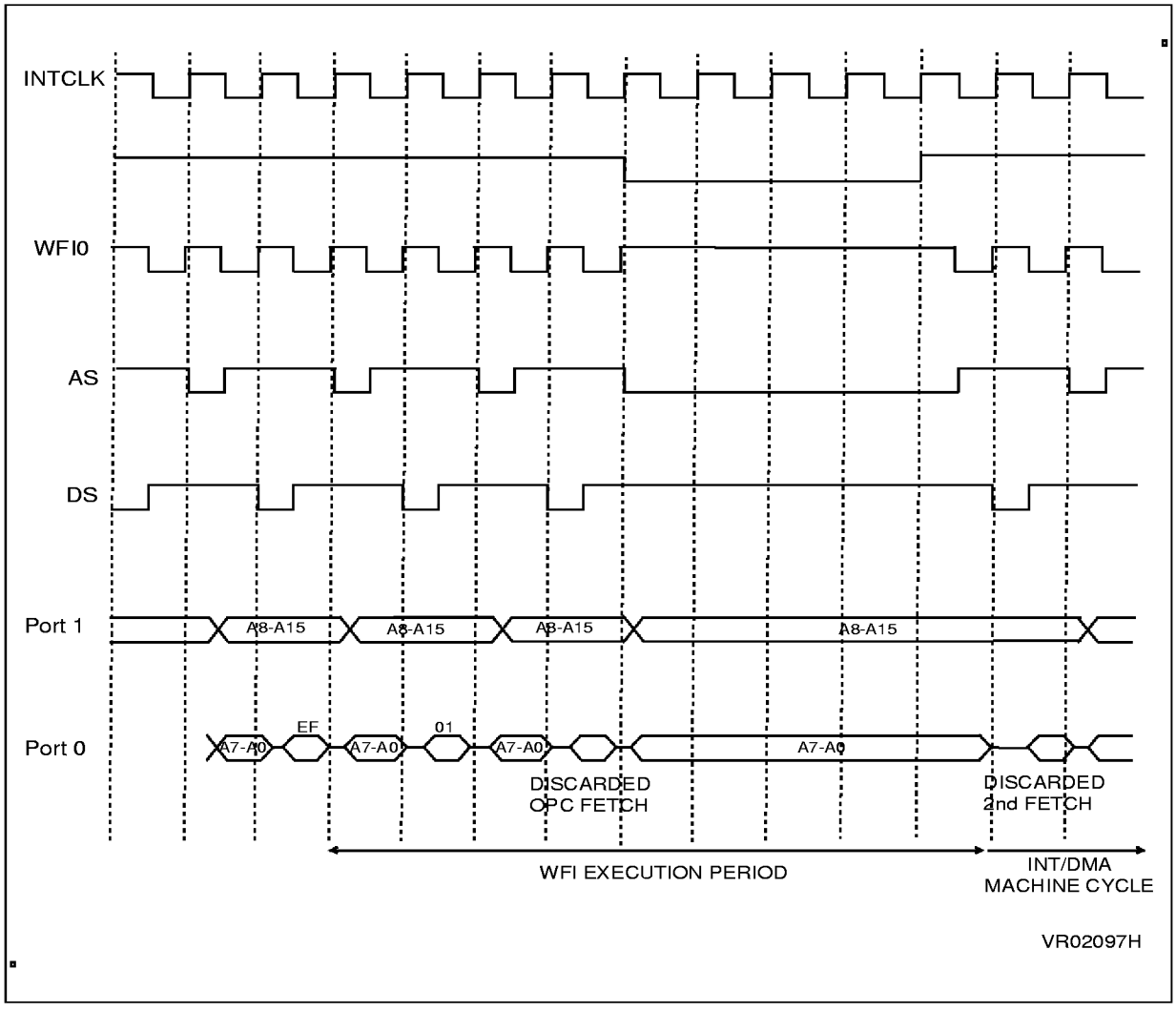
– **Interrupt Pending bit** (IP). Set by hardware when the Trigger Event occurs. Can be set/ cleared by software to generate/cancel pending interrupts and give the status for Interrupt polling.

– **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 25. Wait for Interrupt Instruction Timing**



### 4.8 INTERRUPT RESPONSE TIME

The interrupt arbitration protocol functions completely asynchronously from instruction flow, and requires 6 CPUCLK cycles to resolve the request's priority.

Requests are sampled every 5 CPUCLK cycles.

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

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

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

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

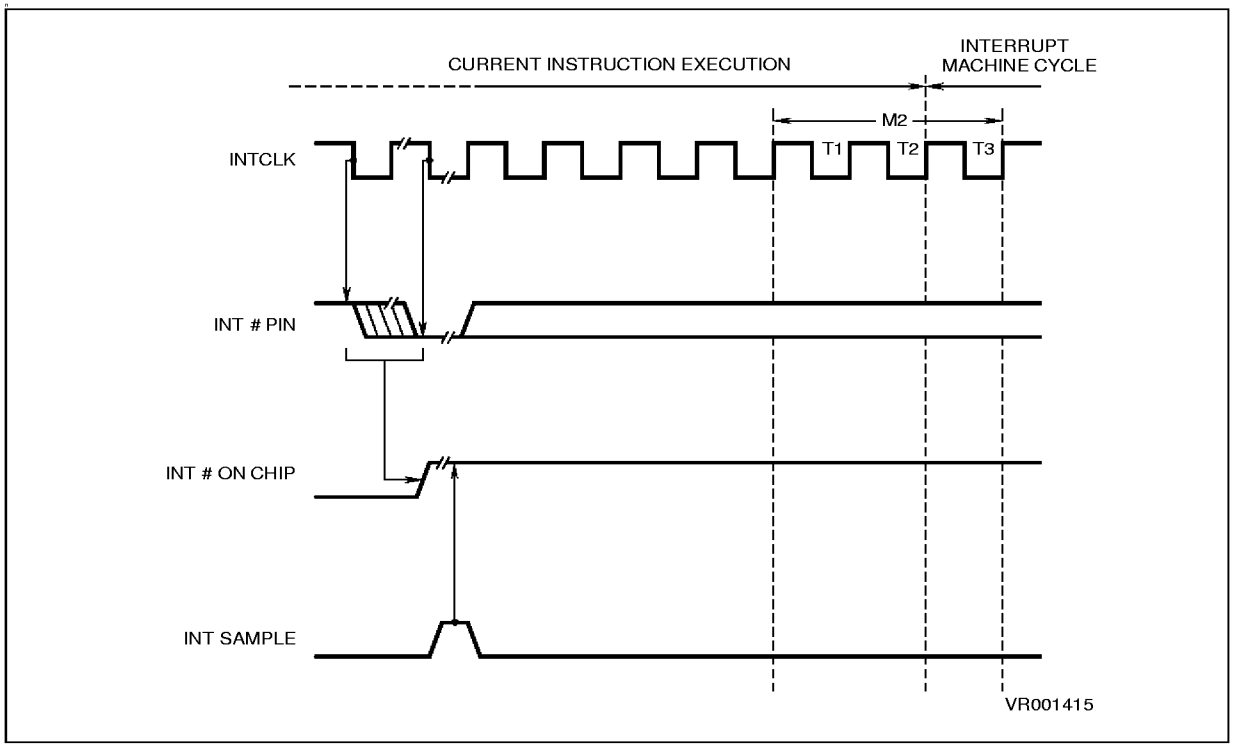
For an interrupt deriving from an external interrupt channel, the response time between a user event and the start of the interrupt service routine can range from a minimum of 26 clock cycles to a maximum of 49 clock cycles.

For a non-maskable Top Level interrupt, the response time between a user event and the start of the interrupt service routine can range from a minimum of 22 clock cycles to a maximum of 51 clock cycles.

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

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

**Figure 26. External Interrupt Response Time**



### 4.9 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.

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

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. Also set and reset by `EI`/`DI` instructions. To clear, use `DI;` to restore, use `DI, POP CICR,` to avoid an interrupt from corrupting the instruction.

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 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
Reset value: 0000 0000 (00h)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| IPD1 | IPD0 | IPDC1 | 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** (Cont'd)

**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

Reset value: 0000 0000 (00h)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| tTLNM | 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 by a 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 DIRECT MEMORY ACCESS (DMA)

## 5.1 INTRODUCTION

The ST9 includes on-chip Direct Memory Access (DMA) in order to provide high-speed data transfer between peripherals and memory or Register File. Multi-channel DMA is fully supported by peripherals having their own controller and DMA channel(s). Each DMA channel transfers data to or from contiguous locations in the Register File, or in Memory. The maximum number of bytes that can be transferred per transaction by each DMA channel is 222 with the Register File, or 65536 with Memory.

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 the Register File is 222, since two Registers are allocated for the Pointer and Counter. Register pairs are used for memory pointers and counters in order to offer the full 65536 byte and count capability.

## 5.2 DMA PRIORITY LEVELS

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.

An interrupt priority request must be higher than the CPL value in order to be acknowledged, whereas, 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 the CPL = 7.

DMA requests do not modify the CPL value, since the DMA transaction is not interruptable.

**Figure 27. DMA Data Transfer**

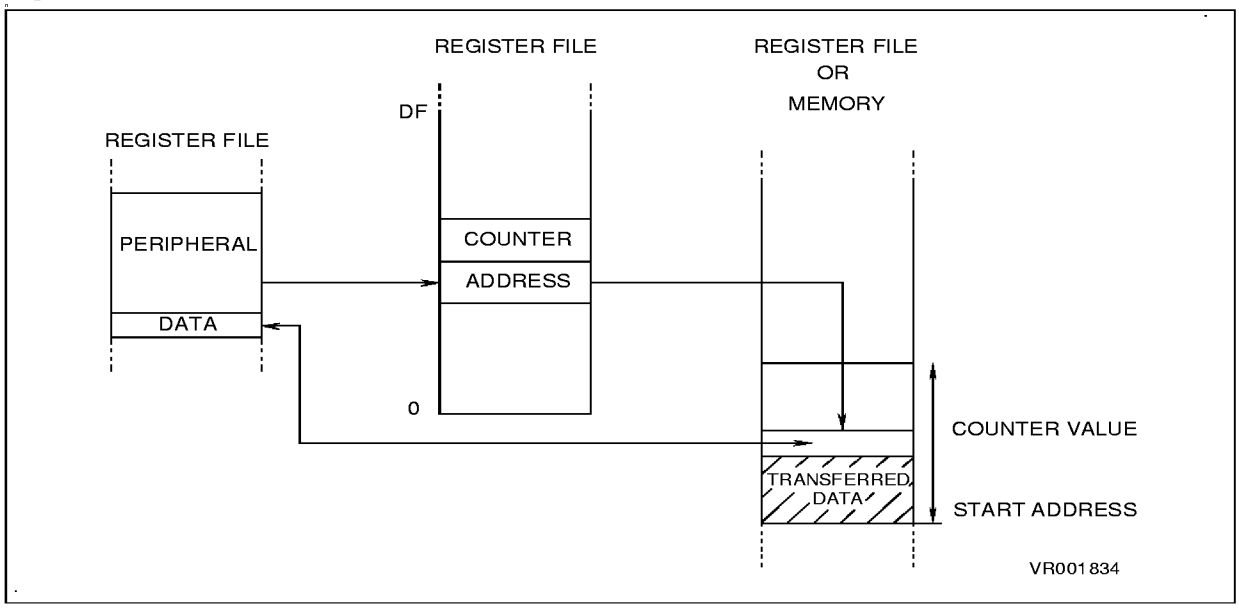


**Figure 28. DMA Transaction to Memory**

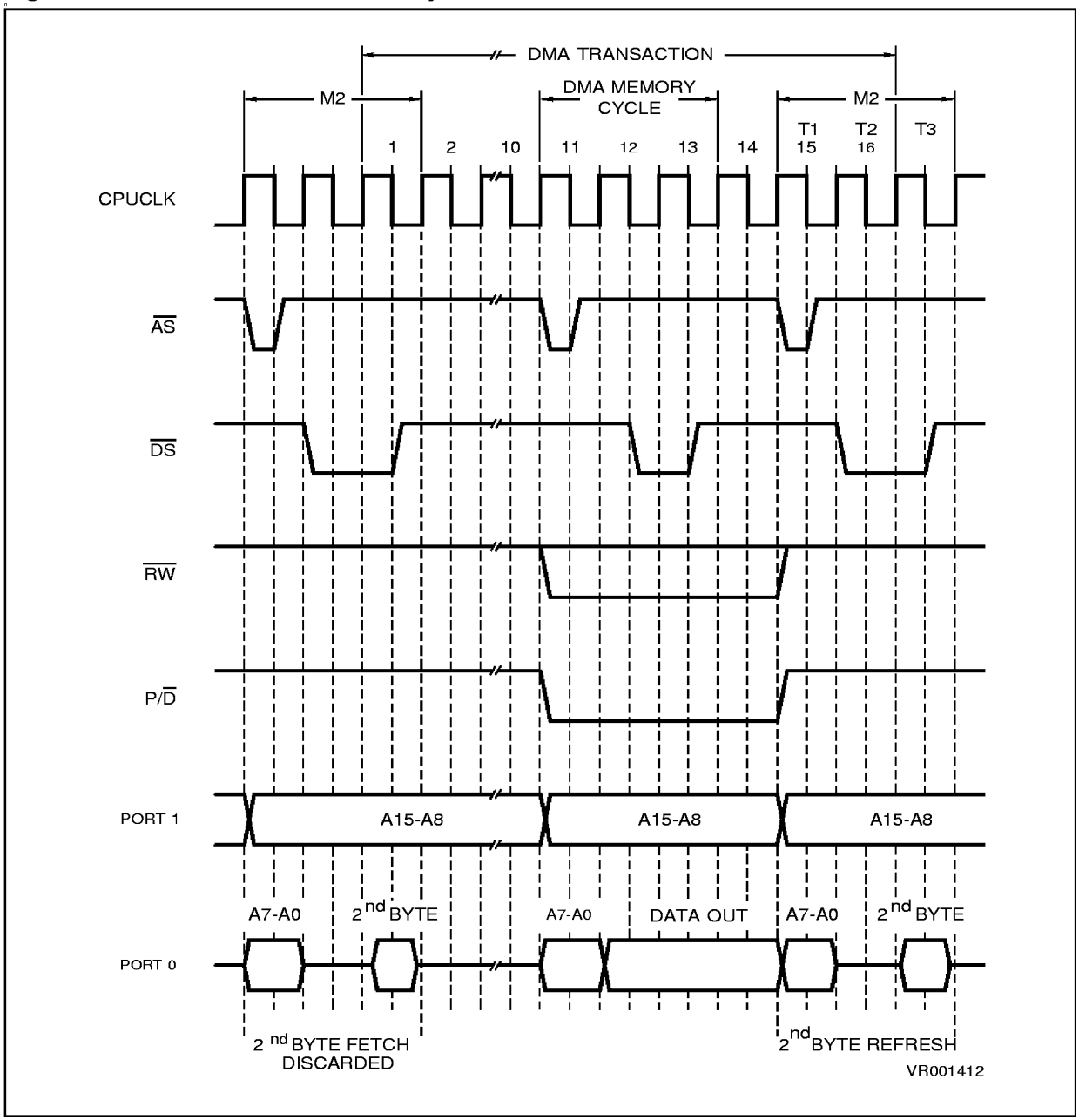
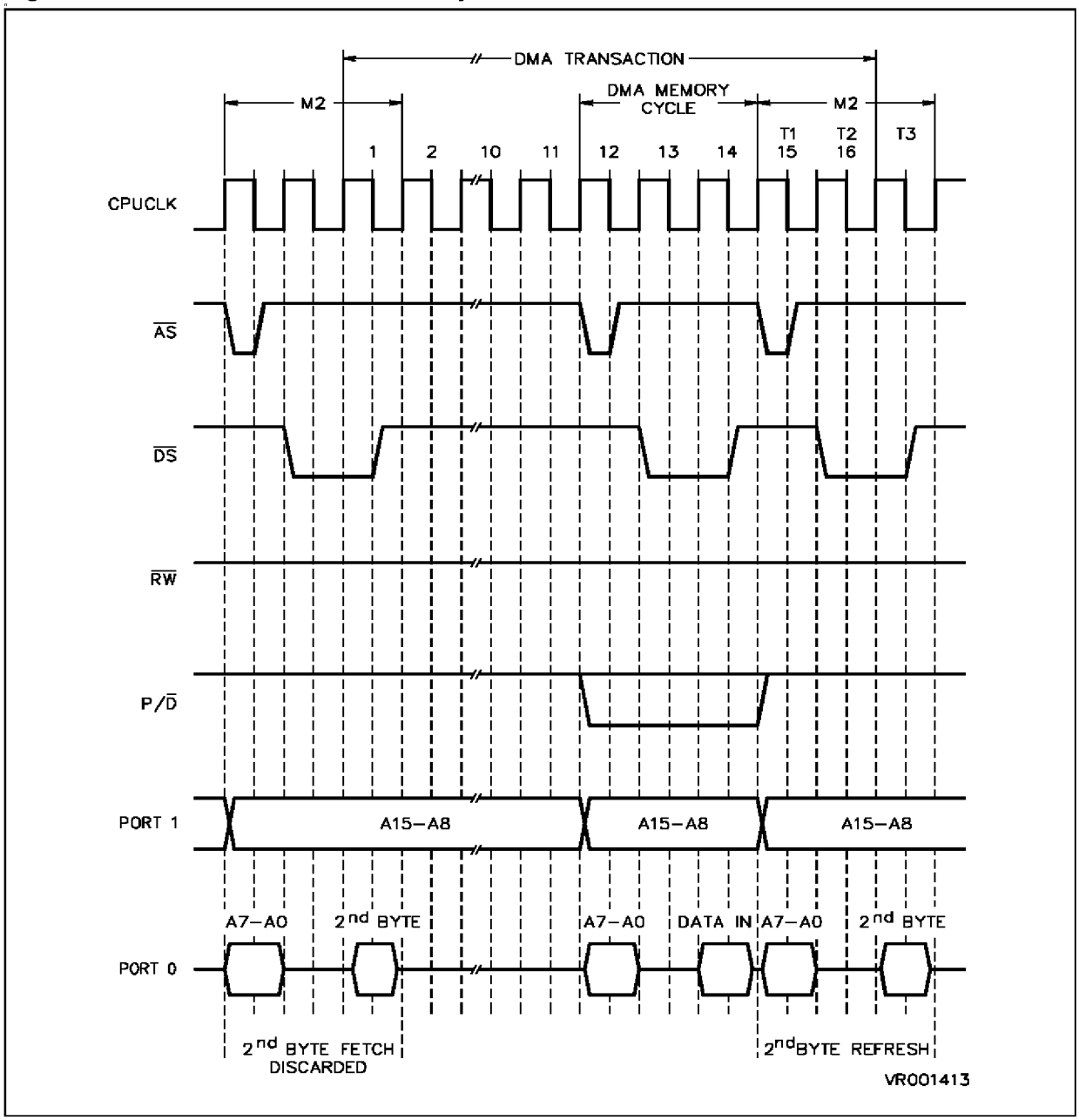


**Figure 29. DMA Transaction from Memory**



### 5.3 DMA TRANSACTIONS

The purpose of an on-chip DMA channel is to transfer a block of data between a peripheral and the 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)

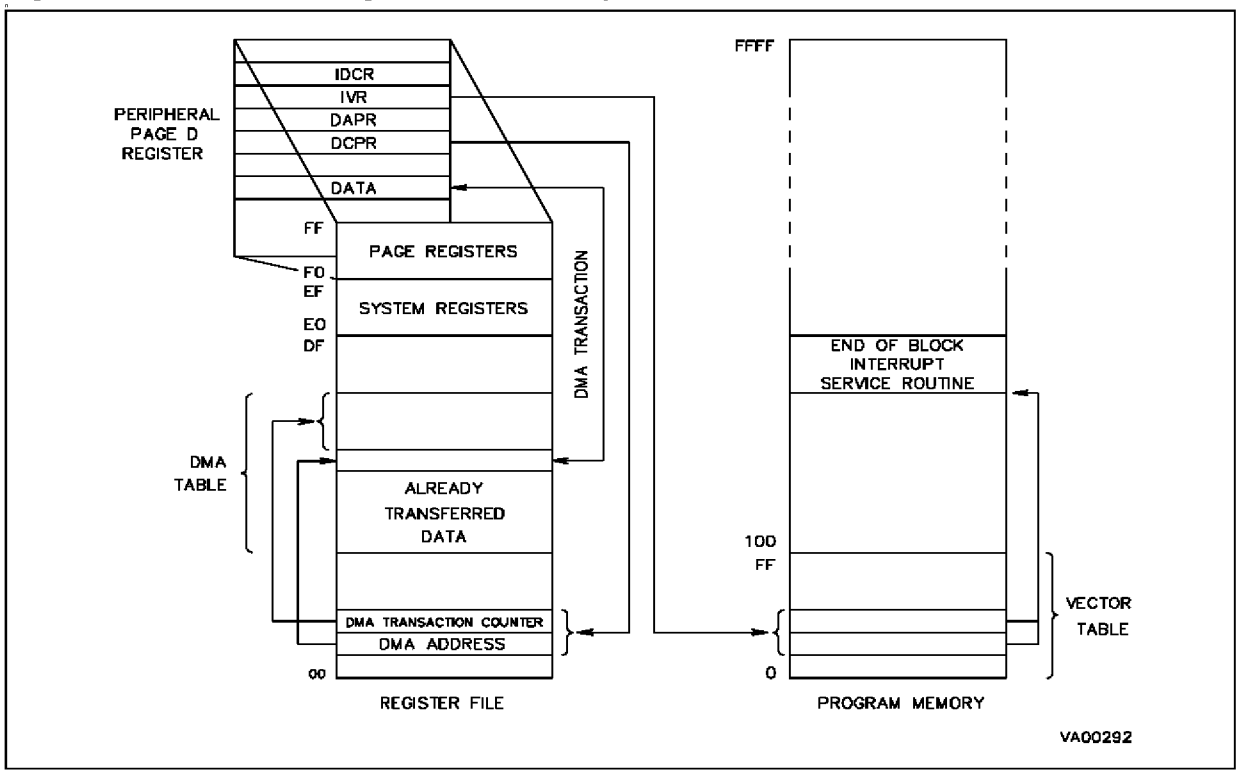
– A post-decrement of the DMA transaction counter, which contains the number of transactions that have still to be performed.

If the DMA transaction is carried out between **the peripheral and the Register File** (Figure 30), 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 address register, and the DMA Transaction Counter in the next register (odd address). They are pointed to by the DMA Transaction Counter Pointer Register (DCPR), located in the peripheral's paged registers. In order to select a 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 0DFh of the Register File.

If the transaction is made between **the peripheral and Memory**, a register pair (16 bits) is required for the DMA Address and for the DMA Transaction Counter (Figure 31). 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.

**Figure 30. DMA Between Register File and Peripheral**



**DMA TRANSACTIONS** (Cont'd)

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

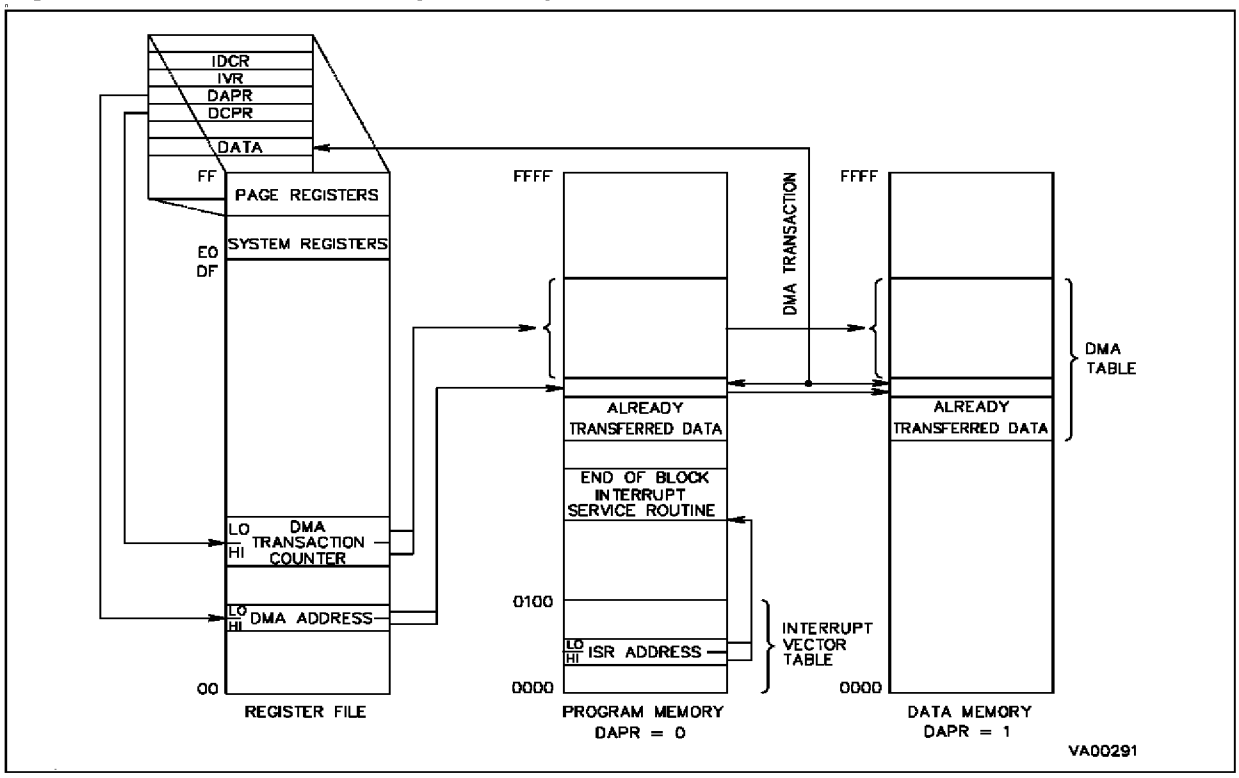
The DMA transaction Counter must be initialized with the number of transactions to perform and will be decremented after each transaction. The DMA Address must be initialized with the starting address of the DMA table and is increased after each transaction. These two register pairs must be located betweeen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

Once a DMA channel is initialized, a transfer can start. The direction of the transfer 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 Current Priority Level (CPL), the DMA transfer is executed at the end of the current instruction. DMA transfers read/write data from/to the location pointed to by the DMA Address Register, the DMA Address register is incremented and the Transaction Counter Register is decremented. When the contents of the Transaction Counter are 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.*

**Figure 31. DMA Between Memory and Peripheral**



**DMA TRANSACTIONS** (Cont'd)

### 5.4 DMA CYCLE TIME

The interrupt and DMA arbitration protocol functions completely asynchronously from instruction flow, and requires 6 CPUCLK cycles to resolve the request's priority.

Requests are sampled every 5 CPUCLK cycles.

DMA transactions are executed if their priority allows it.

A DMA transfer with the Register file requires 8 CPUCLK cycles.

A DMA transfer with memory requires 16 CPUCLK cycles, plus any required wait states.

### 5.5 THE SWAP MODE

An extra feature which may be found on the DMA channels of some peripherals (i.e the MultiFunction TIMER) is the Swap mode. This feature allows transfer 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 to the same space, Register File or Memory). The DMA transaction is programmed to start on one of the two tables (say table 0) and, at the end of the 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 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 specific control registers, the following register list should be considered as a general example. The names and register bit allocations 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 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**: *Memory Segment Selector.*
If set, DMA transactions are carried out in the segment pointed to by DMASR; if reset, DMA transactions are carried out in the segment pointed to by ISR.

# 6 RESET AND CLOCK CONTROL UNIT (RCCU)

## 6.1 INTRODUCTION

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

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

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

On ST9 devices where the external Stop pin is available, this circuit also detects and manages the externally triggered Stop mode, during which all oscillators are frozen in order to achieve the lowest possible power consumption.

## 6.2 CLOCK CONTROL UNIT

The Clock Control Unit generates the internal clocks for the CPU core (CPUCLK) and for the on-chip peripherals (INTCLK). The Clock Control Unit may be driven by an external crystal circuit, connected to the OSCIN and OSCOUT pins, or by an external pulse generator, connected to OSCIN (see Figure 38 and Figure 40). A low frequency external clock may be connected to the CK_AF pin, and this clock source may be selected when low power operation is required.

### 6.2.1 Clock Control Unit Overview

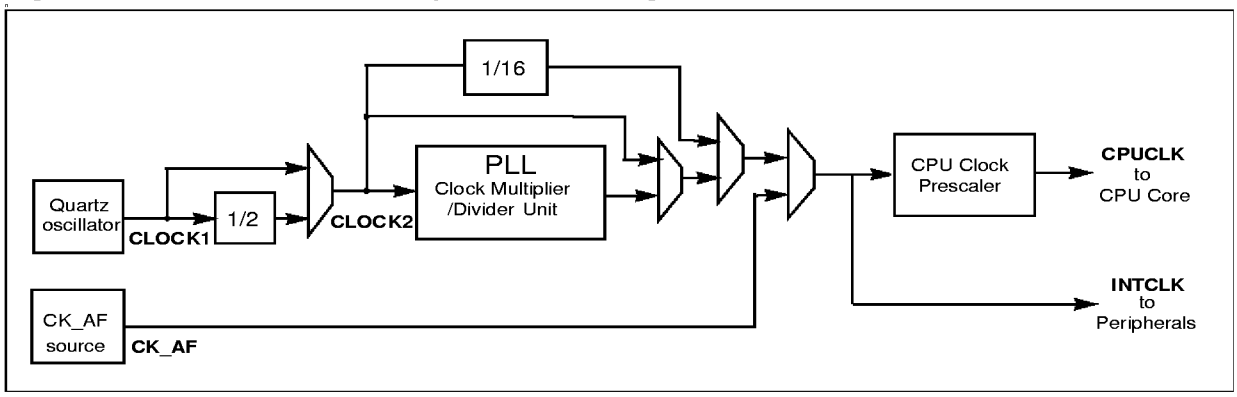
As shown in Figure 32, a programmable divider can divide the CLOCK1 input clock signal by two. In practice, the divide-by-two is virtually always used in order to ensure a 50% duty cycle signal to the PLL multiplier circuit. The resulting signal, CLOCK2, is the reference input clock to the programmable Phase Locked Loop frequency multiplier, which is capable of multiplying the clock frequency by a factor of 6, 8, 10 or 14; the multiplied clock is then divided by a programmable divider, by a factor of 1 to 7. By this means, the ST9 can operate with cheaper, medium frequency (3-5 MHz) crystals, while still providing a high frequency internal clock for maximum system performance; the range of available multiplication and division factors allow a great number of operating clock frequencies to be derived from a single crystal frequency.

For low power operation, especially in Wait for Interrupt mode, the Clock Multiplier unit may be turned off, whereupon the output clock signal may be programmed as CLOCK2 divided by 16. For further power reduction, a low frequency external clock connected to the CK_AF pin may be selected, whereupon the crystal controlled main oscillator may be turned off.

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

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

**Figure 32. Clock Control Unit Simplified Block Diagram**



### 6.3 CLOCK MANAGEMENT

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

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

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

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

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

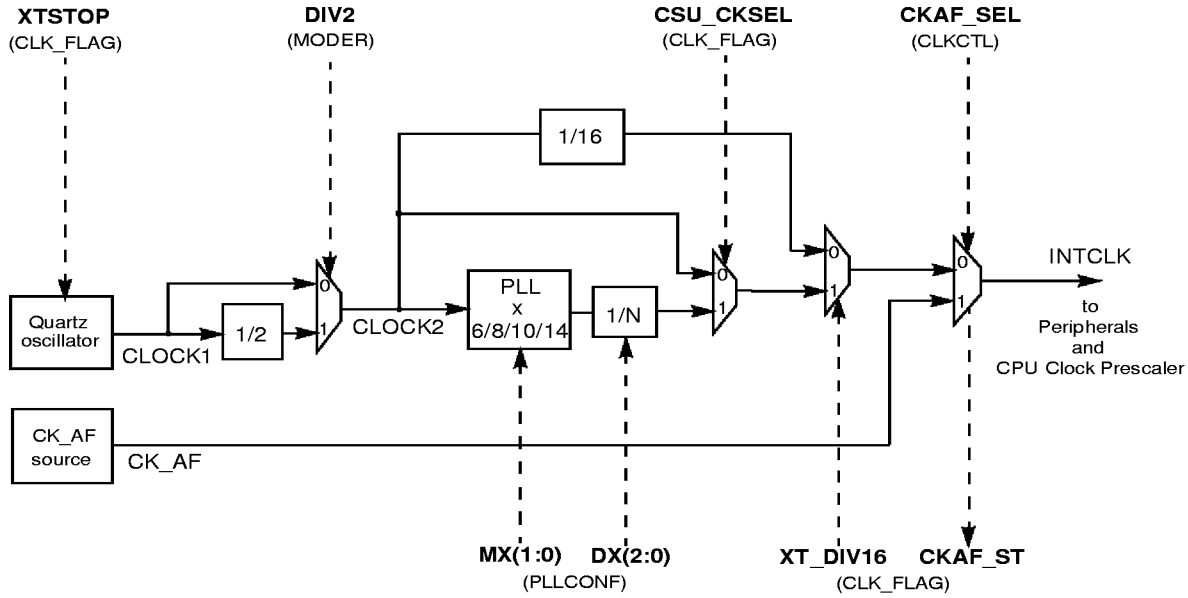
– **CLK_FLAG** (Clock Flag Register)
This is a Paged Register (R242, Page 55).

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

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

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

**Figure 33. Clock Control Unit Programming**




**Wait for Interrupt and Low Power Modes:**

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

**CLOCK MANAGEMENT** (Cont'd)

### 6.3.1 PLL Clock Multiplier Programming

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

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

Following a RESET phase, programming bits DX0-2 to a value different from 111 will turn the PLL on. After allowing a stabilisation period for the PLL, setting the CSU_CKSEL bit in the CLK_FLAG Register selects the multiplier clock.

The maximum frequency allowed for INTCLK is 16MHz for 5V operation, and 12MHz for 3V operation. Care is required, when programming the PLL multiplier and divider factors, not to exceed the maximum permissible operating frequency for INTCLK, according to supply voltage.

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

### 6.3.2 CPU Clock Prescaling

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

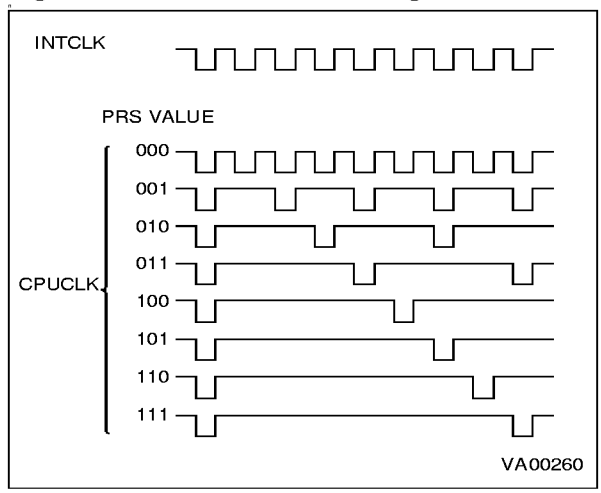
The internal peripherals are not affected by the CPUCLK prescaler and continue to operate at the full INTCLK frequency. This is particularly useful when little processing is being done and the peripherals are doing most of the work.

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

The clock generated is shown in Figure 34, and it will be noted that the prescaling of the clock does not preserve the 50% duty cycle, since the high level is stretched to replace the missing cycles.

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

**Figure 34. CPU Clock Prescaling**



**CLOCK MANAGEMENT** (Cont'd)

### 6.3.3 Peripheral Clock

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

### 6.3.4 Low Power Modes

The user can select an automatic slowdown of clock frequency during Wait for Interrupt operation, thus idling in low power mode while waiting for an interrupt. In WFI operation the clock to the CPU core is stopped, thus suspending program execution, while the clock to the peripherals may be programmed as described in the following paragraphs. Two examples of Low Power operation in WFI are illustrated in Figure 35 and Figure 36.

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

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

It is up to the user program to switch back to a faster clock on the occurrence of an interrupt, taking care to respect the oscillator and PLL stabilisation delays, as appropriate.

It should be noted that any of the low power modes may also be selected explicitly by the user program even when not in Wait for Interrupt mode, by setting the appropriate bits.

### 6.3.5 Interrupt Generation

System clock selection modifies the CLKCTL and CLK_FLAG registers.

The clock control unit generates an external interrupt request when CK_AF and CLOCK2/16 are selected or deselected as system clock source, as well as when the system clock restarts after a hardware stop (when the STOP MODE feature is available on the specific device). This interrupt can be masked by resetting the INT_SEL bit in the CLKCTL register. Note that this is the only case in the ST9 where an an interrupt is generated with a high to low transition.

**Table 13. Summary of Operating Modes using main Crystal Controlled Oscillator**

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

**Figure 35. Example of Low Power mode programming in WFI using CK_AF external clock**

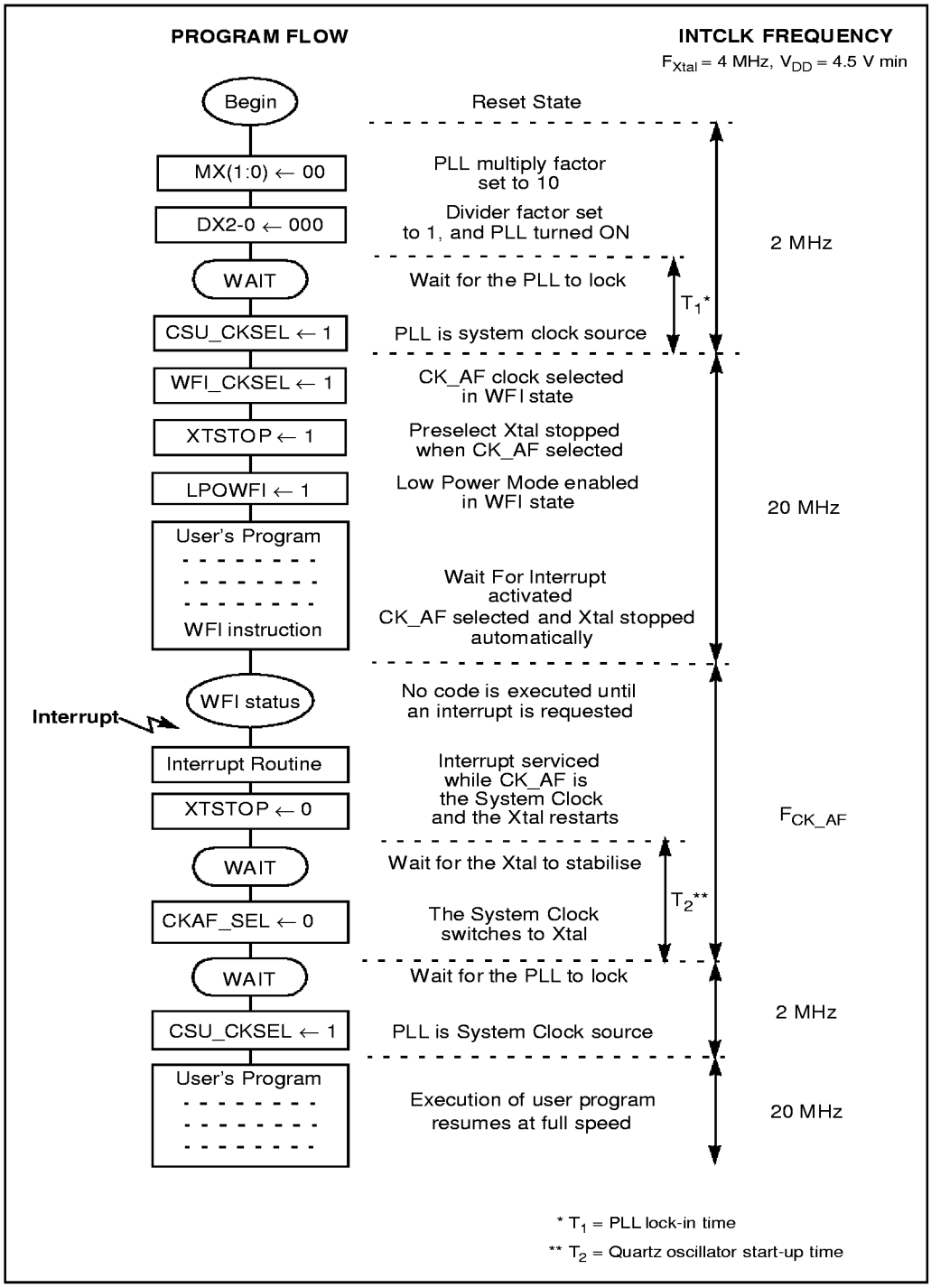
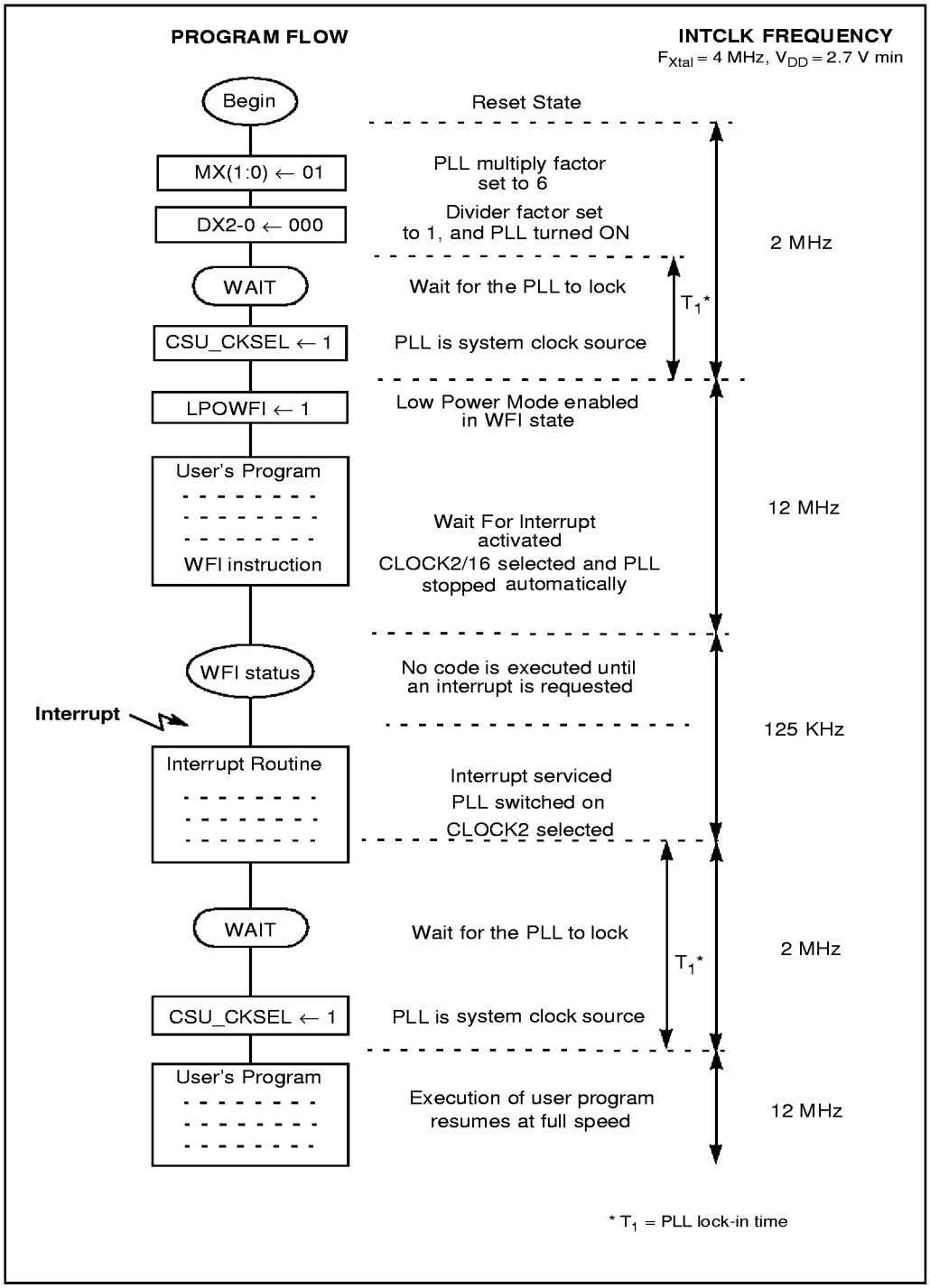


**Figure 36. Example of Low Power mode programming in WFI using CLOCK2/16**



### 6.4 CLOCK CONTROL REGISTERS

**MODER R235** (EBh) System Read/Write Mode Register

Reset Value: 1110 0000 (E0h)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| - | - | DIV2 | PRS2 | PRS1 | PRS0 | - | - |

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

b4-b2 = **PRS2, PRS1, PRS0**: *Clock Prescaling.* These bits define the prescaler value used to prescale CPUCLK from INTCLK. When these three bits are reset, the CPUCLK is not prescaled, and is equal to INTCLK; in all other cases, the internal clock is prescaled by the value of these three bits plus one.

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.

**CLKCTL R240** (0F0h) Page 55 Read Write Clock Control Register

Reset Value: 00000000b (00h)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| INT_SEL | 0 | 0 | 0 | SRESEN | CKAF_SEL | WFI_CKSEL | LPOWFI |

b0 = **LPOWFI**: *Low Power mode during Wait For Interrupt.* If set, the device enters Low Power mode when the WFI instruction is executed. The clock during this state depends on WFI_CKSEL.

b1 = **WFI_CKSEL**: *WFI Clock Select.* If reset the clock during WFI is CLOCK2/16; if set the clock during WFI will be CK_AF, providing it is present. In effect this bit sets CKAF_SEL in WFI mode.

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

b2 = **CKAF_SEL**: *Alternate Function Clock Select* If set, the CK_AF clock is selected. To check if the selection has actually occurred, check that CKAF_ST is set. Note that if no clock is present on the CK_AF pin, the selection will not occur.

b3 = **SRESEN**: If set, generates a Reset when HALT is executed. If reset, the HALT instruction turns off the quartz, the PLL and the CCU.

b4-b6 = **Reserved for test purposes**
Must be kept reset for normal operation.

b7 = **INT_SEL**: *Interrupt Selection.* If set, this bit selects the internal RCCU interrupt as the source of the interrupt request; if reset, the external interrupt channel input signal is carried through unchanged (Reset state).

## CLOCK CONTROL REGISTERS

**CLK_FLAG R242** (0F2h) Page 55 Read/Write

Clock Flag Register

Reset Value: 01001000b after a Watchdog Reset
Reset Value: 00101000b after a Software Reset
Reset Value: 00001000b after a Power-On Reset

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| EX_ STP | WDGRES | SOFTRES | XTSTOP | XT_ DIV16 | CKAF_ ST | - | CSU_ CKSEL |

b0 = **CSU_CKSEL**: When set, the PLL Multiplier provides the system clock. When reset, the system clock is provided by the CLOCK2. This bit is automatically reset when:

– bits DX2-0 (PLLCONF) are set to 111;

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

– WFI is executed while the LOPWFI bit is set;

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

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

b2 = **CKAF_ST**: (Read Only) If set, indicates that the alternate function clock has been selected. If no clock signal is present on the CK_AF pin, the selection will not occur. If reset, the PLL clock, CLOCK2 or CLOCK2/16 is selected (depending on bit 0).

b3 = **XT_DIV16**: If reset, CLOCK2/16 is selected and the PLL is off. If set, the input is CLOCK2. An interrupt is generated when the bit is toggled.

b4 = **XTSTOP**: When this bit is set, the Xtal oscillator will be stopped as soon as the CK_AF clock is present and selected, whether this is done explicitly by the user program, or as a result of WFI, WFI_CKSEL having previously been set to select the CK_AF clock during WFI.

b5 = **SOFTRES**: (Read Only) Set on software reset (HALT instruction).

**WARNING:** If you select the CK_AF as system clock and turn off the oscillator (bits R240.2 and R242.4 at 1), and then switch back to the XT clock by resetting the R240.2 bit, you must wait for the oscillator to restart correctly (12ms).

b6 = **WDGRES**: (Read Only) Set on WDG reset.

b7 = **EX_STP**: This bit is set to indicate that an externally generated Stop condition has occurred.

**WARNING**: if this register is accessed with a logical instruction, such as AND or OR, some bits may not be set as expected. For example, if the program sets the XTSTOP bit, but the CKAF_ST is still reset, XTSTOP will remain reset. A subsequent AND with '1' or an OR with '0' to this bit will reset the bit and the oscillator will not be stopped.

**PLLCONF R246** (0F6h) Page 55 Read/Write

PLL Configuration Register

Reset Value: xx00x111b

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| - | - | MX1 | MX0 | - | DX2 | DX1 | DX0 |

b5-4 = **MX(1-0)**: *PLL Multiplication Factor.* Refer to Figure 14 for multiplier settings.

b2-0 = **DX(2-0)**: *PLL output clock divider factor.* Refer to Figure 15 for divider settings.

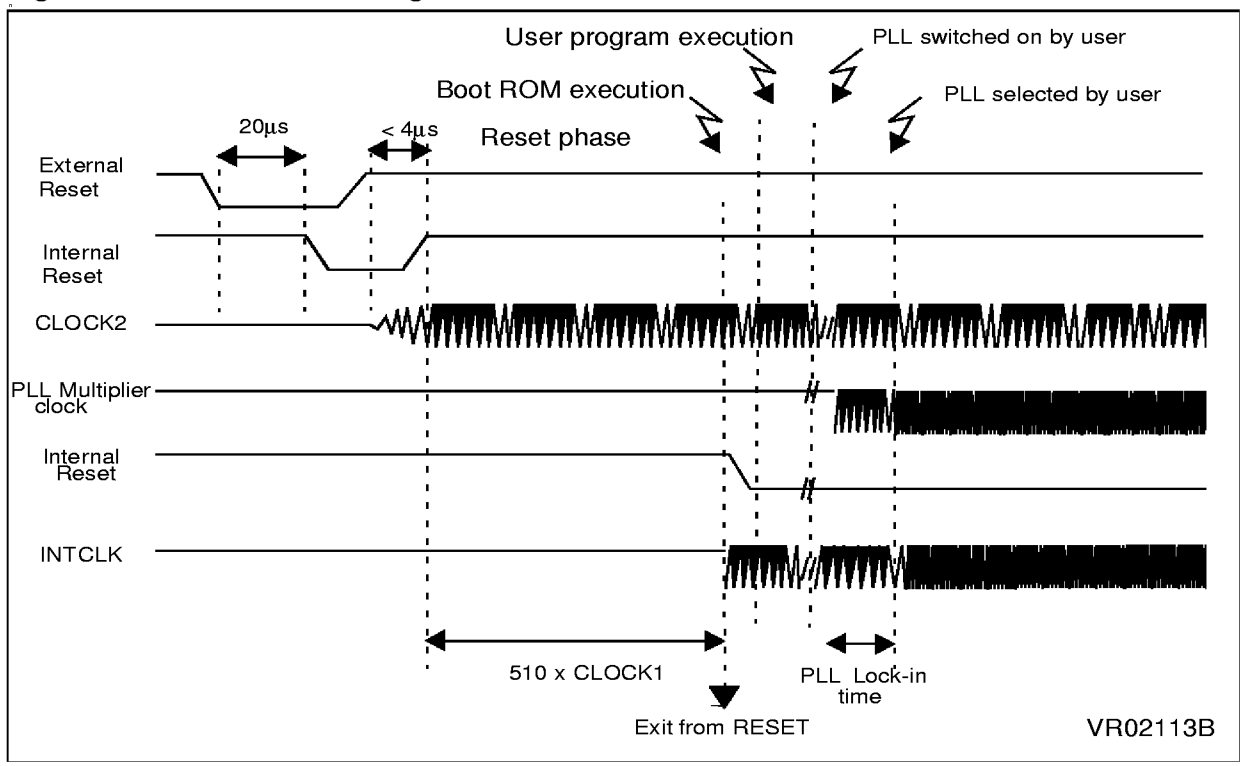
**Table 14. PLL Multiplication Factors**

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

**Table 15. Divider Configuration**

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

**Figure 37. RCCU General Timing**



### 6.5 OSCILLATOR CHARACTERISTICS

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

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

OSCOUT must not be directly used to drive external circuits.

When the oscillator is stopped, OSCOUT goes high impedance.

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 impedance state.

To exit the HALT condition and restart the oscillator, an external RESET pulse is required, having a a minimum duration of 12ms, as illustrated inFigure 41

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

**Table 16. Oscillator Transconductance**

| gm | Min | Typ | Max |
|---|---|---|---|
| mA/V | 0.77 | 1.5 | 2.4 |

**Figure 38. Crystal Oscillator**

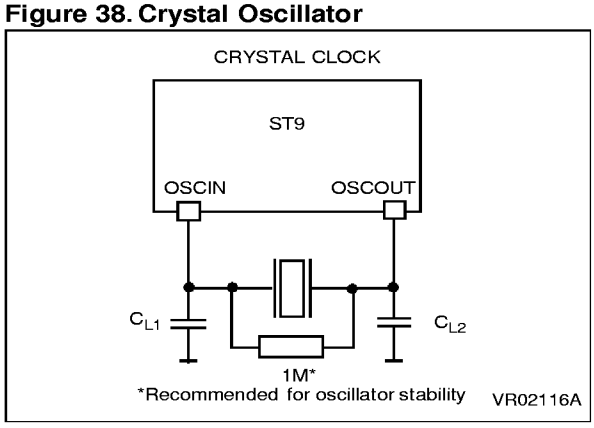



**Table 17. Crystal Specification**

| C1=C2 Freq. | 56p | 47p | 33p | 22p |
|---|---|---|---|---|
| 5 Mhz | 110 | 120 | 210 | 340 |
| 4 Mhz | 150 | 200 | 330 | 510 |
| 3 Mhz | 270 | 350 | 560 | 850 |

**Legend**:
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)

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

**Figure 39. Internal Oscillator Schematic**

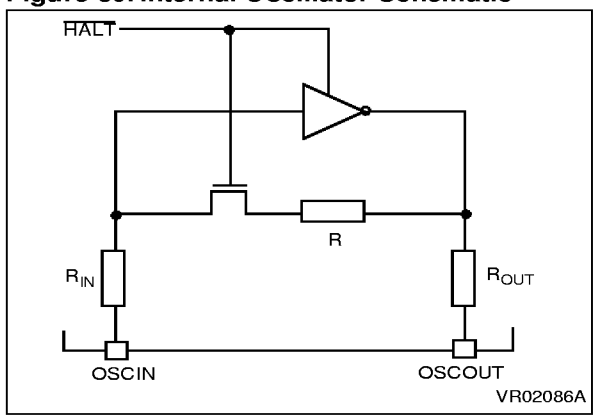
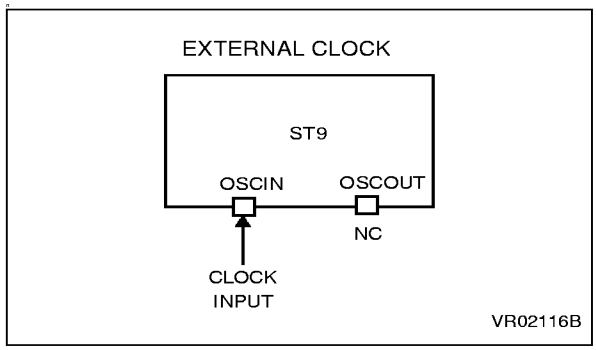



**Figure 40. External Clock**



### 6.6 RESET/STOP MANAGER

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

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

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

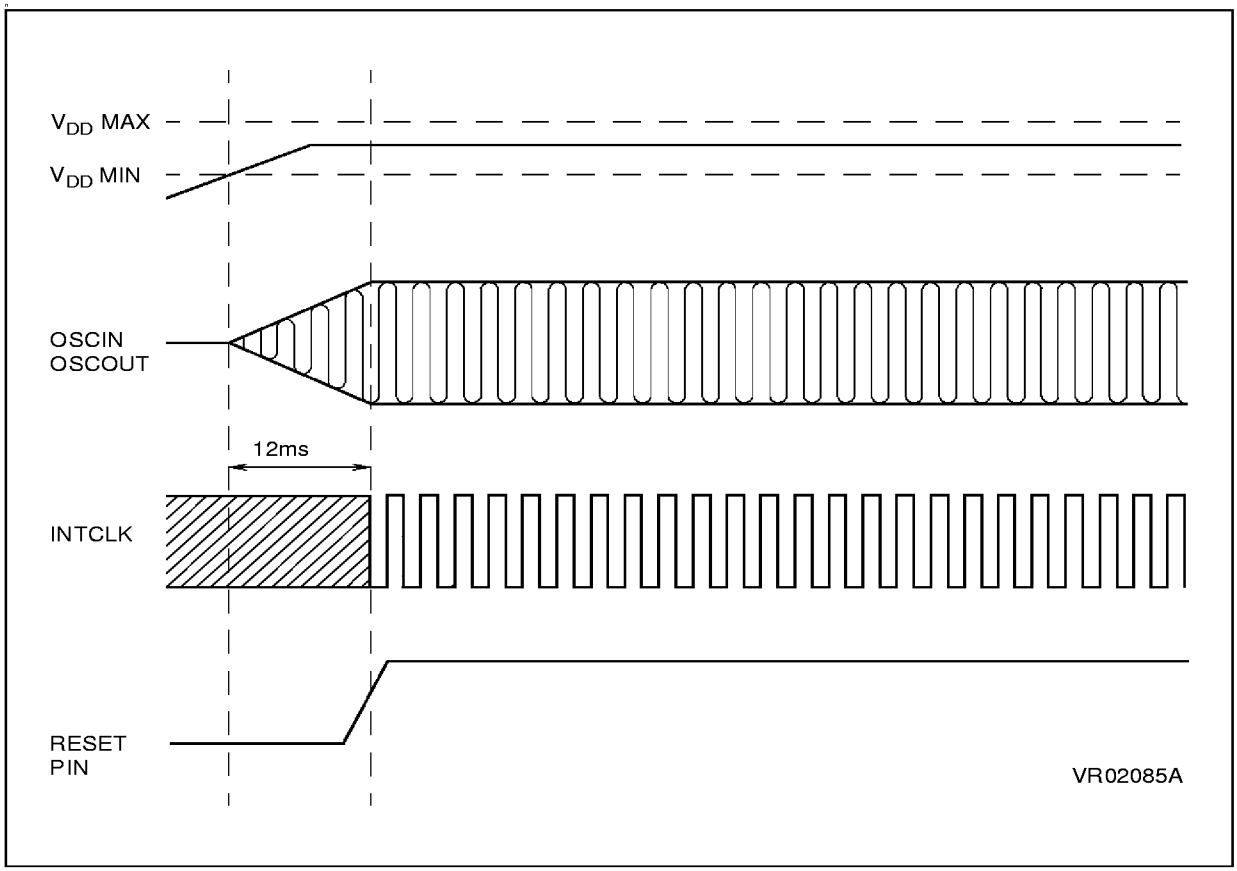
– A Watchdog end of count condition.

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

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

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

**Figure 41. Oscillator Start-up Sequence and Reset Timing**



**RESET/STOP MANAGER** (Cont'd)

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

When the Reset pin goes high again, 510 oscillator clock cycles (CLOCK1) are counted before exiting the Reset state (+-1 CLOCK1 period, depending on the delay between the rising edge of the Reset pin and the first rising edge of CLOCK1). Subsequently a short Boot routine is executed from the device internal Boot ROM, and control then passes to the user program.

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

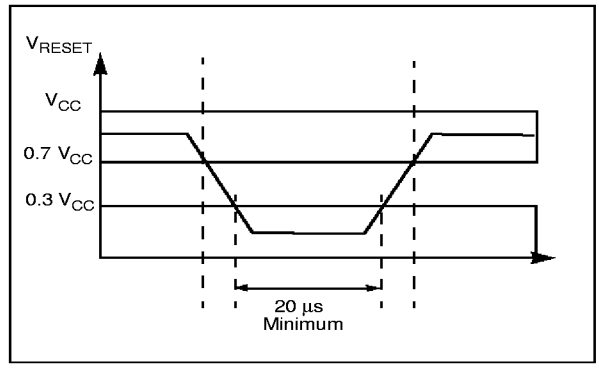
At the end of the Boot routine the Program Counter will be set to the location specified in the Reset Vector located in the lowest two bytes of memory.

### 6.6.1 Reset Pin Timing

To improve the noise immunity of the device, the Reset pin has a Schmitt trigger input circuit with hysteresis. In addition, a filter will prevent an unwanted reset in case of a single glitch of less than 50 ns on the Reset pin. The device is certain to reset if a negative pulse of more than 20ns is applied. When the reset pin goes high again, a delay of up to 4µs will elapse before the RCCU detects this rising front. From this event on, 510 oscillator clock cycles (CLOCK1) are counted before exiting the Reset state (+-1CLOCK1 period depending on the delay between the positive edge the RCCU detects and the first rising edge of CLOCK1)

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

**Figure 42. Recommended Signal to be Applied on Reset Pin**



### 6.7 EXTERNAL STOP MODE

On ST9 devices provided with an external Stop pin, the Reset/Stop Manager can also stop all oscillators without resetting the device.

To enter Stop Mode, the STOP pin must be forced to “0” for a minimum of 4 system clock cycles; while the Stop pin is kept at “0”, the MCU will remain in Stop Mode and all context information will be preserved. During this condition the internal clock will be frozen in the high state.

When the pin is forced back to “1”, the MCU resumes execution of the user program after a delay of 255 CLOCK2 periods.

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

**Table 18. Internal Registers Reset Values**

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

# 7 EXTERNAL MEMORY INTERFACE (EXTMI)

## 7.1 INTRODUCTION

The ST9 External Memory Interface uses two registers (EMR1 and EMR2 mapped in group F, Page 21 of the Register File) to configure external memory accesses. Some interface signals are also affected by WCR - R252 Page 0.

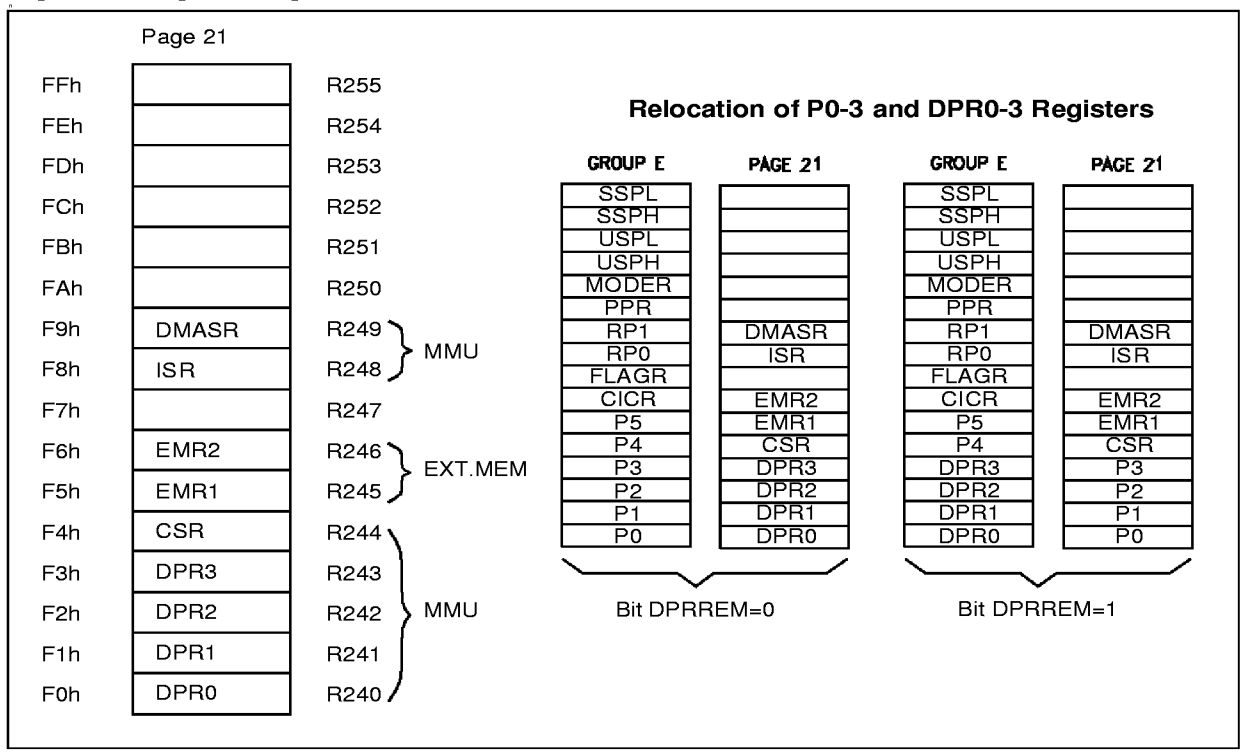
If the two registers EMR1 and EMR2 are set to the proper values, the new ST9 memory access cycle is similar to that of the original ST9, with the only exception that it is composed of just two system clock phases, named T1 and T2.

During phase T1 the memory address is output on the ASN falling edge and is valid on the rising edge of ASN. Port1, Port2, Port6 and PDN maintain the address stable until the following T1 phase.

During phase T2, two forms of behavior are possible. If the memory access is a Read cycle, Port 0 pins are released in high-impedance until the next T1 phase and the data signals are sampled by the ST9 on the rising edge of DSN. If the memory access is a Write cycle, on the falling edge of DSN, Port 0 outputs data to be written in the external memory. Those data signals are valid on the rising edge of DSN and are maintained stable until the next address is output. Note that DSN is pulled low at the beginning of phase T2 only during an external memory access.

The next chapter describes the meaning and the behavior of the external memory signals and the chapter after summarizes the meaning of the bits of the two registers EMR1 and EMR2.

**Figure 43. Page 21 Registers**



## 7.2 EXTERNAL MEMORY SIGNALS

The access to external memory is made using the following signals:

### 7.2.1 ASN: Address Strobe

ASN (Output, Active low, Tristate) is active during the System Clock high-level phase of each T1 memory cycle: an ASN rising edge indicates that Memory Address, Read/Write and Program/Data Memory control signals are valid. ASN is released in high-impedance during the bus acknowledge cycle or under the processor control by setting the HIMP bit (MODER.0, R235). Depending on the device ASN is available as Alternate Function or as a dedicated pin.

Under Reset, ASN is held high with an internal weak pull-up.

The behavior of this signal is affected by the following external memory dedicated register bits:

#### 7.2.1.1 Bit MC of register EMR1 - bit 6 of register R245, Page 21.

If bit MC is reset, ASN keeps the ST9OLD meaning: an ASN rising edge indicates that Memory Address, Read/Write and Program/Data Memory signals are valid.

If bit MC is set, the ASN pin becomes ALE (Address Load Enable) which is ASN inverted. Thus Memory Address, Read/Write and Program/Data Memory signals are valid whenever a falling edge of ALE occurs.

#### 7.2.1.2 Bit BSZ of register EMR1 - bit 1 of register R245, Page 21.

If bit BSZ is set, ASN uses larger, more noisy output buffers. This may limit the operation frequency of the device, unless the clock is slow enough or sufficient wait states are inserted.

#### 7.2.1.3 Bit ASAF of register EMR1 - bit 4 of register R245, Page 21.

Setting bit ASAF enables the ASN alternate function.

#### 7.2.1.4 Bit ETO of register EMR1 - bit 2 of register R245, Page 21.

If bit ETO is reset or internal memory protection enabled, the ASN pin toggles only if an external memory access is performed.

If bit ETO is set and internal memory protection disabled, the ASN pin toggles during both internal and external memory accesses.

#### 7.2.1.5 Bits PAS1-PAS0 of register EMR2 - bits 3 and 2 of register R246, Page 21.

These two bits contain the number of wait cycles (from 0 to 3) to add to the System Clock in order to stretch ASN during either external Program memory accesses (MEMSEL="0", PDN="1") or external ROM accesses (MEMSEL="1", A21="0").

#### 7.2.1.6 Bits DAS1-DAS0 of register EMR2 - bit 1 and 0 of register R246, Page 21.

These two bits contain the number of wait cycles (from 0 to 3) to add to the System Clock in order to stretch ASN during either external Data memory accesses (MEMSEL="0", PDN="0") or external RAM accesses (MEMSEL="1", A21="1").

### 7.2.2 DSN: Data Strobe

DSN (Output, Active low, Tristate) is active during the internal clock high-level phase of each T2 memory cycle. During an external memory read cycle, the data on Port 0 must be valid before the DSN rising edge. During an external memory write cycle, the data on Port 0 are output on the falling edge of DSN and they are valid on the rising edge of DSN. When the internal memory is accessed DSN is kept high during the whole memory cycle. DSN is released in high-impedance during bus acknowledge cycle or under processor control by setting the HIMP bit (MODER.0, R235). Under Reset status, DSN is held high with an internal weak pull-up.

The behavior of this signal is affected by the following external memory dedicated register bits:

**EXTERNAL MEMORY SIGNALS**(Cont'd)

**7.2.2.1 Bit MC of register EMR1 - bit 6 of register R245, Page 21.**

If bit MC is reset, DSN keeps the ST9OLD meaning: a rising edge of DSN indicates that data on PORT0 are valid to be written/read in/from the external memory. When the internal memory is accessed, DSN is kept high during the whole memory cycle.

If bit MC is set, DSN becomes OEN (Output ENable): it keeps the ST9OLD meaning during external read operations but is forced to "1" during external write operations.

**7.2.2.2 Bit DS2EN of register EMR1 - bit 5 of register R245, Page 21.**

If bit DS2EN is reset, the behavior of DSN keeps the behavior described in the previous paragraph and depends on the value of the MC bit.

If bit DS2EN is set, the behavior of DSN depends on which kind of memory is currently used (RAM or ROM, Data or Program). This is defined by the MEMSEL bit. If a ROM/Program memory is used, DSN is forced to "1" during the whole memory cycle. If a RAM/Data memory is used, the behavior of DSN is described in previous paragraphs.

**7.2.2.3 Bit BSZ of register EMR1 - bit 1 of register R245, Page 21.**

If bit BSZ is set, DSN uses larger, more noisy output buffers.

**7.2.3 DS2N: Data Strobe 2**

This additional Data Strobe pin (Alternate Function Output, Active low, Tristate) allows to connect to the micro two external memories (a RAM/Data and a ROM/Program) without any external logic. The selection between RAM/Data and ROM/Program memories depends on the MEMSEL bit value.

The RAM/Data memory is controlled by the DSN pin while the ROM/Program is controlled by the DS2N pin. When the internal memory is addressed DS2N is kept high during the whole memory cycle. DS2N is released in high-impedance during bus acknowledge cycle or under processor control by setting the HIMP bit (MODER.0, R235). DS2N is enabled via software as the Alternate Function output of the associated I/O port bit (refer to specific ST9 version to identify the specific port and pin). Under Reset status, the associated bit of the port is set to the bidirectional weak pull-up mode.

The behavior of this signal is affected by the following external memory dedicated register bits:

**7.2.3.1 Bit DS2EN of register EMR1 - bit 5 of register R245, Page 21.**

If bit DS2EN is reset, DS2N is kept high during the whole memory cycle.

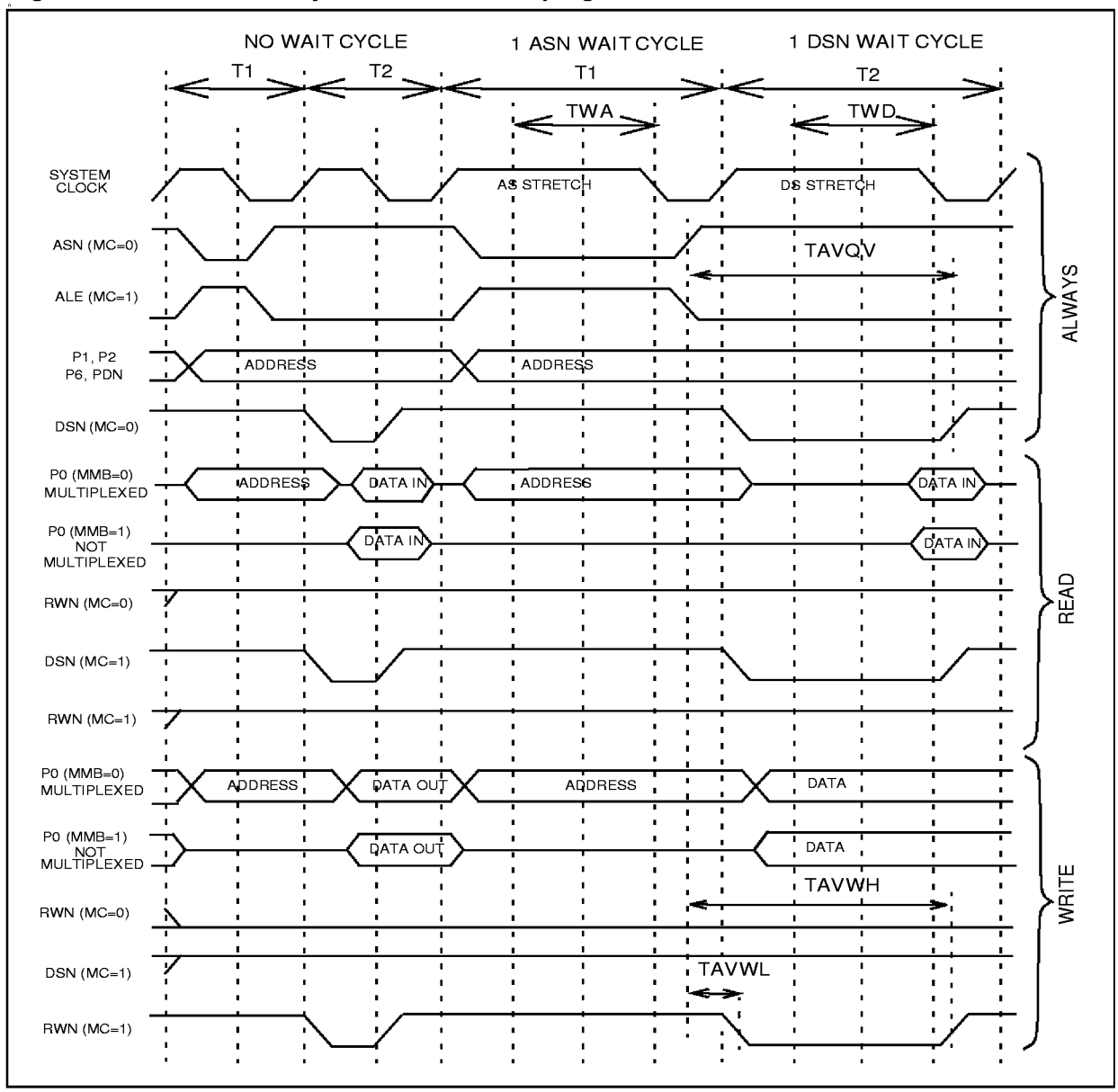
If bit DS2EN is set, the behavior of DS2N depends on which kind of memory is currently used (RAM or ROM, Data or Program), this is defined by the MEMSEL bit. If a RAM/Data memory is used, DS2N is forced to "1" during the whole memory cycle. If a ROM/Program memory is used, DS2N has the same behavior of DSN.

**7.2.3.2 Bit BSZ of register EMR1 - bit 1 of register R245, Page 21.**

If bit BSZ is set, DS2N uses larger, more noisy output buffers.

**EXTERNAL MEMORY SIGNALS**(Cont'd)

**Figure 44. External memory Read/Write with a programmable wait**



**EXTERNAL MEMORY SIGNALS**(Cont'd)

### 7.2.4 RWN: Read/Write

RWN (Alternate Function Output, Active low, Tristate) identifies the type of memory cycle: RWN="1" identifies a memory read cycle, RWN="0" identifies a memory write cycle. It is defined at the beginning of each memory cycle and it remains stable until the following memory cycle. RWN is released in high-impedance during bus acknowledge cycle or under processor control by setting the HIMP bit (MODER). RWN is enabled via software as the Alternate Function output of the associated I/O port bit (refer to specific ST9 device to identify the port and pin). Under Reset status, the associated bit of the port is set into bidirectional weak pull-up mode.

The behavior of this signal is affected by the following external memory dedicated register bits:

#### 7.2.4.1 Bit MC of register EMR1 - bit 6 of register R245, Page 21.

If bit MC is reset, RWN keeps the before mentioned meaning: RWN="1" indicates a reading memory cycle, RWN="0" indicates a writing memory cycle.

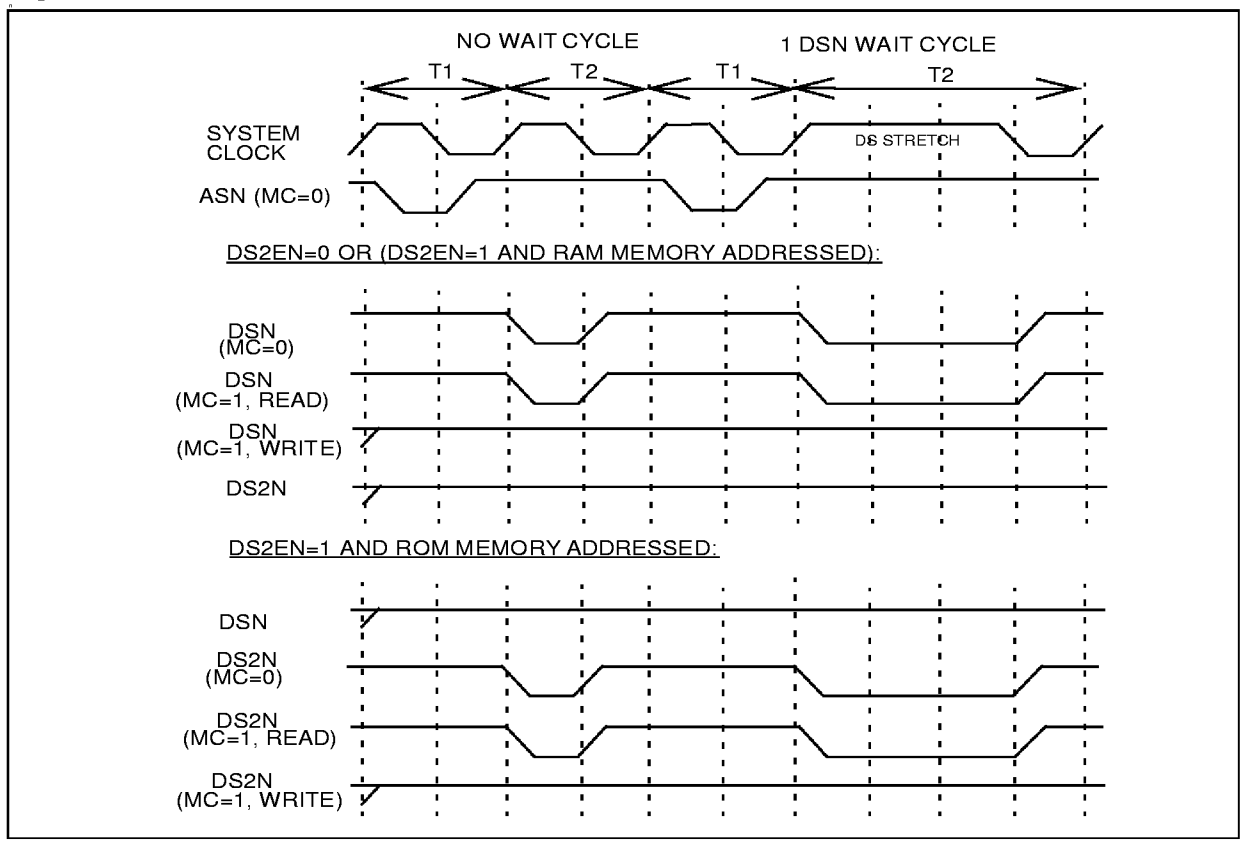
If bit MC is set, RWN becomes WEN (Write ENable): it is forced to "1" during external read operations but follows the original ST9 DSN behavior during external write operations.

#### 7.2.4.2 Bit ETO of register EMR1 - bit 2 of register R245, Page 21.

If bit ETO is reset or internal memory protection enabled, the RWN pin toggles only if an external memory access is performed.

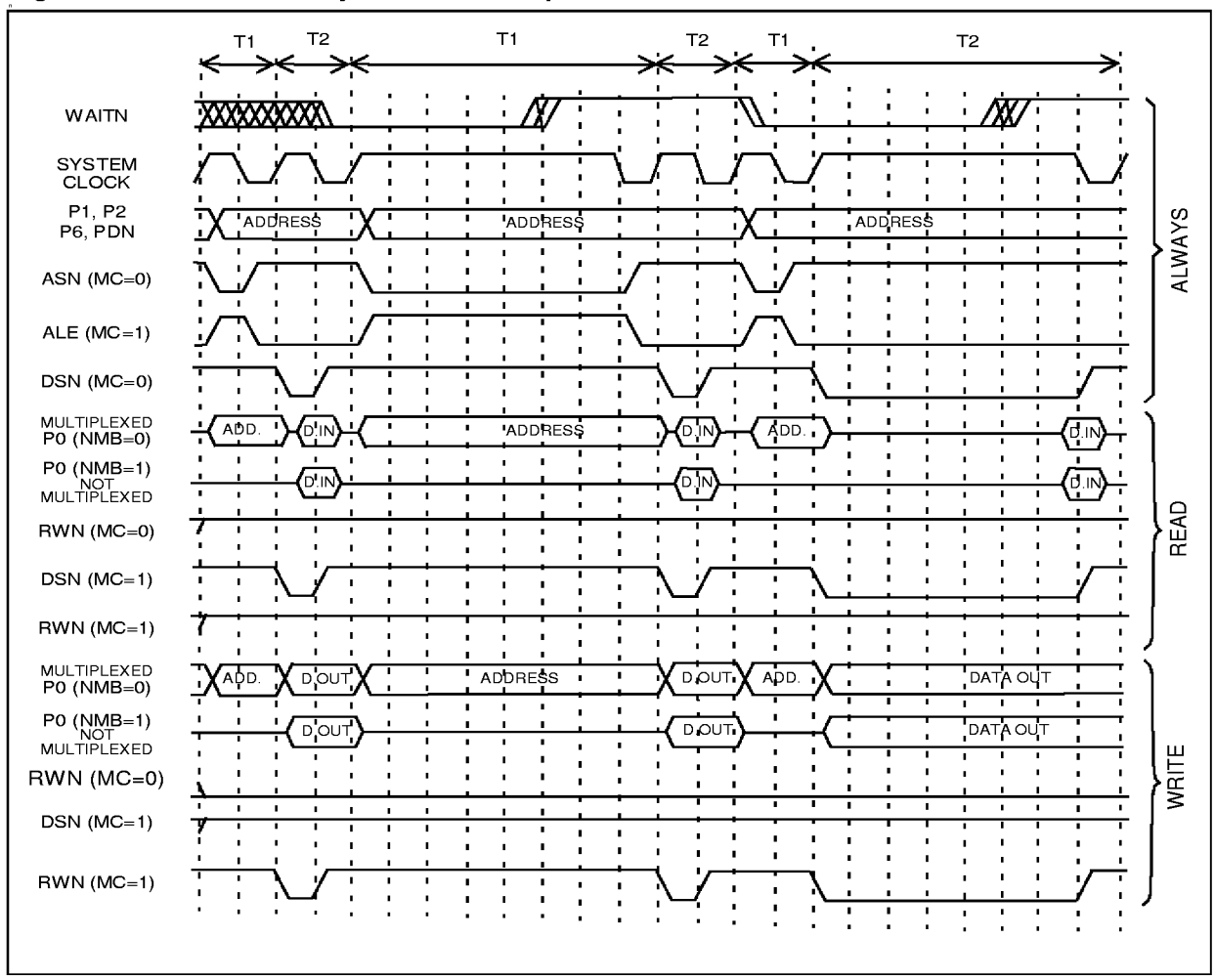
If bit ETO is set and internal memory protection disabled, the RWN pin toggles during both internal and external memory accesses.

**Figure 45. Effects of DS2EN on the behavior of DSN and DS2N**



**EXTERNAL MEMORY SIGNALS**(Cont'd)

**Figure 46. External memory Read/Write sequence with external wait**



**EXTERNAL MEMORY SIGNALS**(Cont'd)

**7.2.4.3 Bit BSZ of register EMR1 - bit 1 of register R245, Page 21.**

If bit BSZ is set, RWN uses larger, more noisy output buffers.

The behavior of this signal is affected by the following external memory dedicated register bits:

**7.2.5 WAITN: External Memory Wait**

WAITN (Alternate Function Input, Active low) indicates to the ST9 that the external memory requires more time to complete the memory access cycle. If bit EWEN (EIVR) is set, the WAITN signal is sampled with the rising edge of the processor internal clock during phase T1 or T2 of every memory cycle. If the signal was sampled active, one more internal clock cycle is added to the memory cycle. On the rising edge of the added internal clock cycle, WAITN is sampled again to continue or finish the memory cycle stretching. Note that if WAITN is sampled active during phase T1 then ASN is stretched, while if WAITN is sampled active during phase T2 then DSN is stretched. WAITN is enabled via software as the Alternate Function input of the associated I/O port bit (refer to specific ST9 version to identify the specific port and pin). Under Reset status, the associated bit of the port is set to the bidirectional weak pull-up mode.

**7.2.6 PORT 0**

If Port 0 (Input/Output, Push-Pull/Open-Drain/ Weak Pull-up) is used as a bit programmable parallel I/O port, it has the same features as a regular port. When set as an Alternate Function, it is used as the External Memory interface: it outputs the multiplexed Address (8 LSB: A7-A0) / Data bus (D7-D0). The behavior of Port 0 pins is affected by the following external memory dedicated register bits:

**7.2.6.1 Bit NMB of register EMR1 - bit 3 of register R245, Page 21.**

If Port 6 is available and bit NMB is set, Port 0 is released in high-impedance whenever it should have output addresses. This is meant to reduce power consumption and EMI.

**7.2.6.2 Bit ETO of register EMR1 - bit 2 of register R245, Page 21.**

If bit ETO is reset or internal memory protection enabled, the Port 0 pins toggle only if an external memory access is performed.

If bit ETO is set and internal memory protection disabled, the Port 0 pins toggle during both internal and external memory accesses.

**7.2.6.3 Bit BSZ of register EMR1 - bit 1 of register R245, Page 21.**

If BSZ is set, the Port 0 pins use larger, more noisy output buffers.

**7.2.6.4 Bit ROMLESS of register EMR1 - bit 0 of register R245, Page 21.**

If ROMLESS is set, the internal ROM is disabled and the external memory is read instead of it.

**7.2.7 PORT 1**

If Port 1 (Input/Output, Push-Pull/Open-Drain/ Weak Pull-up) is used as a bit programmable parallel I/O port, it has the same features as a regular port. When set as an Alternate Function, it is used as the external memory interface to provide the 8 MSB of the address (A15-A8).

The behavior of Port 1 pins is affected by the following external memory dedicated register bits:

**7.2.7.1 Bit ETO of register EMR1 - bit 2 of register R245, Page 21.**

If bit ETO is reset or internal memory protection enabled, the Port 1 pins toggle only if an external memory access is performed.

If bit ETO is set and internal memory protection disabled, the Port 1 pins toggle during both internal and external memory accesses.

**7.2.7.2 Bit BSZ of register EMR1 - bit 1 of register R245, Page 21.**

If BSZ is set, the Port1 pins use larger, more noisy output buffers.

**7.2.7.3 Bit ROMLESS of register EMR1 - bit 0 of register R245, Page 21.**

If ROMLESS is set, the internal ROM is disabled and the external memory is read instead of it.

### 7.3 EXTERNAL MEMORY INTERFACE REGISTERS

Two registers, mapped inside group F, Page 21 of the Register File, allow the user to configure external memory accesses as needed.

#### 7.3.1 EMR1: EXTERNAL MEMORY REGISTER 1

**EMR1 R245** (F5h) Page 21 R/W

External Memory Register 1

Reset value: 1000 0000b (80h)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| X | MC | DS2EN | ASAF | NMB | ETO | BSZ | ROMLESS |

b7 = Reserved.

b6 = **MC**: *Mode Control.* If MC is reset, ASN, DSN and RWN pins keep the ST9OLD meaning. If MC is set, then:

– ASN pin becomes ALE, Address Load Enable (ASN inverted);

– DSN becomes OEN, Output ENable: it keeps the ST9OLD meaning during external read operations, but is forced to "1" during external write operations;

– RWN pin becomes WEN, Write ENable: it follows the ST9OLD DSN meaning during external write operations, but is forced to "1" during external read operations.

b5 = **DS2EN**: *Data Strobe 2 ENable.* If DS2EN is reset, then:

– DS2N pin is forced to "1".

– If bit MC is reset, DSN pin keeps the ST9OLD behavior; if bit MC is set, DSN becomes OEN.

If DS2EN is set the behavior of DSN and DS2N pins depends on the kind of memory (RAM or ROM, Data or Program) that is currently referenced (this is defined by the bit MEMSEL of the EMR2 register):

– ROM/Program memory: DSN pin is equal to "1" during the whole memory cycle. DS2N pin follows the ST9OLD DSN meaning (if MC= "0") or it becomes OEN (if MC= "1").

– RAM/Data memory: DS2N pin is equal to "1" during the whole memory cycle. DSN pin follows the ST9OLD DSN meaning (if MC= "0") or it becomes OEN (if MC= "1").

b4 = **ASAF**: *Address Strobe as Alternate Function.* Depending on the device, ASN can be either a dedicated pin or a port Alternate Function. In this last case, when ASAF is set this Alternate Function is enabled.

b3 = **NMB**: *No Multiplexed Bus.* If NMB is reset, Port 0, when configured as an Alternate Function, outputs the 8 LSB of address multiplexed with the 8 bits of data. If NMB is set, Port 0 outputs data only: it is released in high-impedance whenever it should have output addresses.

b2 = **ETO**: *External TOggle.* If ETO is reset or the memory protection enabled, the external memory interface pins (ASN, DSN, DS2N, RWN, PDN, Port0, Port1, Port2, Port6) toggle only if an access to external memory is performed. If ETO is set and the memory protection is disabled, the above pins (except DSN and DS2N which never toggle during internal memory accesses) toggle during both internal and external memory accesses.

b1 = **BSZ**: *Bus SiZe.* If BSZ is set, all the I/O ports use larger, more noisy output buffers.

b0 = Reserved.

***WARNING**: External memory must be correctly addressed before and after a write operation on the EMR1 register. For example, if code is fetched from external memory using the ST9OLD external memory interface configuration (MC=0), a setting to 1 of MC bit will produce an unpredictable behaviour of the device.*

**EXTERNAL MEMORY INTERFACE REGISTERS**(Cont'd)

### 7.3.2 EMR2: EXTERNAL MEMORY REGISTER 2

**EMR2 R246** (F6h) Page 21 R/W

External Memory Register 2

Reset value: 0000 1111b (0Fh)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| - | ENCSR | DPRREM | MEMSEL | PAS1 | PAS0 | DAS1 | DAS0 |

b7 = Reserved.

b6 = **ENCSR**: *ENable Code Segment Register.* This bit affects the ST9 CPU behavior whenever an interrupt request is issued.

If ENCSR is reset, the CPU works in original ST9 compatibility mode. For the duration of the interrupt service routine, ISR is used instead of CSR, and the interrupt stack frame is identical to that of the original ST9: only the PC and Flags are pushed. This avoids saving the CSR on the stack in the event of an interrupt, thus ensuring a faster interrupt response time. The drawback is that it is not possible for an interrupt service routine to perform inter-segment calls or jumps: these instructions would update the CSR, which, in this case, is not used (ISR is used instead). The code segment size for all interrupt service routines is thus limited to 64K bytes.

If ENCSR is set, ISR is only used to point to the interrupt vector table and to initialize the CSR at the beginning of the interrupt service routine: the old CSR is pushed onto the stack together with the PC and flags, and CSR is then loaded with the contents of ISR. In this case, iret will also restore CSR from the stack. This approach allows interrupt service routines to access the entire 4Mbytes of address space; the drawback is that the interrupt response time is slightly increased, because of the need to also save CSR on the stack. Full compatibility with the original ST9 is lost in this case, because the interrupt stack frame is different; this difference, however, should not affect the vast majority of programs.

The bit is cleared on reset.

b5 = **DPRREM**: *Data Page Registers REMapping.* If DPRREM is set, the locations of the four MMU (Memory Management Unit) Data Page Registers (DPR0, DPR1, DPR2 and DPR3) are swapped with that of the Data Registers of ports 0-3. The bit is cleared at reset.

b4 = **MEMSEL**: *MEMory SELect.* This bit allows to establish the kind of the external memory addressed: this is useful to drive properly the DS2N pin (when enabled to work by setting DS2EN bit) and to establish the proper number of wait cycles to stretch external memory accesses. The information about the sort of referenced external memory is carried either by PDN pin (if the MMU is not available on the device) or by the Most Significant Address Bit (A21) generated by the MMU. If MEMSEL is reset, the PDN pin determinates the memory used:

– PDN= "0" selects external Data memory;
– PDN= "1" selects external Program memory.

If MEMSEL is set, the Most Significant Address Bit (A21) establishes the memory used:

– A21 = "0" selects ROM memory;
– A21 = "1" selects RAM memory.

Whenever either the external Data (MEMSEL="0", PDN="0") or RAM (MEMSEL="1", A21="1") memory is used, the pin DS2N is kept high for the whole memory cycle, while DSN pin is used as external memory interface to control data exchange on Port0. Moreover, the number of wait cycles to stretch the Address Strobe signal during external memory accesses is given by DAS1-0 (b1-0 of EMR2), while the number of wait cycles to stretch the Data Strobe signal is given by WDM2-0.

Whenever either the external Program (MEMSEL="0", PDN ="1") or ROM (MEMSEL="1", A21="0") memory is referenced, the pin DSN is kept high for the whole memory cycle, while DS2N pin (if bit DS2EN is set) is used as external memory interface to control data exchange on Port 0. In this case, the number of wait cycles to stretch the Address Strobe signal during external memory accesses is given by PAS1-0 (b3-2 of register EMR2), while the number of wait cycles to stretch the Data Strobe signal is given by WPM2-0.

The MEMSEL bit is hardware reset if no MMU is available on the specific device, forcing the PDN bit as Program/Data memory select signal. This bit is cleared on reset.

b3-b2 = **PAS1-0**: *Program memory Address strobe Stretch.* These two bits contain the number of wait cycles (from 0 to 3) to add to the System Clock to stretch ASN during either external Program memory accesses (MEMSEL="0", PDN="1") or external ROM accesses (MEMSEL="1", A21="0"). The reset value is 3.

b1-b0 = **DAS1-0**: *Data memory Address strobe Stretch.* These two bits contain the number of wait cycles (from 0 to 3) to add to the System Clock to stretch ASN during either external Data memory accesses (MEMSEL="0", PDN="0") or external RAM accesses (MEMSEL="1", A21="1"). The reset value is 3.

**EXTERNAL MEMORY INTERFACE REGISTERS**(Cont'd)

**WCR R252** (FCh) Page 0 Read/Write

Wait Control Register

Reset Value: 0111 1111 (7Fh)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| 0 | WDGEN | WDM2 | WDM1 | WDM0 | WPM2 | WPM1 | WPM0 |

b7 = Reserved, read as "0".

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

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

b5-b3 = **WDM2-0**: Data Memory 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 (reset condition).

b2-b0 = **WPM2-0**: Program Memory 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 (reset condition).

**Note:** The number of clock cycles added refers to INTCLK and NOT to CPUCLK.

***WARNING.*** The reset value of the Wait Control Register gives the maximum number of Wait cycles for external memory. To get optimum performance from the ST9 when used in single-chip mode (no external memory) the user should write the WDM2,1,0 and WPM2,1,0 bits to "0".

# 8 I/O PORTS

## 8.1 INTRODUCTION

ST9 devices feature flexible individually programmable multifunctional input/output lines. Refer to the Pin Decription Chapter for specific pin allocations. These lines, which are logically grouped as 8-bit ports, can be individually programmed to provide digital and analog input/output, or to connect input/output signals to the on-chip peripherals as alternate pin functions. All ports can be individually configured as an input, bi-directional, output or alternate function. In addition, pull-ups can be turned off for open-drain operation, and weak pull-ups can be turned on in their place, to avoid the need for off-chip resistive pull-ups. Ports (0, 2, 3, 4, and 5), configured as open drain must never have voltage on the port pin exceeding $V_{DD}$ (refer to the Electrical Characteristics section). Input buffers can be either TTL or CMOS compatible.

## 8.2 SPECIFIC PORT CONFIGURATIONS

Refer to the Peripheral Configuration Chapter for a list of the specific port styles and reset values.

## 8.3 PORT CONTROL REGISTERS

Each port is associated with a Data register (PxDR) and three Control registers (PxC0, PxC1, PxC2). These define the port configuration and allow dynamic configuration changes during program execution. Port Data and Control registers are mapped into the Register File as shown inFigure 1. Port Data and Control registers are treated just like any other general purpose register. There are no special instructions for port manipulation: any instruction that can address a register, can address the ports. Data can be directly accessed in the port register, without passing through other memory or “accumulator” locations.

**Figure 47. I/O Register Map**

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

**CONTROL REGISTERS** (Cont'd)

During Reset, ports with weak pull-ups are set in bidirectional/weak pull-up mode and the output Data Register is set to FFh. This condition is also held after Reset, except for Ports 0 and 1 in ROM-less devices, and can be redefined under software control. Ports without weak pull-ups are set in high impedance during reset.

To ensure a proper reset, these ports must be externally connected to either $V_{DD}$ or $V_{SS}$ through external pull-up or pull-down resistors.

### 8.4 INPUT/OUTPUT BIT CONFIGURATION

By programming the control bits PxC0.n and PxC1.n (see Figure 48) it is possible to configure bit Px.n as Input, Output, Bidirectional or Alternate Function Output, where X is the number of the I/O port, and n the bit within the port (n = 0 to 7).

When programmed as input, it is possible to select the input level as TTL or CMOS compatible by programming the relevant PxC2.n control bit, except where the Schmitt trigger option is assigned to the pin.

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

The basic structure of the bit Px.n of a general purpose port Px is shown in Figure 49.

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

**When Px.n is programmed as an Input**
(See Figure 50).

– The Output Buffer is forced tristate.

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

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

**INPUT/OUTPUT BIT CONFIGURATION**(Cont'd)

**Figure 48. Control Bits**

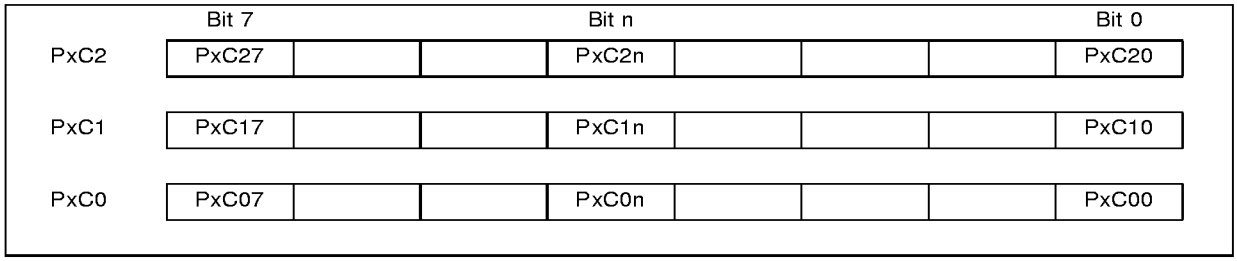



**Table 19. Port Bit Configuration Table (n = 0, 1... 7; x = port number**

| | General Purpose I/O Pins | | | | | | | | ADC Pins |
|---|---|---|---|---|---|---|---|---|---|
| PXC2n<br>PXC1n<br>PXC0n | 0<br>0<br>0 | 1<br>0<br>0 | 0<br>1<br>0 | 1<br>1<br>0 | 0<br>0<br>1 | 1<br>0<br>1 | 0<br>1<br>1 | 1<br>1<br>1 | 1<br>1<br>1 |
| PXn Configuration | BID(1) | BID | OUT | OUT | IN | IN | AF | AF | AF |
| PXn Output Type | WP | OD | PP | OD | HI-Z | HI-Z | PP | OD | HI-Z(2) |
| PXn Input Type | TTL (or Schmitt Trigger) | TTL (or Schmitt Trigger) | TTL (or Schmitt Trigger) | TTL (or Schmitt Trigger) | CMOS (or Schmitt Trigger) | TTL (or Schmitt Trigger) | TTL (or Schmitt Trigger) | TTL (or Schmitt Trigger) | Analog Input |

(1) Reset state of standard I/O port bit with weak pull-up.

(2) For A/D Converter inputs.

**Legend:**

X = Port
n = Bit
AF = Alternate Function
HI-Z= High Impedance

OD = Open Drain
WP = Weak Pull-up
PP = Push-Pull
BID = Bidirectional

TTL = TTL Standard Input Levels
CMOS = CMOS Standard Input Levels
IN = Input
OUT = Output

**INPUT/OUTPUT BIT CONFIGURATION**(Cont'd)

**Figure 49. Basic Structure of an I/O Port Pin**

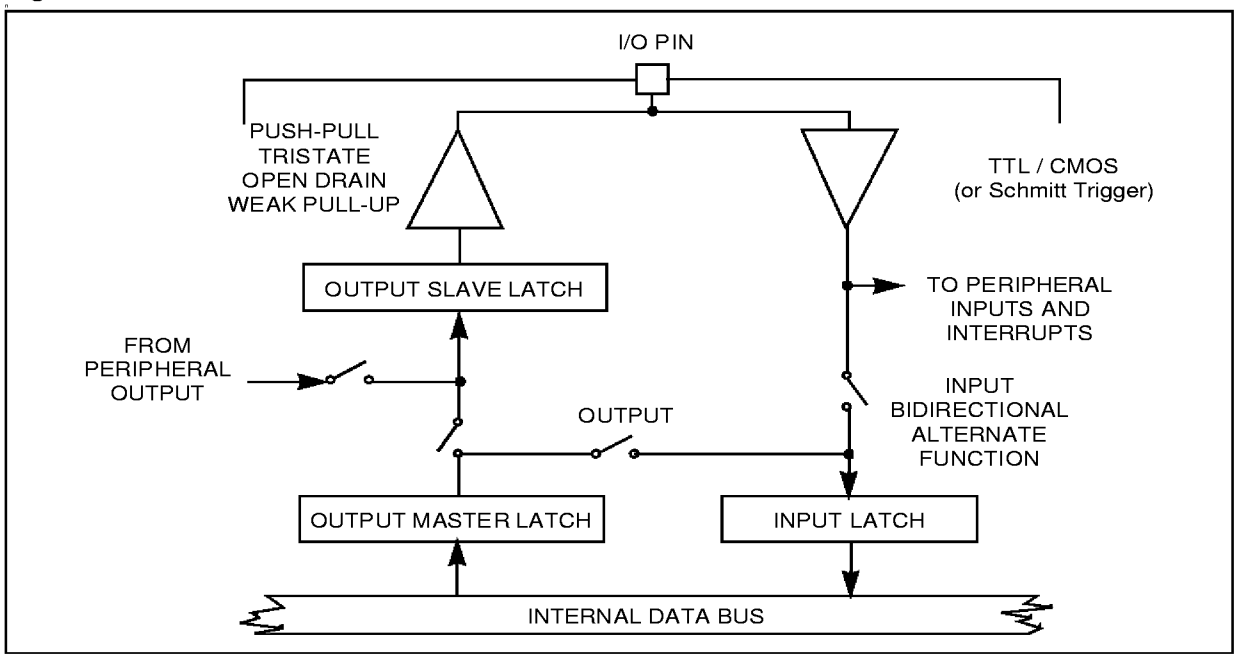



**Figure 50. Input Configuration**

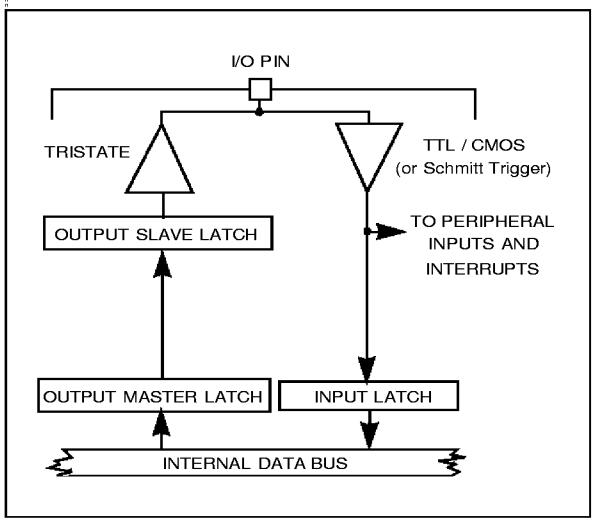
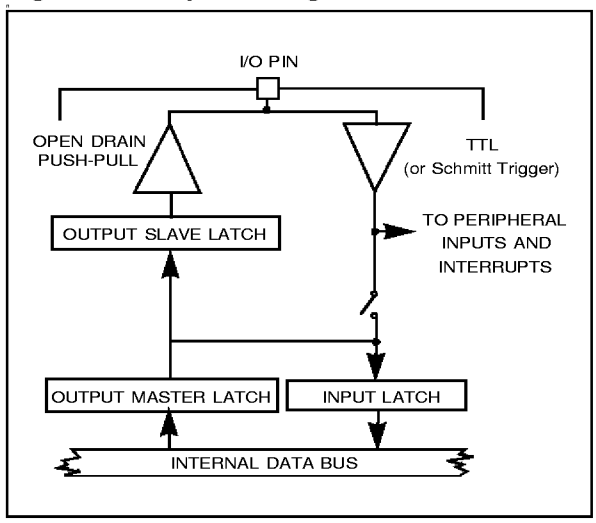



**Figure 51. Output Configuration**



**INPUT/OUTPUT BIT CONFIGURATION**(Cont'd)

**When Px.n is programmed as an Output**
(Figure 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 the instruction.

**When Px.n is programmed as Bidirectional**
(Figure 6)

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

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

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

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

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

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

For example:

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

A `bset` instruction on bit 7 will return:

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

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

**When Px.n is programmed as a digital Alternate Function Output**
(Figure 7)

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

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

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

**Figure 52. Bidirectional Configuration**

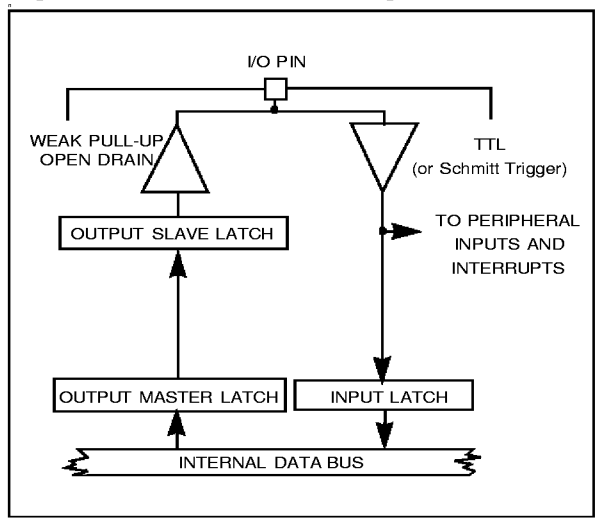
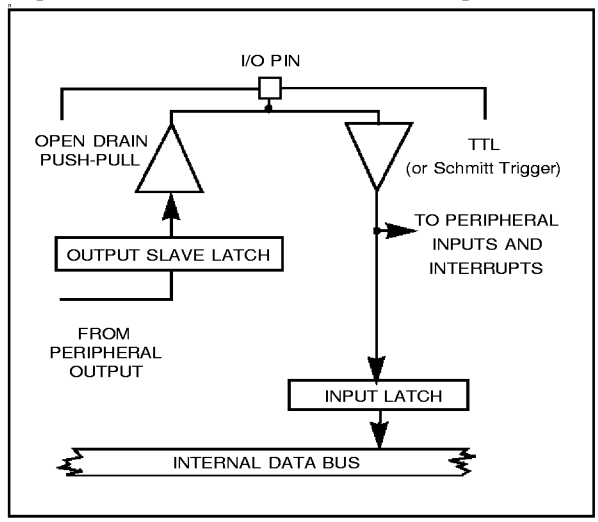



**Figure 53. Alternate Function Configuration**



### 8.5 ALTERNATE FUNCTION ARCHITECTURE

**Alternate Function Inputs and Outputs are different for each specific device. Please refer to the Pin Description Chapter for the specific device configuration.**

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

– Data bus Input/Output

– Alternate Function Input

– Alternate Function Output

#### 8.5.1 Pin Declared as I/O

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

#### 8.5.2 Pin Declared as an Alternate Input

A single pin may be directly connected to several Alternate inputs. In this case, the user must select the required input mode and enable the selected Alternate Function. No specific port configuration is required to enable an Alternate Function input, since the input buffer is directly connected to each alternate function module on the shared pin. As more than one module can use the same input, it is up to the user software to enable the required module as necessary. Parallel I/Os remains operational even when using an Alternate Function input. The exception to this is when an I/O port bit is connected to an analog voltage level (ADC).

#### 8.5.3 Pin Declared as an Alternate Function Output

Several Alternate Function outputs may drive a common pin. In such case, the Alternate Function output signals are logically ANDed before driving the common pin. The user must therefore enable the required Alternate Function Output.

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

### 8.6 I/O STATUS AFTER WFI, HALT AND RESET

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

| Mode | Data Port | Address Port(s) | Other I/O Ports |
|---|---|---|---|
| WFI | Depends on the last memory operation performed on the Port | Next Address | Not Affected (clock outputs running) |
| HALT | | | Not Affected (clock outputs stopped) |
| RESET | Push-Pull / Alternate function | | Bidirectional/Weak Pull-up (High impedance when disabled in hardware; external Pull-ups required). |

# 9 TIMER/WATCHDOG (WDT)

## 9.1 INTRODUCTION

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

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

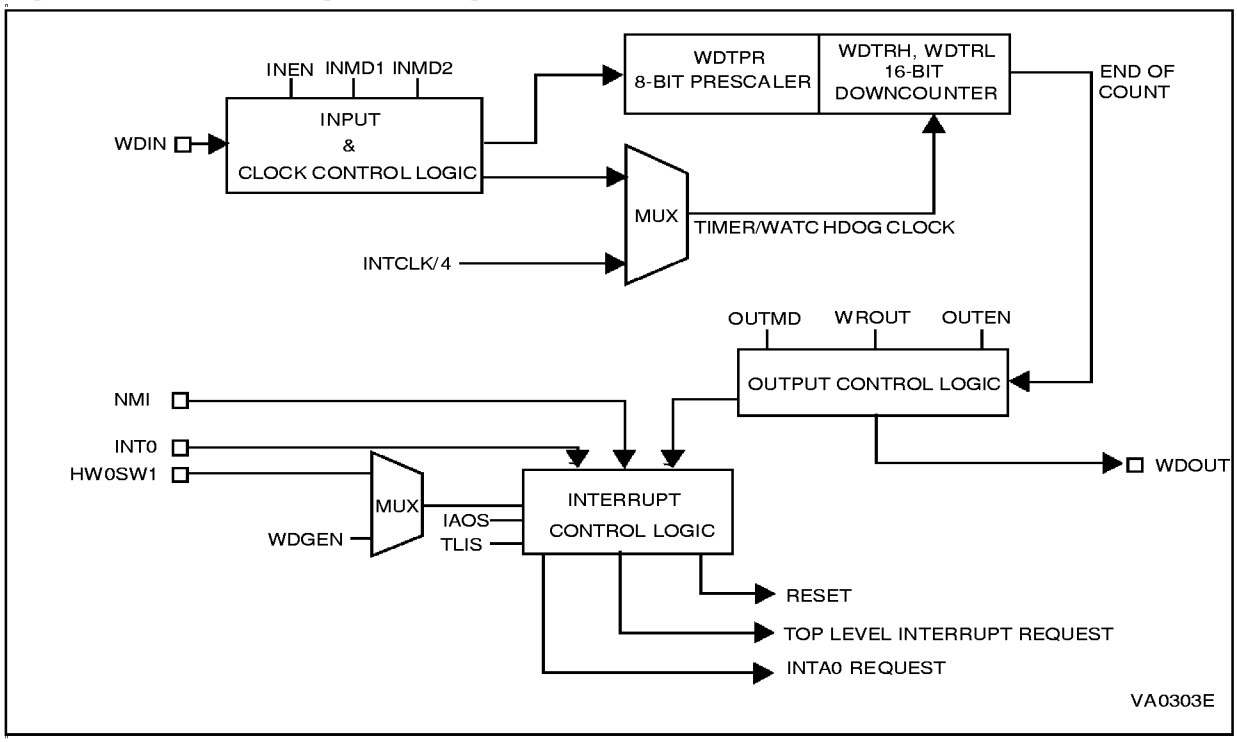
The main WDT registers are:

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

The hardware interface consists of up to five signals:

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

**Figure 1. Timer/Watchdog Block Diagram**




## 9.2 DEVICE-SPECIFIC OPTIONS

Depending on the ST9 variant and package type, some WDT interface signals described in this chapter may not be connected to an external pin. Refer to the signal availability table in the Peripheral Configuration section.

## 9.3 FUNCTIONAL DESCRIPTION

### 9.3.1 External Signals

The HW0SW1 pin can be used to permanently enable Watchdog mode. Refer to section 9.4.1 on page 96.

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

– Event Counter Mode

– Gated External Input Mode

– Triggerable Input Mode

– Retriggerable Input Mode

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

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

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

### 9.3.2 Initialisation

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

### 9.3.3 Start/Stop

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

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

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

End of Count is when the counter is 0.

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

### 9.3.4 Single/Continuous Mode

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

#### 9.3.4.1 Single Mode

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

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

#### 9.3.4.2 Continuous Mode

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

### 9.3.5 Input Section

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

For instance, when INTCLK = 20MHz, the End Of Count rate is:

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

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

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

– Event Counter Mode

– Gated External Input Mode

– Triggerable Input Mode

– Retriggerable Input Mode

The mode is configurable in the WDTCR.

### 9.3.6 Event Counter Mode

In this mode the Timer is driven by the external clock applied to the input pin, thus operating as an event counter. The event is defined as a high to low transition of the input signal. Spacing between trailing edges should be at least 8 INTCLK periods (or 400ns with INTCLK = 20MHz).

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

FUNCTIONAL DESCRIPTION (Cont'd)

### 9.3.7 Gated Input Mode

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

### 9.3.8 Triggerable Input Mode

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

– setting the Start-Stop bit, followed by

– a High to Low transition on the input pin.

To stop the Timer, reset the ST_SP bit.

### 9.3.9 Retriggerable Input Mode

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

### 9.3.10 Timer/Counter Output Modes

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

**No Output Mode**
(OUTEN = "0")

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

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

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

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

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

## 9.4 WATCHDOG TIMER OPERATION

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

### 9.4.1 Hardware Watchdog/Software Watchdog

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

If HW0SW1 is held low:

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

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

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

If HW0SW1 is held high, or is not present:

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

### 9.4.2 Starting the Watchdog

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

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

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

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

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

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

**WATCHDOG TIMER OPERATION (Cont'd)**

### 9.4.3 Preventing Watchdog System Reset

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

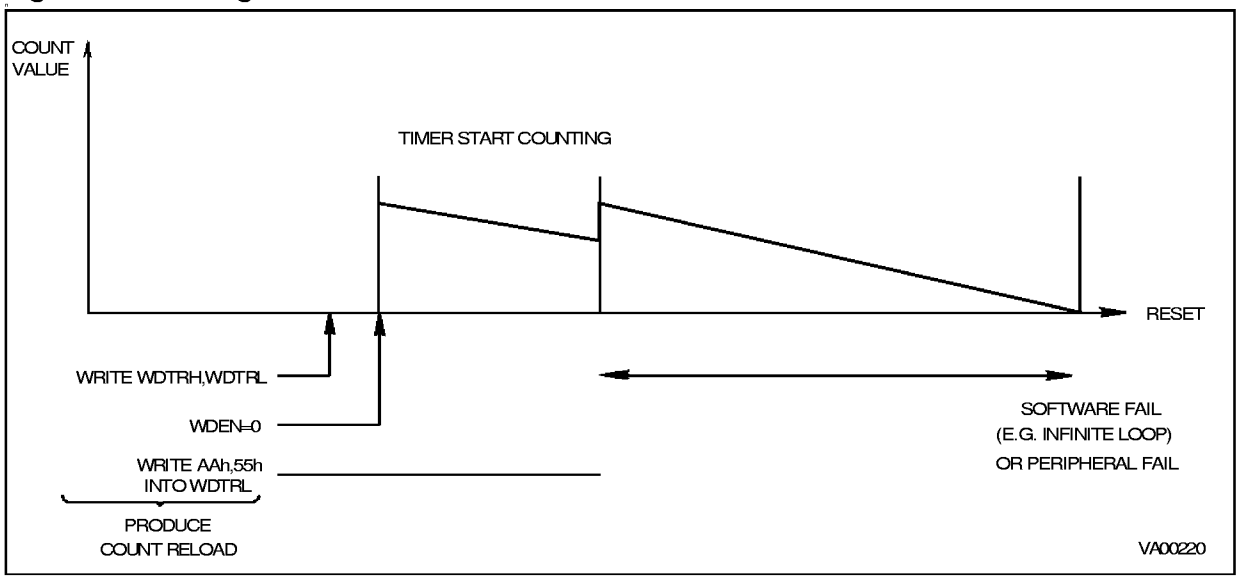
To reload the counter, the two writing operations must be performed sequentially without inserting other instructions that modify the value of the WDTLR register between the writing operations. The maximum allowed time between two reloads of the counter depends on the Watchdog timeout period.

### 9.4.4 Non-Stop Operation

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

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

**Figure 2. Watchdog Timer Mode**



### 9.5 WDT INTERRUPTS

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

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

A block diagram of the interrupt logic is given in Figure Figure 3.

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

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

A reset caused by the watchdog will set bit 6, WDGRES of R242 (Clock Flag Register). Seesection CLOCK CONTROL REGISTERS

**Figure 3. Interrupt Sources**

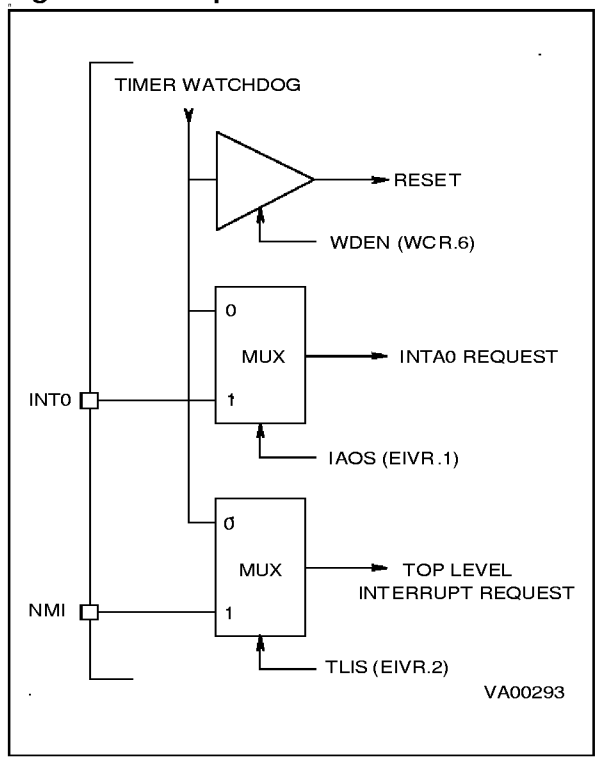



**Table 20. Interrupt Configuration**

| Control Bits | | | Enabled Sources | | | Operating Mode |
|---|---|---|---|---|---|---|
| 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

### 9.6 TIMER/WATCHDOG REGISTERS

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

**WDTHR**: Timer/Watchdog High Register

**WDTLR**: Timer/Watchdog Low Register

**WDTPR**: Timer/Watchdog Prescaler Register

**WDTCR**: Timer/Watchdog Control Register

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

Watchdog Mode Enable, (WCR.6)

Top Level Interrupt Selection, (EIVR.2)

Interrupt A0 Channel Selection, (EIVR.1)

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

**Counter Register**

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

**WDTHR R248** (F8h) Page 0 Read/Write

Timer/Watchdog High Register

Reset value: 1111 1111b (FFh)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| R15 | R14 | R13 | R12 | R11 | R10 | R9 | R8 |

b7-b0 = **R[15-8]** Counter Most Significant Bits.

**WDTLR R249** (F9h) Page 0 Read/Write

Timer/Watchdog Low Register

Reset value: 1111 1111b (FFh)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |

b7-b0 = **R[7-0]** Counter Least Significant Bits.

**WDTPR R250** (FAh) Page 0 Read/Write

Timer/Watchdog Prescaler Register

Reset value: 1111 1111b (FFh)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| PR7 | PR6 | PR5 | PR4 | PR3 | PR2 | PR1 | PR0 |

b7-b0 = **PR[7-0]** Prescaler value from 1 (00h) to 256 (FFh).

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

**TIMER/WATCHDOG REGISTERS**(Cont'd)

**WDTCR R251**(FBh) Page 0 Read/Write
Watchdog Timer Control Register
Reset value: 0001 0010b (12h)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| ST_SP | S_C | INMD1 | INMD2 | INEN | OUTMD | WROUT | OUTEN |

b7 = **ST_SP**: *Start/Stop Bit* Setting this bit, starts the counting operation (see Warning above). When this bit is reset, 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 selected when this bit is reset.

b5-b4 = **INMD1, INMD2**: *Input mode selection bits*.
These bits select the input mode:

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

b3 = **INEN**: *Input Enable*. This bit enables the input section when set, and disables it when reset.

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

b1 = **WROUT**: *Write Out* The status of this bit is transferred to the Output pin when OUTMD is set; it is user definable to allow PWM output (on Reset WROUT is set).

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

**WCR R252** (FCh) Page 0 Read/Write
Wait Control Register
Reset value: 0111 1111b (7Fh)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| x | WDGEN | x | x | x | x | x | x |

b6 = **WDGEN**: *Watchdog Enable* (active low). Resetting this bit via software enters the Watchdog mode. Once reset, it cannot be set anymore by the user program. At System Reset, the Watchdog mode is disabled.

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

**EIVR R246** (F6h) Page 0 Read/Write
External Interrupt Vector Register
Reset value: xxxx 0110b (x6h)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| x | x | x | x | x | TLIS | IA0S | x |

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.

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

Other bits are described in the Interrupt section.

# 10 MULTIFUNCTION TIMER (MFT)

## 10.1 INTRODUCTION

The Multifunction Timer (MFT) peripheral offers powerful timing capabilities and features 12 operating modes, including automatic PWM generation and frequency measurement.

The MFT comprises a 16-bit Up/Down counter driven by an 8-bit programmable prescaler. The input clock may be INTCLK/3 or an external source. The timer features two 16-bit Comparison Registers, and two 16-bit Capture/Load/Reload Registers. Two input pins and two alternate function output pins are available.

Several functional configurations are possible, for instance:

– 2 input captures on separate external lines, and 2 independent output compare functions with the counter in free-running mode, or 1 output compare at 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 at a fixed repetition rate.

When two timers are present in an ST9 device, a combined operating mode is available.

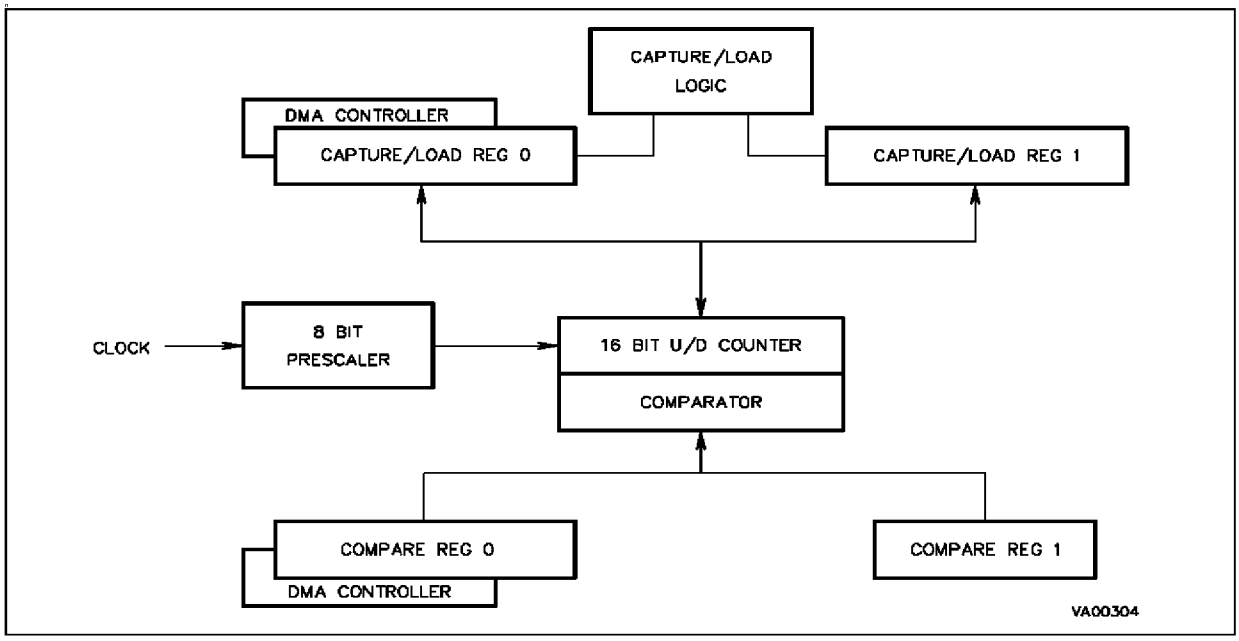
Four internal signals are also available for timing on-chip functions: the On-Chip Event signal can be used to control other peripherals on the chip itself, and 3 other signals, can be internally connected to I/O ports, to allow automatic, timed, DMA transfers. The two external inputs may be individually programmed to detect any of the following:

– rising edges

– falling edges

– both rising and falling edges

**Figure 4. MFT Simplified Block Diagram**



**INTRODUCTION** (Cont'd)

The configuration of each input is programmed in the Input Control Register.

Each of the two output pins 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 actions, independently, on each of the two outputs:

– Nop, Set, Reset, Toggle

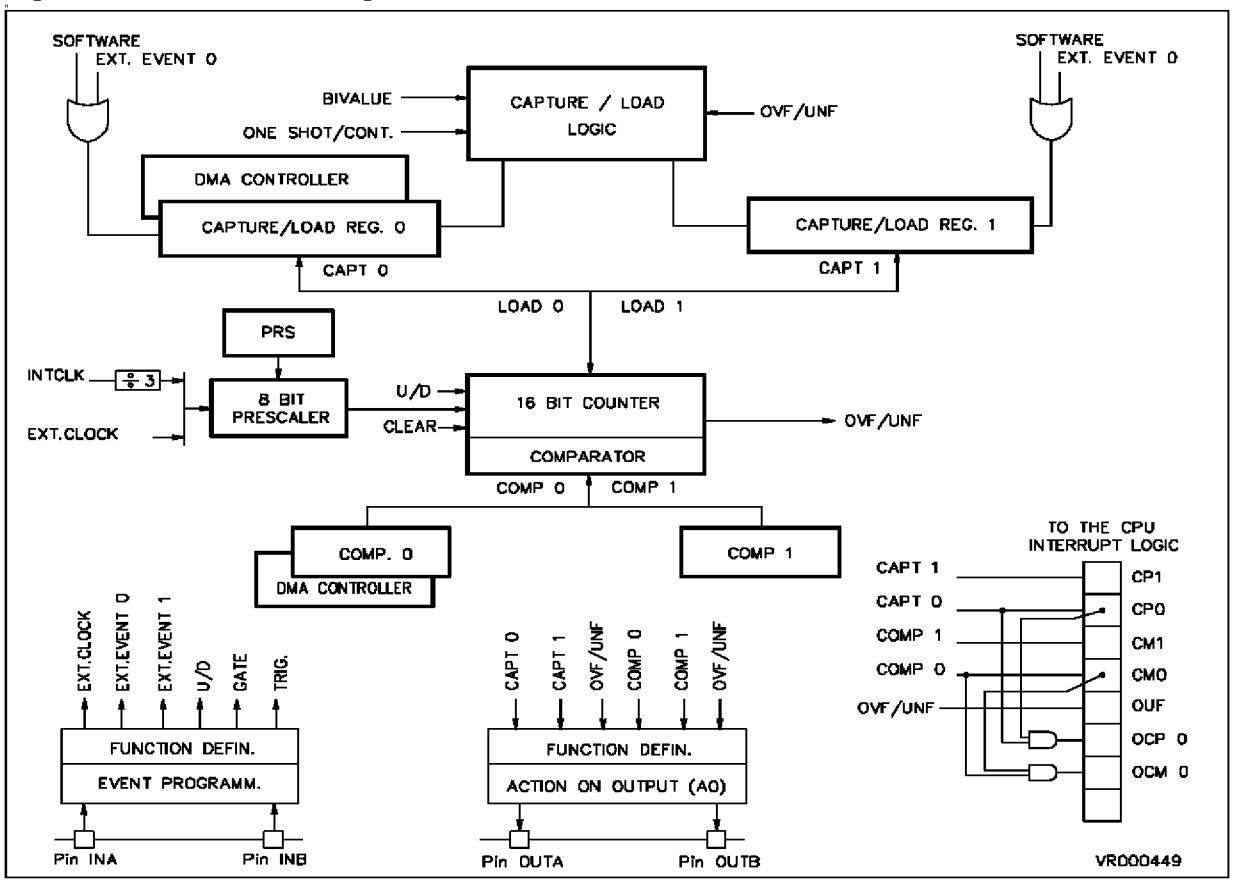
In addition, an additional On-Chip Event signal can be generated by two of the three sources mentioned above, i.e. Over/Underflow event and Compare 0 event. This signal can be used internally to synchronise another on-chip peripheral, or as a strobe for an I/O port (see I/O port chapter).Five maskable 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 rapid data transfer operations. Each DMA request (associated with a capture on the REG0R register, or with a compare on the CMP0R register) has priority over an interrupt request generated by the same source.

Each DMA channel may be involved in external transfers between memory and an I/O port, using three internal lines (one for setting the data flow direction, and two for transfer synchronisation).

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

**Figure 5. Detailed Block Diagram**



## 10.2 FUNCTIONAL DESCRIPTION

The MFT operating modes are selected by programming the Timer Control Register (TCR) and the Timer Mode Register (TMR).

### 10.2.1 One Shot Mode

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

### 10.2.2 Continuous Mode

Whenever the counter reaches an End Of Count condition, the counting sequence is automatically restarted and the counter is reloaded from REG0R (or from REG1R, when selected in Biload Mode). Continuous Mode is entered by resetting the C0 bit in TMR.

### 10.2.3 Triggered And Retriggered Modes

A triggered event may be generated by software (by setting either the CP0 or the CP1 bit in the FLAGR timer register), or by an external source which may be programmed to respond to the rising edge, the falling edge or both, by programming bits A0-A1 and B0-B1 in ICR.

In One Shot and Triggered Mode, every trigger event arriving before an End Of Count, is masked. In One Shot and Retriggered Mode, every trigger received while the counter is running, automatically reloads the counter from REG0R. Triggered/Retriggered Mode is set by the REN bit in TMR.

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

**WARNING**. If the Triggered Mode is selected when the counter is in Continuous Mode, every trigger is disabled, it is not therefore possible to synchronise the counting cycle by hardware or software.

### 10.2.4 Gated Mode

In this mode, counting takes place only when the external gate input is at a logic low level. The selection of TxINA or TxINB as the gate input is made by programming the IN0-IN3 bits in ICR.

### 10.2.5 Capture Mode

The 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 on REG1R, initiated by software (by setting CP0 or CP1 in the FLAGR register) or by an event on the external input pins.

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

### 10.2.6 Up/Down Mode

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

**FUNCTIONAL DESCRIPTION** (Cont'd)

### 10.2.7 Free Running Mode

The timer counts continuously (in up or down mode) and the counter value simply overflows or underflows through 0FFFh or zero; there is no End Of Count condition as such, and no reloading takes place. This mode is automatically selected either in Bicapture Mode or by setting REG0R for a capture function (Continuous Mode must also be set). In Autoclear Mode, free running operation can be had, with the possibility of choosing a maximum count value before overflow or underflow which is less than $2^{16}$ (see Autoclear Mode).

### 10.2.8 Monitor Mode

When the RM1 bit in TMR is reset, and the timer is not in Bivalue Mode, REG1R acts as a monitor, duplicating the current up or down counter contents, thus allowing the counter to be read "on the fly".

### 10.2.9 Autoclear Mode

A clear command forces the counter either to 0000h or to 0FFFFh, depending on whether upcounting or downcounting is selected. The counter reset may be obtained either directly, through the CCL bit in TCR, or by entering the Autoclear Mode, through the CCP0 and CCMP0 bits in TCR.

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

The Clear On Capture mode allows direct measurement of delta time between successive captures on REG0R, while the Clear On Compare mode allows free running with the possibility of choosing a maximum count value before overflow or underflow which is less than $2^{16}$ (see Free Running Mode).

### 10.2.10 Bivalue Mode

Depending on the value of the RM0 bit in TMR, the Biload Mode (RM0 reset) or the Bicapture Mode (RM0 set) can be selected as illustrated inFigure 21 below:

**Table 21. Bivalue 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 set in TMR) and programming REG0R as a reload register (RM0 reset in TMR).

At any End Of Count, 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).

**FUNCTIONAL DESCRIPTION** (Cont'd)

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

B) Bicapture Mode

The Bicapture Mode is entered by selecting the Bivalue Mode (the BM bit in TMR is set) and by programming REG0R as a capture register (the RM0 bit in TMR is set).

Every capture event, software simulated (by setting the CP0 flag) or coming directly from the TxINA input line, captures the current counter value alternately into REG0R and REG1R. When the BM bit is reset, REG0R is the current register, so that the first capture, after resetting the BM bit, is always into REG0R.

### 10.2.11 Parallel Mode

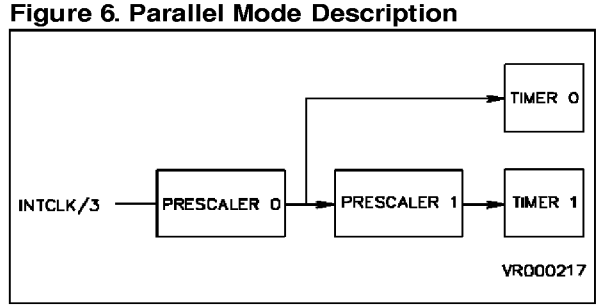
When two timers are present on an ST9 device, the parallel mode is entered when the ECK bit in the TMR register of Timer 1 is set. The Timer 1 prescaler input is internally connected to the Timer 0 prescaler output. Timer 0 prescaler input is connected to the system clock line.

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. In this mode the clock frequency may be divided by a factor in the range from 1 to $2^{16}$.

### 10.2.12 Autodiscriminator Mode

The phase difference sign of two overlapping 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.

**Figure 6. Parallel Mode Description**



### 10.3 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, or both 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 bits IN0 - IN3 of the ICR, as illustrated inFigure 22

**Table 22. 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 relating to the external input pin assignment are defined in conjunction with the RM0 and RM1 bits in TMR.

For input pin assignment codes which use the input pins as Trigger Inputs (except for code 1010, Trigger Up:Trigger Down), the following conditions apply:

– a trigger signal on the TxINA input pin performs an U/D counter load if RM0 is reset, or an external capture if RM0 is set.

– a trigger signal on the 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 must not 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 must not be less than the prescaler clock period (INTCLK/3) if the input pin is programmed as rising and falling edge sensitive (valid also in Auto discrimination 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 and the edge of the input pin B pulse, must be at least equal to the system clock (INTCLK) period.

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

**INPUT PIN ASSIGNMENT** (Cont'd)

**10.3.1 TxINA = I/O - TxINB = I/O**

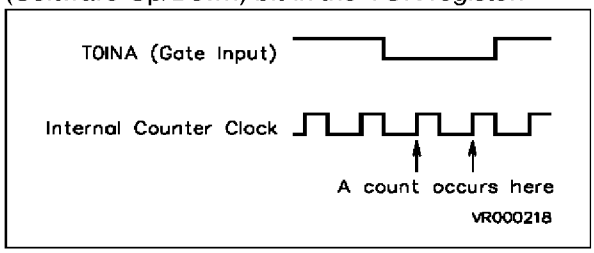
Input pins A and B are not used by the Timer. The counter clock is internally generated and the up/down selection may be made only by software via the UDC (Software Up/Down) bit in the TCR register.

**10.3.2 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 selection may be made only by software via the UDC (Software Up/Down) bit in the TCR register.

**10.3.3 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 via the UDC (Software Up/Down) bit in the TCR register.




**10.3.4 TxINA = Gate - TxINB = Trigger**

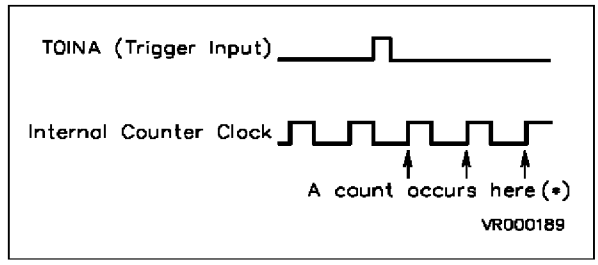
Both input pins A and B are connected to the timer, with the resulting effect of combining the actions relating to the previously described configurations.

**10.3.5 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 selection may be made only by software via the UDC (Software Up/Down) bit in the TCR register.

**10.3.6 TxINA = Trigger - TxINB = I/O**

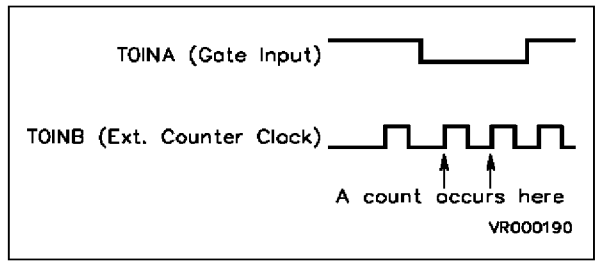
The signal applied to input pin A acts as a trigger for REG0R, initiating the action for which the register was programmed (i.e. a reload or capture). The prescaler clock is internally generated and the up/down selection may be made only by software via the UDC (Software Up/Down) bit in the TCR register.




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

**10.3.7 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.



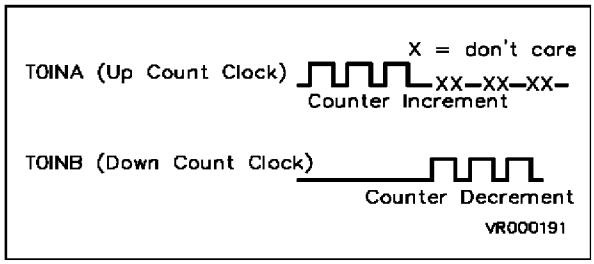

**10.3.8 TxINA = Trigger - TxINB = Trigger**

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

**INPUT PIN ASSIGNMENT** (Cont'd)

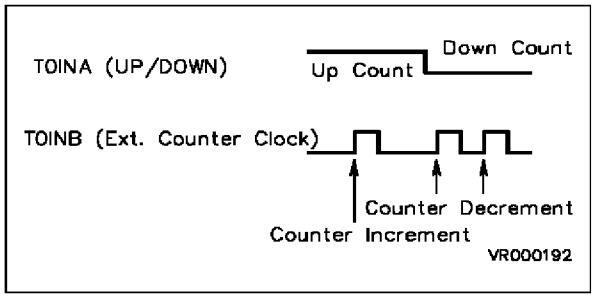
### 10.3.9 TxINA = Clock Up - TxINB = Clock Down

The edge 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, and input pin B has priority on input pin A.



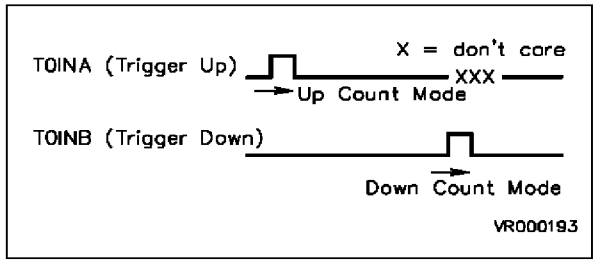

### 10.3.10 TxINA = Up/Down - TxINB = Ext Clock

An High (or Low) level applied to 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.



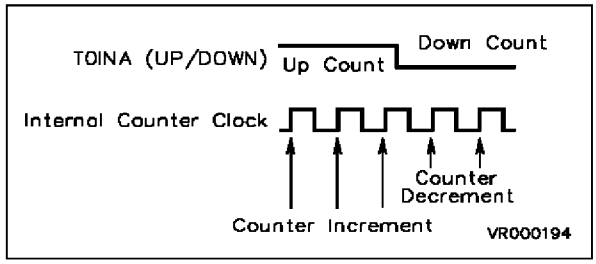

### 10.3.11 TxINA = Trigger Up - TxINB = Trigger Down

Up/down control is performed through both input pins A and B. A edge on input pin A sets the up count mode, while a edge on input pin B (which has priority on input pin A) sets the down count mode. The counter clock is internally generated, and setting the UDC bit in the TCR register has no effect in this configuration.




### 10.3.12 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.

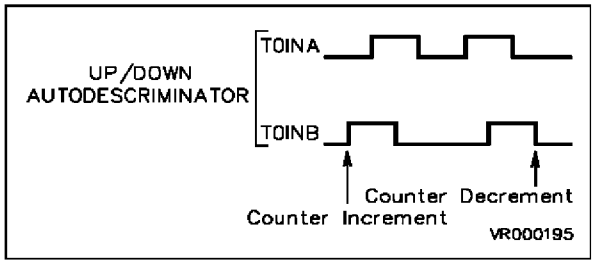


**INPUT PIN ASSIGNMENT** (Cont'd)

### 10.3.13 Autodiscrimination Mode

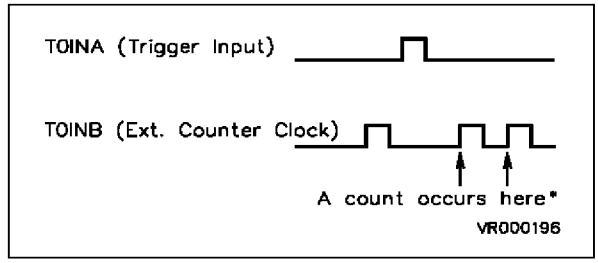
The phase between two pulses (respectively on input pin B and input pin A) generates a one step up (or down) count, so that the up/down control and the counter clock are both external. Thus, if the rising edge of TxINB arrives when TxINA is at a low level, the timer is incremented (no action if the rising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

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




### 10.3.14 TxINA = Trigger - TxINB = Ext. Clock

The signal applied to input pin A acts as a trigger signal on REG0R, initiating 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 the clock for the prescaler.




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

### 10.3.15 TxINA = Ext. Clock - TxINB = Trigger

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

### 10.3.16 TxINA = Trigger - TxINB = Gate

The signal applied to input pin A acts as a trigger signal on REG0R, initiating 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).

### 10.4 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 actions on any of the two outputs:

– Nop

– Set

– Reset

– Toggle

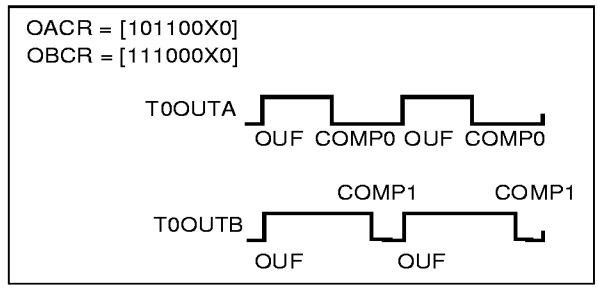
Furthermore an On Chip Event signal can be driven by two of the three sources: the Over/Underflow event and Compare 0 event by programming the CEV bit of the OACR register and the OEV bit of OBCR register respectively. This signal can be used internally to synchronise another on-chip peripheral or as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

Output Waveforms

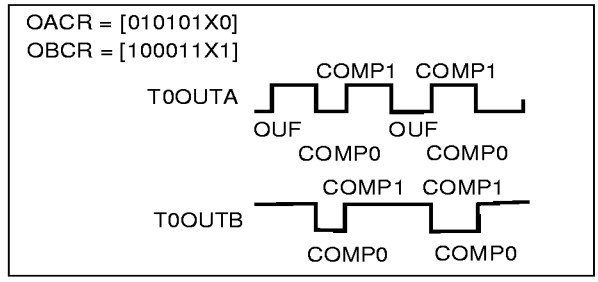
Depending on the programming of OACR and OBCR, the following example waveforms can be generated on TxOUTA and TxOUTB pins.

For a configuration where TxOUTA is driven by the Over/Underflow (OUF) and the Compare 0 event (CM0), and 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.

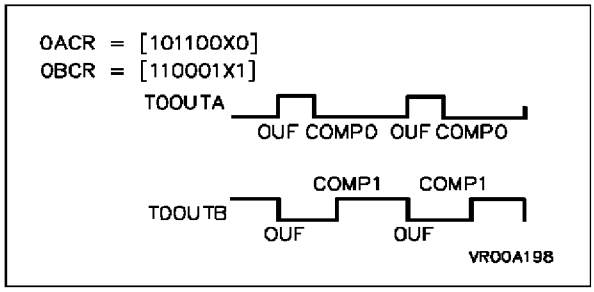



For a configuration where TxOUTA is driven by the Over/Underflow, by Compare 0 and by Compare 1; TxOUTB is driven by both Compare 0 and Compare 1. OACR is programmed with TxOUTA preset to “0”. OUF toggles Output 0, as do CM0 and CM1. OBCR is programmed with TxOUTB preset to “1”. OUF does not affect the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

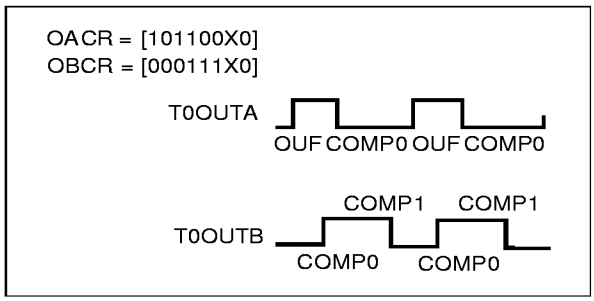


**OUTPUT PIN ASSIGNMENT** (Cont'd)

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



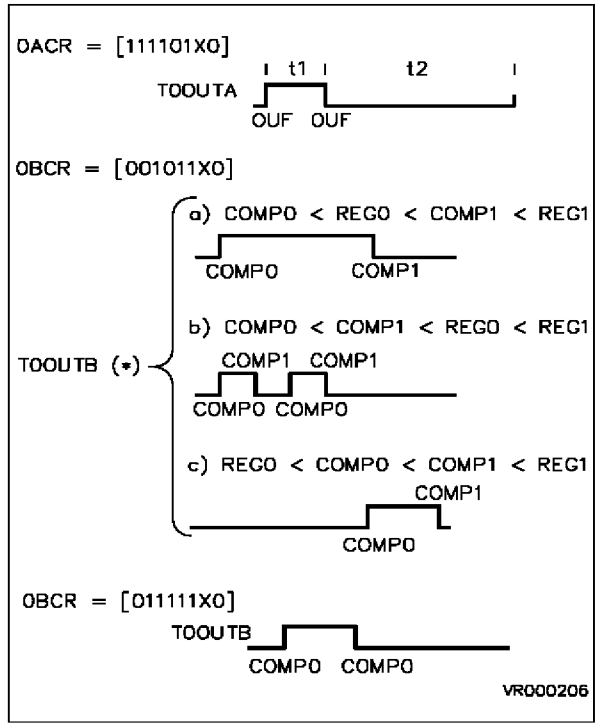

For a configuration where TxOUTA is driven by the Over/Underflow and by Compare 0, and TxOUTB is driven by Compare 0 and 1. OACR is programmed with TxOUTA preset to "0". OUF sets TxOUTA, CM0 resets it and CM1 has no effect. OBCR is programmed with TxOUTB preset to "0". OUF has no effect, CM0 sets TxOUTB 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 of 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 sample waveforms have been drawn based on 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

## 10.5 INTERRUPT AND DMA

### 10.5.1 Timer Interrupt

The timer has 5 different Interrupt sources, belonging to 3 independent groups, which are assigned to the following Interrupt vectors:

**Table 23. Timer Interrupt Structure**

| Interrupt Source | Vector Address |
|---|---|
| COMP 0<br>COMP 1 | xxxx x110 |
| CAPT 0<br>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, where 000 represents the highest priority level. These priorities are fixed by hardware, according to the source which generates the interrupt request. The 5 most significant bits are programmed by the user in the Interrupt Vector Register (IVR) associated with each Timer.

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

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

### 10.5.2 Timer DMA

Two Independent DMA channels, associated with Comp0 and Capt0 respectively, allow DMA transfers from Register File or Memory to the Comp0 Register, and from the Capt0 Register to Register File or Memory (transfers between Memory and an I/O port are also available). Their priority is set by hardware 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 with each Interrupt mask and DMA mask combination, follow the standard architecture described in the Interrupt and DMA chapters.

### 10.5.3 DMA Pointers

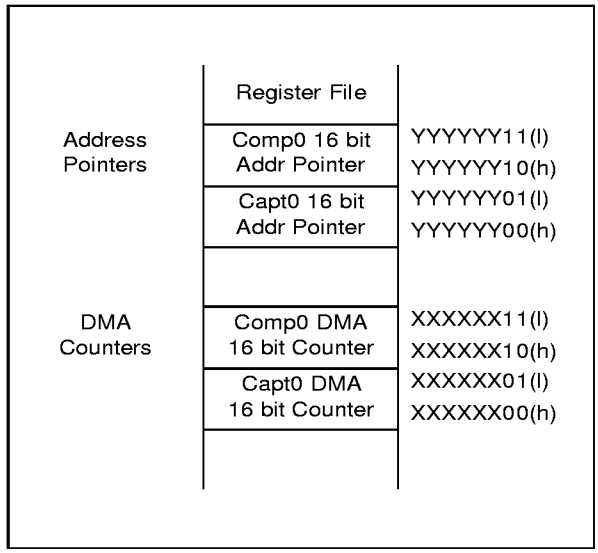
The 6 programmable most significant bits of the DMA Counter Pointer Register (DCPR) and of the DMA Address Pointer Register (DAPR) are common to both channels (Comp0 and Capt0). The Comp0 and Capt0 Address Pointers are mapped as a pair in the Register File, as are the Comp0 and Capt0 DMA Counter pair.

In order to specify either the Capt0 or the Comp0 pointers, according to the channel being serviced, the Timer resets address bit 1 for CAPT0 and sets it for COMP0, when the D0 bit in the DCPR register is equal to zero (Word address in Register File). In this case (transfers between registers and memory), the pointers are split into two groups of adjacent Address and Counter pairs respectively.

For register to register transfers (selected by programming “1” into bit 0 of the DCPR register), only one pair of pointers is required, and the pointers are mapped into one group of adjacent positions.

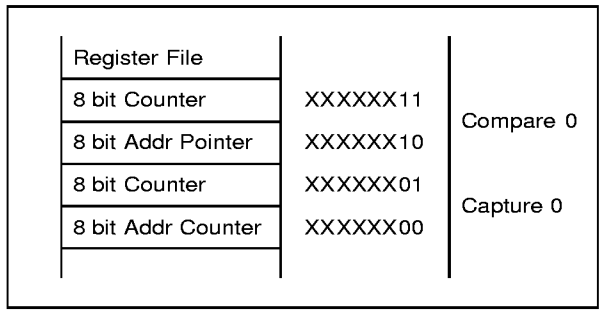
The DMA Address Pointer Register (DAPR) is not used in this case, but must be considered reserved.

**Figure 7. Pointer Mapping for Transfers between Registers and Memory**



**INTERRUPT AND DMA** (Cont'd)

**Figure 8. Pointer Mapping for Register to Register Transfers**




### 10.5.4 DMA Transaction Priorities

Each Timer DMA transaction is a 16-bit operation, therefore two bytes must be transferred sequentially, by means of two DMA transfers. In order to speed up each word transfer, the second byte transfer is executed by automatically forcing the peripheral priority to the highest level (000), regardless of the previously set level. It is then restored to its original value after executing the transfer. Thus, once a request is being serviced, its hardware priority is kept at the highest level regardless of the other Timer internal sources, i.e. once a Comp0 request is being serviced, it maintains a higher priority, even if a Capt0 request occurs between the two byte transfers.

### 10.5.5 DMA Swap Mode

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

The End Of Block Interrupt routine must at this point reload both address and counter pointers of the channel referred to 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 required for the reload routine.

The SWAP feature overcomes this drawback, allowing high speed continuous transfers. Bit 2 of the DMA Counter Pointer Register (DCPR) and of the DMA Address Pointer Register (DAPR), toggles after every 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 update or read 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 channels (CM0 and CP0).*

### 10.5.6 DMA End Of Block Interrupt Routine

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

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

– Test Toggle bit and Jump.
– Reload Pointers (odd or even depending on toggle bit status).
– Reset EOB bit: this bit must be reset only after the old pair of pointers has been restored, so that, if a new EOB condition occurs, the next pair of pointers is ready for swapping.
– Verify the software protection condition.
– Read the corresponding Overrun bit: this confirms that no DMA request has been lost in the meantime.
– Return.

**WARNING**: The EOB bits are read/write **only** for test purposes. Writing a logical "1" by software (when the SWEN bit is set) will cause a spurious interrupt request. These bits are normally only reset by software.

### 10.5.7 DMA Software Protection

A second EOB condition may occur before the first EOB routine is completed, this would cause a not yet updated pointer pair 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), thus blocking any further DMA operation. As shown above, this mask bit should always be checked in each EOB routine, to ensure that all DMA transfers are properly served.

## 10.6 TIMER DMA EXTERNAL I/O PORT MODES

Each Timer DMA channel can also be employed in external transfers between memory and an I/O port with the handshake capability (available on some specific device types). In this case only Byte transfers are executed for any request. Two control bits (DCTS and DCTD) in the Interrupt/DMA Control Register (IDCR) set each channel in INT/EXT mode.

Internal = between Register File or Memory and the MFT

External = between Register File or Memory and I/O ports.

The relevant I/O port must then be programmed in DMA mode and the port direction selected by programming the HDCxR register of that port.

The two modes, however, are not the same for both channels, as explained in the following section.

### 10.6.1 CM0 Channel External Mode

This mode is enabled when the DCTD (DMA Compare Transaction Destination) bit is set in the IDCR register.

This mode allows only Output transfers, from Register File/Memory to the I/O port, as a result of a request generated by a CM0 event or of a software request (setting the CM0 flag). A typical application would be a DMA data flow which needs to be output at fixed times.

Synchronization with the I/O port is accomplished by an internal signal, active when the data to be transfered is present on the internal Data Bus. If programmed, the on-chip event pulse can also be generated and used to strobe the output data on the selected handshake port.

In either case the DMA Output mode must be selected in the HDCTL Register of the port (see Handshake chapter).

### 10.6.2 CP0 Channel In External Mode

This mode is enabled when the DCTS (DMA Capture Transaction Source) bit is set in the IDCR register.

This mode allows bi-directional transfers controlled (when the I/O port is programmed in DMA Input/Output mode in the HDCTL register) by the value of the DD bit of the HDCTL register (the DD bit selects the DMA input or the DMA Output mode).

The DMA request can be either an External CPT0 request (Timer External input A) or a software request (by setting the CP0 Flag).

This, along with a further internal synchronization signal, generated by the Timer Unit, allows handshake operations managed by the I/O port while the direction of the data to read or write on the I/O port is fixed by the value of the DD bit in the HDCTL register.

### 10.6.3 DMA Channel Synchronization

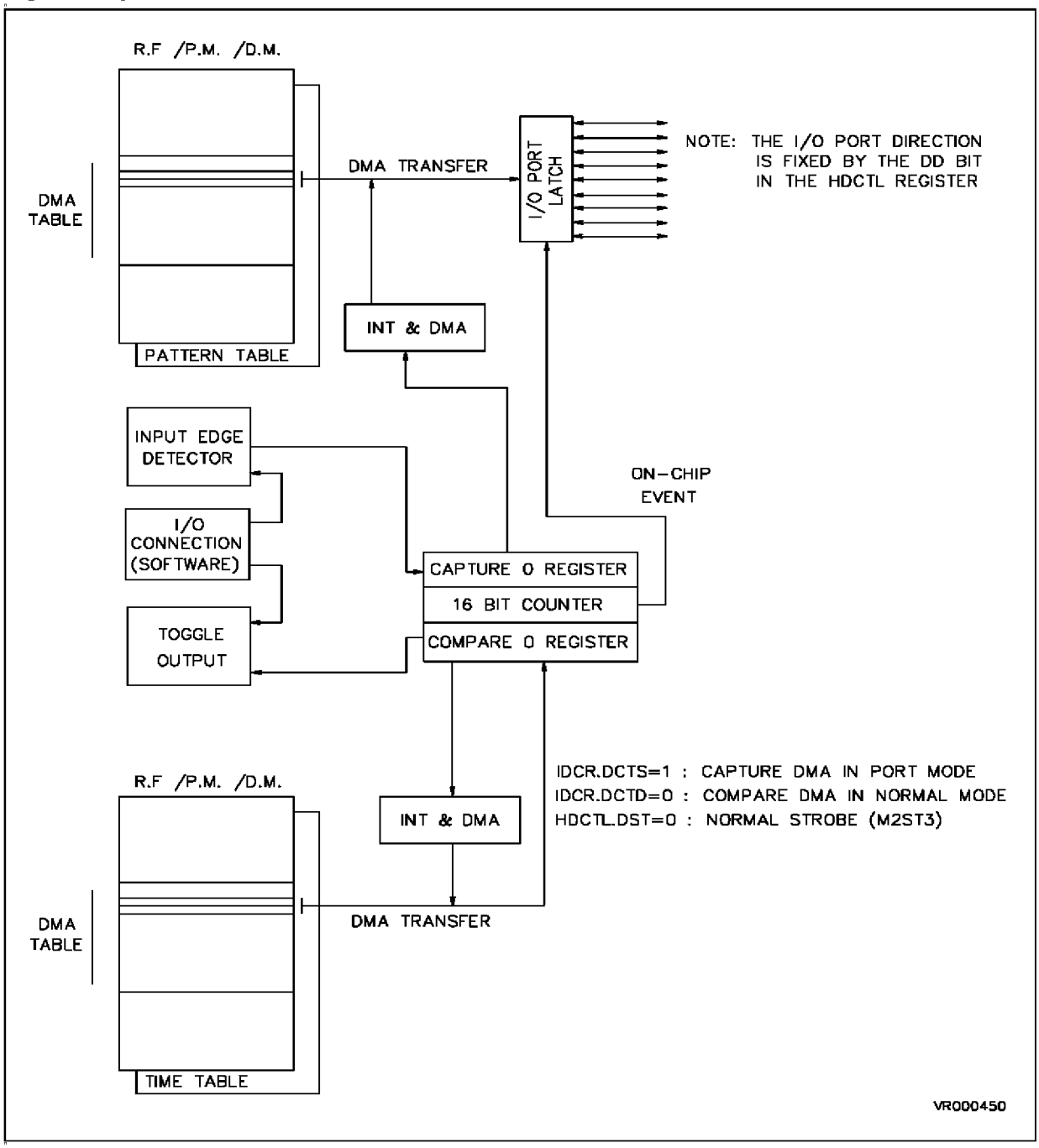
A CP0 DMA request can also be generated by a CM0 event, simply by setting the Timer External Input A to be sensitive to both rising and falling edges, connecting it by hardware or software (though the IOCR register) to the Timer OUT 0, and programming the CM0 action as output toggle.

This will cause a CP0 request to be generated after each CM0 condition, thus synchronizing the 2 DMA channels (see the following application example).

The DCTS bit must be set and the DCTD bit must be reset, in the IDCR register. Figure 9 shows an example of two channel synchronisation. A new byte will be sent out through the I/O port at an interval specified by the COMP0 value mapped in the look-up table.

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

**Figure 9. Synchronization of Timer DMA Channels**



### 10.7 REGISTER DESCRIPTION

Twentyone control and data registers are associated with each Multifunction Timer, and are located in the Group F I/O pages of the ST9 Register File, as shown in Figure 24below.

Note that unused registers must be considered as reserved.

Each register is described in detail in the following pages, together with the meaning and function of every bit. The register is referred to without the absolute address which is dependent on the number of the timer used.

**Table 24. Multifunction Timer Register Map (Group F)**

| | Page 8 | Page 9 | Page 10<br>Page 0Ah | Page 12<br>Page 0Ch | Page 13<br>Page 0Dh | |
|---|---|---|---|---|---|---|
| R255 | IDMR - TIM1 | | IDMR - TIM0 | IDMR - TIM3 | | FFh |
| R254 | FLAGR-TIM1 | | FLAGR - TIM0 | FLAGR - TIM3 | | FEh |
| R253 | OBCR - TIM1 | | OBCR - TIM0 | OBCR - TIM3 | | FDh |
| R252 | OACR - TIM1 | | OACR - TIM0 | OACR - TIM3 | | FCh |
| R251 | PRSR - TIM1 | | PRSR - TIM0 | PRSR - TIM3 | | FBh |
| R250 | ICR - TIM1 | | ICR - TIM0 | ICR - TIM3 | | FAh |
| R249 | TMR - TIM1 | | TMR - TIM0 | TMR - TIM3 | | F9h |
| R248 | TCR - TIM1 | IOCR - TIM0, TIM1 | TCR - TIM0 | TCR - TIM3 | IOCR - TIM3 | F8h |
| R247 | CMP1LR - TIM1 | IDCR - TIM1 | CMP1LR - TIM0 | CMP1LR - TIM3 | IDCR - TIM3 | F7h |
| R246 | CMP1HR - TIM1 | IVR - TIM1 | CMP1HR - TIM0 | CMP1HR - TIM3 | IVR - TIM3 | F6h |
| R245 | CMP0LR - TIM1 | DAPR - TIM1 | CMP0LR - TIM0 | CMP0LR - TIM3 | DAPR - TIM3 | F5h |
| R244 | CMP0HR - TIM1 | DCPR - TIM1 | CMP0HR - TIM0 | CMP0HR - TIM3 | DCPR - TIM3 | F4h |
| R243 | REG1LR - TIM1 | IDCR - TIM0 | REG1LR - TIM0 | REG1LR - TIM3 | | F3h |
| R242 | REG1HR - TIM1 | IVR - TIM0 | REG1HR - TIM0 | REG1HR - TIM3 | | F2h |
| R241 | REG0LR - TIM1 | DAPR - TIM0 | REG0LR - TIM0 | REG0LR - TIM3 | | F1h |
| R240 | REG0HR - TIM1 | DCPR - TIM0 | REG0HR - TIM0 | REG0HR - TIM3 | | F0h |

**REGISTER DESCRIPTION** (Cont'd)

### 10.7.1 Register 0 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 |

### 10.7.2 Register 1 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 High Register 1

Reset value: **undefined**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| R15 | R14 | R13 | R12 | R11 | R10 | R9 | R8 |

**REG1LR R243** (F3h) Read/Write

Capture Load Low Register 1

Reset value: **undefined**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |

### 10.7.3 Compare 0 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 High Register

Reset value: **00**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| R15 | R14 | R13 | R12 | R11 | R10 | R9 | R8 |

**CMP0LR R245** (F5h) Read/Write

Compare 0 Register Low

Reset value: **00**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |

### 10.7.4 Compare 1 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 High Register

Reset value: **00**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| R15 | R14 | R13 | R12 | R11 | R10 | R9 | R8 |

**CMP1LR R247** (F7h) Read/Write

Compare 1 Low Register

Reset value: **00**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |

**REGISTER DESCRIPTION** (Cont'd)

### 10.7.5 Timer Control Register (TCR)

This register is used to control the Timer's status.

**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 generating an interrupt request and is automatically rest after being cleared. No effect when this bit is reset. This bit always returns "0" when read.

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 counting. Resetting this bit selects Down counting.

b2 = **UDCS**: *Up/Down Count status*. This bit is read only and monitors the direction of the counter. When the bit is set, the counter is using Up mode counting. When the bit is reset, 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. When the bit is reset, this indicates that the counter is halted.

### 10.7.6 Timer Mode Register (TMR)

This register is used to select the Timer's operating mode.

**TMR R249** (F9h) Read/Write

Timer Mode Register

Reset value: **0000 0000b (00h)**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| OE1 | OE0 | BM | RM1 | RM0 | ECK | REN | C0 |

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 high. 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 set. When the bit is reset, the Bivalue mode is disabled. After that, depending on the value of the RM0 bit (TMR - bit 3), the Biload or Bicapture mode will be 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, REG1R monitors the value of the counter. This bit has no effect when the Bivalue Mode is enabled.

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

b2 = **ECK**: *Timer clock source*. 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 the configuration of the IN0 - IN3 bits in ICR. When the ECK bit is set, different functions are performed depending on the number of the relevant timer. For Timer 0 and Timer 3, setting the ECK bit stops the counter.

**REGISTER DESCRIPTION** (Cont'd)

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 the RM0, RM1 and BM bits.

**Table 25. 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 |

### 10.7.7 External Input Control Register (ICR)

This register allows the function and operation of the TxINA and TxINB inputs to be programmed.

**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,IN1,IN0**: *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** (Cont'd)

### 10.7.8 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). Following a RESET condition, the prescaler is automatically loaded with 00h, 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 programmed in the PRSR register is equal to the divider value minus one. For example, loading PRSR with 24 causes the prescaler to divide by 25.

### 10.7.9 Output A Control Register (OACR)

This register selects the sources that can drive a TxOUTA pin:

– OVF/UNF, being an Overflow or Underflow event on the U/D counter,

– COMP0, being a successful compare event on the CMP0R register, and

– COMP1, being a successful compare event on the CMP1R register.

By programming bits B0 and B1 of the relevant source, one of the following actions may be applied to the TxOUTA pin on the occurrence of each of the events described above:

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

**Note**: Whenever more than one event occurs simultaneously, the action taken will be the result of 'ANDing' the B1-B0 fields. This register also allows the action of COMP0 on the on-chip event to be selected, via bit 1.

**OACR R252** (FCh) Read/Write

Output A Control Register

Reset value: **xxxx xx0xb**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| B0 | B1 | B0 | B1 | B0 | B1 | CEV | 0P |

< 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** (Cont'd)

**10.7.10 Output B Control Register (OBCR)**

This register selects the sources that can drive a TxOUTB pin:

– OVF/UNF, being an Overflow or Underflow event on the U/D counter,

– COMP0, being a successful compare event on the CMP0R register, and

– COMP1, being a successful compare event on the CMP1R register.

By programming bits B0 and B1 of the relevant source, one of the following actions may be applied to the TxOUTB pin on the occurrence of each of the events described above:

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

**Note**: Whenever more than one event occurs simultaneously, the action taken will be the result of 'ANDing' the B1-B0 fields. This register also allows the action of Overflow/Underflow on the on-chip event to be selected, via bit 1.

**OBCR R253** (FDh) Read/Write

Output B Control Register

Reset value: **xxxx xx0xb**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| B0 | B1 | B0 | B1 | B0 | B1 | OEV | 0P |

< COMP0 >< COMP1 ><OVF/UNF>

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

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

b3-b2 = **B0, B1**: *OVF/UNF Control bits*. 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**: *TxOUTB preset Control bit*. The value of this bit is the preset value of the TxOUTB output pin. Reading this bit returns the current state of the TxOUTA output pin (useful when this output is selected in toggle mode).

**10.7.11 Flag Register (FLAGR)**

This register contains the flags of the successful captures or comparisons, together with the Overflow/Underflow and Overrun indications. The Interrupt mode on capture can also 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, the user can see 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**: *Compare 0*. This bit is set after a successful compare on CMP0R register.

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

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

b2 = **OCP0**: *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**: *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** (Cont'd)

### 10.7.12 Interrupt/DMA Mask Register (IDMR)

This register contains the Global Timer Interrupt enable bit and the INT/DMA enable bits related to 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 |

b7 = **GTIEN**: *Global Timer Interrupt Enable* When this bit is set, all Timer Interrupts from enabled sources are enabled. When the bit is reset, all Timer Interrupts 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".

b2 = **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".

### 10.7.13 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 | REF MEM |

b7-b2 = **DCP7-DCP2**: *MSBs 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 tothe Compare 0 channel.

b1 = **DMA-SRCE**: *DMA Source selection.* 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 Memory.

**REGISTER DESCRIPTION** (Cont'd)

### 10.7.14 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 ) [R245 (F5h)] 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.* These bits contain the most significant bits of the DMA Address register location and are user programmable. Through user programmable, bit D2 may be hardware toggled if the Swap mode is set for the Timer DMA section related to Capture 0 channel.

Note: During a DMA transfer with the Register File, the DAPR is not used; however, in SWAP mode, DAPR(2) is used to point to the correct table.

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

b0 = **PRG/DAT:** *DMA memory selection.* This concept is meaningless on the ST9PLUS, where memory is all mapped in one address space. The bit allows compatibility with original ST9 software. When this bit is set, it selects the Source/Destination of the DMA area as Data Memory, while when it is reset, the Source/Destination of the DMA area is External Program Memory (according to the value of D0 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 |

### 10.7.15 Interrupt Vector Register (IVR)

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

Only one Interrupt Vector Register is available for each timer, and it is able to manage 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 within a group, the FLAGR register can be used to check the relevant interrupt source.

**IVR R242** (F2h) [R246 (F6h)] 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 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 memory locations (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 memory. They are fixed by hardware, depending on the group of sources which generated the interrupt request as follows:

.

| W1 | W0 | Interrupt Source |
|---|---|---|
| 0 | 0 | Overflow/Underflow even interrupt |
| 0 | 1 | Not available |
| 1 | 0 | Capture event interrupt |
| 1 | 1 | Compare event interrupt |

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

REGISTER DESCRIPTION (Cont'd)

### 10.7.16 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 also contains the two End Of Block bits.

**IDCR R243** (F3h) [R247 (F7h)] 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 Swap mode is disabled (SWEN bit = "0"), the CPE bit is set by hardware.

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 set by hardware.

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

b4 = **DCTD**: *DMA Compare Transfer Destination*. This bit selects the destination of the DMA operation related to the channel associated with Compare 0. When this bit is reset, the selected destination is the CMP0R register. When the bit is set, the I/O port is selected as the 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 - PL0:** *Interrupt/DMA priority level*. With these three bits it is possible to select the Interrupt and DMA priority level of each timer, as one of eight levels (see Interrupt/DMA chapter).

### 10.7.17 I/O Connection Register (IOCR)

This register allows the user to select an on-chip connection between input A and output A on the 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 Timer & and Timer 3 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 Timer 0 is made on-chip or externally (physically on pins):

SC0="0": TxOUTA and TxINA unconnected

SC0="1": TxOUTA and TxINA connected internally

# 11 STANDARD TIMER (STIM)

## 11.1 INTRODUCTION

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

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

– event counter,

– gated external input mode,

– triggerable input mode,

– retriggerable input mode.

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

The Standard Timer is composed of a 16-bit down counter with an 8-bit prescaler. The input clock to the prescaler can be driven either by an internal clock equal to INTCLK divided by 4, or by CLOCK2 derived directly from the external oscillator, divided by 64 or 128, thus providing a stable time reference independent from the PLL programming.

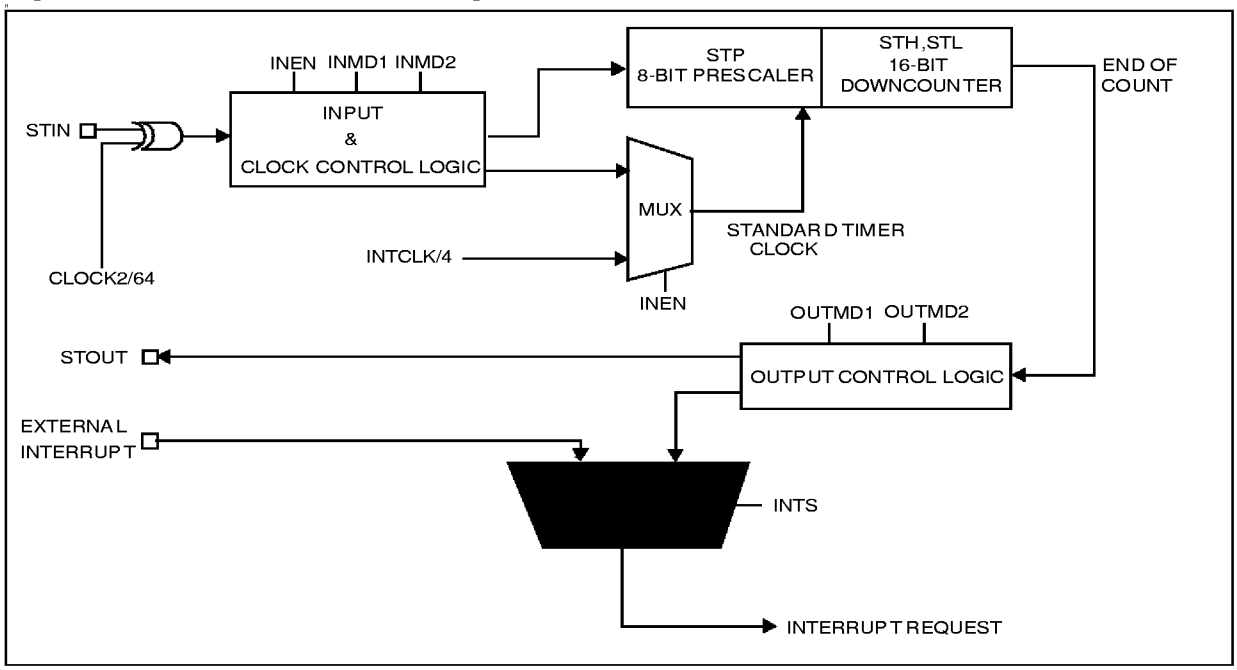
The Standard Timer End Of Count condition is able to generate an interrupt which is connected to one of the external interrupt channels.

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

### 11.1.1 Device-Specific Options

Depending on the ST9 variant and package type, some STIM interface signals described in this chapter may not be connected to an external pin. Refer to the signal availability table in the Peripheral Configuration section.

**Figure 10. Standard Timer Block Diagram**



## 11.2 STANDARD TIMER FUNCTIONS

### 11.2.1 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 Standard Timer to start counting at the beginning of the next instruction. Resetting this bit will stop the counter.

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

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

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

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

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

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

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

### 11.2.2 Standard Timer Input Modes (ST9 devices with Standard Timer Input STIN)

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

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

The Standard Timer is driven by the signal applied to the input pin (STIN) which acts as an external clock. The unit works therefore as an event counter. The event is a high to low transition on STIN. Spacing between trailing edges should be at least the period of INTCLK multiplied by 8 (i.e. the maximum Standard Timer input frequency is 2.5 MHz with INTCLK = 20MHz).

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

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

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

The Standard Timer is started by:

a) setting the Start-Stop bit, AND

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

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

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

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

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

For devices where STIN is replaced by a connection to CLOCK2/64, the condition (INMD1 = "0", INMD2 = "0") will allow the Standard Timer to generate a stable time base independent from the PLL programming.

**STANDARD TIMER FUNCTIONS (Cont'd)**

**11.2.4 Standard 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 Standard 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 Standard Timer toggles the state of the STOUT 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.

**PWM Output Mode** (OUTMD1 = "1")

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

**11.2.5 Interrupt Selection**

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

The STCR bit 2 (INTS) selects the interrupt source between the Standard Timer interrupt and the external interrupt pin. Thus the Standard Timer Interrupt uses the interrupt channel and takes the priority and vector of the external interrupt channel.

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

**Note:** When disabling the Standard Timer Interrupt (writing INTS=1 in the STCR register) a rising edge may be generated on the interrupt channel, causing an unwanted interrupt.

### 11.3 STANDARD TIMER REGISTERS

The ST9 can have up to 4 Standard Timers.

Each Standard Timer has 4 registers mapped into page 0Bh in Group F of the Register File

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

**STH R240** (F0h) Page 0B Read/Write

Counter High Byte Register

Reset value: 1111 1111b (FFh)

| 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: 1111 1111b (FFh)

| 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 Standard 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

Standard Timer Prescaler Register

Reset value: 1111 1111b (FFh)

| 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 Standard Timer is programmed into this register. When reading the STP register, the returned value corresponds to the programmed data instead of the current data.
00h: No prescaler
01h: Divide by 2
FFh: Divide by 256

**STC R243** (F3h) Page 0B Read/Write

Standard Timer Control Register

Reset value: 0001 0100b (14h)

| 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 Standard Timer.

b6 = **S-C**: Single-Continuous Mode Select. Setting S-C to "1" sets the Standard 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 Section 11.2.2, when enabled by INEN.

b3 = **INEN**: Input Enable.

If the STIN pin is not present, INEN must be 0.
0: Input section disabled.
1: Input section enabled.

b2 = **INTS**: Interrupt Selection.
0: Standard Timer interrupt enabled
1: Standard Timer interrupt disabled (reset status).

b1-b0 = **OUTMD1, OUTMD2** Output Mode Selection. These bits select the output functions as described in Section 11.2.4.

# 12 SERIAL PERIPHERAL INTERFACE (SPI)

## 12.1 INTRODUCTION

The Serial Peripheral Interface (SPI) is a general purpose on-chip shift register peripheral. It allows communication with external peripherals via an SPI protocol bus.

In addition, special operating modes allow reduced software overhead when implementing $I^2C$-bus and IM-bus communication standards.

The SPI uses up to 3 pins: Serial Data In (SDI), Serial Data Out (SDO) and Synchronous Serial Clock (SCK). Additional I/O pins may act as device selects or IM-bus address identifier signals.

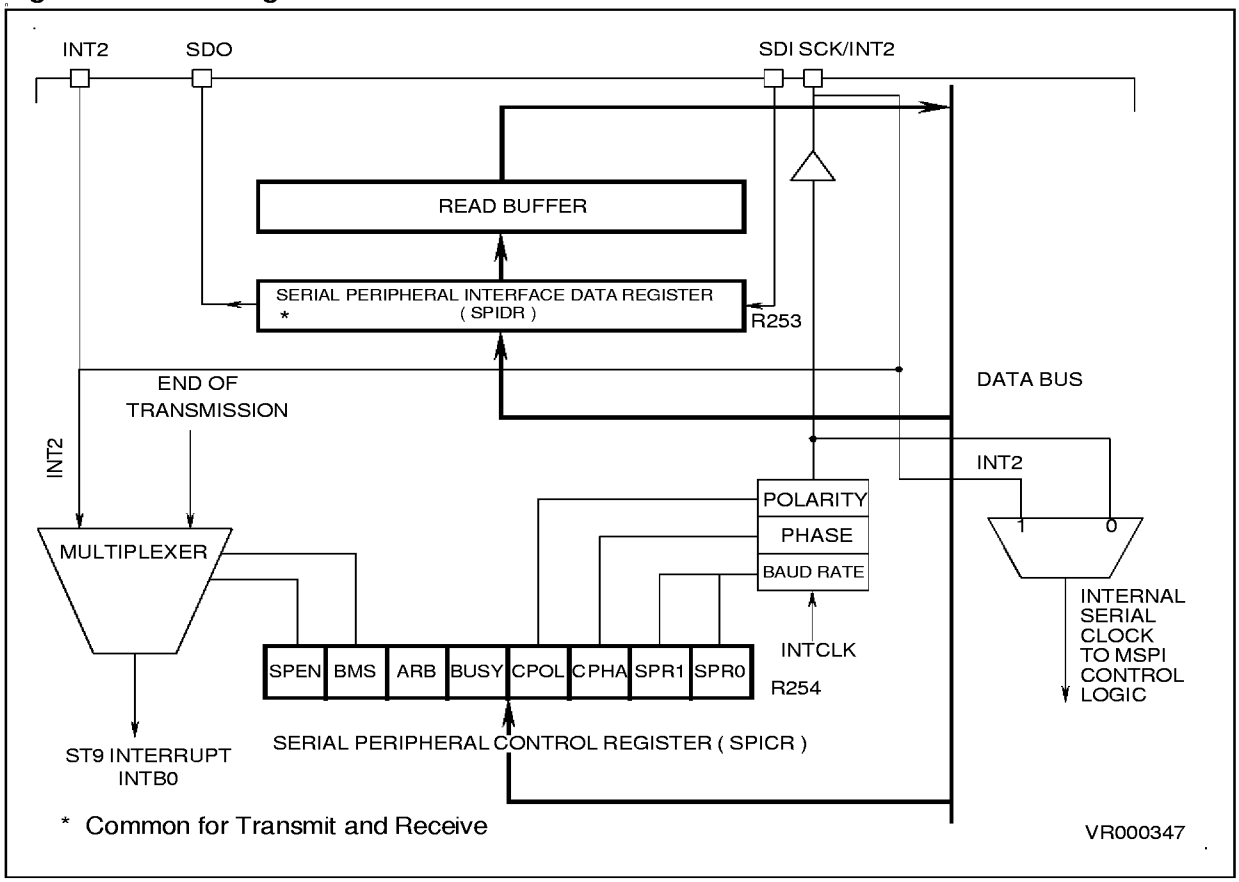
The main features are:

- Full duplex synchronous transfer if 3 I/O pins are used
- Master operation only
- 4 Programmable bit rates
- Programmable clock polarity and phase
- Busy Flag
- End of transmission interrupt
- Additional hardware to facilitate more complex protocols

### 12.1.1 DEVICE-SPECIFIC OPTIONS

Depending on the ST9 variant and package type, the SPI interface signals may not be connected to separate external pins. Refer to the Peripheral Configuration Chapter for the device pin-out.

**Figure 11. Block Diagram**



## 12.2 FUNCTIONAL DESCRIPTION

The SPI, when enabled, receives input data from the internal data bus to the SPI Data Register (SPIDR). A Serial Clock (SCK) is generated by controlling through software two bits in the SPI Control Register (SPICR). The data is parallel loaded into the 8 bit shift register during a write cycle. This is shifted out serially via the SDO pin, MSB first, to the slave device, which responds by sending its data to the master device via the SDI pin. This implies full duplex transmission if 3 I/O pins are used with both the data-out and data-in synchronized with the same clock signal, SCK. Thus the transmitted byte is replaced by the received byte, eliminating the need for separate "Tx empty" and "Rx full" status bits.

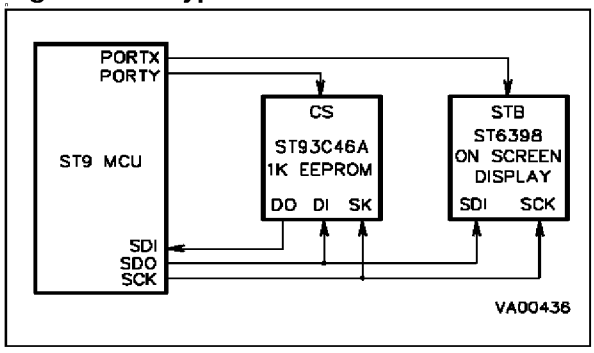
When the shift register is loaded, data is parallel transferred to the read buffer and becomes available to the CPU during a subsequent read cycle.

The SPI requires three I/O port pins:

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

An additional I/O port output bit may be used as a slave chip select signal. Data and Clock pins I C Bus protocol are open-drain to allow arbitration and multiplexing.

**Figure 12. A Typical SPI Network**




### 12.2.1 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/$I^2C$-bus configuration, the SDI line senses the value forced on the data line (by SDO or by another peripheral connected to the S-bus/$I^2C$-bus).

### 12.2.2 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. 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.

During an S-Bus or $I^2C$-Bus protocol, the SDO pin is forced to high impedance when the SPI is disabled. When arbitration is lost, SDO is set to one.

**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 the SCK clock from INTCLK. The SCK clock is used to synchronize data transfer, both in to and out of the device, through its SDI and SDO pins. The SCK clock type, and its relationship with data is controlled by the CPOL (Clock Polarity) and CPHA (Clock Phase) bits in the Serial Peripheral Control Register (SPICR). This input is provided with a digital filter which eliminates spikes lasting less than one INTCLK period.

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

### 12.3 INTERRUPT STRUCTURE

The SPI peripheral is associated with external interrupt channel B0 (pin INT2). Multiplexing between the external pin and the SPI internal source is controlled by the SPEN and BMS bits, as shown in Table 26 Interrupt Configuration.

The two possible SPI interrupt sources are:

– End of transmission (after each byte).

– S-bus/$I^2C$-bus start or stop condition.

Care should be taken when toggling the SPEN and/or BMS bits from the "0,0" condition. Before changing the interrupt source from the external pin to the internal function, the B0 interrupt channel should be masked. EIMR.2 (External Interrupt Mask Register, bit 2, IMBO) and EIPR.2 (External Interrupt Pending Register bit 2, IMP0) should be "0" before changing the source. This sequence of events is to avoid the generating and reading of spurious interrupts.

A delay instruction lasting at least 4 clock cycles (e.g. 2 NOPs) should be inserted between the SPEN toggle instruction and the Interrupt Pending bit reset instruction.

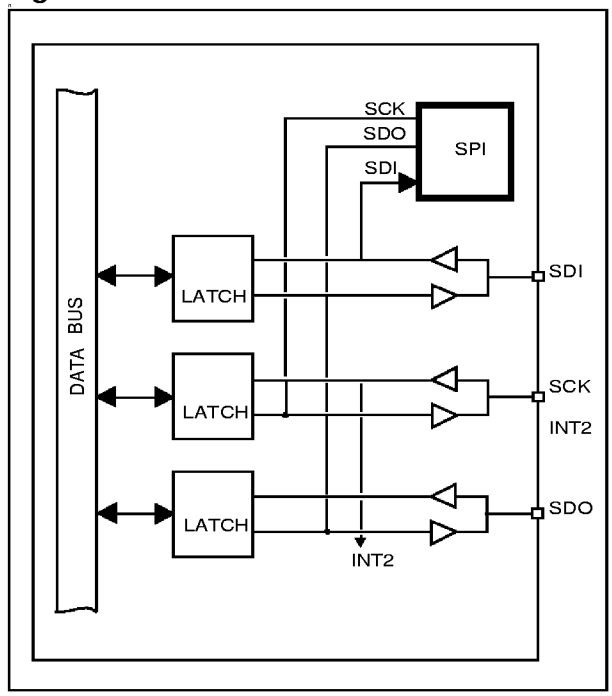
The INT2 input Function is always mapped together with the SCK input Function, to allow Start/Stop bit detection when using S-bus/$I^2C$-bus protocols.

A start condition occurs when SDI goes from "1" to "0" and SCK is "1". The Stop condition occurs when SDI goes from "0" to "1" and SCK is "1". For both Stop and Start conditions, SPEN = "0" and BMS = "1".

**Table 26. Interrupt Configuration**

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

**Figure 13. SPI I/O Pins**



### 12.4 SPI REGISTERS

The SPI uses two registers, mapped in page 0 of the register file:

**SPIDR R253** (FDh) Page 0 Read/Write

SPI Data Register

Reset Value: undefined

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

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

Note: SPIDR state remains undefined until the end of transmission of the first byte.

**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 tristate. Furthermore, SPEN affects the selection of the source for interrupt channel B0. Transmission starts data is written to the SPIDR Register.

b6 = **BMS**: *S-bus/I²C-bus Mode Selector.*
This bit should be set when the SPI is used in an S-bus/I²C-bus protocol; this enables S-bus/I²C-bus arbitration, clock synchronization and Start/Stop detection. When the bit is reset, a re-initialisation of the SPI logic is performed, thus allowing recovery procedures after a RX/TX failure. BMS (and SPEN) affect the selection of the source for interrupt channel B0.

b5 = **ARB**: *Arbitration flag bit.*
This bit is set when the SPI loses arbitration in S-bus/I²C-bus mode, 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 a high value until a write instruction on SPIDR is performed.

b4 = **BUSY**: *SPI Busy Flag.*
The BUSY flag is set when a transmission is in progress. 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 to $V_{CC}$ through a resistor, depending on the CPOL state. Polarity should be set during the initialisation routine, in accordance with the setting of all peripherals, and should 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 signal on the SCK pin. The CPHA bit selects the clock edge used to capture data. 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 15 shows the relationship between CPHA, CPOL and SCK, and indicates active clock edges and strobe times.

| CPOL | CPHA | SCK (in Figure 15) |
|---|---|---|
| 0 | 0 | (a) |
| 0 | 1 | (b) |
| 1 | 0 | (c) |
| 1 | 1 | (d) |

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

| SPR 1 | SPR 0 | 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) |

### 12.5 WORKING WITH OTHER PROTOCOLS

The SPI peripheral offers the following facilities for operation with S-bus/$I^2C$-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 with 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 on how to manage these protocols.

#### 12.5.1 $I^2C$-bus Interface

The $I^2C$-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 16, data is toggled with clock low. An I C bus start condition is the transition on SDI from 1 to 0 with the SCK held high. In a stop condition, the SCK is also high and the transition on SDI is from 0 to 1. During both of these conditions, SPEN= 0 and BMS = 1.

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

**Figure 14. S-Bus / $I^2C$-bus Peripheral Compatibility without S-Bus Chip Select**

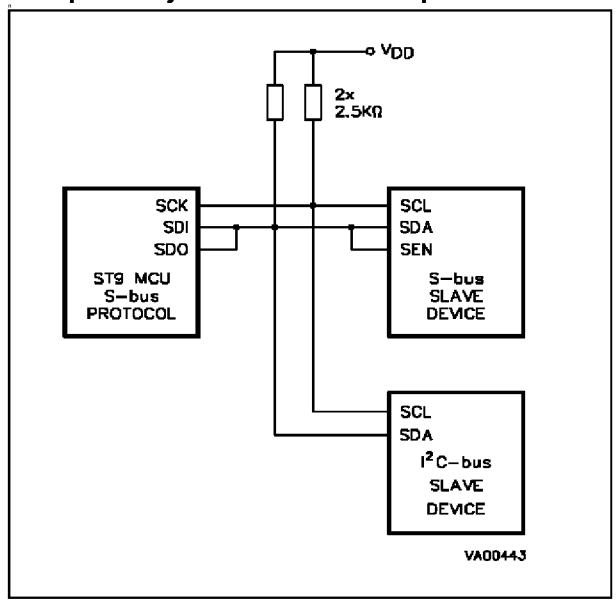
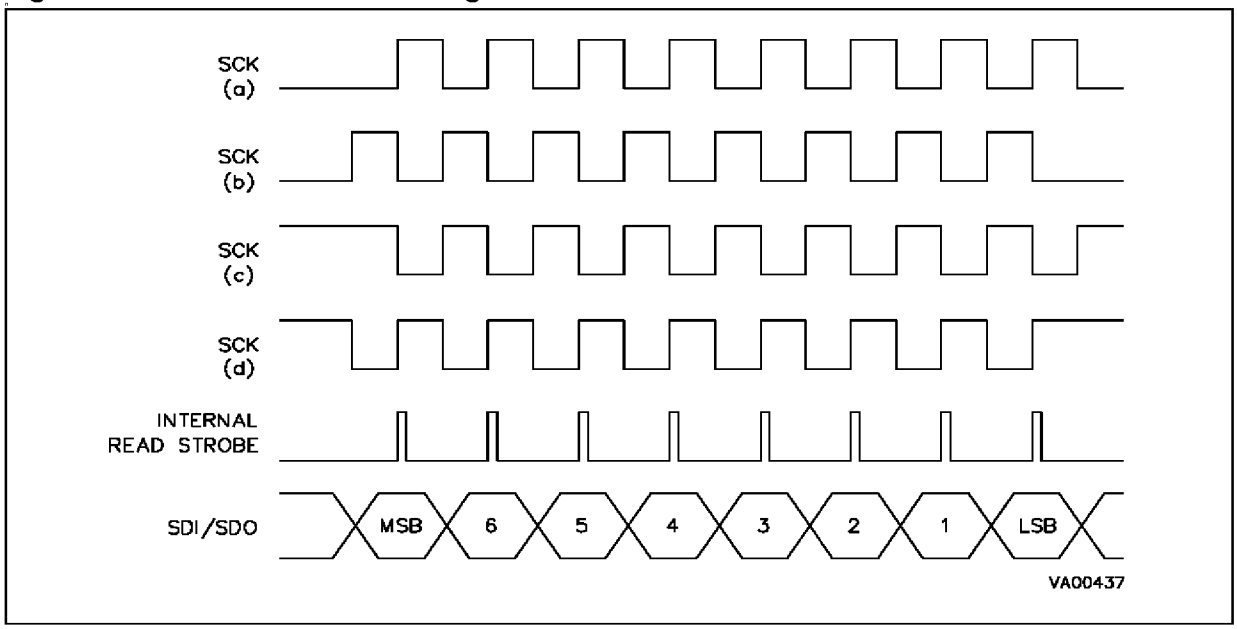



**Figure 15. SPI Data and Clock Timing**

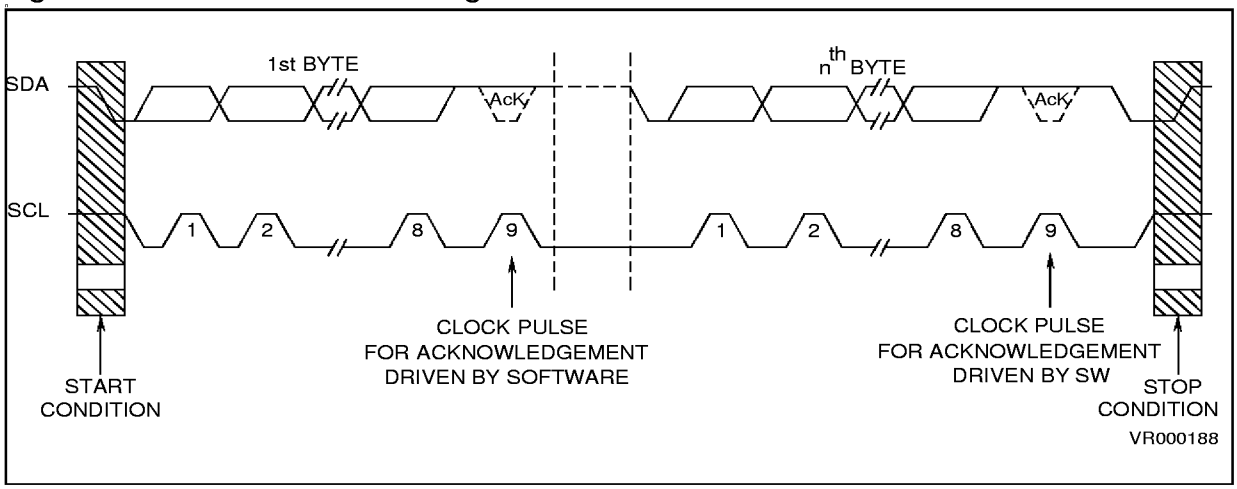


**WORKING WITH OTHER PROTOCOLS**(Continued)

**Table 27. Typical $I^2C$-bus Sequences**

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

**Figure 16. SPI Data and Clock Timing**



**WORKING WITH OTHER PROTOCOLS**(Continued)

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

**SPI working with an I2C-bus**

To use the SPI with the I²C-bus protocol, the SCK line is used as SCL; the SDI and SDO lines, externally wire-ORed, are used as SDA. All output pins must be configured as open drain (seeFigure 14).

Figure 27 illustrates the typical I²C-bus sequence, comprising 5 phases: Initialization, Start, Transmission, Acknowledge and Stop. It should be noted that only the first 8 bits are handled by the SPI peripheral; the ACKNOWLEDGE bit must be managed by software, by polling or forcing the SCL and SDO lines via the corresponding I/O port bits.

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

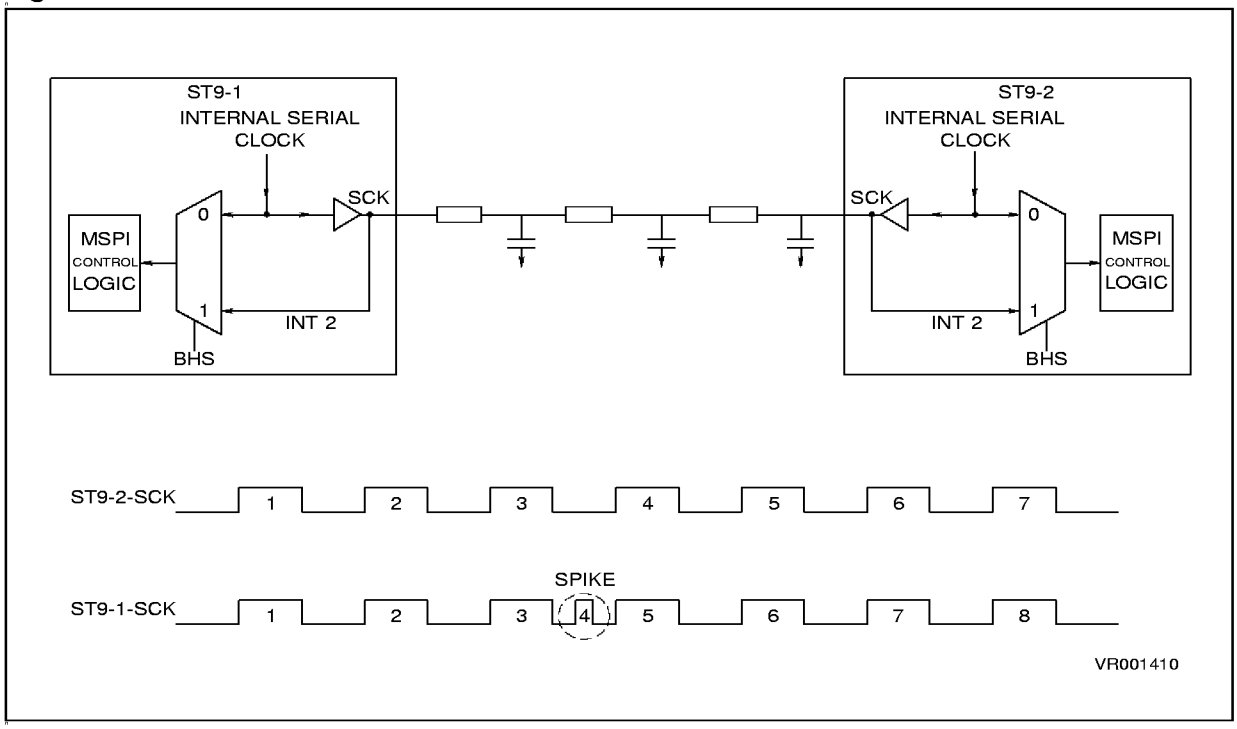
**Clock Synchronization**

In a multimaster I²C-bus system, when several masters generate their own clock, synchronization is required. 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 manner, devices using different clock sources and different frequencies can be interfaced.

**Arbitration Lost**

When several masters are sending data on the SDA line, the following takes place: if the transmitter sends a “1” and the SDA line is forced low by another device, the ARB flag (SPICR.5) is set and the SDO buffer is disabled (ARB is reset and the SDO buffer is enabled when SPIDR is written to again). When BMS is set, 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 17, if device ST9-1 detects a noise spike and therefore gains a clock pulse, it will stop its transmission early and hold the clock line low, causing device ST9-2 to freeze on the 7th bit. To exit and recover from this condition, the BMS bit must be reset; this will cause the SPI logic to be reset, thus aborting the current transmission. An End of Transmission interrupt is generated following this reset sequence.

**Figure 17. SPI Arbitration**



**DIFFERENT PROTOCOLS**(Continued)

### 12.5.2 S-Bus Interface

The S-bus is a three-wire bidirectional data-bus, possessing functional features similar to the I²C-bus. As opposed to the I²C-bus, the Start/Stop conditions are determined by encoding the information on 3 wires rather than on 2, as shown in Figure 18. The additional line is referred as SEN.

### SPI Working with S-bus

The S-bus protocol uses the same pin configuration as the I²C-bus for generating the SCL and SDA lines. The additional SEN line is managed through a standard ST9 I/O port line, under software control (see Figure 14).

**Figure 18. S-bus Configuration**

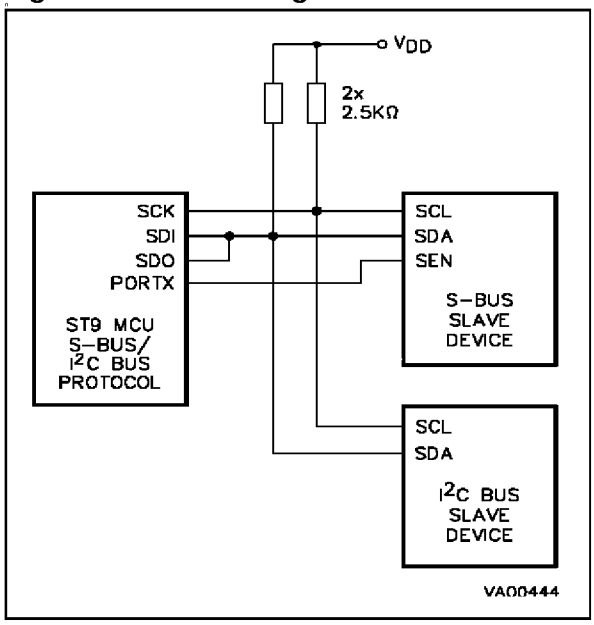



**Figure 19. Mixed S-bus and I²C-bus System**

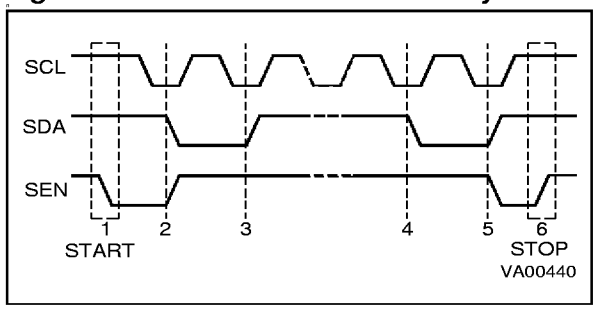
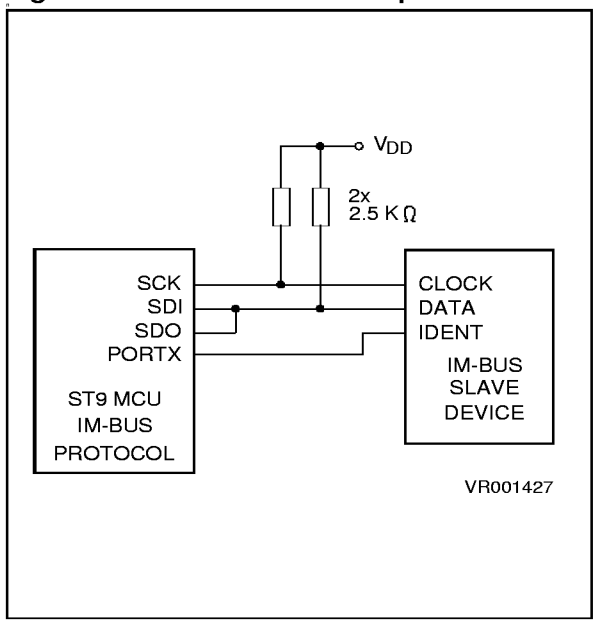



**Figure 20. ST9 and IM-bus Peripheral**



**DIFFERENT PROTOCOLS**(Continued)

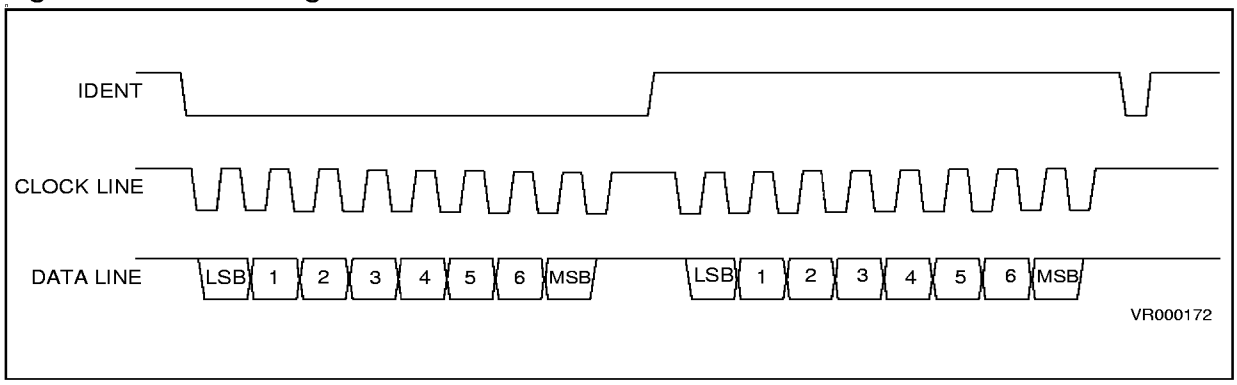
### 12.5.3 IM-bus Interface

The IM-bus features a bidirectional data line and a clock line; in addition, it requires an IDENT line to distinguish an address byte from a data byte (Figure 21). Unlike the $I^2C$-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 20 shows the connections between an IM-bus peripheral and an ST9 SPI. The SDO and SDI pins are connected to the bidirectional data pin of the peripheral device. The SDO alternate function is configured as 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 the SPIDR register. To receive data from the peripheral, the user should write FFh to the SPIDR register, 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 "1" do not affect the SDO/SDI line status (which defaults to a high level due to the FFh value 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 by setting both CPOL and CPHA to "1". By using a general purpose I/O as the IDENT line, and forcing it to a logical "0" when writing to the SPIDR register, an address is sent (or read). Then, by setting this bit to "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. This will generate the stop condition.

**Figure 21. IM bus Timing**



# 13 SERIAL COMMUNICATIONS INTERFACE (SCI)

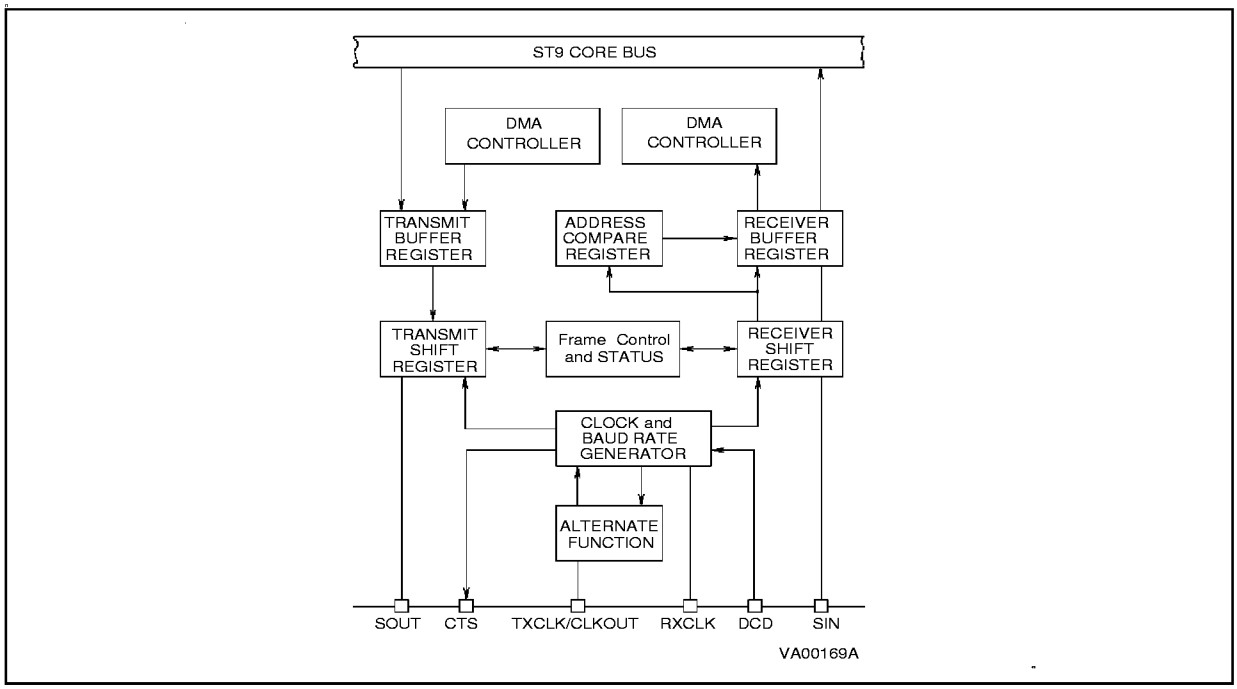
## 13.1 INTRODUCTION

The Serial Communications Interface (SCI) offers full-duplex serial data exchange with a wide range of external equipment. The SCI offers four operating modes: Asynchronous, Asynchronous with synchrounous clock, Serial expansion and Synchronous.

The SCI offers the following principal features:

- Full duplex synchronous and asynchronous operation.
- Transmit, receive, line status, and device address interrupt generation.
- Integral Baud Rate Generator capable of dividing the input clock by any value from 2 to $2^{16}$-1 (16 bit word) and generating the internal 16 X data sampling clock for asynchronous operation or the 1X clock for synchronous operation.
- Fully programmable serial interface :
  - 5, 6, 7, or 8 bit word length.
  - Even, odd, or no parity generation and detection.
  - 0, 1, 1-1/2, 2, 2-1/2 stop bit generation.
  - False start bit detection.
  - Complete status reporting capabilities.
  - Line break generation and detection.
- Programmable address indication bit (wake-up bit) and user invisible compare logic to support multiple microcomputer networking. Optional character search function.
- Internal diagnostic capabilities:
  - Local loopback for communications link fault isolation.
  - Auto-echo for communications link fault isolation.
- Separate interrupt/DMA channels for transmit and receive.
- In addition, a Synchronous mode supports:
  - High speed communication
  - Possibility of hardware synchronization (CTS/DCD signals).
  - Programmable polarity and stand-by level for data SIN/SOUT.
  - Programmable active edge and stand-by level for clocks CLKOUT/RXCL.
  - Programmable active levels of CTS/DCD signals.
  - Full Loop-Back and Auto-Echo modes for DATA, CLOCKs and CONTROLs.

**Figure 22. SCI Block Diagram**



## 13.2 FUNCTIONAL DESCRIPTION

The SCI offers four operating modes:

– Asynchronous mode

– Asynchronous mode with synchronous clock

– Serial expansion mode

– Synchronous mode

Asynchronous mode, Asynchronous mode with synchronous clock and Serial expansion mode output data with the same serial frame format. The differences lie in the data sampling clock rates (1X, 16X) and in the operating modes.

### 13.2.1 SCI Operating Modes

#### 13.2.1.1 Asynchronous Mode

In this mode, data and clock can be asynchronous (the transmitter and receiver can use their own clocks to sample received data), each data bit is sampled 16 times per clock period.

The baud rate clock should be set to the ÷16 Mode and the frequency of the input clock (from an external source or from the internal baud-rate generator output) is set to suit.

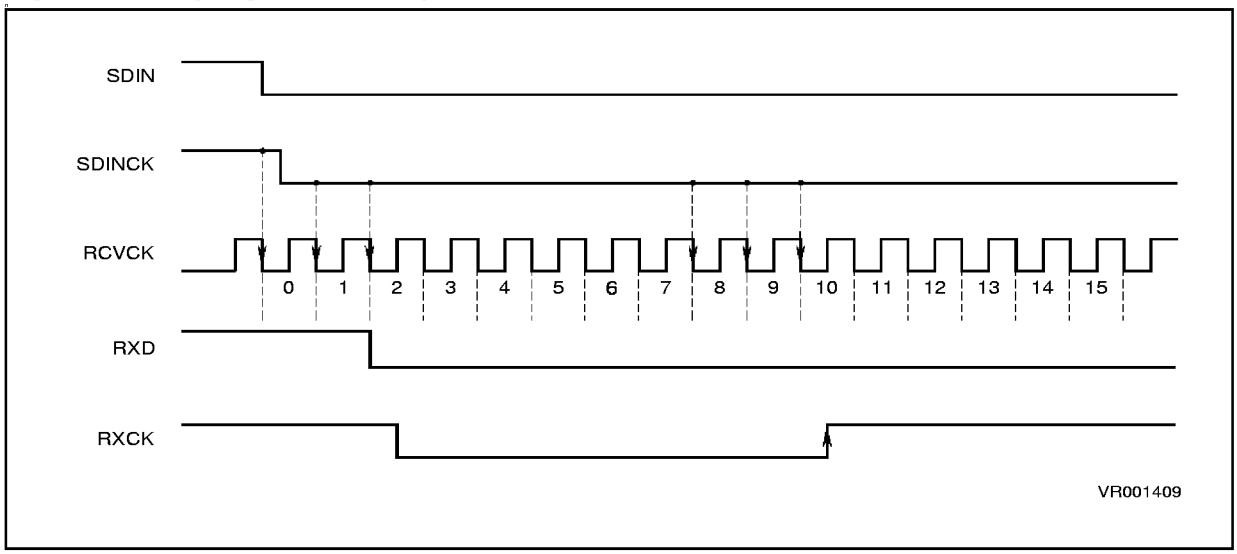
#### 13.2.1.2 Asynchronous Mode with Synchronous Clock

In this mode, data and clock are synchronous, each data bit is sampled once per clock period.

For transmit operation, a general purpose I/O port pin can be programmed to output the CLKOUT signal from the baud rate generator. If the SCI is provided with an external transmission clock source, there will be a skew equivalent to two INTCLK periods between clock and data.

Data will be transmitted on the falling edge of the transmit clock. Received data will be latched into the SCI on the rising edge of the receive clock.

**Figure 23. Sampling Times in Asynchronous Format**



**FUNCTIONAL DESCRIPTION** (Cont'd)

**13.2.1.3 Serial Expansion Mode**

This mode is used to communicate with an external synchronous peripheral.

The transmitter only provides the clock waveform during the period that data is being transmitted on the CLKOUT pin (the Data Envelope). Data is latched on the rising edge of this clock.

Whenever the SCI is to receive data in serial port expansion mode, the clock must be supplied externally, and be synchronous with the transmitted data. The SCI latches the incoming data on the rising edge of the received clock, which is input on the RXCLK pin.

**13.2.1.4 Synchronous Mode**

This mode is used to access an external synchronous peripheral, dummy start/stops bits are not included in the data frame. Polarity, stand-by level and active edges of I/O signals are fully and separately programmable for both inputs and outputs.

It's necessary to set the SMEN bit of the Synchronous Input Control Register (SICR) to enable this mode and all the related extra features (otherwise disabled).

The transmitter will provide the clock waveform only during the period when the data is being transmitted via the CLKOUT pin, which can be enabled by setting both the XTCLK and OCLK bits of the Clock Configuration Register. Whenever the SCI is to receive data in synchronous mode, the clock waveform must be supplied externally via the RXCLK pin and be synchronous with the data. For correct receiver operation, the XRX bit of the Clock Configuration Register must be set.

Two external signals, Clear-To-Send and Data-Carrier-Detect (CTS/DCD), can be enabled to synchronise the data exchange between two serial units. The CTS output becomes active just before the first active edge of CLKOUT and indicates to the target device that the MCU is about to send a synchronous frame; it returns to its stand-by state following the last active edge of CLKOUT (MSB transmitted).

The DCD input can be considered as a gate that filters RXCLK and informs the MCU that an transmitting device is transmitting a data frame. Polarity of CTS/DCD is individually programmable, as for clocks and data.

Loop-Back and Auto-Echo modes are available for all I/O data, clocks and control signals; as in asyncronous modes, they can be enabled by setting the AEN/LBEN bits of the Clock Configuration Register.

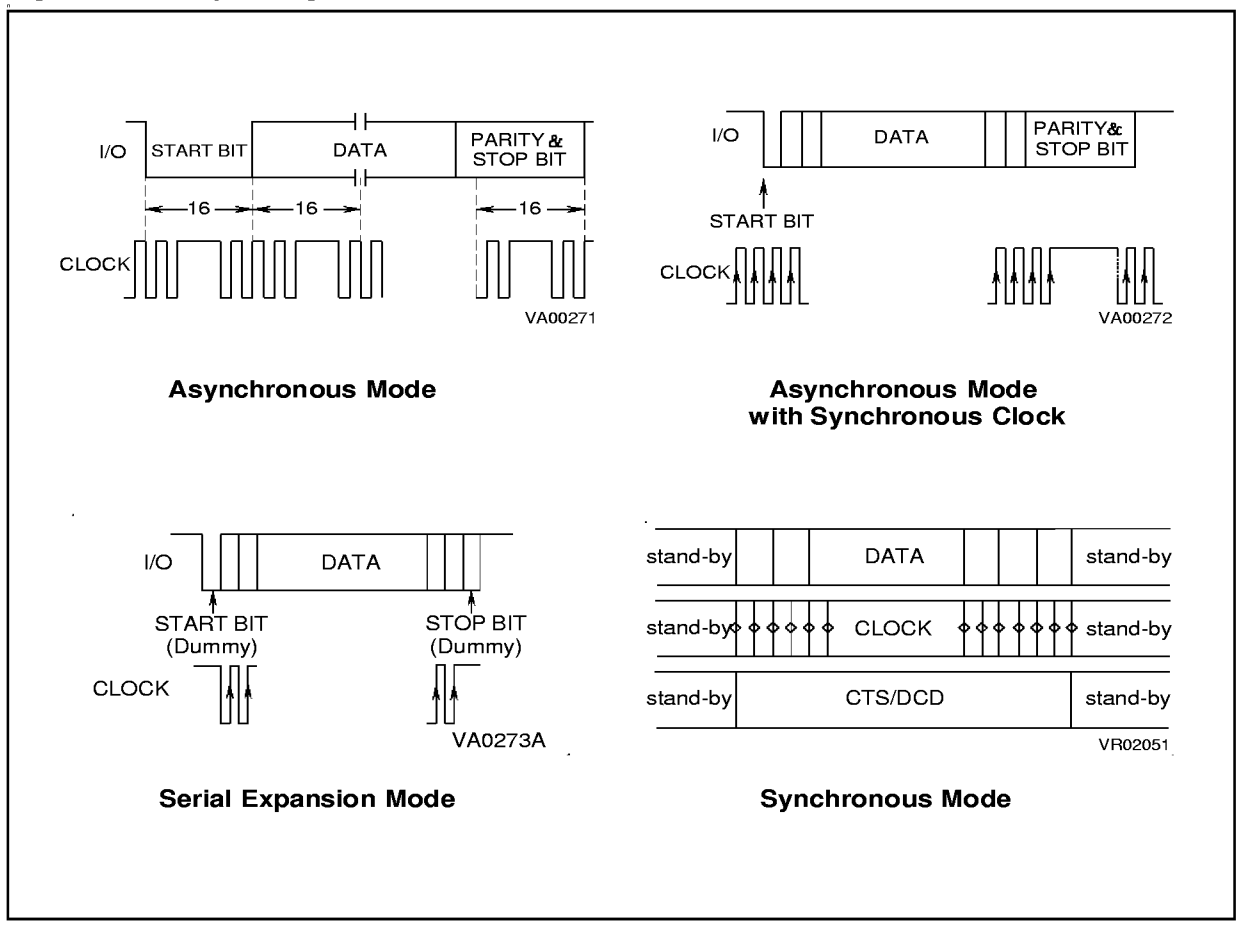
In Synchronous Mode, when AEN is set, the transmitter outputs (data, clock and control) are disconnected from the I/O pins, which are driven directly by the receiver input pins. When LBEN is set, the receiver inputs (data,clock and controls) are disconnected and the transmitter outputs are looped-back into the receiver section. SOUT, CKOUT and CTS will be set to their programmed stand-by levels and the status of the INPL, XCKPL, DCDPL, OUTPL, OCKPL and CTSPL bits in the SICR register are irrelevant .

The data word is programmable from 5 to 8 bits, as for the other modes; parity, address/9th, stop bits and break cannot be inserted into the transmitted frame. Programming of the related bits of the SCI control registers are irrelevant in Synchronous Mode: all the corresponding interrupt requests must, in any case, be masked in order to avoid incorrect operation during data reception.

**FUNCTIONAL DESCRIPTION** (Cont'd)

As a direct consequence of the data format, the only Address Interrupt Mode available in Synchronous Mode is 'Character Search'. This is a powerful feature which can generate an interrupt on receiving a predetermined character (Character Match Interrupt). To enable this function, the AMEN and AM bits must be reset and set, respectively ("01").

**Figure 24. SCI Operating Modes**



**FUNCTIONAL DESCRIPTION** (Cont'd)

### 13.2.2 Serial Frame Format

Characters sent or received by the SCI can have some or all of the features in the following format, depending on the operating mode:

**START**: the START bit indicates the beginning of a data frame in Asynchronous mode. The START condition is detected as a high to low transition. A dummy START bit is generated in Serial Expansion mode.

**DATA**: the DATA word length is programmable from 5 to 8 bits, for both Synchronous and Asynchronous modes.

**PARITY**: The Parity Bit (not available in Synchronous mode) is optional, and can be used with any word length. It is used for error checking and is set so as to make the total number of high bits in DATA plus PARITY odd or even, depending on the number of "1"s in the DATA field.

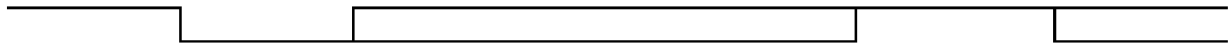
**ADDRESS/9TH**: The Address/9th Bit is optional and may be added to any word format. It is used in both Serial Expansion and Asynchronous modes to indicate that the data is an address (bit set).

The ADDRESS/9TH bit is useful when several microcontrollers are exchanging data on the same serial bus. Individual microcontrollers can stay idle on the serial bus, waiting for a transmitted address. When a microcontroller recognizes its own address, it can begin Data Reception, likewise, on the transmit side, the microcontroller can transmit another address to begin communication with a different microcontroller.

The ADDRESS/9TH bit can be used as an additional data bit or to mark control words (9th bit).

**STOP**: Indicates the end of a data frame in Asynchronous mode. A dummy STOP bit is generated in Serial Expansion mode. The STOP bit can be programmed to be 0, 1, 1.5, 2, 2.5 or 3 bits long. It returns the SCI to the quiescent marking state (i.e., a constant high-state condition) which lasts until a new start bit indicates an incoming word.

**Figure 25. SCI Character Formats**



| | START(2) | DATA(1) | PARITY(2) | ADDRESS(2) | STOP(2) | |
|---|---|---|---|---|---|---|
| **# bits** | 0, 1 | 5, 6, 7, 8 | 0, 1 | 0, 1 | 0, 1, 1.5, 2, 2.5, 1, 2, 3 | 16X 1X |
| **states** | | | NONE ODD EVEN | ON OFF | | |

(1) LSB First

(2) Not available in Synchronous mode

**FUNCTIONAL DESCRIPTION** (Cont'd)

**13.2.2.1 Data transfer**

Data to be transmitted by the SCI is first loaded by the program into the Transmitter Buffer Register. The SCI will transfer the data into the Transmitter Shift Register when the Shift Register becomes available (empty). The Transmitter Shift Register converts the parallel data into serial format for transmission via the SCI Alternate Function output, Serial Data Out. On completion of the transfer, the transmitter buffer register interrupt pending bit will be updated. If the selected word length is less than 8 bits, the unused most significant bits do not need to be defined.

Incoming serial data from the Serial Data Input pin is converted into parallel format by the Receiver Shift Register. At the end of the input rdata frame, the valid data portion of the received word is transferred from the Receiver Shift Register into the Receiver Buffer Register. All Receiver interrupt conditions are updated at the time of transfer. If the selected character format is less than 8 bits, the unused most significant bits will be set.

The Frame Control and Status block creates and checks the character configuration (Data length and number of Stop bits), as well as the source of the transmitter/receiver clock.

The internal Baud Rate Generator contains a programmable divide by "N" counter which can be used to generate the clocks for the transmitter and/or receiver. The baud rate generator can use INTCLK or the Receiver clock input via RXCLK.

The Address bit/D9 is optional and may be added to any word in Asynchronous and Serial Expansion modes. It is commonly used in network or machine control applications. When enabled (AB set), an address or ninth data bit can be added to a transmitted word by setting the Set Address bit (SA). This is then appended to the next word entered into the (empty) Transmitter Buffer Register and then cleared by hardware. On character input, a set Address Bit can indicate that the data preceeding the bit is an address which may be compared in hardware with the value in the Address Compare Register (ACR) to generate an Address Match interrupt when equal.

The Address bit and Address Comparison Register can also be combined to generate four differnt types of Address Interrupt to suit different protocols, based on the status of the Address Mode Enable bit (AMEN) and the Address Mode bit (AM) in the CHCR register.

The character match Address Interrupt mode may be used as a powerful character search mode, generating an interrupt on reception of a predetermined character e.g. Carriage Return or End of Block codes.

The Line Break condition is fully supported for both transmission and reception. Line Break is sent by setting the SB bit (IDPR). This causes the transmitter output to be held low (after all buffered data has been transmitted) for a minimum of one complete word length and until the SB bit is Reset.

Testing of the communications channel may be performed using the built-in facilities of the SCI peripheral. Auto-Echo mode and Loop-Back mode may be used individually or together. In Asynchronous , Asynchronous with Synchronous Clock and Serial Expansion modes they are available only on SIN/SOUT pins through the programming of AEN/LBEN bits in CCR. In Synchronous mode (SMEN set) the above configurations are available on SIN/SOUT, RXCLK/CLKOUT and DCD/CTS pins by programming the AEN/LBEN bits. In this case, the I/O pins and the receiver/transmitter sections will be connected together, depending on the status of AEN/LBEN, but independently of the programmed polarity (in Auto-Echo mode SOUT=SIN, CLKOUT=RXCLK and CTS=DCD, even if they act on the internal receiver with the programmed polarity/edge; in Loop-Back mode SIN=SOUT, RXCLK=CLKOUT, DCD=CTS and the output pins are locked to their programmed stand-by level ) .

**Table 28. Address Interrupt Modes**

| |
|---|
| If 9th Data Bit is set (1) |
| If Character Match |
| If Character Match and 9th Data Bit is set(1) |
| If Character Match Immediately Follows BREAK (1) |

(1) Not available in Synchronous mode

**FUNCTIONAL DESCRIPTION** (Cont'd)

### 13.2.3 Clocks And Serial Transmission Rates

The communication bit rate of the SCI transmitter and receiver sections can be provided from the internal Baud Rate Generator or from external sources. The bit rate clock is divided by 16 in Asynchronous mode (CD in CCR reset), or undivided in Synchronous mode (CD set). With INTCLK running at 20MHz, or with a 10MHz external clock, a maximum bit rate of 5MBaud is available in undivided mode and 625KBaud or 312.5KBaud respectively in divided by 16 mode.

**External Clock Sources.** The External Clock input pin TXCLK may be programmed by the TXCLK and OCLK bits in the CCR register as: the transmit clock input, Baud Rate Generator output (allowing an external divider circuit to provide the receive clock for split rate transmit and receive), or as CLKOUT output in Synchronous and Serial Expansion modes. The RXCLK Receive clock input is enabled by the XRX bit, this input should be set in accordance with the setting of the CD bit.

**Baud Rate Generator.** The internal Baud Rate Generator consists of a 16-bit programmable divide by "N" counter which can be used to generate the transmitter and/or receiver clocks. The minimum baud rate divisor is 2 and the maximum divisor is $2^{16}$-1. After initialising the baud rate generator, the divisor value is immediately loaded into the counter. This prevents potentially long random counts on the initial load.
The Baud Rate generator frequency is equal to the Input Clock frequency divided by the Divisor value.

***WARNING:*** *Programming the baud rate divider to 0 or 1 will stop the divider.*

The output of the baud generator has aprecise 50% duty cycle. The Baud Rate generator can use INTCLK for the input clock source. In this case, INTCLK (and therefore the MCU Xtal) should be chosen to provide a suitable frequency for division by the Baud Rate Generator to give the required transmit and receive bit rates. Suitable INTCLK frequencies and the respective divider values for standard Baud rates are shown inTable 29 Practical Example of SCI Baud Rate Generator Divider Values.

### 13.2.4 SCI Initialization Procedure

Writing to either of the two Baud Rate Generator Registers immediatly disables and resets the SCI baud rate generator, as well as the transmitter and receiver circuitry.

After writing to the second Baud Rate Generator Register, the transmitter and receiver circuits are enabled. The Baud Rate Generator will load the new value and start counting.

To initialize the SCI, the user should first initialize the most significant byte of the Baud Rate Generator Register; this will reset all SCI circuitry. The user should then initialize all other SCI registers (SICR/SOCR included) for the desired operating mode and then, to enable the SCI, he should initialize the least significant byte Baud Rate Generator Register.

'On-the-Fly' modifications of the control registers' content during transmitter/receiver operations, although possible, can corrupt data and produce undesirable spikes on the I/O lines (data, clock and control). Furthermore, modifying the control registers' content without reinitialising the SCI circuitry (during stand-by cycles, waiting to transmit or receive data) must be kept carefully under control by software to avoid spurious data being transmitted or received.

**Note**: For synchronous receive operation, the data and receive clock must not exhibit significant skew between clock and data. The received data and clock are internally synchronized to INTCLK.

**FUNCTIONAL DESCRIPTION** (Cont'd)

**Figure 26. Auto Echo Configuration**

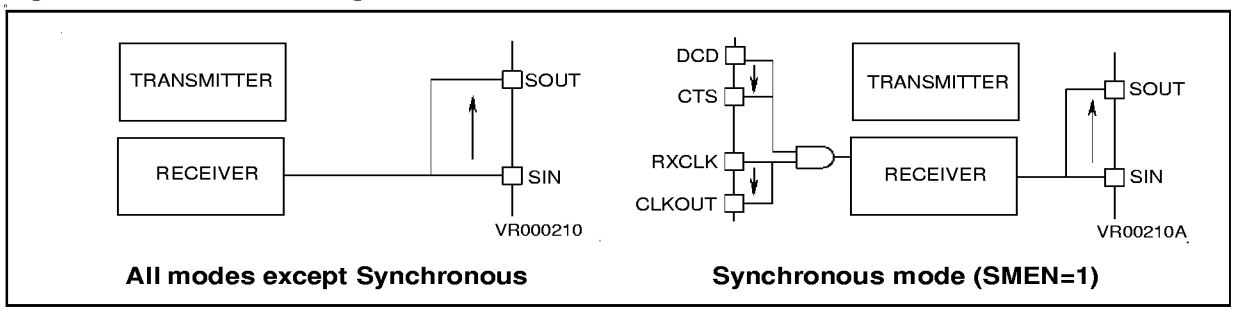



**Figure 27. Loop Back Configuration**

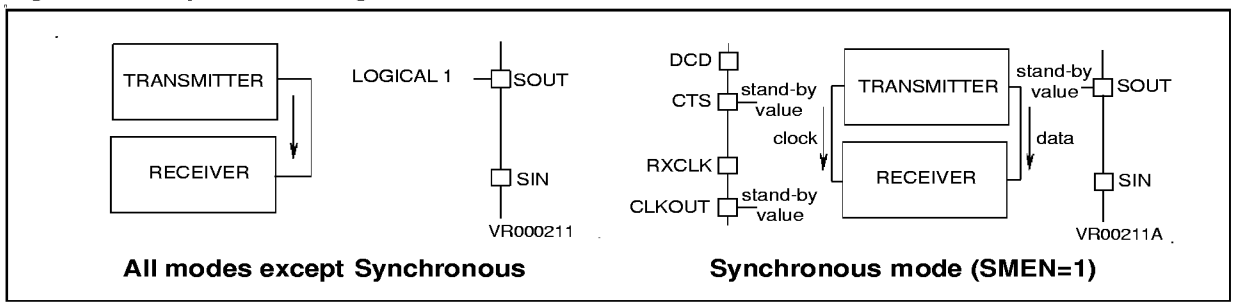
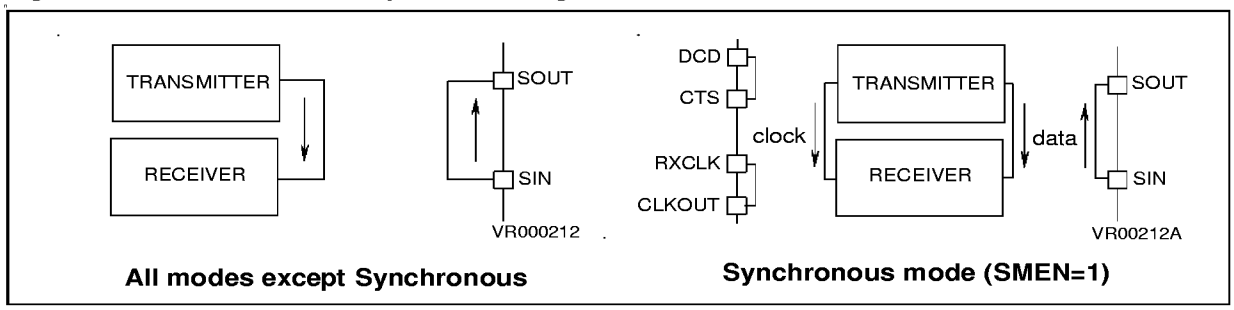



**Figure 28. Auto Echo and Loop-Back Configuration**

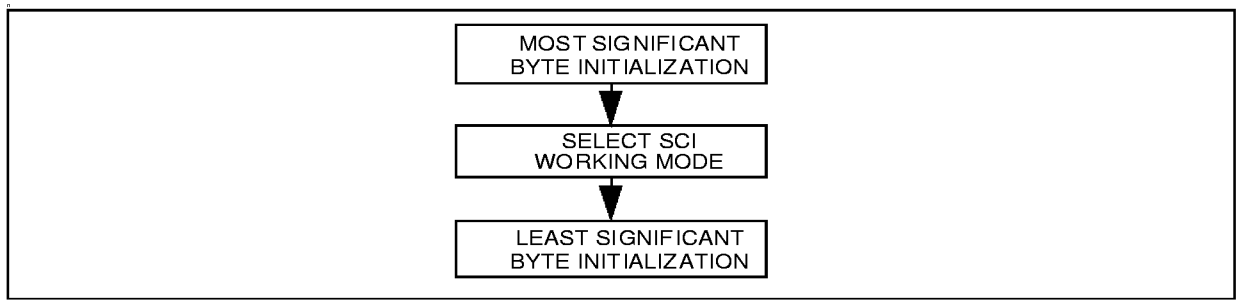


**FUNCTIONAL DESCRIPTION** (Cont'd)

**Table 29. Practical Example of SCI Baud Rate Generator Divider Values**

| INTCLK: 19660.800 KHz | | | | | | | |
|---|---|---|---|---|---|---|---|
| Baud Rate | Clock Factor | Desired Freq (kHz) | Divisor | | Actual Baud Rate | Actual Freq (kHz) | Deviation |
| | | | Dec | Hex | | | |
| 50.00 | 16 X | 0.80000 | 24576 | 6000 | 50.00 | 0.80000 | 0.0000% |
| 75.00 | 16 X | 1.20000 | 16384 | 4000 | 75.00 | 1.20000 | 0.0000% |
| 110.00 | 16 X | 1.76000 | 11170 | 2BA2 | 110.01 | 1.76014 | -0.00081% |
| 300.00 | 16 X | 4.80000 | 4096 | 1000 | 300.00 | 4.80000 | 0.0000% |
| 600.00 | 16 X | 9.60000 | 2048 | 800 | 600.00 | 9.60000 | 0.0000% |
| 1200.00 | 16 X | 19.20000 | 1024 | 400 | 1200.00 | 19.20000 | 0.0000% |
| 2400.00 | 16 X | 38.40000 | 512 | 200 | 2400.00 | 38.40000 | 0.0000% |
| 4800.00 | 16 X | 76.80000 | 256 | 100 | 4800.00 | 76.80000 | 0.0000% |
| 9600.00 | 16 X | 153.60000 | 128 | 80 | 9600.00 | 153.60000 | 0.0000% |
| 19200.00 | 16 X | 307.20000 | 64 | 40 | 19200.00 | 307.20000 | 0.0000% |
| 38400.00 | 16 X | 614.40000 | 32 | 20 | 38400.00 | 614.40000 | 0.0000% |
| 76800.00 | 16 X | 1228.80000 | 16 | 10 | 76800.00 | 1228.80000 | 0.0000% |

**Figure 29. SCI Baud Rate Generator Initialization Sequence**



**FUNCTIONAL DESCRIPTION** (Cont'd)

**13.2.5 Input Signals**

**SIN: Serial Data Input** This pin is the serial data input to the SCI receiver shift register.

**TXCLK: External Transmitter Clock Input** This pin is the external input clock driving the SCI transmitter. The TXCLK frequency must be greater than or equal to 16 times the transmitter data rate (depending whether the X16 or the X1 clock have been selected) and must have a period of at least twice INTCLK. The use of the TXCLK pin is optional.

**RXCLK: External Receiver Clock Input.** This input is the clock to the SCI receiver when using an external clock source connected to the baud rate generator. INTCLK is normally the clock source. A 50% duty cycle is not required for this input, however, the shortest period must last more than two INTCLK periods. Use of RXCLK is optional.

**DCD: Data Carrier Detect .** This input is enabled only in Synchronous mode; it works as a gate for the RXCLK clock and informs the MCU that an emitting device is transmitting a synchronous frame. The active level can be programmed as 1 or 0 and must be provided at least one INTCK period before the first active edge of the input clock.

**13.2.6 Output Signals**

**SOUT: Serial Data Output.** This Alternate Function output signal is the serial data output for the SCI transmitter in all operating modes.

**CLKOUT: Clock Output** The alternate Function of this pin outputs either the data clock from the transmitter in Serial Expansion or Synchronous modes, or the clock output from the Baud Rate Generator. In Serial expansion mode it will clock only the data portion of the frame and its stand-by state is high: data is valid on the rising edge of the clock. Even in Synchronous mode CLKOUT will only clock the data portion of the frame, but the stand-by level and active edge polarity are programmable by the user.

When Synchronous mode is disabled (SMEN in SICR is reset), the state of the XTCLK and OCLK bits in CCR determine the source of CLKOUT; '11' enables the Serial Expansion Mode.

When the Synchronous mode is enabled (SMEN in SICR is set), the state of the XTCLK and OCLK bits in CCR determine the source of CLKOUT; '00' disables it for PLM applications.

**CTS: Clear To Send.** This output Alternate Function is only enabled in Synchronous mode; it becomes active when the Least Significant Bit of the data frame is sent to the Serial Output Pin (DATOUT) and indicates to the target device that the MCU is about to send a synchronous frame; it returns to its stand-by value just after the last active edge of CLKOUT (MSB transmitted). The active level can be programmed high or low.

## 13.3 INTERRUPTS AND DMA

### 13.3.1 Interrupts

The SCI can generate interrupts as a result of several conditions. Receiver interrupts include data pending, receive errors (overrun, framing and parity), as well as address or break pending. Transmitter interrupts are software selectable for either Transmit Holding Register Empty (HSN set) or for Transmit Shift Register Empty (HSN reset) conditions.

Typical Usage of the Interrupts generated by the SCI peripheral are illustrated in Figure 30.

The SCI peripheral is able to generate interrupt requests as a result of a number of events, several of which share the same interrupt vector. It is therefore necessary to poll ISR, the Interrupt Status Register, in order to determine the active trigger. These bits should be reset by the programmer during the Interrupt Service routine.

The four major levels of interrupt are encoded in hardware to provide two bits of the interrupt vector register, allowing the position of the block of pointer vectors to be resolved to an 8 byte block size.

The SCI interrupts have an internal priority structure in order to resolve simultaneous events. Refer also to Section 13.2.1 SCI Operating Modes for more details relating to Synchronous mode

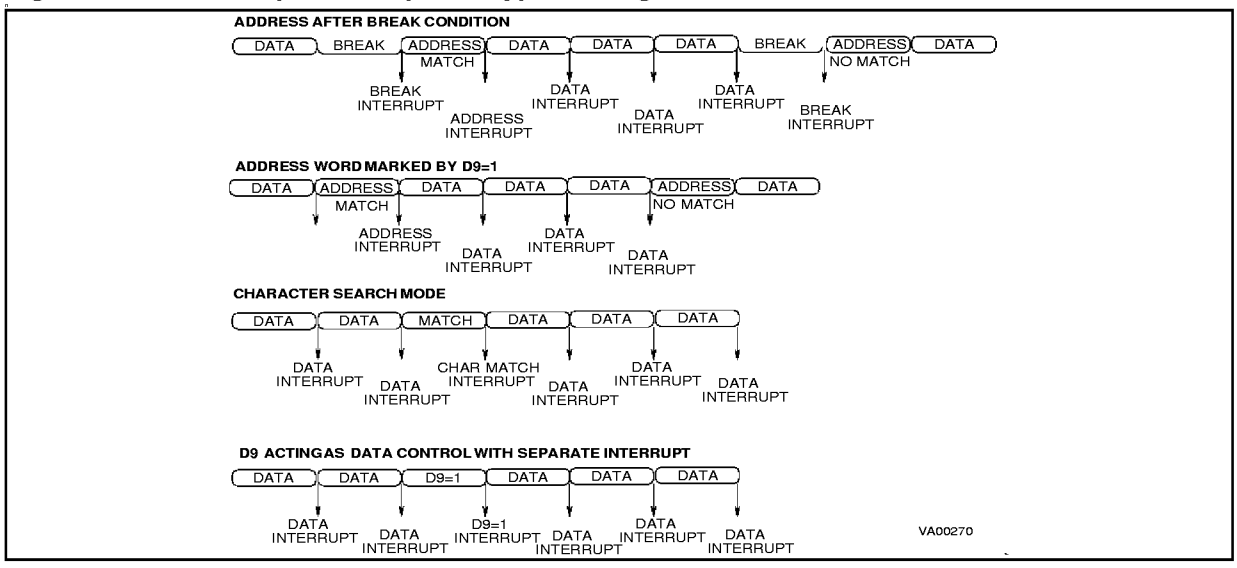
**Table 30. SCI Interrupt Internal Priority**

| | |
|---|---|
| Receive DMA Request | Highest Priority |
| Transmit DMA Request | |
| Receive Interrupt | |
| Transmit Interrupt | Lowest Priority |

**Table 31. SCI I nterrupt Vector**

| Interrupt Source | Vector Address |
|---|---|
| Transmitter Buffer or Shift Register Empty<br>Transmit DMA end of Block | xxx x110 |
| Received Ready<br>Receive DMA end of Block | xxxx x100 |
| Break Detector<br>Address Word Match | xxxx x010 |
| Receiver Error | xxxx x000 |

**Figure 30. SCI Interrupts: Example of Typical Usage**



**INTERRUPTS and DMA** (Cont'd)

### 13.3.2 DMA

Two DMA channels are associated with the SCI, for transmit and for receive. These follow the register scheme as described in DMA chapter. It should be noted that, after initializing the DMA counter and pointer registers and enabling DMA, data transmission is triggered by a character written into the Transmit Holding register.

When DMA is active the Receive Data Pending bit (RXDP in ISR), and the Transmit status bit interrupt sources are replaced by the DMA End Of Block Interrupt sources for transmit and receive, respectively.

The last DMA data word of a block of data will cause a DMA cycle followed by a transmit interrupt. This sequence will signal to the ST9 core to reinitialize the transmit DMA block counter. The Transmit End of Block status bit (TXEOB) should be reset by software in order to avoid undesired interrupt routines, especially in the initialisation routine (after reset) and after entering the End Of Block interrrupt routine.

Similarly the last DMA data word of a block of data will cause a DMA cycle followed by a receiver data ready interrupt. This sequence will signal to the ST9 core to reinitialize the receiver DMA block counter. The Received End of Block status bit (RXEOB) should be reset by software in order to avoid undesired interrupt routines, especially in the initialisation routine (after reset) and after entering the End Of Block interrupt routine.

Remark: If properly initialized, the DMA controller starts a data transfer after and only if the running program has loaded the Transmitter Buffer Register with a value. In order to execute properly a DMA transmission, the End Of Block interrupt routine must include the following actions:

– Load the Transmitter Buffer Register (TXBR) with the first byte to transmit.
– Restore the DMA counter (TDCPR)
– Restore the DMA pointer (TDAPR)
– Reset the transmitter end of block bit TXEOB (IMR.5)
– Reset the transmitter holding empty bit TXHEM (ISR.1)
– Enable DMA

## 13.4 CONTROL REGISTERS

The SCI registers are located in the following pages in the ST9:

SCI number 0: page 24 (18h)

SCI number 1: page 25 (19h) (when present)

The SCI is controlled by the following registers

:

| Address | Register |
|---|---|
| R240 (F0h) | Receiver DMA Transaction Counter Pointer Register |
| R241 (F1h) | Receiver DMA Source Address Pointer Register |
| R242 (F2h) | Transmitter DMA Transaction Counter Pointer Register |
| R243 (F3h) | Transmitter DMA Destination Address Pointer Register |
| R244 (F4h) | Interrupt Vector Register |
| R245 (F5h) | Address Compare Register |
| R246 (F6h) | Interrupt Mask Register |
| R247 (F7h) | Interrupt Status Register |
| R248 (F8h) | Receive Buffer Register same Address as Transmitter Buffer Register (Read Only) |
| R248 (F8h) | Transmitter Buffer Register same Address as Receive Buffer Register (Write only) |
| R249 (F9h) | Interrupt/DMA Priority Register |
| R250 (FAh) | Character Configuration Register |
| R251 (FBh) | Clock Configuration Register |
| R252 (FCh) | Baud Rate Generator Register |
| R253 (FDh) | Baud Rate Generator Register |
| R254 (FEh) | Synchronous Input controlregister |
| R255 (FFh) | Synchronous Output controlregister |

**CONTROL REGISTERS** (Cont'd)

**RDCPR R240** (F0h) Read/Write

Receiver DMA Transaction Counter Pointer

Reset value: **undefined**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| RC7 | RC6 | RC5 | RC4 | RC3 | RC2 | RC1 | RR/M |

b7-b1 = **RC7-RC1**: *Receive DMA Counter Pointer.* RDCPR contains the address of the pointer (in the Register File) of the DMA receiver transaction counter.

b0 = **RR/M**: *Receiver Register File/Memory Selector.* If this bit = "1" the Register File will be selected as Destination, if this bit = "0" the Memory space will be selected.

**RDAPR R241** (F1h) Read/Write

Receiver DMA Source Address Pointer

Reset value: **undefined**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| RA7 | RA6 | RA5 | RA4 | RA3 | RA2 | RA1 | RD/P |

b7-b1 = **RA7-RA1**: *Receive DMA Address Pointer.* RDAPR contains the address of the pointer (in the Register File) of the receiver DMA data source.

b0 = **RD/P**: *Receive DMA Data/Program Memory Selector.* If memory (RR/M = "0") has been selected for DMA transfers, when this bit = "1" receiver DMA transfers will go to Data Memory. If this bit = "0" receiver DMA transfers will go to Program Memory.

**TDCPR R242** (F2h) Read/Write

Transmitter DMA Transaction Counter Pointer

Reset value: **undefined**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| TC7 | TC6 | TC5 | TC4 | TC3 | TC2 | TC1 | TR/M |

b7-b1 = **TC7-TC1**: *Transmitter DMA Counter Pointer.* TDCPR contains the address of the pointer (in the Register File) of the DMA transmitter transaction counter.

b0 = **TR/M**: *Transmitter Register File/Memory Selector.* If this bit = "1" the Register File will be selected as Source, if this bit = "0" the Memory space will be selected.

**TDAPR R243** (F3h) Read/Write

Transmitter DMA Destination Address Pointer

Reset value: undefined

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| TA7 | TA6 | TA5 | TA4 | TA3 | TA2 | TA1 | TD/P |

b7-b1 = **TA7-TA1**: *Transmitter DMA Address Pointer.* TDAPR contains the address of the pointer (in the Register File) of the transmitter DMA data source.

b0 = **TD/P**: *Transmitter DMA Data/Program Memory Selector.* If memory (TR/M = "0") has been selected for DMA transfers, when this bit = "1" transmitter DMA transfers come from Data Memory. If this bit = "0" transmitter DMA transfers come from Program Memory

**CONTROL REGISTERS** (Cont'd)

**IVR R244** (F4h) Read/Write

Interrupt Vector Register

Reset value: **undefined**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| V7 | V6 | V5 | V4 | V3 | EV2 | EV1 | 0 |

b7-b3 = **V7-V3**: *SCI Interrupt Vector Base Address.* User programmable interrupt vector bits for transmitter and receiver

b2-b1 = **EV2-EV1**: *Encoded Interrupt Source (Read only).* EV2 and EV1 are set by hardware according to the interrupt source

.

| EV2 | EV1 | Interrupt source |
|---|---|---|
| 0 | 0 | Receiver Error (Overrun, Framing, Parity) |
| 0 | 1 | Break detect or address match |
| 1 | 0 | Receiver data ready/receiver DMA End of Block |
| 1 | 1 | Transmitter buffer or shift register empty transmitter DMA End of Block |

b0 = **D0**: This bit is fixed by hardware: always "0" when read.

**ACR R245** (F5h) Read/Write

Address/Data Compare Register

Reset value: **undefined**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |

b7-b0 = **AC7-AC0:** *Address/Compare Character.* With either 9th bit address mode, address after break mode, or character search, the received address will be compared to the value stored in this register. When a valid address matches this register content, the Receive Address Pending bit is set. After the RXAP bit is set in an addressed mode all received data words will be transferred to the Receiver Buffer Register.

**IMR R246** (F6h) Read/Write

Interrupt Mask Register

Reset value: 0xx0 0000b

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| HSN | RXEOB | TXEOB | RXE | RXA | RXB | RXDI | TXDI |

b7 = **HSN**: *Holding or shift register empty interrupt* Selects the interrupt/DMA source as the transmitter register empty event. If set, a holding register empty will generate a transmitter register empty interrupt. If reset, a shift register empty will generate a transmitter register empty interrupt.

b6 = **RXEOB**: *Received End of Block.* Set after a receiver DMA cycle to mark the end of a block of data. The last DMA data word will cause a DMA cycle followed by a receiver data ready interrupt. This sequence instructs the ST9 core to reinitialize the receiver DMA block counter. RXEOB should be reset by software in order to avoid undesired interrupt routines, especially in the initialisation routine (after reset) and after entering the End Of Block interrupt routine. Resetting this bit will cancel the interrupt request.
RXEOB can only be reset, the core sets this bit.

b5 = **TXEOB**: *Transmitter End of Block.* Set in a transmitter DMA cycle to mark the end of a data block. The last DMA data word will cause a DMA cycle followed by a transmitter interrupt. This sequence instructs the ST9 core to reinitialize the transmitter DMA block counter. The same comments made for RXEOB apply.

b4 = **RXE**: *Receiver Error Mask.* When this bit is reset, the receiver error bits, Overrun Error (OE), Parity Error (PE), and Framing Error (FE), cannot generate an interrupt.

b3 = **RXA**: *Receiver Address Mask* When this bit is reset, the Receiver Address Pending (RXAP) bit cannot generate an interrupt.

b2 = **RXB**: *Receiver Break Mask* When this bit is reset, the Receiver Break Pending (RBP) bit cannot generate an interrupt.

b1 = **RXDI**: *Receiver Data Interrupt Mask* When this bit is reset, the Receiver Data Pending (RDP) bit and the Receiver End of Block (RXEOB) bit cannot generate an interrupt. RXDI has no effect on DMA transfers.

b0 = **TXDI**: *Transmitter Data Interrupt Mask* When this bit is reset, neither the Transmitter Holding or Shift Register Empty (TXHEM) bit or the Transmitter End of Block (TXEOB) bit can generate an interrupt. TXDI has no effect on DMA transfers.

**CONTROL REGISTERS** (Cont'd)

**ISR R247** (F7h) Read/Write

Interrupt Status Register

Reset value: undefined

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| OE | FE | PE | RXAP | RXBP | RXDP | TXHEM | TXSEM |

b7 = **OE**: *Overrun Error Pending.* This bit is set if the data in the Receiver Buffer Register was not read by the CPU before the next character was transferred into the Receiver Buffer Register (the previous data is lost).

b6 = **FE**: *Framing Error Pending bit* This bit is set if the received data word did not have a valid stop bit. In the case where a framing error occurs when the SCI is programmed in address mode and is monitoring an address, the interrupt is asserted and the corrupted data element is transferred to the Receiver Buffer Register.

b5 = **PE**: *Parity Error Pending.* This bit is set if the received word did not have the correct even or odd parity bit.

b4 = **RXAP**: *Receiver Address Pending.* RXAP is set after an interrupt acknowledged in the address mode. The source of this interrupt is given by the couple of bits (AMEN, AM) as detailed in the "Interrupt/DMA Priority Register" description.

b3 = **RXBP**: *Receiver Break Pending bit* This bit is set if the received data input is held low for the full word transmission time (start bit, data bits, parity bit, stop bit).

b2 = **RXDP**: *Receiver Data Pending bit.* This bit is set when data is loaded into the Receiver Holding Register.

b1 = **TXHEM**: *Transmitter buffer register Empty.* This bit is set if the Holding Register is empty.

b0 = **TXSEM**: *Transmitter Shift Register Empty.* This bit is set if the Shift Register has completed the transmission of the available data.

**Note**: The Interrupt Status Register bits can be reset but cannot be set by the user. The interrupt source must be cleared by resetting the related bit when executing the interrupt service routine (naturally the other pending bits should not be reset).

**RXBR R248** (F8h) Read only

Receive Buffer Register

Reset value: undefined

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| RD7 | RD6 | RD5 | RD4 | RD3 | RD2 | RD1 | RD0 |

b7-b0 = **RD7-RD0**: *Received Data.* This register stores the data portion of the received word. The data will be transferred from the Receiver Shift Register into the Receiver Buffer Register at the end of the word. All receiver interrupt conditions will be updated at the time of transfer. If the selected character format is less than 8 bits, unused most significant bits will forced to "1".

**Note**. RXBR and TXBR are two physicaly different registers located at the same address.

**TXBR R248** (F8h) Write only

Transmitter Buffer Register

Reset value: undefined

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| TD7 | TD6 | TD5 | TD4 | TD3 | TD2 | TD1 | TD0 |

b7-b0 = **TD7-TD0**: *Transmit Data.* The ST9 core will load the data for transmission into this register. The SCI will transfer the data from the buffer into the Shift Register when available. At the transfer, the Transmitter Buffer Register interrupt is updated. If the selected word format is less than 8 bits, the unused most significant bits are not significant.

**Note**. TXBR and RXBR are two physicaly different registers located at the same address.

**CONTROL REGISTERS** (Cont'd)

**IDPR R249** (F9h) Read/Write

Interrupt/DMA Priority Register

Reset value: undefined

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| AMEN | SB | SA | RXD | TXD | PRL2 | PRL1 | PRL0 |

b7 = **AMEN**: *Address Mode Enable.* This bit, together with AM (R250), decodes the desired addressing/9th data bit/character match operation.

In addressed mode the SCI monitors the input serial data until its address is detected

.

| AMEN | AM | |
|---|---|---|
| 0 | 0 | Address interrupt if 9th data bit = 1 |
| 0 | 1 | Address interrupt if character match |
| 1 | 0 | Address interrupt if character match and 9th data bit =1 |
| 1 | 1 | Address interrupt if character match with word immediately following Break |

Upon reception of address, the RXAP bit (in the Interrupt Status Register) is set and an interrupt cycle can begin. The address character will not be transferred into the Receiver Buffer Register but, all data following the matched SCI address and preceeding the next address word will be transferred to the Receiver Buffer Register and the proper interrupts updated. If the address does not match, all data following this unmatched address will not be transferred to the Receiver Buffer Register.

In any of the cases the RXAP bit must be reset by software before the next word is transferred into the Buffer Register.

When AMEN is reset and AM is set, a useful character search function is performed. This allows the SCI to generate an interrupt whenever a specific character is encountered (e.g. Carriage Return).

b6 = **SB**: *Set Break.* If this bit is set, a break will be transmitted following the transmission of all data in the Transmitter Shift Register and the Buffer Register. The break will be a low level on the transmitter data output for at least one complete word format. If software does not reset SB before the minimum break length has finished, the break condition will continue until software resets SB. The SCI terminates the break condition with a high level on the transmitter data output for one transmission clock period.

b5 = **SA**: *Set Address.* If an address/9th data bit mode is selected, SA value will be loaded for transmission. Setting this bit indicates an address word. SA will be cleared by hardware after it is loaded into the Shift Register. Proper procedure would be, when the Transmitter Buffer Register is empty, to load the value of SA and then load the data into the Transmitter Buffer Register.

b4 = **RXD**: *Receiver DMA Mask* If this bit is reset, no receiver DMA request will be generated, and the RXDP bit in the Interrupt Status Register can request an interrupt. If RXD is set to "1", the RXDP bit can request a DMA transfer. This bit is reset by hardware when the transaction counter value decrements to zero. At that time a receiver "end of block" interrupt can occur.

b3 = **TXD**: *Transmitter DMA Mask* If this bit is "0" no transmitter DMA request will be generated and the TXHEM (or TXSEM) bit in the Interrupt Status Register can request an interrupt. If TXD is set, the TXHEM (or TXSEM) bit can request a DMA transfer. This bit is reset by hardware when the transaction counter value decrements to zero. At that time a transmitter End Of Block interrupt can occur.

b2-b0 = **PRL2, PRL2, PRL0**: *SCI Interrupt/DMA Priority bits.* The priority for the SCI is encoded with (PRL2,PRL1,PRL0). Priority level 0 is the highest, while level 7 represents no priority.

When the user has defined a priority level for the SCI, priorities within the SCI are hardware defined. These SCI internal priorities are

:

| | |
|---|---|
| receiver DMA request | highest priority |
| transmitter DMA request | |
| receiver interrupt | |
| transmitter interrupt | lowest priority |

**CONTROL REGISTERS** (Cont'd)

**CHCR R250** (FAh) Read/Write

Character Configuration Register

Reset value: undefined

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| AM | EP | PEN | AB | SB1 | SB0 | WL1 | WL0 |

b7 = **AM**: *Address Mode*. decodes the desired addressing/9th data bit/character match operation in conjunction with AMEN (IDPR register).

b6 = **EP**: *Even Parity*. When parity is enabled, this bit selects between even or odd parity. If this bit is reset, odd parity will be selected. If this bit is set, even parity will be selected.

b5 = **PEN**: *Parity Enable*. When this bit is set, a parity bit is generated (transmit data) or checked (received data). If the address/9th bit is enabled, the parity bit will precede the address/9th bit (the 9th bit is never included in the parity calculation).

b4 = **AB**: *Address/9th Bit*. If this bit is set, the character format will include a bit between the parity bit and the first stop bit. This bit can be used to address the SCI or as a ninth data bit.

b3-b2 = **SB1-SB2**: *Number of Stop Bits*. .

| SB2 | SB1 | Number of stop bits | |
|---|---|---|---|
| | | in 16X mode | in 1X mode |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1.5 | 2 |
| 1 | 0 | 2 | 2 |
| 1 | 1 | 2.5 | 3 |

b1-b0 = **WL1, WL0**: *Number of Data Bits*

| WL1 | WL0 | Data Length |
|---|---|---|
| 0 | 0 | 5 bits |
| 0 | 1 | 6 bits |
| 1 | 0 | 7 bits |
| 1 | 1 | 8 bits |

**CONTROL REGISTERS** (Cont'd)

**CCR R251** (FBh) Read/Write

Clock Configuration Register

Reset value: 0000 0000 (00h)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| XTCLK | OCLK | XRX | XBRG | CD | AEN | LBEN | STPEN |

b7 = **XTCLK**, b6 = **OCLK**: These two bits select the source for the transmitter clock. The following table shows the coding of XTCLK and OCLK.

| XTCLK | OCLK | Pin Function |
|---|---|---|
| 0 | 0 | Pin is used as a general I/O |
| 0 | 1 | Pin = TXCLK (used as an input) |
| 1 | 0 | Pin = CLKOUT (outputs the Baud Rate Generator clock) |
| 1 | 1 | Pin = CLKOUT (outputs the Serial exp. mode clock) |

b5 = **XRX**: *External Receiver Clock Source*. If set, the receiver will use the external receiver clock source, whose frequency must be 16 times the data rate, or equal to the data rate, depending on the status of the CD bit.

b4 = **XBRG**: *Baud Rate Generator Clock Source*. If this bit is set, the baud rate generator will use the external receiver clock. If reset, the baud rate generator will useINTCLK.

b3 = **CD**: *Clock Divisor*. If CD is set, both the receiver and the transmitter will be in 1X clock mode. In 1X clock mode, the transmitter will transmit data at one data bit per clock period. If this bit is reset, both the receiver and the transmitter will be in 16X mode. In 16X mode each data bit period will be 16 clock periods long. The status of CD will determine the SCI configuration (synch/asynch).

b2 = **AEN**: *Auto Echo Enable*. If AEN is set, the SCI is in auto echo mode: the SCI transmitter is disconnected from the data-out pin SOUT, which is driven directly by the receiver data-in pin, SIN. The receiver remains connected to SIN and is operational, unless loopback mode is also selected.

b1 = **LBEN**: *Loopback Enable*. If this bit is set, loopback mode is enabled. In this mode the transmitter output is set to a high level, the receiver input is disconnected, and the output of the Transmitter Shift Register is looped back into the Receiver Shift Register input. All interrupt sources, (transmitter and receiver) are operational.

b0 = **STPEN**: *Stick Parity Enable*. If this bit is set, the transmitter and the receiver will use the opposite parity type selected by the even parity bit (EP).

| EP | SPEN | Parity(Transmitter & Receiver) |
|---|---|---|
| 0 (odd) | 0 | Odd |
| 1 (even) | 0 | Even |
| 0 (odd) | 1 | Even |
| 1 (even) | 1 | Odd |

**CONTROL REGISTERS** (Cont'd)

**SICR R254** (FEh) Read/Write

Synchronous Input Control

Reset value : 0000 0011b (03h)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| SMEN | INPL | XCKPL | DCDEN | DCDPL | INPEN | X | X |

b7 = **SMEN**: *Synchronous Mode Enable*. When SMEN is set, Synchronous mode is selected with its programmed I/O configuration. If SMEN is reset, all features relating to Synchronous mode are disabled (the contents of SICR and SOCR are ignored).

b6 = **INPL**: *SIN Input Polarity*. When INPL is set the polarity is inverted, when reset it is not. INPL only affects received data. In Auto-Echo mode SOUT = SIN even if INPLis set. In Loop-Back mode the state of the INPL bit is irrelevant.

b5 = **XCKPL**: *Receiver Clock Polarity*. When XCKPL is set, RXCLK is active on the falling edge, while when the bit is reset RXCLK is active on the rising edge. XCKPL only affects the receiver clock. In Auto-Echo mode CKOUT = RXCLK independently of the XCKPL status. In Loop-Back the state of the XCKPL bit is irrelevant.

b4 = **DCDEN**: *DCD Input Enable* . When DCDEN is set, hardware synchronization is enabled; when it is reset, hardware synchronization is disabled.

When DCDEN is set, RXCLK drives the receiver section only during the active level of the DCD input (DCD works as a gate on RXCLK, informing the MCU that atransmitting device is sending a synchronous frame to it).

b3 = **DCDPL**: *DCD Input Polarity*. When DCDPL is set, the DCD input is active when HIGH; when the bit is reset, the DCD input is active when LOW. DCDPL only affects the gating activity ot the receiver clock. In Auto-Echo mode CTS= DCD independently of DCDPL. In Loop-Back mode, the state of DCDPL is irrelevant.

b2 = **INPEN**: *All Input Disable*. When INPEN is set, SIN/RXCLK/DCD are disabled; when the bit is reset, SIN/RXCLK/DCD are enabled

**SOCR R255** (FFh) Read/Write

Synchronous Output Control

Reset value: 0000 0001 (01h)

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| OUTPL | OUTSB | OCKPL | OCKSB | CTSEN | CTSPL | - | X |

b7 = **OUTPL**: *SOUT Output Polarity*. When OUTPOL is set , polarity is inverted; when reset it is not. OUTPL affects only the data sent by the transmitter section. In Auto-Echo mode SOUT = SIN even if OUTPL=1. In Loop-Back mode, the state of OUTPL is irrelevant.

b6 = **OUTSB**: *SOUT Output Stand-By Level*. When OUTSP is set, SOUT stand-by level is LOW; when reset, SOUT stand-by level is HIGH.

b5 = **OCKPL**: *Transmitter Clock Polarity*. When OCKPL is set, CLKOUT is active on the falling edge; when the bit is reset, CLKOUT is active on therising edge. XCKPL only affects the transmitter clock. In Auto-Echo mode CKOUT = RXCLK independently of the state of OCKPL. In Loop-Back mode the state of OCKPL is irrelevant.

b4 = **OCKSB**: *Transmitter Clock Stand-By Level*. When OCKSB is set, the CLKOUT stand-by level is LOW; when the bit is reset, the CLKOUT stand-by level is HIGH.

b3 = **CTSEN**: *CTS Output Enable*. When CTSEN is set, the CTS hardware synchronization is enabled; when the bit is reset, the synchronisation is disabled.

When CTSEN is set, the CTS output becomes active just before the first active edge of CLKOUT and indicates to target device that the MCU is about to send a synchronous frame; it returns to its stand-by value just after the last active edge of CLKOUT ( MSB transmitted).

b2 = **CTSPL**: *CTS Output Polarity*. When CTSPL is set, the CTS output is active when HIGH; when the bit is reset, the CTS output is active when LOW.

CTSPL affects only the CTS activity on the output pin. In Auto-Echo mode CTS= DCD independently from the CTSPL value . In Loop-Back mode CTSPL value is 'Don't Care' .

b1 = **Reserved**.

b0 = **"Don't Care"**

**CONTROL REGISTERS** (Cont'd)

**BRGHR R252** (FCh) Read/Write

Baud Rate Generator High Register ()

Reset value: **undefined**

| 15 | | | | | | | 8 |
|---|---|---|---|---|---|---|---|
| BG15 | BG14 | BG13 | BG12 | BG11 | BG10 | BG9 | BG8 |

**BRGLR R253** (FDh) Read/Write

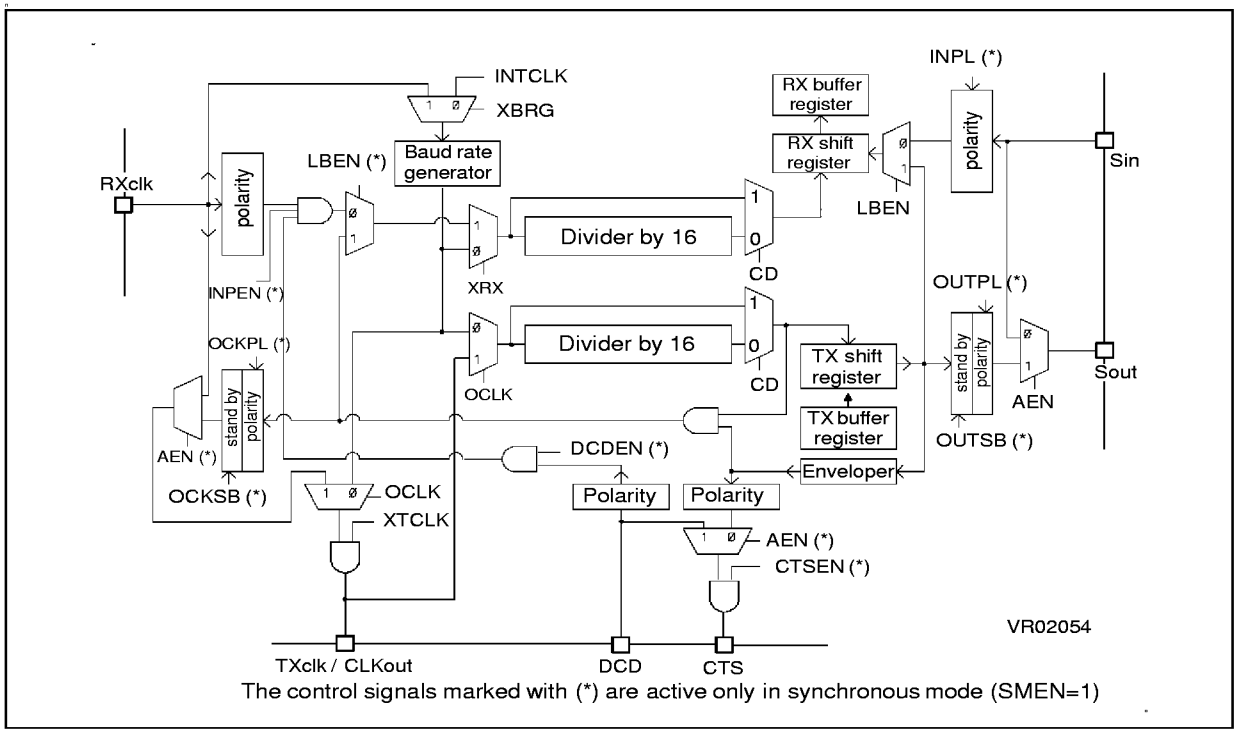
Baud Rate Generator Low Register

Reset value: **undefined**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| BG7 | BG6 | BG5 | BG4 | BG3 | BG2 | BG1 | BG0 |

b15-b0: The Baud Rate generator is a programmable divide by "N" counter which can be used to generate the clocks for the transmitter and/or receiver. This counter divides the clock input by the value in the Baud Rate Generator Register. The minimum baud rate divisor is 2 and the maximum divisor is $2^{16}$-1. After initialization of the baud rate generator, the divisor value is immediately loaded into the counter. This prevents potentially long random counts on the initial load. If set to 0 or 1, the Baud Rate Generator is stopped.

**Figure 31. SCI Functional Schematic**



# 14 EIGHT-CHANNEL ANALOG TO DIGITAL CONVERTER (ADC8)

## 14.1 INTRODUCTION

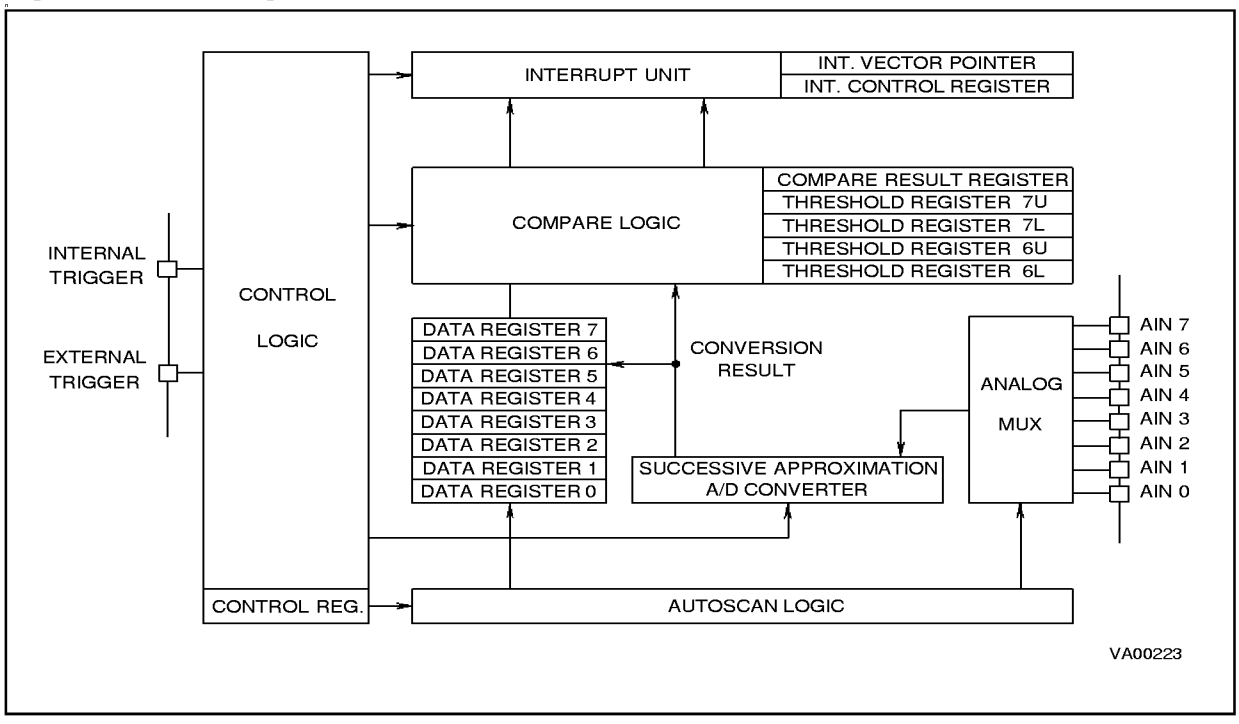
The 8-Channel Analog to Digital Converter (ADC8) comprises an input multiplex channel selector feeding a successive approximation converter. Conversion requires 138 INTCLK cycles (of which 85 are required for sampling), conversion time is thus a function of the INTCLK frequency; for instance, for a 20MHz clock rate, conversion of the selected channel requires 6.9µs. This time includes the 4.25µs required by the built-in Sample and Hold circuitry, which minimizes the need for external components and allows quick sampling of the signal to minimise warping and conversion error. Conversion resolution is 8 bits, with ±1/2 LSB maximum non-linearity error between $V_{SS}$ and the analog $V_{DD}$ reference.

The converter uses a fully differential analog input configuration for the best noise immunity and precision performance. Two separate supply references are provided to ensure the best possible supply noise rejection and to allow the use of analog reference voltages lower than the digital $V_{DD}$ supply. In fact, the converted digital value, is referred to the analog reference voltage which determines the full scale converted value. Naturally, Analog and Digital $V_{SS}$ MUST be common.

Up to 8 multiplexed Analog Inputs are available, depending on the specific device type. A group of signals can be converted sequentially by simply programming the starting address of the first analog channel to be converted and using the AUTOSCAN feature.

Two Analog Watchdogs are provided, allowing continuous hardware monitoring of two input channels. An Interrupt request is generated whenever the converted value of either of these two analog inputs is outside the upper or lower programmed threshold values. The comparison result is stored in a dedicated register.

**Figure 32. Block Diagram**



**INTRODUCTION** (Cont'd)

Single and continuous conversion modes are available. Conversion may be triggered by an external signal or, internally, by the Multifunction Timer.

A Power-Down programmable bit allows the ADC8 to be set in low-power idle mode.

The ADC8's Interrupt Unit provides two maskable channels (Analog Watchdog and End of Conversion) with hardware fixed priority, and up to 7 programmable priority levels.

**CAUTION**: ADC8 INPUT PIN CONFIGURATION

The input Analog channel is selected by using the I/O pin Alternate Function setting (PXC2, PXC1, PXC0 = 1,1,1) as described in the I/O ports section. The I/O pin configuration of the port connected to the A/D converter is modified in order to prevent the analog voltage present on the I/O pin from causing high power dissipation across the input buffer. Deselected analog channels should also be maintained in Alternate function configuration for the same reason.

## 14.2 FUNCTIONAL DESCRIPTION

### 14.2.1 Operating Modes

Two operating modes are available: Continuous Mode and Single Mode. To enter one of these modes it is necessary to program the CONT bit of the Control Logic Register, the Continuous Mode is selected when CONT is set, while Single Mode is selected when CONT is reset.

Both modes can operate in AUTOSCAN configuration, allowing sequential conversion of the input channels. The number of analog inputs to be converted may be set by software, by setting the number of the first channel to be converted into the Control Register (SC2, SC1, SC0 bits). As each conversion is completed, the channel number is automatically incremented, up to channel 7. For example, if SC2, SC1, SC0 are set to 0,1,1, conversion will proceed from channel 3 to channel 7, whereas, if SC2, SC1, SC0 are set to 1,1,1, only channel 7 will be converted.

When the ST bit of the Control Logic Register is set, either by software or by hardware (by an internal or external synchronisation trigger signal), the analog inputs are sequentially converted (from the first selected channel up to channel 7) and the results are stored in the relevant Data Registers.

In **Single Mode** (CONT = "0"), the ST bit is reset by hardware following conversion of channel 7; an End of Conversion (ECV) interrupt request is issued and the ADC8 waits for a new start event.

In **Continuous Mode** (CONT = "1"), a continuous conversion flow is initiated by the start event. When conversion of channel 7 is complete, conversion of channel 's' is initiated (where 's' is specified by the setting of the SC2, SC1 and SC0 bits); this will continue until the ST bit is reset by software. In all cases, an ECV interrupt is issued each time channel 7 conversion ends.

When channel 'i' is converted ('s' <'i' <7), the related Data Register is reloaded with the new conversion result and the previous value is lost. The End of Conversion (ECV) interrupt service routine can be used to save the current values before a new conversion sequence (so as to create signal sample tables in the Register File or in Memory).

### 14.2.2 Triggering and Synchronisation

In both modes, conversion may be triggered by internal or external conditions; externally this may be tied to ADTRG, as an Alternate Function input on an I/O port pin, and internally, it may be tied to INTRG, generated by a Multifunction Timer peripheral. Both external and internal events can be seperately masked by programming the EXTG/INTG bits of the Control Logic Register (CLR). The events are internally ORed, thus avoiding potential hardware conflicts. However, the correct procedure is to enable only one alternate synchronisation condition at any time.

The effect either of these synchronisation modes is to set the ST bit by hardware. This bit is reset, in Single Mode only, at the end of each group of conversions. In Continuous Mode, all trigger pulses after the first are ignored.

The synchronisation sources must be at a logic low level for at least the duration of one INTCLK cycle and, in Single Mode, the period between trigger pulses must be greater than the total time required for a group of conversions. If a trigger occurs when the ST bit is still set, i.e. when conversion is still in progress, it will be ignored.

### 14.2.3 Analog Watchdogs

Two internal Analog Watchdogs are available for highly flexible automatic threshold monitoring of external analog signal levels. Analog channels 6 and 7 monitor an acceptable voltage level window for the converted analog inputs. The external voltages applied to inputs 6 and 7 are considered normal while they remain below their respective Upper thresholds, and above or at their respective Lower thresholds.

**FUNCTIONAL DESCRIPTION** (Cont'd)

When the external signal voltage level is greater than, or equal to, the upper programmed voltage limit, or when it is less than the lower programmed voltage limit, a maskable interrupt request is generated and the Compare Results Register is updated in order to flag the threshold (Upper or Lower) and channel (6 or 7) responsible for the interrupt. The four threshold voltages are user programmable in dedicated registers (08h to 0Bh) of the ADC8 register page. Only the 4 MSBs of the Compare Results Register are used as flags (the 4 LSBs always return "1" if read), each of the four MSBs being associated with a threshold condition.

Following a hardware reset, these flags are reset. During normal ADC8 operation, the CRR bits are set, in order to flag an out of range condition and are automatically reset by hardware after a software reset of the Analog Watchdog Request flag in the ICR Register.

### 14.2.4 Power Down Mode

Before enabling an A/D conversion, the POW bit of the Control Logic Register must be set; this must be done at least 60µs before the first conversion start, in order to correctly bias the analog section of the converter circuitry.

When the ADC8 is not required, the POW bit may be reset in order to reduce the total power consumption. This is the reset configuration, and this state is also selected automatically when the ST9 is placed in Halt Mode (following the execution of the `halt` instruction).

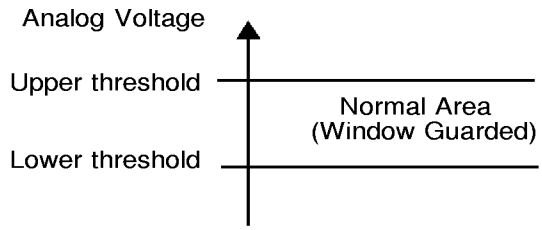
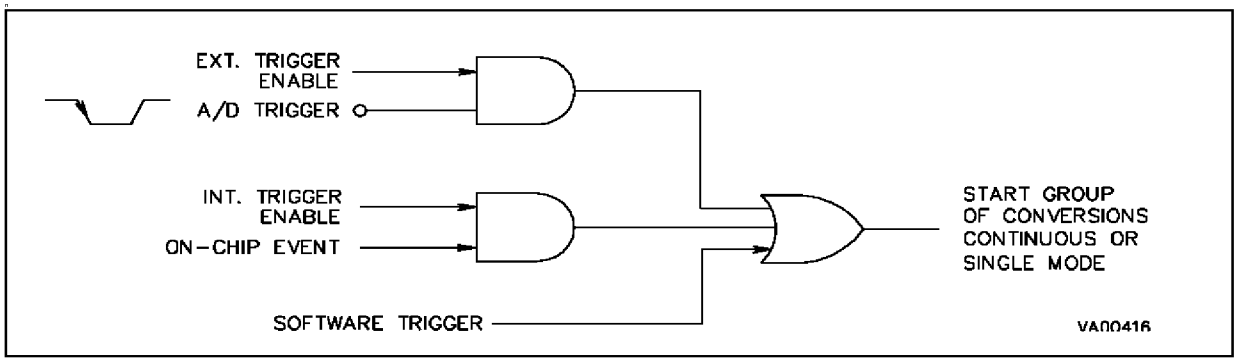
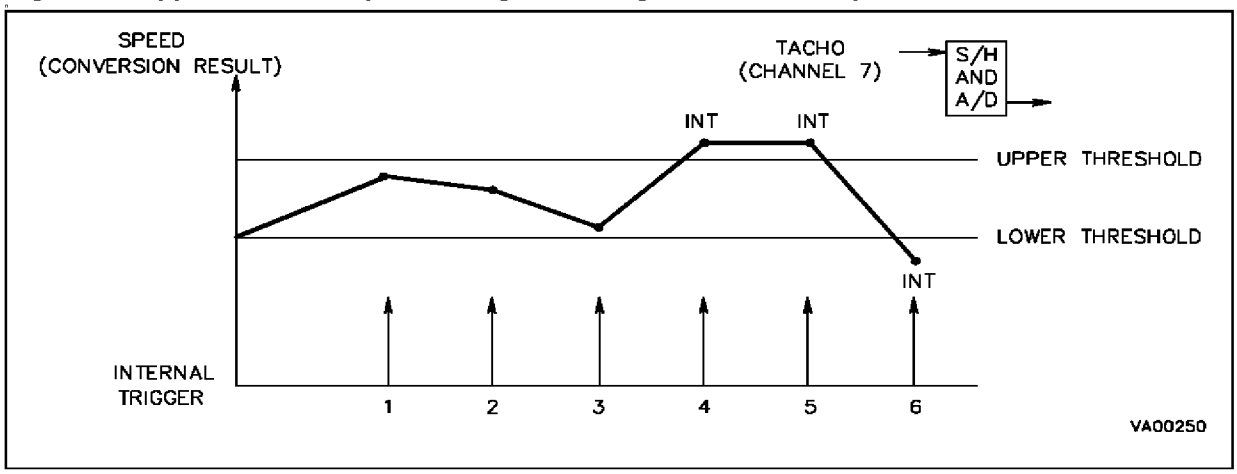



**Figure 33. A/D Trigger Source**



**FUNCTIONAL DESCRIPTION** (Cont'd)

**Figure 34. Application Example: Analog Watchdog used in Motorspeed Control**



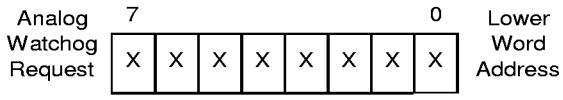

### 14.3 INTERRUPTS

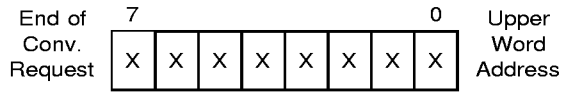
The ADC8 provides two interrupt sources:

– End of Conversion

– Analog Watchdog Request

The A/D Interrupt Vector Register (IVR) provides hardware generated flags which indicate the interrupt source, thus allowing automatic selection of the correct interrupt service routine.




The A/D Interrupt vector should be programmed by the User to point to the first memory location in the Interrupt Vector table containing the base address of the four byte area of the interrupt vector table in which the address of the A/D interrupt service routines are stored.

The Analog Watchdog Interrupt Pending bit (AWD, ICR.6), is automatically set by hardware whenever any of the two guarded analog inputs go out of range. The Compare Result Register (CRR) tracks the analog inputs which exceed their programmed thresholds.

When two requests occur simultaneously, the Analog Watchdog Request has priority over the End of Conversion request, which is held pending.

The Analog Watchdog Request requires the user to poll the Compare Result Register (CRR) to determine which of the four thresholds has been exceeded. The threshold status bits are set to flag an out of range condition, and are automatically reset by hardware after a software reset of the Analog Watchdog Request flag in the ICR Register. The interrupt pending flags, ECV and AWD, should be reset by the user within the interrupt service routine. Setting either of these two bits by software will cause an interrupt request to be generated.

## 14.4 ADC8 REGISTERS

### 14.4.1 Data Registers (DiR)

The conversion results for the 8 available channels are loaded into the 8 Data registers following conversion of the corresponding analog input.

**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**
b7-b0 = D3.7-D3.0: Channel 3 Data

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| D3.7 | D3.6 | D3.5 | D3.4 | D3.3 | D3.2 | D3.1 | D3.0 |

**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**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| D7.7 | D7.6 | D7.5 | D7.4 | D7.3 | D7.2 | D7.1 | D7.0 |

**ADC8 REGISTERS** (Cont'd)

### 14.4.2 Lower Threshold Registers (LTiR)

The two Lower Threshold registers are used to store the user programmable lower threshold 8-bit values, to be compared with the current conversion results, thus setting the lower window limit.

**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 | LT6.7 | LT7.5 | LT7.4 | LT7.3 | LT7.2 | LT7.1 | LT7.0 |

b7-b0 = **LT7.7-LT7.0:** *Channel 7 Lower Threshold*

### 14.4.3 Upper Threshold Registers (UTiR)

The two Upper Threshold registers are used to store the user programmable upper threshold 8-bit values, to be compared with the current conversion results, thus setting the upper window limit.

**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 value*

**UT7R R251** (FBh) Page 63 Read/Write

Channel 7 Upper Threshold Register

Reset Value: **undefined**

| 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|
| UT7.7 | UT6.7 | UT7.5 | UT7.4 | UT7.3 | UT7.2 | UT7.1 | UT7.0 |

b7-b0 = **UT7.7-UT7.0:** *Channel 7 Upper Threshold value*

### 14.4.4 Compare Result Register (CRR)

The result of the 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 when converted data is greater than or equal to the threshold value. Not affected otherwise.

b6 = **C6U**: *Compare Reg 6 Upper threshold*
Set when converted data is greater than or equal to the threshold value. Not affected otherwise.

b5 = **C7L**: *Compare Reg 7 Lower threshold*
Set when converted data is less than the threshold value. Not affected otherwise.

b4 = **C6L**: *Compare Reg 6 Lower threshold*
Set when converted data is less than the threshold value. Not affected otherwise.

These bits should be reset at the end of the "Out of Range" interrupt service routine.

b3-b0 = undefined, return "1" when read.

**Note**: Any software reset request of the ICR, will also cause all the compare status bits to be hardware forced to zero, in order to prevent possible overwriting if an interrupt request occurs between reset and the Interrupt request software reset.

**REGISTERS** (Cont'd)

### 14.4.5 Control Logic Register (CLR)

The Control Logic Register (CLR) manages the ADC8's logic. Writing to this register will cause the current conversion to be aborted and the autoscan logic to be re-initialized. CLR is programmable as follows:

**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 - SC0:** *Start Conversion Address.*

These 3 bits define the starting analog input channel in Autoscan mode. The first channel addressed by SC2-SC0 is converted, then the channel number is incremented for the successive conversion, until channel 7 (111) is converted. When SC2, SC1 and SC0 are all set, only channel 7 will be converted.

b4 = **EXTG**: *External Trigger.*

When set, this bit allows a conversion sequence to be started on the subsequent edge of the external signal applied to the ADTRG pin (when enabled as an Alternate Function).

b3 = **INTG**: *Internal Trigger.*

When set, this bit allows a conversion sequence to be started, synchronized by an internal signal (On-chip Event signal) from a Multifunction Timer peripheral.

Both External and Internal Trigger inputs are internally OR'ed, thus avoiding Hardware conflicts; however, the correct procedure is to enable only one alternate synchronization input at a time.

b2 = **POW**: *Power Up/Power Down.*

When this bit is set, the A/D converter logic and analog circuitry is enabled. When the bit is reset, all power consuming logic is disabled, thus selecting a low power idle mode.

b1 = **CONT**: *Continuous/Single.*

When this bit is set (Continuous Mode), the first group of conversions are started, either by software (by setting the ST bit), or by hardware (on an internal or external trigger, depending on the setting of the INTG and EXTG bits); a continuous conversion sequence is then initiated.

When this bit is reset (Single Mode), a single sequence of conversions is initiated whenever an external (or internal) trigger occurs, or when the ST bit is set by software.

The effect of the either synchronization mode is to set the START/STOP bit, which is hardware reset when in SINGLE mode, at the end of each sequence of conversions.

Requirements: The External Synchronisation Input must receive a low level pulse wider than an INTCLK period and, for both External and On-Chip Event synchronisation, the repetition period must be greater than the time required for the selected sequence of conversions.

b0 = **ST**: *Start/Stop.*

When this bit is set, a group of conversions is initiated; when the bit is reset, conversion is halted. When the A/D converter is running in Single Mode, this bit is hardware reset at the end of a sequence of conversions.

**REGISTERS** (Cont'd)

### 14.4.6 Interrupt Control Register (ICR)

The Interrupt Control Register contains the three priority level bits, the two source 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 set by hardware whenever either of the two monitored analog inputs goes out of bounds. The threshold values are stored in registers F8h and FAh for channel 6, and in registers F9h and FBh for channel 7 respectively. The Compare Result Register (CRR) keeps track of the analog inputs exceeding the thresholds.

AWD and ECV must be reset by the user, before returning from the Interrupt Service Routine. Setting either of these bits 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. When set, it enables the request; when reset, it masks the request.

b4 = **AWDI**: *Analog Watchdog Interrupt Enable*
This bit masks or enables the Analog Watchdog interrupt request. When set, it enables the request; when reset, it masks the request.

b3 = **D3**: Undefined

b2-b0 = **PL2 - PL0:** *A/D Interrupt Priority Level*

These three bits allow selection of the Interrupt priority level for the ADC8.

### 14.4.7 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.*

his bit is set by hardware, according to the A/D interrupt source. It is reset 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 if the source is the End of Conversion interrupt, thus pointing to the upper word.

When two requests occur simultaneously, the Analog Watchdog Request has priority over the End of Conversion request, which is held pending.

b0 = **D0**: Fixed

This bit is fixed by hardware. It always returns the value "0" when read.

# 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 and Analog Input | -5 to +5 | mA |
| | Maximum Accumulated Pin injection Current in the device | -50 to +50 | mA |
| $AV_{DD}$ | A/D Converter Analog Reference | -0.3 to 7.0 | V |
| $AV_{SS}$ | A/D Converter $V_{SS}$ | $V_{SS}$ | |

**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 | -40 | 85 | °C |
| $V_{DD}$ | Operating Supply Voltage | 2.7 | 5.5 | V |
| $f_{INTCLK}$ | Internal Clock Frequency @ 4.5V - 5.5V<br>Internal Clock Frequency @ 2.7V - 3.3V | 0(1) | 16<br>12 | MHz |

**Note 1.** 1MHz when A/D is used

## PACKAGE THERMAL CHARACTERISTICS

| Symbol | Parameter | Package | Value | | | Unit |
|---|---|---|---|---|---|---|
| | | | Min. | Typ. | Max. | |
| Rth | Thermal junction to ambiant | PLCC84 | | 35 | | °C/W |
| | | PQFP80 | | 40 | | |

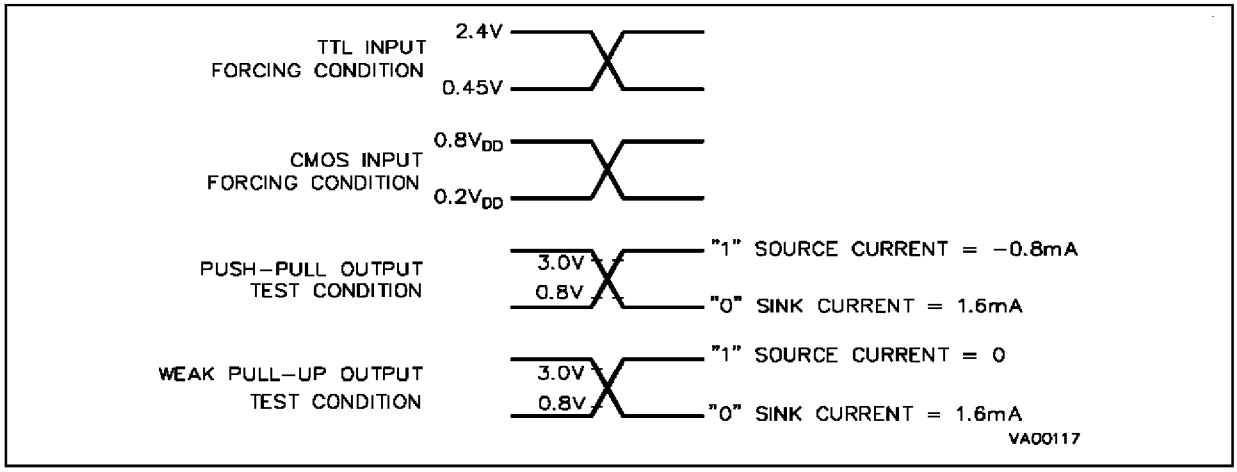
## DC ELECTRICAL CHARACTERISTICS

($V_{DD}$ = 5V ± 10% $T_A$ = -40°C + 85°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 |
| $I_{LKA/D}$ | A/D Conv. Input Leakage | | – 3 | | + 3 | µA |
| $I_{LKAP}$ | Active Pull-up Input Leakage | 0V < $V_{IN}$ < 0.8V | – 10 | | + 10 | µA |
| $I_{LKOS}$ | OSCIN Pin Input Leakage | 0V < $V_{IN}$ < $V_{DD}$ | | | ±3 | 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.

## AC TEST CONDITIONS



## AC ELECTRICAL CHARACTERISTICS

($V_{DD}$ = 5V ± 10% $T_A$ = -40°C + 85°C, unless otherwise specified)

| Symbol | Parameter | VDD | | Unit |
|---|---|---|---|---|
| | | INTCLK | MAX | |
| $I_{DD}$ | Run Mode Current no CPUCLK prescale, Clock divide by 2, PLLx10 | 16MHz | 40(*) | mA |
| | | 4MHz | 10(*) | mA |
| $I_{WFI}$ | WFI Mode Current Clock divide by 2, PLLOFF, XTAL=4MHz | 125KHz | 1.5(*) | mA |
| $I_{HALT}$ | HALT Mode Current | | 10 | µA |

(*)Preliminary Value

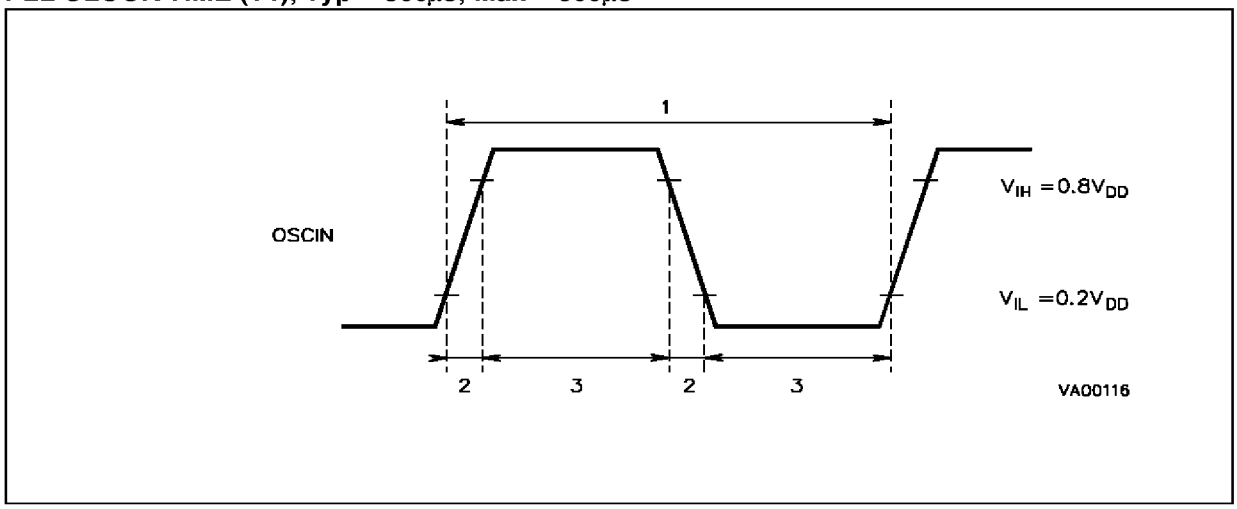
**Note:** All I/O Ports are configured in bidirectional weak pull-up mode with no DC load, external clock pin (OSCIN) is driven by square wave external clock

## CLOCK TIMING TABLE

($V_{DD}$ = 5V ± 10%, $T_A$ = -40°C + 85°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 | | ns | 1 |
| | | | 38 | | ns | 2 |
| | PLL_T1 | PLL Locking Time | | 500 | ms | |

### PLL CLOCK TIME (T1), Typ = 300µs, Max = 500µs



## EXTERNAL BUS TIMING TABLE

($V_{DD}$ = 5V ± 10%, $T_A$ = -40°C + 85°C, Cload = 50pF,INTCLK = 16MHz, 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 $\overline{AS}$ ↑ | TpC (2P+1) –22 | TWCH+PTpC –18 | 20 | | ns |
| 2 | ThAS (A) | Address Hold Time after $\overline{AS}$ ↑ | TpC –17 | TwCL –13 | 25 | | ns |
| 3 | TdAS (DR) | $\overline{AS}$ ↑ to Data Available (read) | TpC (4P+2W+4) –52 | TpC (2P+W+2) –51 | | 115 | ns |
| 4 | TwAS | $\overline{AS}$ Low Pulse Width | TpC (2P+1) –7 | TwCH+PTpC –3 | 35 | | ns |
| 5 | TdAz (DS) | $\overline{\text{Address Float}}$ to DS ↓ | 12 | 12 | 12 | | ns |
| 6 | TwDSR | $\overline{DS}$ Low Pulse Width (read) | TpC (4P+2W+3) –20 | TwCH+TpC (2P+W+1) –16 | 105 | | ns |
| 7 | TwDSW | $\overline{DS}$ Low Pulse Width (write) | TpC (2P+2W+2) –13 | TpC (P+W+1) –13 | 70 | | ns |
| 8 | TdDSR (DR) | $\overline{DS}$ ↓ to Data Valid Delay (read) | TpC (4P+2W-3) –50 | TwCH+TpC(2P+W+1) –46 | | 75 | ns |
| 9 | ThDR (DS) | Data to $\overline{DS}$ ↑ Hold Time (read) | 0 | 0 | 0 | | ns |
| 10 | TdDS (A) | $\overline{DS}$ ↑ to Address Active Delay | TpC –7 | TwCL –3 | 35 | | ns |
| 11 | TdDS (AS) | $\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay | TpC –18 | TwCL –14 | 24 | | ns |
| 12 | TsR/W (AS) | R/$\overline{W}$ Set-up Time before $\overline{AS}$ ↑ | TpC (2P+1) –22 | TwCH+PTpC –18 | 20 | | ns |
| 13 | TdDSR (R/W) | $\overline{DS}$ ↑ to R/$\overline{W}$ and Address Not Valid Delay | TpC –9 | TwCL –5 | 33 | | ns |
| 14 | TdDW (DSW) | Write Data Valid to $\overline{DS}$ ↓ Delay (write) | TpC (2P+1) –32 | TwCH+PTpC –28 | 10 | | ns |
| 15 | ThDS (DW) | Data Hold Time after $\overline{DS}$ ↑ (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) | $\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay | TpC –18 | TwCL –14 | 24 | | ns |

## EXTERNAL WAIT TIMING TABLE

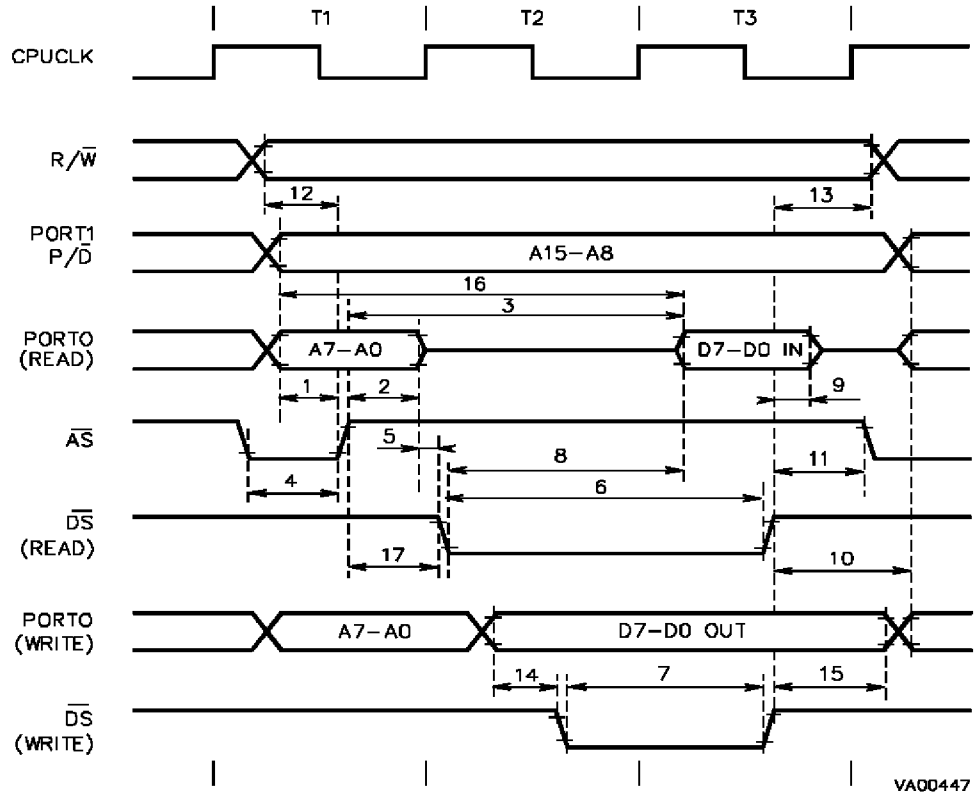
(VDD = 5V ± 10%, TA = -40°C +85°C, Cload = 50pF, INTCLK = 12MHz, Push-pull output configuration, unless otherwise specified)

| N° | Symbol | Parameter | Value (Note) | | | | Unit |
|---|---|---|---|---|---|---|---|
| | | | OSCIN Divided By 2 | OSCIN Not Divided By 2 | Min. | Max. | |
| 1 | TdAs (WAIT) | $\overline{AS}$ ↑ to $\overline{WAIT}$ ↓ Delay | 2(P+1)TpC –29 | 2(P+1)TpC –29 | | 40 | ns |
| 2 | TdAs (WAIT) | $\overline{AS}$ ↑ to $\overline{WAIT}$ ↓ Min. Delay | 2(P+W+1)TpC –4 | 2(P+W+1)TpC –4 | 80 | | ns |
| 3 | TdAs (WAIT) | $\overline{AS}$ ↑ to $\overline{WAIT}$ ↓ Max. Delay | 2(P+W+1)TpC –29 | 2(P+W+1)TpC –29 | | 83W+40 | ns |

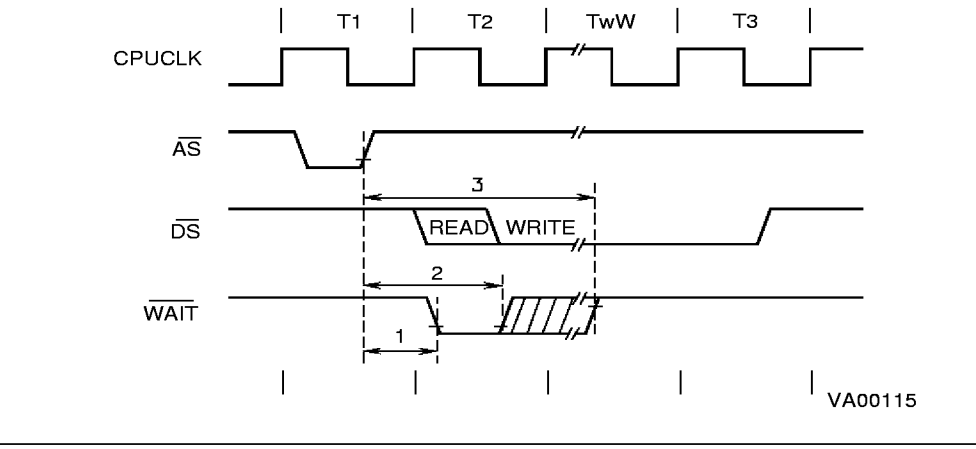
**Note:** (for both tables) The value in the left hand two columns show the formula used to calculate the timing minimum or maximum from the oscillator clock period, prescale value and number of wait cycles inserted.
The value in the right hand two columns show the timing minimum and maximum for an external clock at 24 MHz divided by 2, prescaler value of zero and zero wait status.

**Legend**:
P = Clock Prescaling Value
W = Wait Cycles
TpC = OSCIN Period
TwCH = High Level OSCIN half period
TwCL = Low Level OSCIN half period

## EXTERNAL BUS TIMING




## EXTERNAL WAIT TIMING



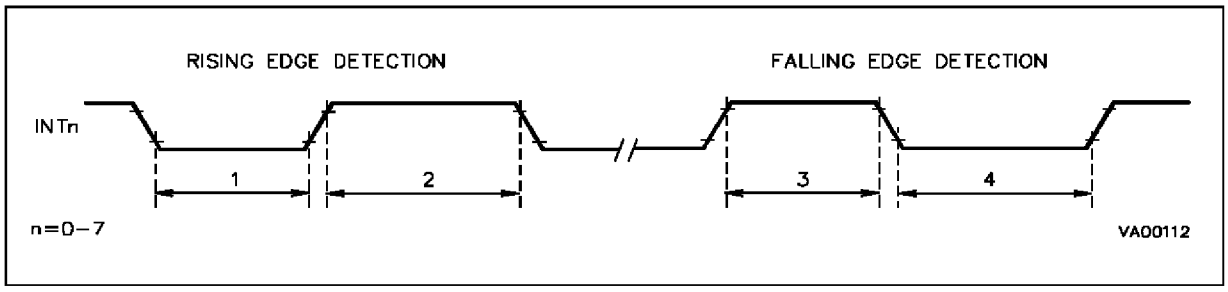
## EXTERNAL INTERRUPT TIMING TABLE

($V_{DD}$ = 5V ± 10%, $T_A$ = -40°C +85°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



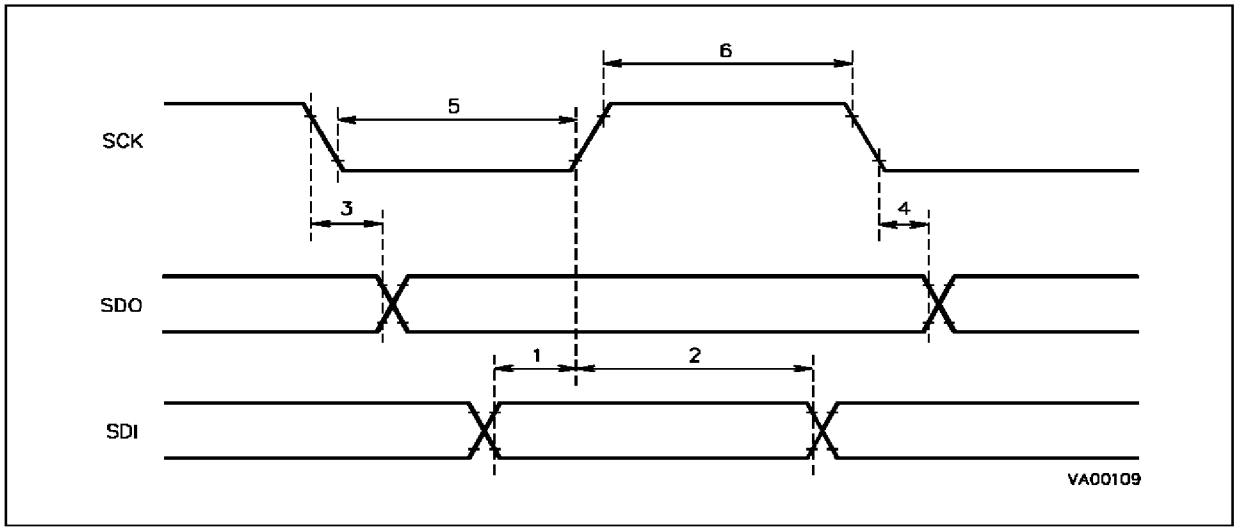
### SPI TIMING TABLE

($V_{DD}$ = 5V ± 10%, $T_A$ = -40°C + 85°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

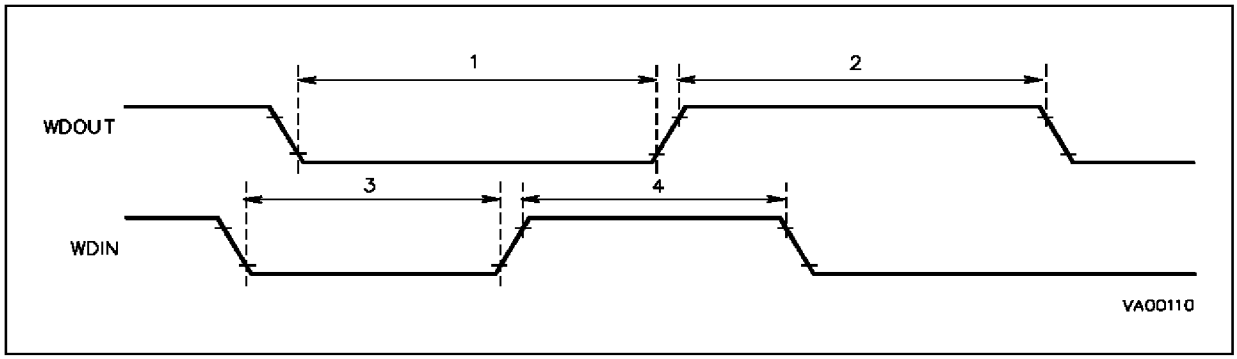


### WATCHDOG TIMING TABLE

($V_{DD}$ = 5V ± 10%, $T_A$ = -40°C + 85°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 |

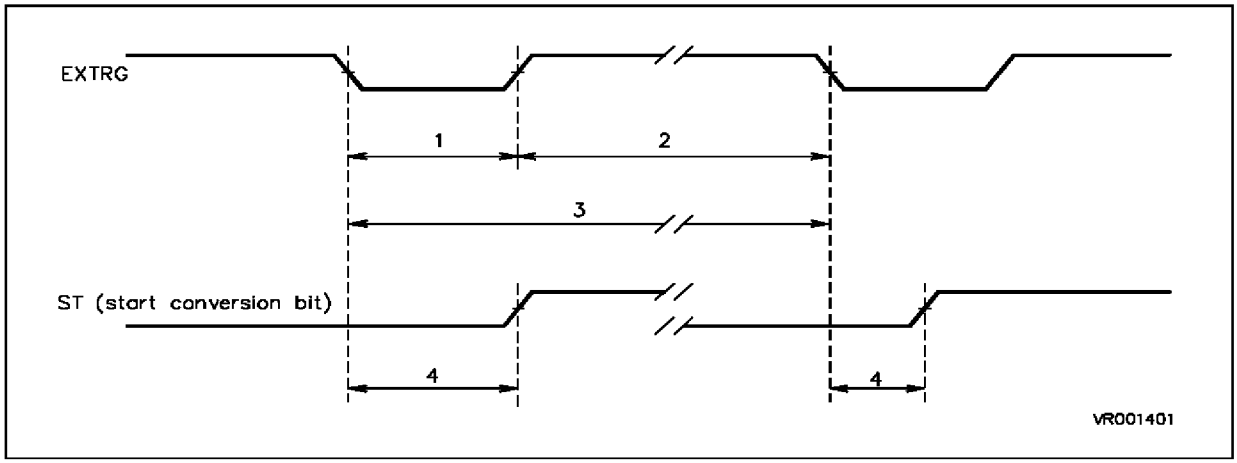
### WATCHDOG TIMING



### A/D EXTERNAL TRIGGER TIMING TABLE

| N° | Symbol | Parameter | OSCIN Divided by 2 (2) | | OSCIN Not Divided by 2 (2) | | Value (3) | | Unit |
|---|---|---|---|---|---|---|---|---|---|
| | | | Min. | Max. | Min. | Max. | Min. | Max. | |
| 1 | $Tw_{LOW}$ | External trigger pulse width | 2 x Tpc | | Tpc | | 83 | - | ns |
| 2 | $Tw_{HIGH}$ | External trigger pulse distance | 2 x Tpc | | Tpc | | 83 | - | ns |
| 3 | $Tw_{EXT}$ | External trigger active edges distance (1) | 276n x Tpc | | 138n x Tpc | | n x 11.5 | - | µs |
| 4 | $Td_{STR}$ | ADTRG falling edge and first conversion start | Tpc | 3 x Tpc | .5 x Tpc | 1.5 x Tpc | 41.5 | 125 | ns |

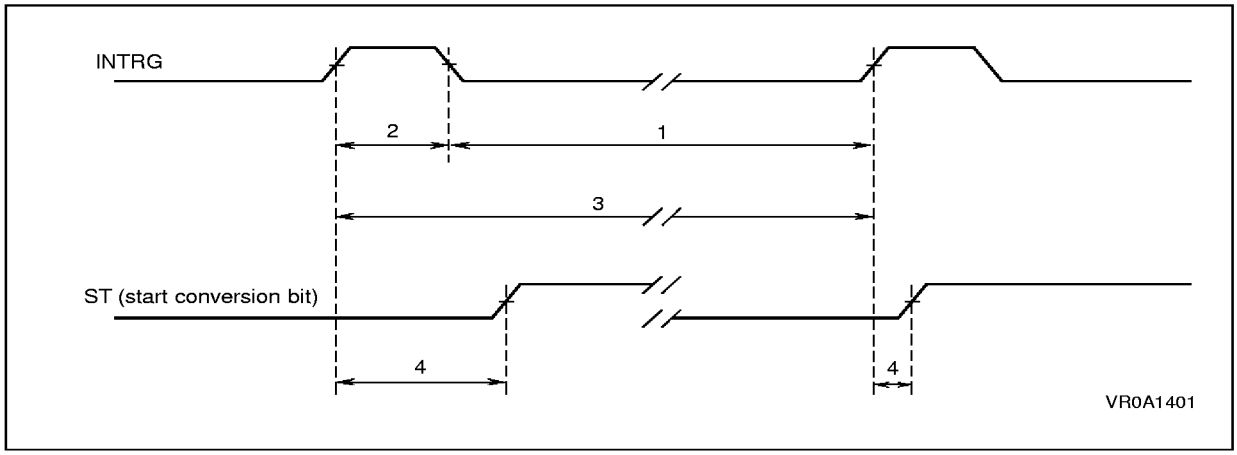
### A/D EXTERNAL TRIGGER TIMING



### A/D INTERNAL TRIGGER TIMING TABLE

| N° | Symbol | Parameter | OSCIN Divided by 2 (2) | | OSCIN Not Divided by 2 (2) | | Value (3) | | Unit |
|---|---|---|---|---|---|---|---|---|---|
| | | | Min. | Max. | Min. | Max. | Min. | Max. | |
| 1 | $Tw_{HIGH}$ | Internal trigger pulse width | Tpc | | .5 x Tpc | | 41.5 | - | ns |
| 2 | $Tw_{LOW}$ | Internal trigger pulse distance | 6 x Tpc | | 3 x Tpc | | 250 | - | ns |
| 3 | $Tw_{EXT}$ | Internal trigger active edges distance (1) | 276n x Tpc | | 138n x Tpc | | n x 11.5 | - | µs |
| 4 | $Tw_{STR}$ | Internal delay between INTRG rising edge and first conversion start | Tpc | 3 x Tpc | .5 x Tpc | 1.5 x Tpc | 41.5 | 125 | ns |

### A/D INTERNAL TRIGGER TIMING

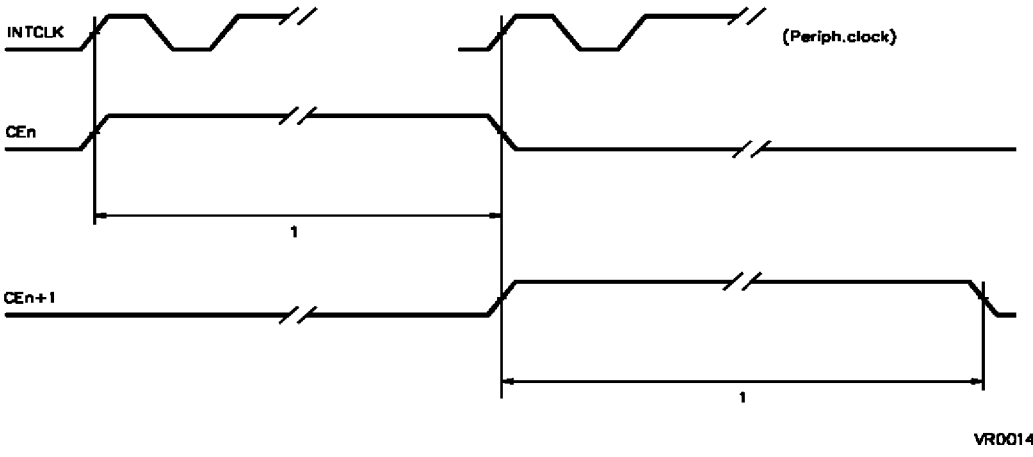


### A/D CHANNEL ENABLE TIMING TABLE

| N° | Symbol | Parameter | OSCIN Divided by 2 (2) | | OSCIN Not Divided by 2 (2) | | Value (3) | | Unit |
|---|---|---|---|---|---|---|---|---|---|
| | | | Min. | Max. | Min. | Max. | Min. | Max. | |
| 1 | $Tw_{EXT}$ | CEn Pulse width (1) | 276n x Tpc | | 138n x Tpc | | n x 11.5 | - | µs |

Notes:
1. n = number of autoscanned channels (1 < n < 8)
2. Variable clock (Tpc = OSCIN clock period)
3. INTCLK = 12MHz

### A/D CHANNEL ENABLE TIMING



## A/D ANALOG SPECIFICATIONS (VDD = 4.5V TO 5.0 V)

| Parameter | Typical | Minimum | Maximum | Units (1) | Notes |
|---|---|---|---|---|---|
| Analog Input Range<br>$A_{VDD}$ | | VDD - 0.3 V | $A_{VDD}$<br>$V_{DD}$ | V | |
| Conversion time | | 138 | | INTCLK | (2) |
| Sample time | | 87.51 | | INTCLK | |
| Power-up time | | 60 | | µs | |
| Resolution | 8 | 8 | | µs | |
| Monotonicity | GUARANTEED | + | | bits | |
| No missing codes | GUARANTEED | + | | | |
| Zero input reading | | 00 | | Hex | |
| Full scale reading | | | FF | Hex | |
| Offset error | .5 | | 1 | LSBs | (1,4) |
| Gain error | .5 | | 1 | LSBs | (4) |
| Diff. Non Linearity | ±.3 | ±.2 | ±.5 | LSBs | (4) |
| Int. Non Linearity | | | 1 | LSBs | (4) |
| Absolute Accuracy | | | 1 | LSBs | (4) |
| $A_{VCC}/A_{VSS}$ Resistance | 13.5 | 16 | 11 | KΩ | |
| Input Resistance | 12 | 8 | 15 | KΩ | (3) |
| Hold Capacitance | | | 30 | pF | |
| Input Leakage | | | ±3 | µA | |

**Notes**:

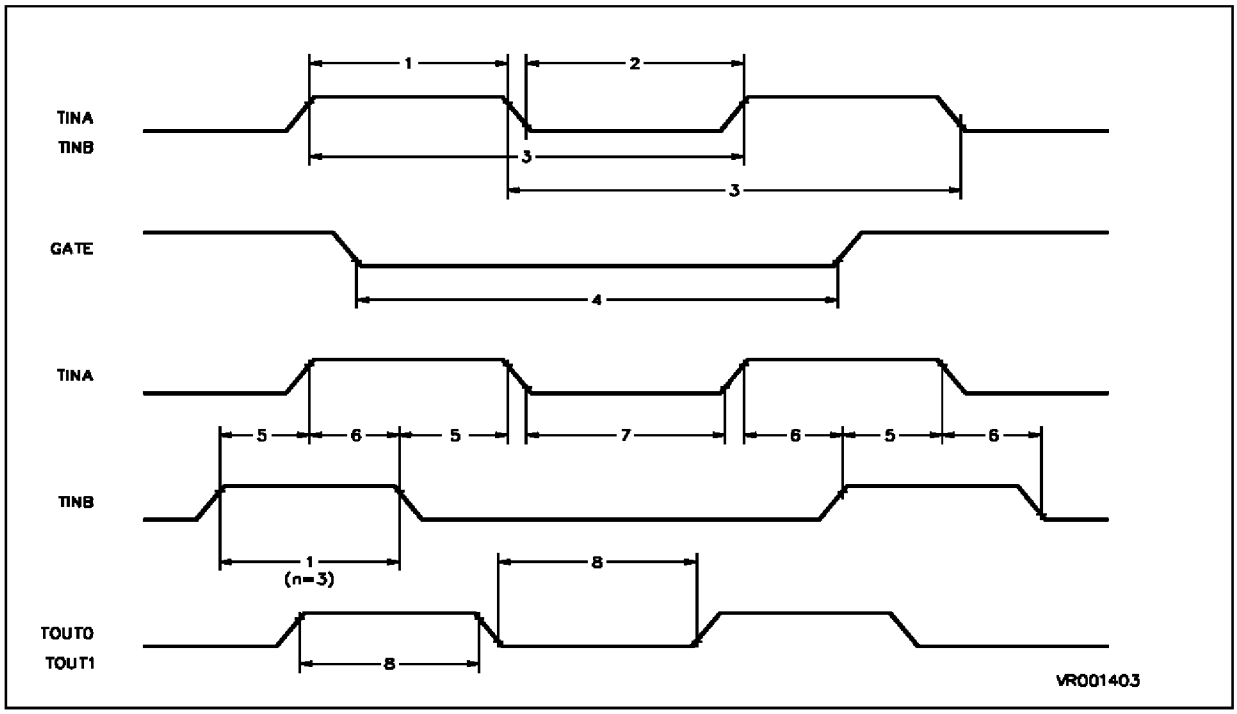
1. "LSBs", as used here, has a value of AVDD/256
2. Including sample time
3. It must be intended as the internal series resistance before the sampling capacitor
4. This is a typical expected value, but not a tested production parameter.
   If V(i) is the value of the i-th transition level (0 < i < 254), the performance of the A/D converter has been valued as follows:
   OFFSET ERROR= deviation between the actual V(0) and the ideal V(0) (=1/2 LSB)
   GAIN ERROR= deviation between the actual V(254) and the ideal V(254) (=AVCC-3/2 LSB)
   DNL ERROR= max {[V(i) - V(i-1)]/LSB - 1}
   INL ERROR= max {[V(i) - V(0)]/LSB - i}
   ABS. ACCURACY= overall max conversion error
   S/N ratio has been valued by sampling a sinusoidal input waveform and then calculating its Fast Fourier Transform.

## MULTIFUNCTION TIMER UNIT EXTERNAL TIMING TABLE

| N° | Symbol | Parameter | OSCIN Divided by 2 (3) | OSCIN Not Divided by 2 (3) | Value (4) | | Unit | Note |
|---|---|---|---|---|---|---|---|---|
| | | | | | Min. | Max. | | |
| 1 | $Tw_{CTW}$ | External clock/trigger pulse width | 2n x Tpc | n x Tpc | n x 83 | - | ns | 1 |
| 2 | $Tw_{CTD}$ | External clock/trigger pulse distance | 2n x Tpc | n x Tpc | n x 83 | - | ns | 1 |
| 3 | $Tw_{AED}$ | Distance between two active edges | 6 x Tpc | 3 x Tpc | 249 | - | ns | |
| 4 | $Tw_{GW}$ | Gate pulse width | 12 x Tpc | 6 x Tpc | 498 | - | ns | |
| 5 | $Tw_{LBA}$ | Distance between TINB pulse edge and the following TINA pulse edge | 2 x Tpc | Tpc | 83 | - | ns | 2 |
| 6 | $Tw_{LAB}$ | Distance between TINA pulse edge and the following TINB pulse edge | 0 | | 0 | - | ns | 2 |
| 7 | $Tw_{AD}$ | Distance between two TxINA pulses | 0 | | 0 | - | ns | 2 |
| 8 | $Tw_{OWD}$ | Minimum output pulse width/distance | 6 x Tpc | 3 x Tpc | 249 | - | ns | |

**Notes**:
1. n = 1 if the input is rising OR falling edge sensitive
   n = 3 if the input is rising AND falling edge sensitive
2. In Autodiscrimination mode
3. Variable clock ( Tpc = OSCIN period )
4. INTCLK = 12 MHz

## MULTIFUNCTION TIMER UNIT EXTERNAL TIMING



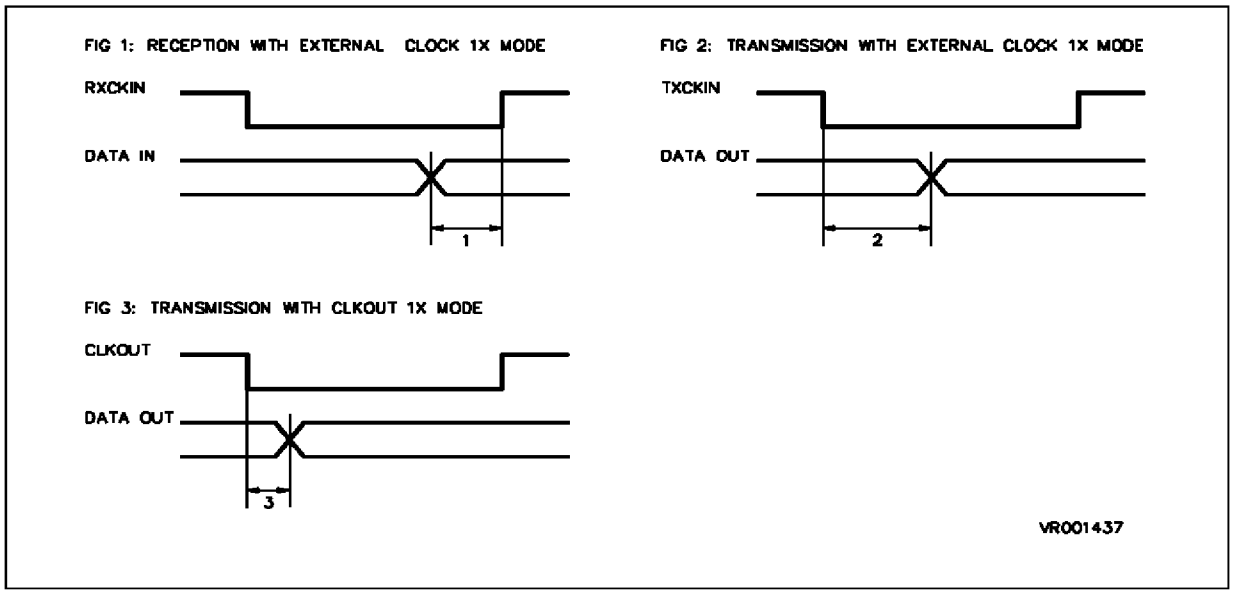
## SCI TIMING TABLE

($V_{DD}$ = 5V ± 10%, $T_A$ = - 40°C to +85°C, Cload = 50pF, INTCLK = 12MHz, Output Alternate Function set as Push-pull)

| N° | Symbol | Parameter | Condition | Value | | Unit |
|---|---|---|---|---|---|---|
| | | | | Min. | Max. | |
| | $F_{RxCKIN}$ | Frequency of RxCKIN | 1 x mode | | $F_{CK}/8$ | Hz |
| | | | 16 x mode | | $F_{CK}/4$ | Hz |
| | $Tw_{RxCKIN}$ | RxCKIN shortest pulse | 1 x mode | 4 $T_{CK}$ | | s |
| | | | 16 x mode | 2 $T_{CK}$ | | s |
| | $F_{TxCKIN}$ | Frequency of TxCKIN | 1 x mode | | $F_{CK}/8$ | Hz |
| | | | 16 x mode | | $F_{CK}/4$ | Hz |
| | $Tw_{TxCKIN}$ | TxCKIN shortest pulse | 1 x mode | 4 $T_{CK}$ | | s |
| | | | 16 x mode | 2 $T_{CK}$ | | s |
| 1 | $Ts_{DS}$ | DS (Data Stable) before rising edge of RxCKIN | 1 x mode reception with RxCKIN | $T_{PC}/2$ | | ns |
| 2 | $Td_{D1}$ | TxCKIN to Data out delay Time | 1 x mode transmission with external clock C load <100pF | | 2.5 $T_{PC}$ | ns |
| 3 | $Td_{D2}$ | CLKOUT to Data out delay Time | 1 x mode transmission with CLKOUT | 350 | | ns |

**Note**: FCK = 1/TCK

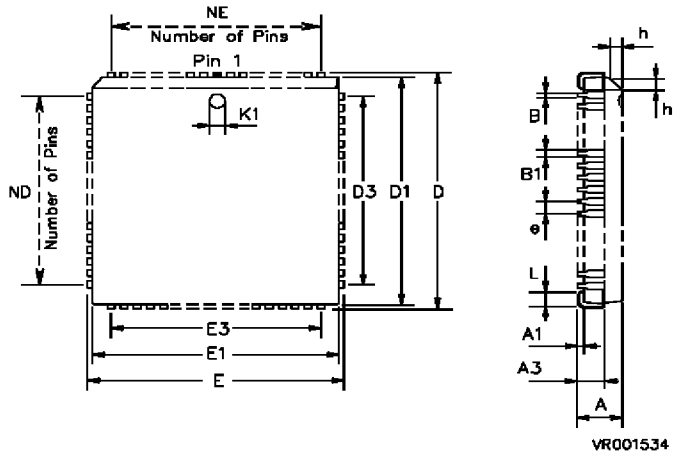
## SCI TIMING



# 16 GENERAL INFORMATION

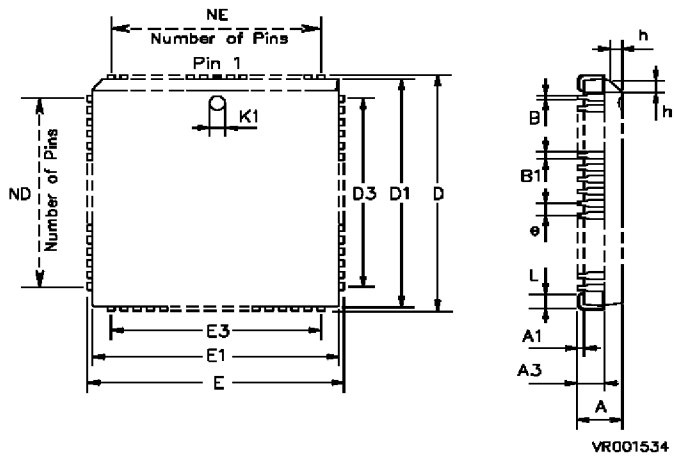
## PACKAGE MECHANICAL DATA

### 84-PIN PLASTIC LEADED CHIP CARRIER PACKAGE



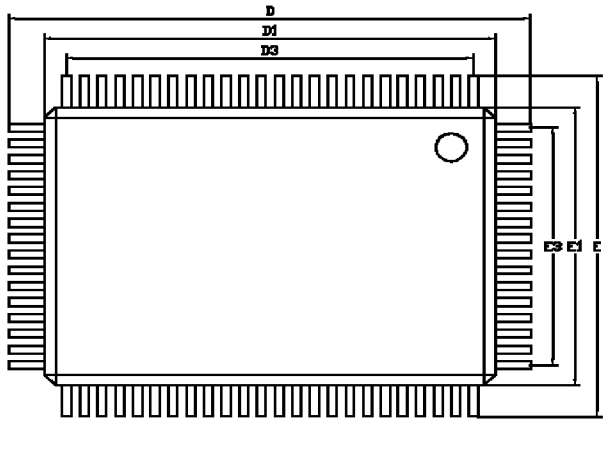

| Dim. | mm | | | inches | | |
|---|---|---|---|---|---|---|
| | Min | Typ | Max | Min | Typ | Max |
| A | 4.20 | | 5.08 | 0.165 | | 0.200 |
| A1 | | 1.00 | | | 0.039 | |
| A3 | 2.29 | | 3.30 | 0.090 | | 0.130 |
| B | | 0.38 | | | 0.015 | |
| B1 | - | - | - | - | - | - |
| D | 30.10 | | 30.35 | 1.185 | | 1.195 |
| D1 | 29.20 | | 19.41 | 1.150 | | 0.764 |
| D3 | 27.69 | | 28.70 | 1.090 | | 1.130 |
| E | 30.10 | | 30.35 | 1.185 | | 1.195 |
| E1 | 29.20 | | 19.41 | 1.150 | | 0.764 |
| E3 | 27.69 | | 28.70 | 1.090 | | 1.130 |
| K1 | | | | | | |
| e | | 1.27 | | | 0.050 | |
| h | | 1.17 | | | 0.046 | |
| | Number of Pins | | | | | |
| N | 84 | | | | | |
| ND | 21 | | | | | |
| NE | 21 | | | | | |

### 84-PIN CERAMIC LEADED CHIP CARRIER PACKAGE




| Dim. | mm | | | inches | | |
|---|---|---|---|---|---|---|
| | Min | Typ | Max | Min | Typ | Max |
| A | | | | | | |
| A1 | | | | | | |
| B | | 0.45 | | | 0.018 | |
| B1 | | 0.63 | | | 0.025 | |
| D | 29.97 | | 30.48 | 1.180 | | 1.200 |
| D1 | 28.83 | | 29.59 | 1.135 | | 1.165 |
| D3 | 25.15 | | 25.65 | 0.990 | | 1.010 |
| E | 29.97 | | 30.48 | 1.180 | | 1.200 |
| E1 | 28.83 | | 29.59 | 1.135 | | 1.165 |
| E3 | 25.15 | | 25.65 | 0.990 | | 1.010 |
| L | | | | | | |
| e | | 1.27 | | | 0.050 | |
| | Number of Pins | | | | | |
| N | 84 | | | | | |
| ND | 21 | | | | | |
| NE | 21 | | | | | |

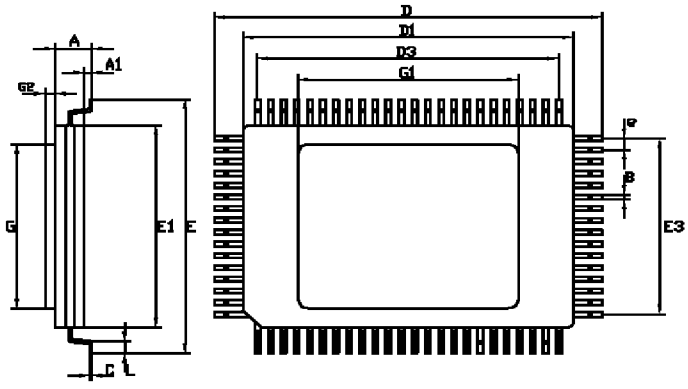
## 80-PIN PLASTIC QUAD FLAT PACKAGE




PQFP080

| Dim | mm | | | inches | | |
|---|---|---|---|---|---|---|
| | Min | Typ | Max | Min | Typ | Max |
| A | | | 3.40 | | | 0.134 |
| A1 | 0.25 | | | 0.010 | | |
| A2 | 2.55 | 2.80 | 3.05 | 0.100 | 0.110 | 0.120 |
| B | 0.30 | | 0.45 | 0.012 | | 0.018 |
| C | 0.13 | | 0.23 | 0.005 | | 0.009 |
| D | 22.95 | 23.20 | 23.45 | 0.904 | 0.913 | 0.923 |
| D1 | 19.90 | 20.00 | 20.10 | 0.783 | 0.787 | 0.791 |
| D3 | | 18.40 | | | 0.724 | |
| E | 16.95 | 17.20 | 17.45 | 0.667 | 0.677 | 0.687 |
| E1 | 13.90 | 14.00 | 14.10 | 0.547 | 0.551 | 0.555 |
| E3 | | 12.00 | | | 0.472 | |
| e | | 0.80 | | | 0.031 | |
| K | 0° | | 7° | | | |
| L | 0.65 | 0.80 | 0.95 | 0.026 | 0.031 | 0.037 |
| L1 | | 1.60 | | | 0.063 | |
| | Number of Pins | | | | | |
| N | 80 | | ND | 24 | NE | 16 |

## 80-PIN CERAMIC QUAD FLAT PACKAGE




CQFP080

| Dim | mm | | | inches | | |
|---|---|---|---|---|---|---|
| | Min | Typ | Max | Min | Typ | Max |
| A | | | 3.24 | | | 0.128 |
| A1 | | 0.20 | | | 0.008 | |
| B | 0.30 | 0.35 | 0.45 | 0.012 | 0.014 | 0.018 |
| C | 0.13 | 0.15 | 0.23 | 0.005 | 0.006 | 0.009 |
| D | 23.35 | 23.90 | 24.45 | 0.919 | 0.941 | 0.963 |
| D1 | 19.57 | 20.00 | 20.43 | 0.770 | 0.787 | 0.804 |
| D3 | | 18.40 | | | 0.724 | |
| E | 17.35 | 17.90 | 18.45 | 0.683 | 0.705 | 0.726 |
| E1 | 13.61 | 14.00 | 14.39 | 0.536 | 0.551 | 0.567 |
| E3 | | 12.00 | | | 0.472 | |
| e | | 0.80 | | | 0.031 | |
| G | 13.75 | 14.00 | 14.25 | 0.541 | 0.551 | 0.561 |
| G1 | 19.75 | 20.00 | 20.25 | 0.778 | 0.787 | 0.797 |
| G2 | | 1.06 | | | 0.042 | |
| L | 0.35 | 0.80 | | 0.014 | 0.031 | |
| | Number of Pins | | | | | |
| N | 80 | | | | | |