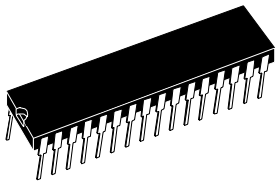
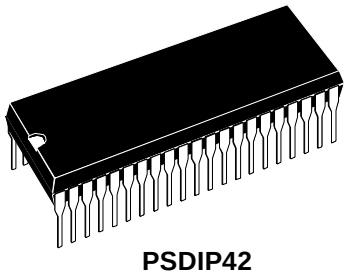
ST92186B

8/16-BIT MCU FOR TV APPLICATIONS WITH UP TO 32K ROM AND ENHANCED ON-SCREEN-DISPLAY

- Register file based 8/16 bit Core Architecture with RUN, WFI, and HALT modes
- -10 to 70°C Operating Temperature Range
- 24 MHz Operation @5V+-10%
- Minimum instruction cycle time: 250ns at 16 MHz internal clock, 165ns at 24 MHz internal clock
- 24 or 32 Kbytes ROM
- 640 bytes of on-chip static RAM
- 256 bytes of Register file
- 256 bytes of display RAM (OSDRAM)
- 32-pin Shrink DIP and 42-pin Shrink DIP packages
- 26 (SDIP42) or 17 (SDIP32) fully programmable I/O pins
- Flexible Clock controller for OSD and Core clocks, running from one single low frequency external crystal
- Enhanced Display Controller with rows of up to 63 characters per row
  - 50/60Hz and 100/120 Hz operation
  - 525/625 lines operation, 4/3 or 16/9 format
  - Interlaced and progressive scanning
  - 18x26 or 9x13 character matrix
  - 256 (18x26) characters, 1024 (9x13) characters definable in ROM by user
  - 512 possible colors, in 4x16-entry palettes
  - 2 x 16-entry palettes for Foreground, and 2 x 16-entry palettes for Background
  - 8 levels of translucency on Fast Blanking
  - Serial, Parallel, and Extended Parallel Attributes modes
  - 7 character sizes in 18x26 mode, 4 in 9x13
  - Rounding, Fringe, Scrolling, Flashing, Shadowing, Italics, Semi-transparent
- 5-channel (SDIP42) or 3-channel (SDIP32) Analog-to-Digital converter with 6-bit accuracy
- 16-bit Watchdog timer with 8-bit prescaler
- 14-bit Voltage Synthesis for tuning reference voltage with 2 outputs (SDIP42) for 2 tuners or 1 output (SDIP32) for 1 tuner



PSDIP32



PSDIP42

- 16-bit standard timer with 8-bit prescaler
- 6 (SDIP42) or 4 (SDIP32) 8-bit programmable PWM outputs
- NMI and 8 (SDIP42) or 6 (SDIP32) external interrupts
- Infra-Red signal digital pre-processor
- Rich instruction set and 14 addressing modes
- Versatile Development Tools, including C-Compiler, Assembler, Linker
- Source Level Debugger, Emulator and Real-Time Operating Systems available from third-parties
- Windows Based OSD Font and Screen Editor
- EPROM and OTP devices available (ST92E196A9 and ST92T196A9)

DEVICE SUMMARY

Device	Program Memory	Package
ST92186B3	24K	SDIP32/SDIP42
ST92186B4	32K	SDIP32/SDIP42

Table of Contents

Table of Contents

Table of Contents

1 GENERAL DESCRIPTION

1.1 INTRODUCTION

The ST92186B family brings the enhanced ST9 register-based architecture to a new range of high-performance microcontrollers specifically designed for TV applications. Their performance derives from the use of a flexible 256-register programming model for ultra-fast context switching and real-time event response. The intelligent on-chip peripherals offload the ST9 core from I/O and data management processing tasks allowing critical application tasks to get the maximum use of core resources. The ST9 MCU devices support low power consumption and low voltage operation for power-efficient and low-cost embedded systems.

1.1.1 Core Architecture

The nucleus of the ST92186B is the enhanced ST9 Core that includes the Central Processing Unit (CPU), the register file and the interrupt controller.

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

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

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

For interrupts or subroutine calls, the CPU uses a system stack in conjunction with the stack pointer (SP). A separate user stack has its own SP. The separate stacks, without size limitations, can be in on-chip RAM (or in Register File) or off-chip memory.

1.1.2 Instruction Set

The ST9 instruction set consists of 94 instruction types, including instructions for bit handling, byte (8-bit) and word (16-bit) data, as well as BCD and Boolean formats. Instructions have been added to facilitate large program and data handling through the MMU, as well as to improve the performance and code density of C Function calls. 14 addressing modes are available, including powerful indirect addressing capabilities.

The ST9's bit-manipulation instructions are set, clear, complement, test and set, load, and various logic instructions (AND, OR, and XOR). Math functions include add, subtract, increment, decrement, decimal adjust, multiply, and divide.

1.1.3 Operating Modes

To optimize performance versus the power consumption of the device, ST9 devices now support a range of 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).

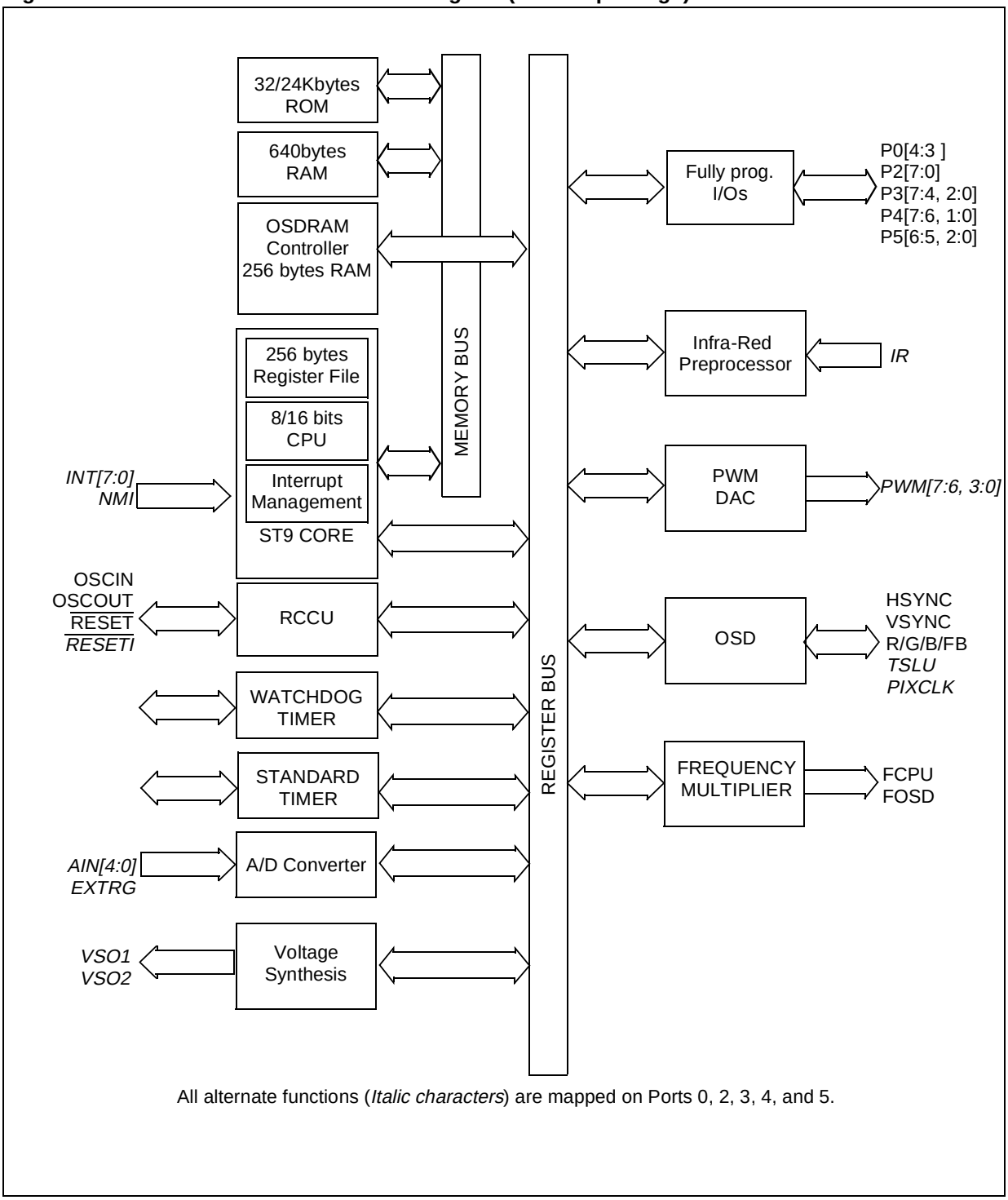
Slow Mode. Power consumption can be significantly reduced by running the CPU and the peripherals at reduced clock speed using the CPU Prescaler and CCU Clock Divider.

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 programmable via the CCU. In 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 is not enabled, the CPU and its peripherals stop operating and the status of the machine remains frozen (the clock is also stopped). A reset is necessary to exit from Halt mode.

INTRODUCTION (Cont'd)

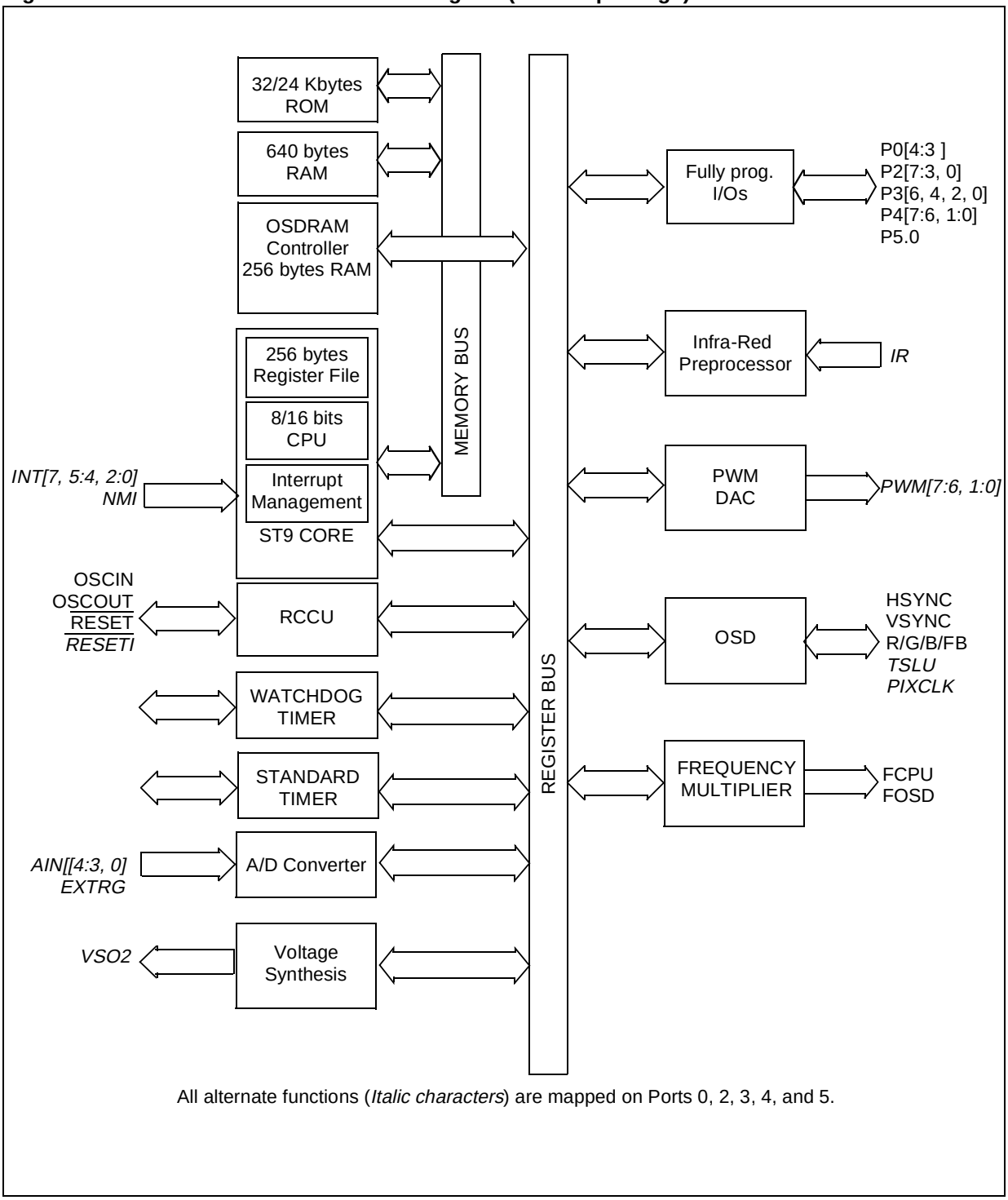
Figure 1. ST92186B Architectural Block Diagram (SDIP42 package)



All alternate functions (*Italic characters*) are mapped on Ports 0, 2, 3, 4, and 5.

INTRODUCTION (Cont'd)

Figure 2. ST92186B Architectural Block Diagram (SDIP32 package)



All alternate functions (*Italic characters*) are mapped on Ports 0, 2, 3, 4, and 5.

INTRODUCTION (Cont'd)

1.1.4 On-chip Peripherals

OSD Controller

The On Screen Display displays any text or menu data generated by the application. Rows of up to 63 characters can be displayed with two user-definable fonts. Colors, character shape and other attributes are software programmable.

Parallel I/O Ports

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

Analog/Digital Converter

The ADC provides up to 5 (SDIP42) or 3 (SDIP32) analog inputs with on-chip sample and hold.

1.2 PIN DESCRIPTION

Figure 3. 32-Pin Package Pin-Out

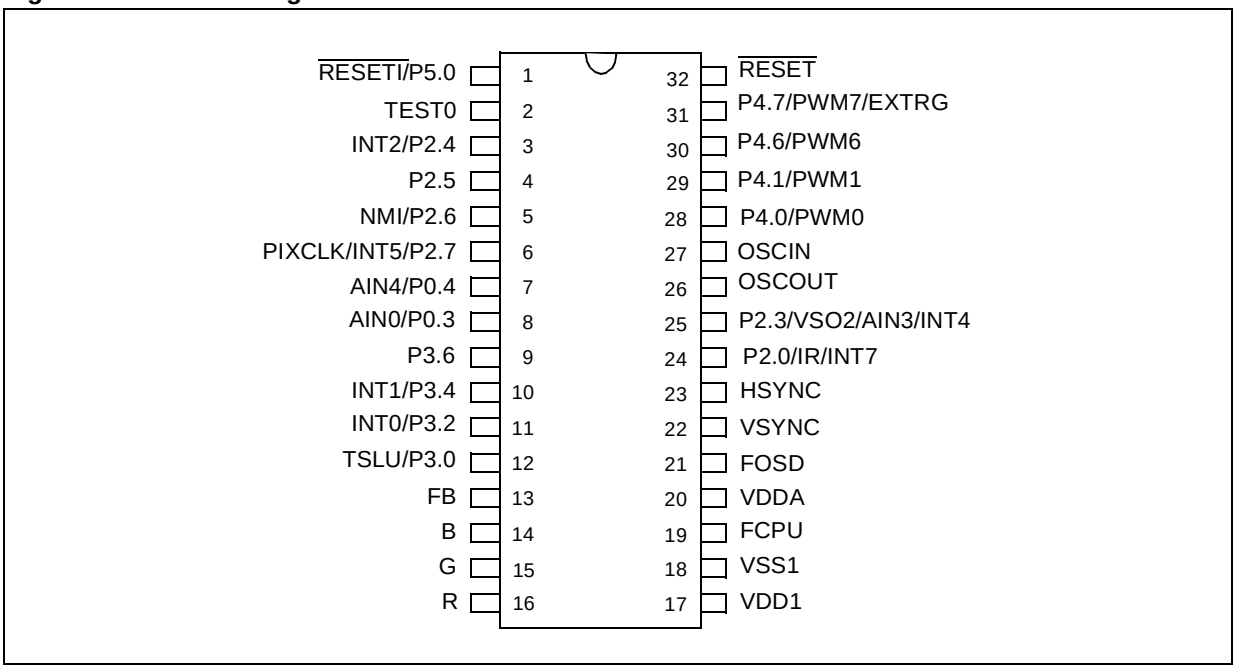
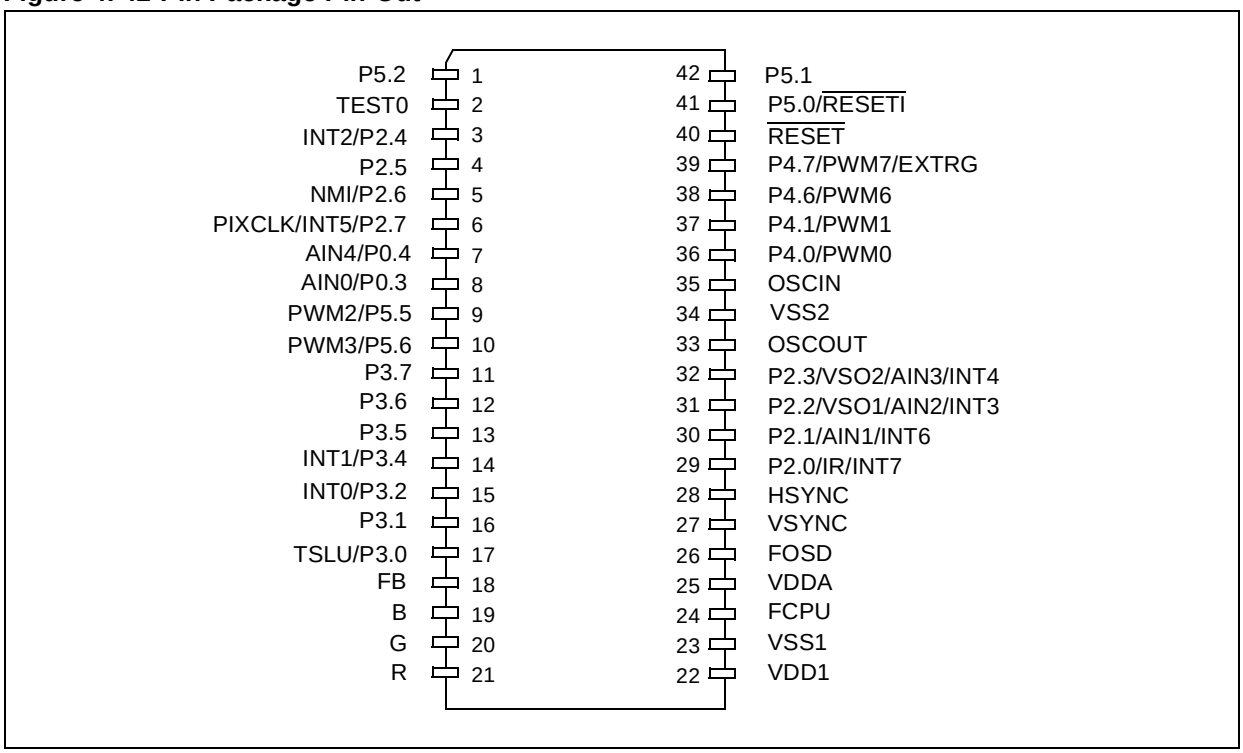


Figure 4. 42-Pin Package Pin-Out



PIN DESCRIPTION (Cont'd)

Table 1. Power Supply Pins

Name	Function	PSDIP32	PSDIP42
V_{DD1}	Main Power Supply Voltage	17	22
V_{SS1}	Analog and Digital Circuit Ground	18	23
V_{SS2}		-	34
V_{DDA}	Analog Circuit Supply Voltage	20	25

Table 2. Primary Function pins

Name	Function	PSDIP32	PSDIP42
OSCIN	Oscillator input	27	35
OSCOUT	Oscillator output	26	33
$\overline{\text{RESET}}$	Reset to initialize the ST9	32	40
HSYNC	Video Horizontal Sync Input (Schmitt trigger)	23	28
VSYNC	Video Vertical Sync input (Schmitt trigger)	22	27
R	Red video analog DAC output	16	21
G	Green video analog DAC output	15	20
B	Blue video analog DAC output	14	19
FB	Fast Blanking analog DAC output	13	18
FCPU	CPU frequency multiplier filter output	19	24
FOSD	OSD frequency multiplier filter output	21	26
TEST0	Test input (must be tied to V_{DD})	2	2

PIN DESCRIPTION (Cont'd)

1.2.1 I/O Port Configuration

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

No I/O pins have any physical weak pull-up capability (they will show no pull-up if they are programmed in the "weak pull-up" software mode).

Input levels can be selected on a bit basis by choosing between TTL or CMOS input levels for I/O port pin except for P2.(5:4,0), P3.(6:4,1:0), P4.(1:0) which are implemented with a Schmitt trigger function.

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

Warning:

Some I/Os are not bonded in the SDIP32 package. You must configure these unbonded I/Os (i.e. P2.1, P2.2, P3.1, P3.5, P3.7, P5.1, P5.2, P5.5 and P5.6) as output push-pull at the very first beginning of your software and never change this configuration after initialization. This will avoid unpredictable software behavior.

1.2.2 I/O Port Reset State

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

All I/Os are forced by the Reset in "weak pull-up" configuration mode (but some are in high impedance due to the lack of physical pull-up) except P5.0 (refer to the Reset section) which is forced into the "Push-Pull Alternate Function" mode until being reconfigured by software.

Warning:

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

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

Table 3. I/O Port Characteristics

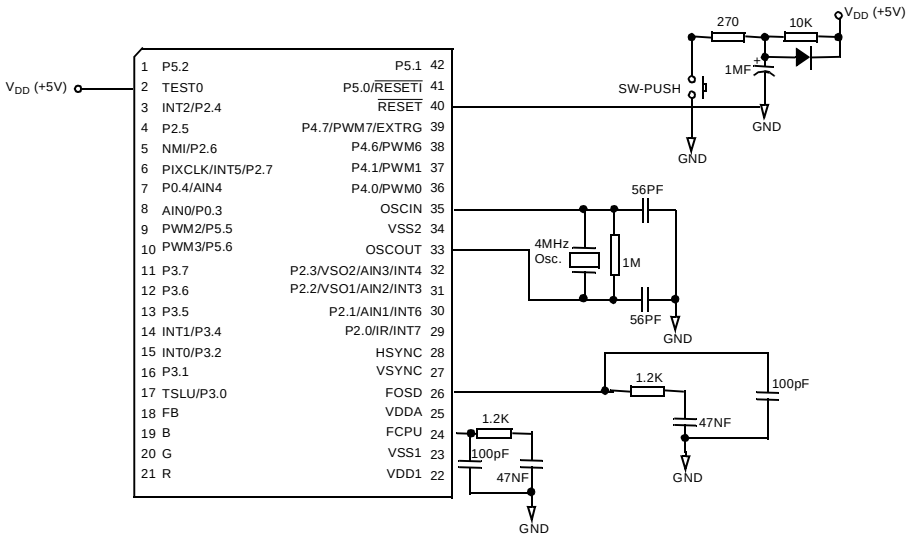
	Input	Output	Weak Pull-Up	Reset State
Port 0[4:3]	TTL/CMOS	Push-Pull/OD	No	Bidirectional
Port 2.0	Schmitt trigger	Push-Pull/OD	No	Bidirectional
Port 2[3:1]	TTL/CMOS	Push-Pull/OD	No	Bidirectional
Port 2[5:4]	Schmitt trigger	Push-Pull/OD	No	Bidirectional
Port 2[7:6]	TTL/CMOS	Push-Pull/OD	No	Bidirectional
Port 3[0:1]	Schmitt trigger	Push-Pull/OD	No	Bidirectional
Port 3.2	TTL/CMOS	Push-Pull/OD	No	Bidirectional
Port 3[6:4]	Schmitt trigger	Push-Pull/OD	No	Bidirectional
Port 3.7	TTL/CMOS	Push-Pull/OD	No	Bidirectional
Port 4[1:0]	Schmitt trigger	Push-Pull/OD	No	Bidirectional
Port 4[7:6]	TTL/CMOS	Push-Pull/OD	No	Bidirectional
Port 5.0	TTL/CMOS	Push-Pull/OD	No	Push-Pull AF Out
Port 5[2:1]	TTL/CMOS	Push-Pull/OD	No	Bidirectional
Port 5[6:5]	TTL/CMOS	Push-Pull/OD	No	Bidirectional

Legend: OD = Open Drain, AF = Alternate Function

Table 4. ST92186B Alternate Functions

Port Name	General Purpose I/O	Pin No.		Alternate Function		
		SDIP32	SDIP42			
P0.3	All ports useable for general purpose I/O (input, output or bidirectional)	8	8	AIN0	I	A/D Analog Data Input 0
P0.4		7	7	AIN4	I	A/D Analog Data Input 4
P2.0		24	29	IR	I	IFR Infrared input
				INT7		External Interrupt 7
P2.1		-	30	AIN1	I	A/D Analog Data Input 1
				INT6		External Interrupt 6
P2.2		-	31	INT3	I	External Interrupt 3
				AIN2		A/D Analog Data Input 2
				VSO1	O	Voltage Synthesis Converter Output 1
P2.3		25	32	INT4	I	External Interrupt 4
				AIN3		A/D Analog Data Input 3
				VSO2	O	Voltage Synthesis Converter Output 2
P2.4		3	3	INT2	I	External Interrupt 2
P2.5		4	4		I/O	
P2.6		5	5	NMI	I	Non Maskable Interrupt Input
P2.7		6	6	INT5	I	External Interrupt 5
				PIXCLK	O	Pixel Clock (after divide-by-2) output
P3.0		12	17	TSLU	O	Translucency Digital Video Output
P3.1		-	16		I/O	
P3.2		11	15	INT0	I	External Interrupt 0
P3.4		10	14	INT1	I	External Interrupt 1
P3.5		-	13		I/O	
P3.6		9	12		I/O	
P3.7		-	11		I/O	
P4.0		28	36	PWM0	O	PWM D/A Converter Output 0
P4.1		29	37	PWM1	O	PWM D/A Converter Output 1
P4.6		30	38	PWM6	O	PWM D/A Converter Output 6
P4.7		31	39	EXTRG	I	A/D Converter External Trigger Input
				PWM7	O	PWM D/A Converter Output 7
P5.0		1	41	$\overline{\text{RESETI}}$	O	Internal Delayed Reset Output
P5.1		-	42		I/O	
P5.2		-	1		I/O	
P5.5		-	9	PWM2	O	PWM D/A Converter Output 2
P5.6		-	10	PWM3	O	PWM D/A Converter Output 3

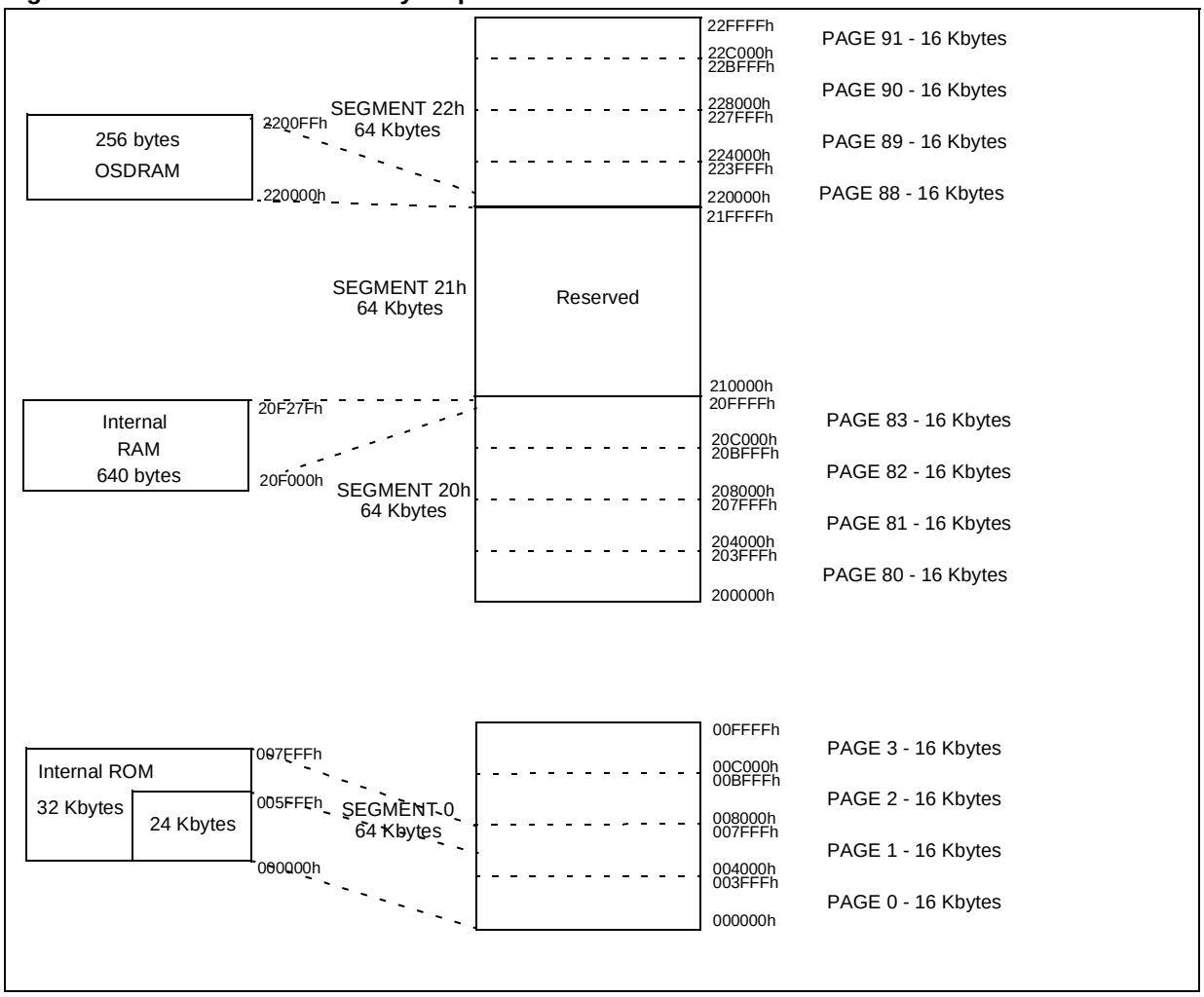
1.3 REQUIRED EXTERNAL COMPONENTS



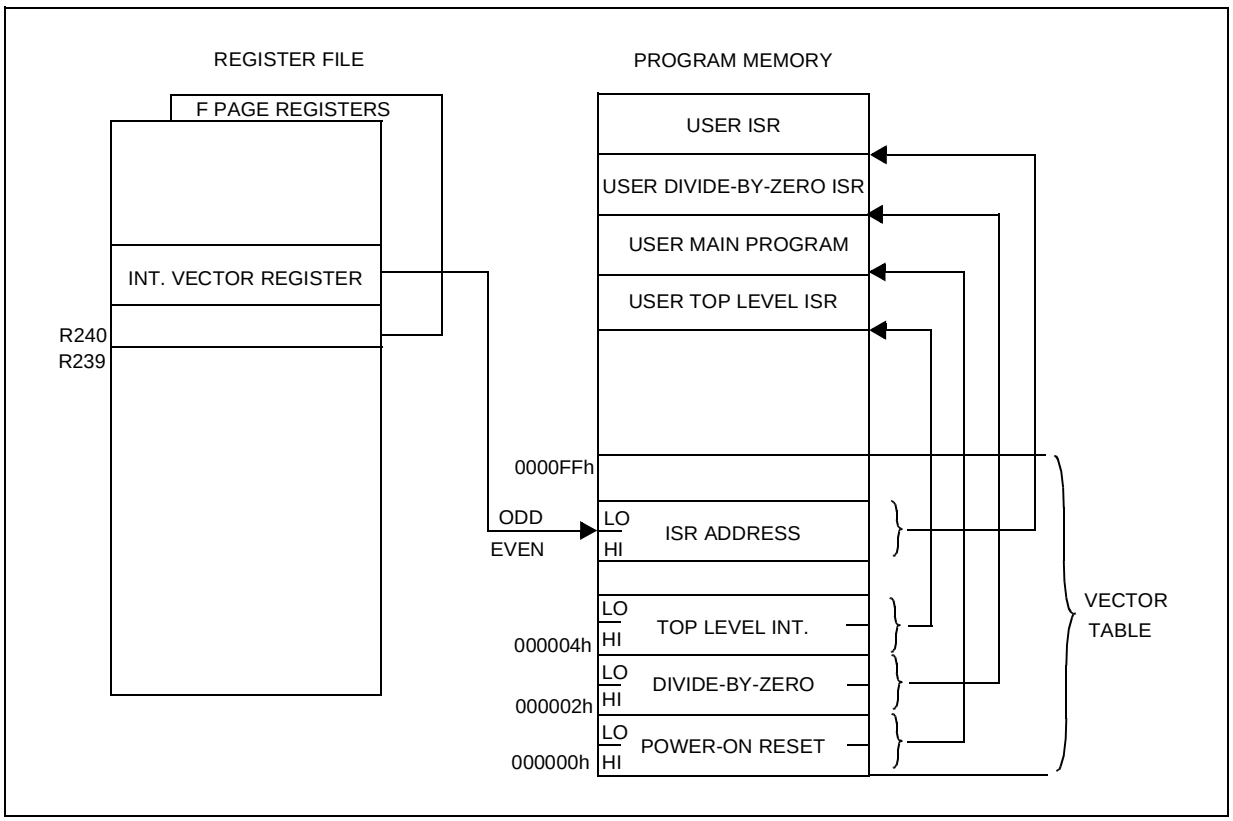
Warning: The decoupling capacitors between analog and digital +5V (V_{DDA}, V_{DD1}) and ground (V_{SS1}, V_{SS2}) are not shown.
Add a 100NF and a 4.7µF capacitor close to the corresponding pins if needed.

1.4 MEMORY MAP

Figure 5. ST92186B User Memory Map



1.5 INTERRUPT VECTOR TABLE



1.6 ST92186B REGISTER MAP

Table 6 contains the map of the group F peripheral pages.

The common registers used by each peripheral are listed in Table 5.

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 5. Common Registers

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

Table 6. Group F Pages Register Map

Resources available on the ST92186B device:

Register	Page									
	0	2	3	11	21	42	43	55	59	62
R255	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	VS	Res.
R254		Port 3					TCC			
R253									Res.	
R252	WCR					OSD	Res.		PWM	
R251	WDT	Res.								
R250		Port 2					IR			
R249							Res.			
R248					MMU		IR			
R247	EXT INT	Res.			Res		Res.			
R246			Port 5		EXTMI					
R245					Res	Res.			Res	
R244					MMU					
R243			Res.	STIM					PWM [1]	
R242		Port 0	Port 4					RCCU		ADC
R241	Res.							Res		
R240								RCCU		

Note 1: page 59, R242 and R243 are reserved on SDIP32

Table 7. Detailed Register Map

Group F Page Dec.	Block	Reg. No.	Register Name	Description	Reset Value Hex.	Doc. Page
N/A	I/O Port 0:5	R224	P0DR	Port 0 Data Register	FF	62
		R226	P2DR	Port 2 Data Register	FF	
		R227	P3DR	Port 3 Data Register	FF	
		R228	P4DR	Port 4 Data Register	FF	
		R229	P5DR	Port 5 Data Register	FF	
	Core	R230	CICR	Central Interrupt Control Register	87	52
		R231	FLAGR	Flag Register	00	26
		R232	RP0	Pointer 0 Register	00	28
		R233	RP1	Pointer 1 Register	00	28
		R234	PPR	Page Pointer Register	54	30
		R235	MODER	Mode Register	E0	30
		R236	USPHR	User Stack Pointer High Register	xx	32
		R237	USPLR	User Stack Pointer Low Register	xx	32
		R238	SSPHR	System Stack Pointer High Reg.	xx	32
		R239	SSPLR	System Stack Pointer Low Reg.	xx	32
0	INT	R242	EITR	External Interrupt Trigger Register	00	52
		R243	EIPR	External Interrupt Pending Reg.	00	53
		R244	EIMR	External Interrupt Mask-bit Reg.	00	53
		R245	EIPLR	External Interrupt Priority Level Reg.	FF	53
		R246	EIVR	External Interrupt Vector Register	x6	54
		R247	NICR	Nested Interrupt Control	00	54
	WDT	R248	WDTHR	Watchdog Timer High Register	FF	72
		R249	WDTLR	Watchdog Timer Low Register	FF	72
		R250	WDTPR	Watchdog Timer Prescaler Reg.	FF	73
		R251	WDTCR	Watchdog Timer Control Register	12	73
		R252	WCR	Wait Control Register	7F	74
2	I/O Port 0	R240	P0C0	Port 0 Configuration Register 0	00	62
		R241	P0C1	Port 0 Configuration Register 1	00	
		R242	P0C2	Port 0 Configuration Register 2	00	
	I/O Port 2	R248	P2C0	Port 2 Configuration Register 0	00	
		R249	P2C1	Port 2 Configuration Register 1	00	
		R250	P2C2	Port 2 Configuration Register 2	00	
	I/O Port 3	R252	P3C0	Port 3 Configuration Register 0	00	
		R253	P3C1	Port 3 Configuration Register 1	00	
		R254	P3C2	Port 3 Configuration Register 2	00	

Group F Page Dec.	Block	Reg. No.	Register Name	Description	Reset Value Hex.	Doc. Page
3	I/O Port 4	R240	P4C0	Port 4 Configuration Register 0	00	62
		R241	P4C1	Port 4 Configuration Register 1	00	
		R242	P4C2	Port 4 Configuration Register 2	00	
	I/O Port 5	R244	P5C0	Port 5 Configuration Register 0	00	
		R245	P5C1	Port 5 Configuration Register 1	00	
		R246	P5C2	Port 5 Configuration Register 2	00	
11	STIM	R240	STH	Counter High Byte Register	FF	77
		R241	STL	Counter Low Byte Register	FF	77
		R242	STP	Standard Timer Prescaler Register	FF	77
		R243	STC	Standard Timer Control Register	14	77
21	MMU	R240	DPR0	Data Page Register 0	00	37
		R241	DPR1	Data Page Register 1	01	37
		R242	DPR2	Data Page Register 2	02	37
		R243	DPR3	Data Page Register 3	83	37
		R244	CSR	Code Segment Register	00	38
		R248	ISR	Interrupt Segment Register	x0	38
		R249	Reserved			
	EXTMI	R246	EMR2	External Memory Register 2	0F	55
42	OSD	R246	OSDBCR2	Border Color Register 2	x0	115
		R247	OSDBCR1	Border Color Register 1	x0	115
		R248	OSDER	Enable Register	00	116
		R249	OSDDR	Delay Register	xx	118
		R250	OSDFBR	Flag Bit Register	xx	119
		R251	OSDSLR	Scan Line Register	xx	118
		R252	OSDMR	Mute Register	xx	120
43	IR	R248	IRPR	Infrared Pulse Register	00	122
		R250	IRSCR	Infrared / Sync Control Register	00	122
	TCC	R253	MCCR	Main Clock Control Register	00	61
		R254	SKCCR	Skew Clock Control Register	00	61
55	RCCU	R240	CLKCTL	Clock Control Register	00	56
		R242	CLK_FLAG	Clock Flag Register	48, 28 or 08	56

Group F Page Dec.	Block	Reg. No.	Register Name	Description	Reset Value Hex.	Doc. Page
59	PWM	R240	CM0	Compare Register 0	00	130
		R241	CM1	Compare Register 1	00	130
		R242	CM2 [1)]	Compare Register 2	00	130
		R243	CM3 [1)]	Compare Register 3	00	130
		R244	Reserved			
		R245				
		R246	CM6	Compare Register 6	00	130
		R247	CM7	Compare Register 7	00	130
		R248	ACR	Autoclear Register	FF	131
		R249	CCR	Counter Register	00	131
		R250	PCTL	Prescaler and Control Register	0C	131
		R251	OCPL	Output Complement Register	00	132
		R252	OER	Output Enable Register	00	132
	VS	R254	VSDR1	Data and Control Register 1	00	127
		R255	VSDR2	Data Register 2	00	127
62	ADC	R240	ADDTR	Channel i Data Register	xx	135
		R241	ADCLR	Control Logic Register	00	135
		R242	ADINT	AD Interrupt Register	01	136

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

Note 1: R242 and R243 are reserved on SDIP32

2 DEVICE ARCHITECTURE

2.1 CORE ARCHITECTURE

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

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

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

2.2 MEMORY SPACES

There are two separate memory spaces:

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

– A single linear memory space accommodating both program and data. All of the physically separate memory areas, including the internal ROM, internal RAM and external memory are mapped in this common address space. The total addressable memory space of 4 Mbytes (limited by the size of on-chip memory and the number of external address pins) is arranged as 64 segments of 64 Kbytes. Each segment is further subdivided into four pages of 16 Kbytes, as illustrated in Figure 6. A Memory Management Unit uses a set of pointer registers to address a 22-bit memory field using 16-bit address-based instructions.

2.2.1 Register File

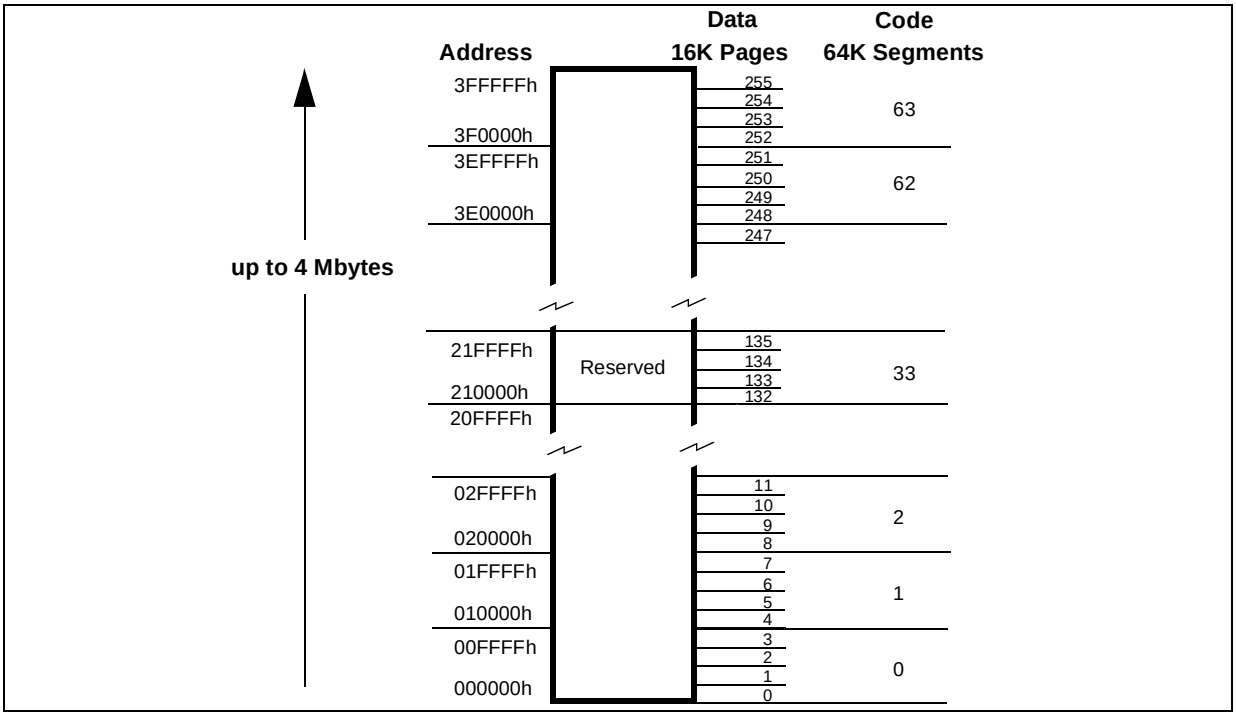
The Register File consists of (see Figure 7):

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

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

– Up to 64 pages, depending on device configuration, each containing up to 16 registers, mapped to Group F (R240 to R255), see Figure 8.

Figure 6. Single Program and Data Memory Address Space



MEMORY SPACES (Cont'd)

Figure 7. Register Groups

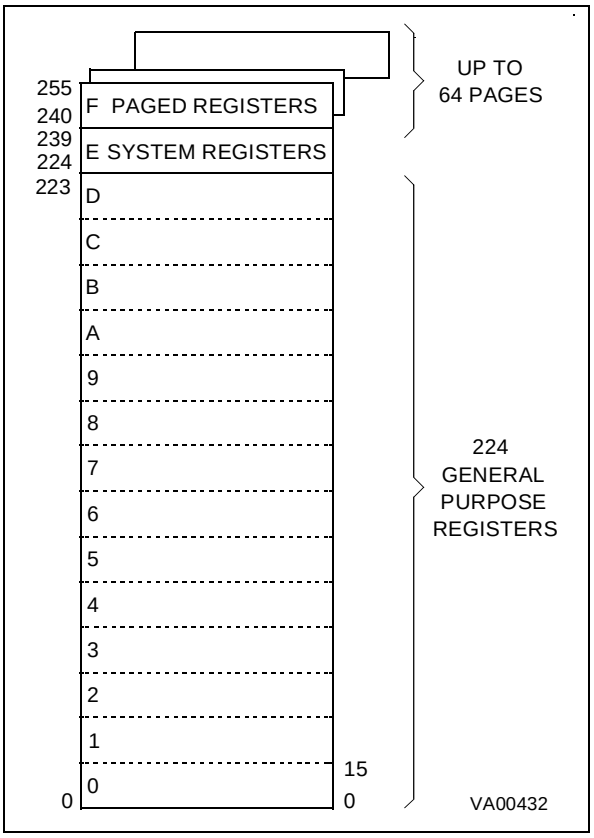


Figure 8. Page Pointer for Group F mapping

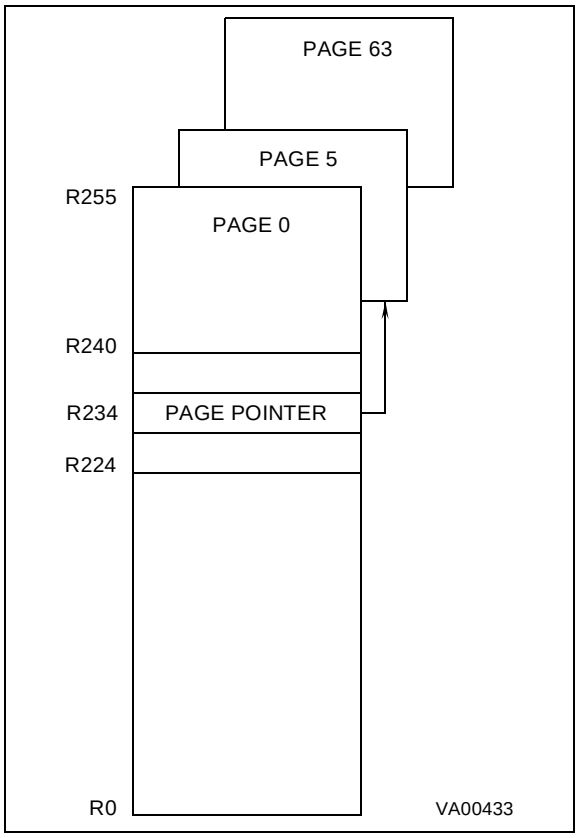
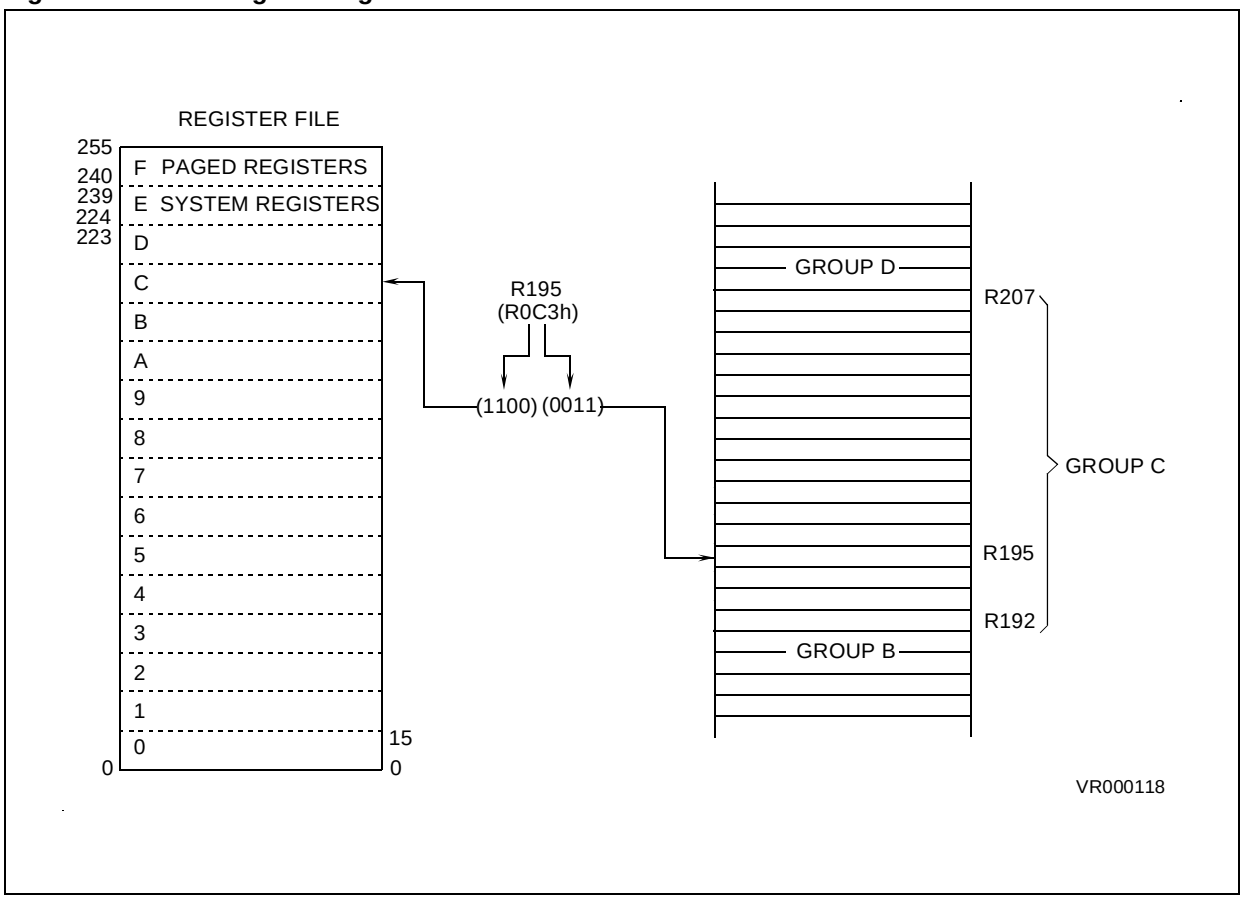


Figure 9. Addressing the Register File



MEMORY SPACES (Cont'd)

2.2.2 Register Addressing

Register File registers, including Group F paged registers (but excluding Group D), may be addressed explicitly by means of a decimal, hexadecimal or binary address; thus R231, RE7h and R11100111b represent the same register (see Figure 9). 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, and illustrated in Figure 10 and in Figure 11.

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.

Paged Registers

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

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

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

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

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

Table 8. 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 9. They are used to perform all the important system settings. Their purpose is described in the following pages. Refer to the chapter dealing with I/O for a description of the Port[5:2] and Port0 Data registers.

Table 9. 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)	Reserved
R224 (E0h)	PORT0 DATA REG.

2.3.1 Central Interrupt Control Register

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

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

7							0
-	TLIP	TLI	IEN	IAM	CPL2	CPL1	CPL0

Bit 7 = Reserved.

This bit must be kept at 1.

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

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

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

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

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

SYSTEM REGISTERS (Cont'd)

2.3.2 Flag Register

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

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

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

7							0
C	Z	S	V	DA	H	-	DP

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

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

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

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

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

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

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

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

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

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

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

Bit 1 = Reserved bit (must be 0).

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

SYSTEM REGISTERS (Cont'd)

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

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

2.3.3 Register Pointing Techniques

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

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

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

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

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

Thus:

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

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

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

SYSTEM REGISTERS (Cont'd)

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

7							0
RG4	RG3	RG2	RG1	RG0	RPS	0	0

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

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

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

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

7							0
RG4	RG3	RG2	RG1	RG0	RPS	0	0

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

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

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

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

SYSTEM REGISTERS (Cont'd)

Figure 10. Pointing to a single group of 16 registers

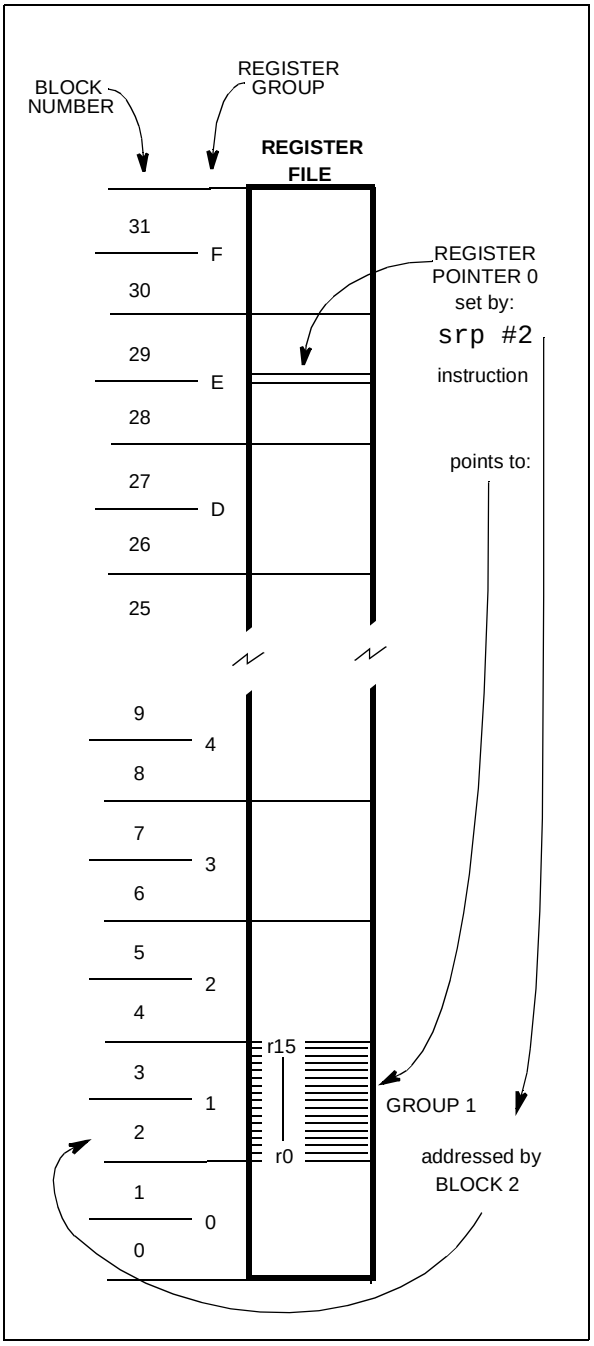
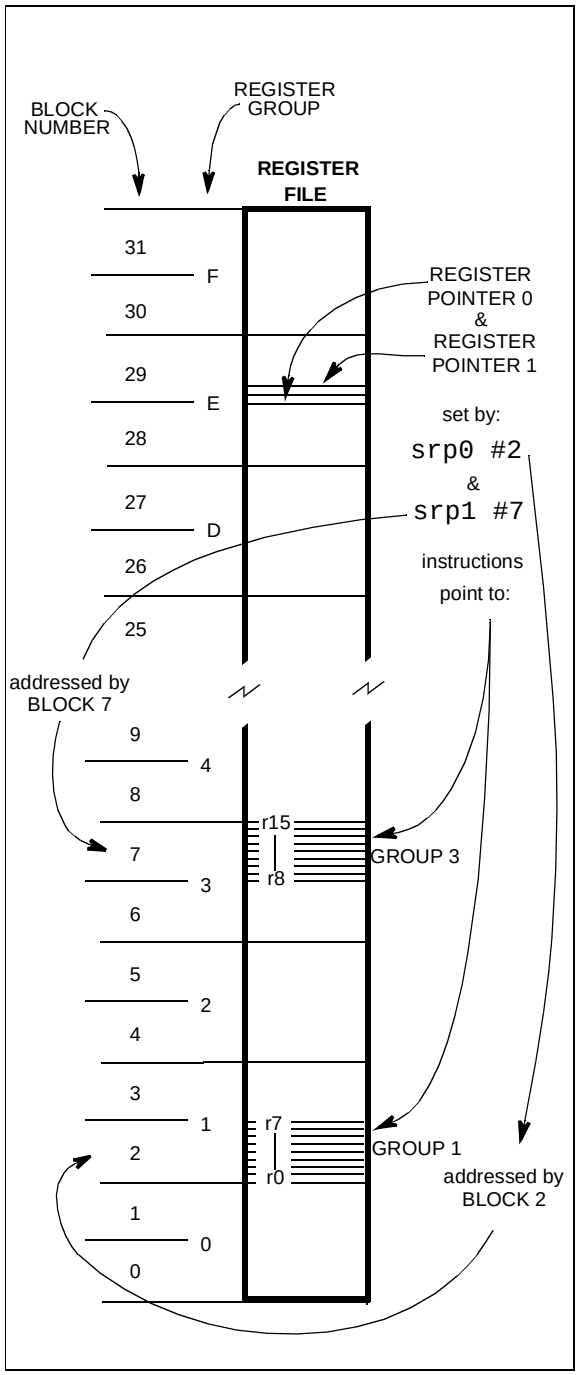


Figure 11. Pointing to two groups of 8 registers



SYSTEM REGISTERS (Cont'd)

2.3.4 Paged Registers

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

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

Thus the instructions:

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

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

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

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

7							0
PP5	PP4	PP3	PP2	PP1	PP0	0	0

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

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

2.3.5 Mode Register

The Mode Register allows control of the following operating parameters:

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

– Management of the clock frequency,

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

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

7							0
SSP	USP	DIV2	PRS2	PRS1	PRS0	BRQEN	HIMP

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

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

Bit 5 = **DIV2**: *OSCIN Clock Divided by 2*.
This bit controls the divide-by-2 circuit operating on OSCIN.
0: Clock divided by 1
1: Clock divided by 2

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

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

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

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

SYSTEM REGISTERS (Cont'd)

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

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

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

2.3.6 Stack Pointers

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

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

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

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

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

System Stack

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

The following automatically push data onto the System Stack:

– Interrupts

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

– Subroutine Calls

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

– Link Instruction

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

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

User Stack

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

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

Stack Pointers

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

The stack pointer registers are located in the System Group of the Register File, this is illustrated in Table 9.

Stack location

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

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

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

SYSTEM REGISTERS (Cont'd)

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

7							0
USP15	USP14	USP13	USP12	USP11	USP10	USP9	USP8

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

7							0
USP7	USP6	USP5	USP4	USP3	USP2	USP1	USP0

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

7							0
SSP15	SSP14	SSP13	SSP12	SSP11	SSP10	SSP9	SSP8

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

7							0
SSP7	SSP6	SSP5	SSP4	SSP3	SSP2	SSP1	SSP0

Figure 12. Internal Stack Mode

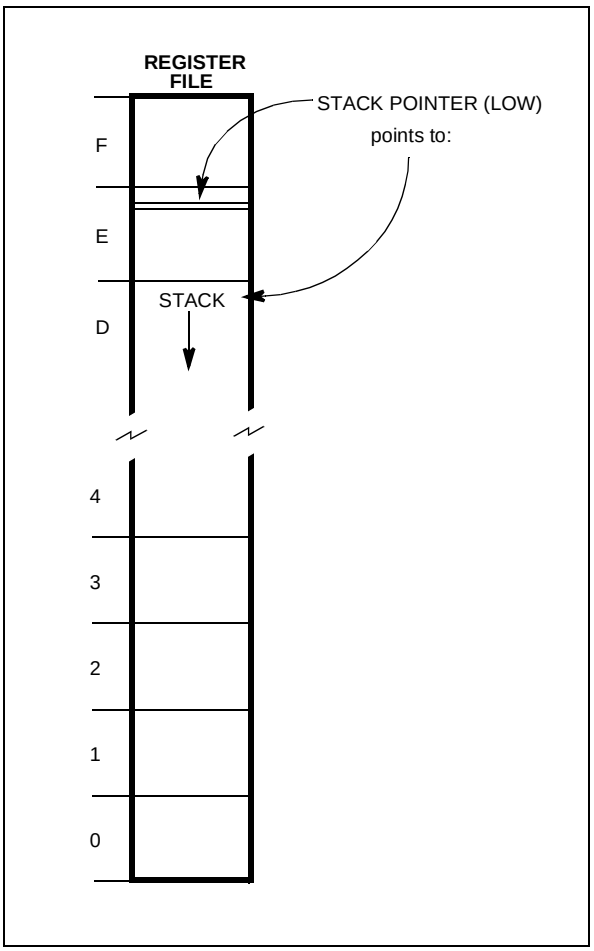
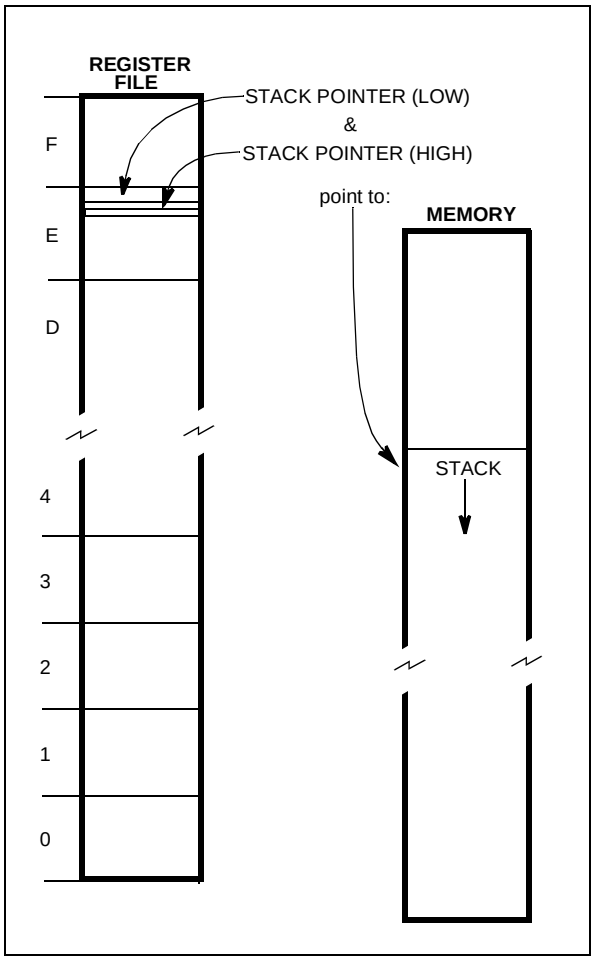


Figure 13. External Stack Mode



2.4 MEMORY ORGANIZATION

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

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

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

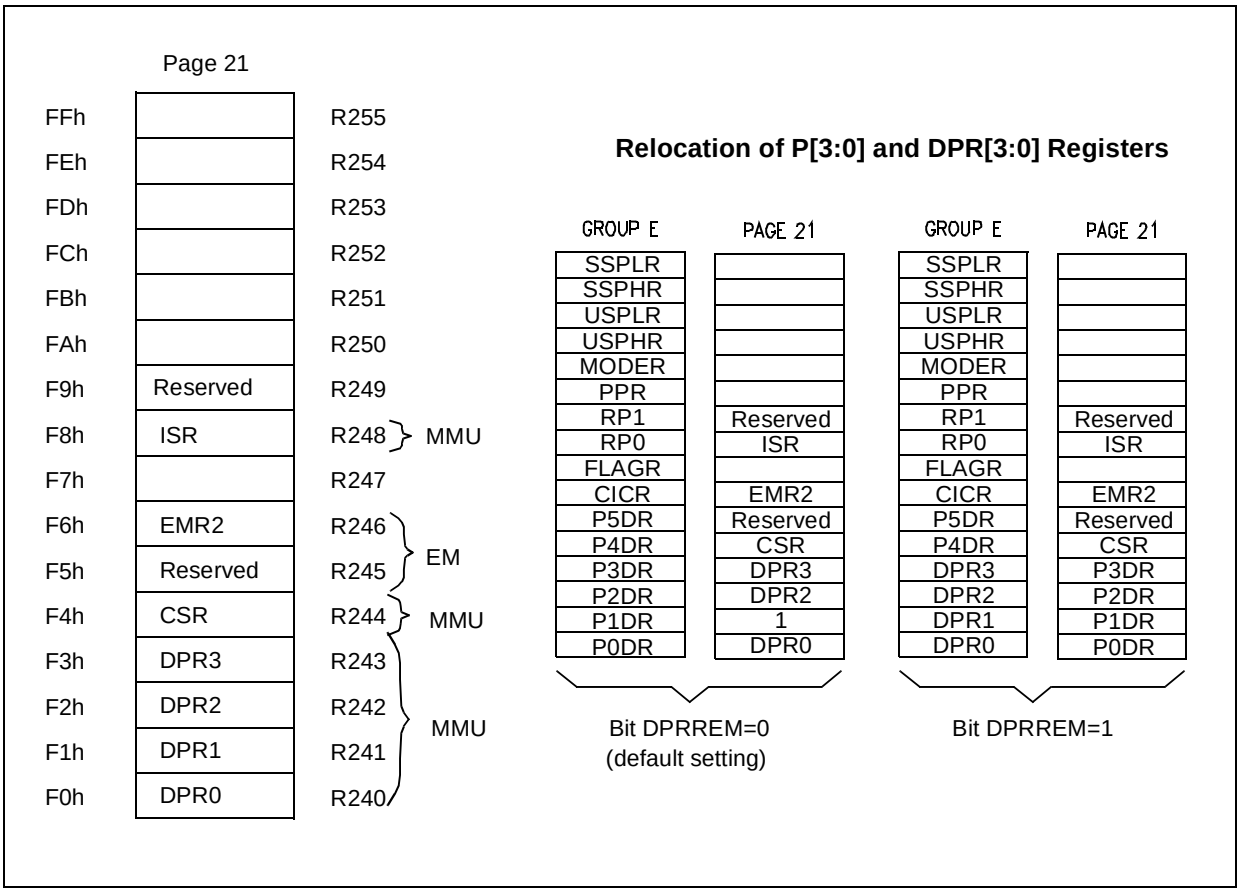
Refer to the Register and Memory Map Chapter for more details on the memory map.

2.5 MEMORY MANAGEMENT UNIT

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

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

Figure 14. 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.

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

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

DPR0: from 0000h to 3FFFh;

DPR1: from 4000h to 7FFFh;

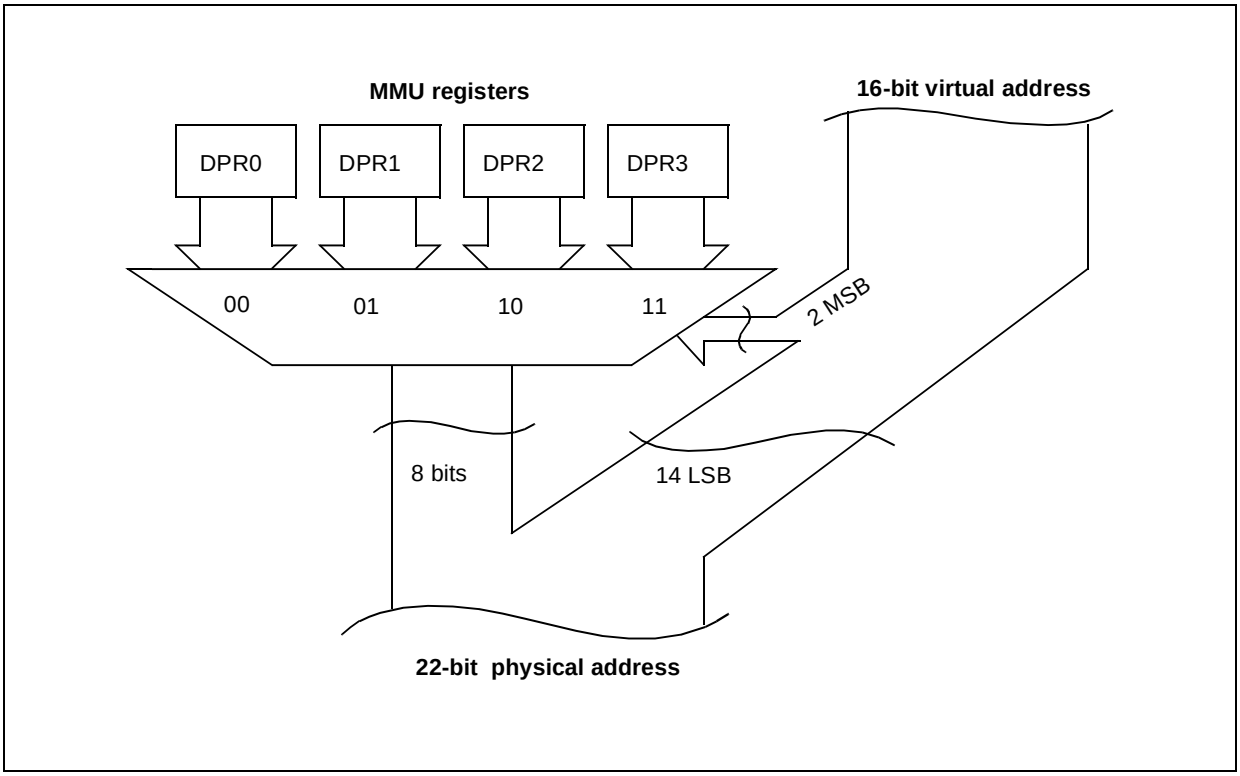
DPR2: from 8000h to BFFFh;

DPR3: from C000h to FFFFh.

The contents of the selected DPR register specify one of the 256 possible data memory pages. This 8-bit data page number, in addition to the remaining 14-bit page offset address forms the physical 22-bit address (see Figure 15).

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 15. Addressing via DPR[3:0]



ADDRESS SPACE EXTENSION (Cont'd)

2.6.2 Addressing 64-Kbyte Segments

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

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

2.7 MMU REGISTERS

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

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

2.7.1 DPR[3:0]: Data Page Registers

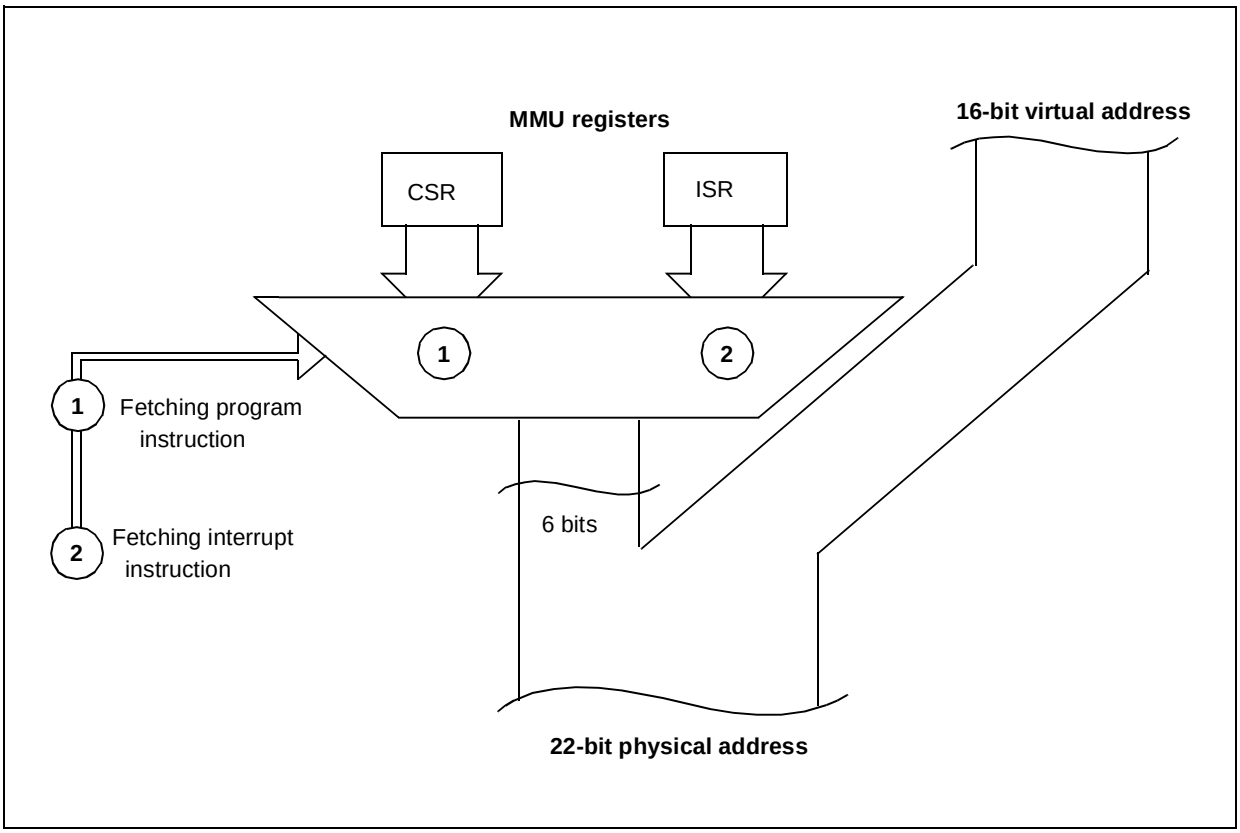
The DPR[3:0] registers allow access to the entire 4 Mbyte memory space composed of 256 pages of 16 Kbytes.

2.7.1.1 Data Page Register Relocation

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

Data Page Register relocation is illustrated in Figure 14.

Figure 16. Addressing via CSR and ISR



MMU REGISTERS (Cont'd)

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

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

7							0
DPR0_7	DPR0_6	DPR0_5	DPR0_4	DPR0_3	DPR0_2	DPR0_1	DPR0_0

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

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

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

7							0
DPR1_7	DPR1_6	DPR1_5	DPR1_4	DPR1_3	DPR1_2	DPR1_1	DPR1_0

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

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

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

7							0
DPR2_7	DPR2_6	DPR2_5	DPR2_4	DPR2_3	DPR2_2	DPR2_1	DPR2_0

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

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

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

7							0
DPR3_7	DPR3_6	DPR3_5	DPR3_4	DPR3_3	DPR3_2	DPR3_1	DPR3_0

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

MMU REGISTERS (Cont'd)

2.7.2 CSR: Code Segment Register

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

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

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

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

7							0
0	0	CSR_5	CSR_4	CSR_3	CSR_2	CSR_1	CSR_0

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

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

2.7.3 ISR: Interrupt Segment Register

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

7							0
0	0	ISR_5	ISR_4	ISR_3	ISR_2	ISR_1	ISR_0

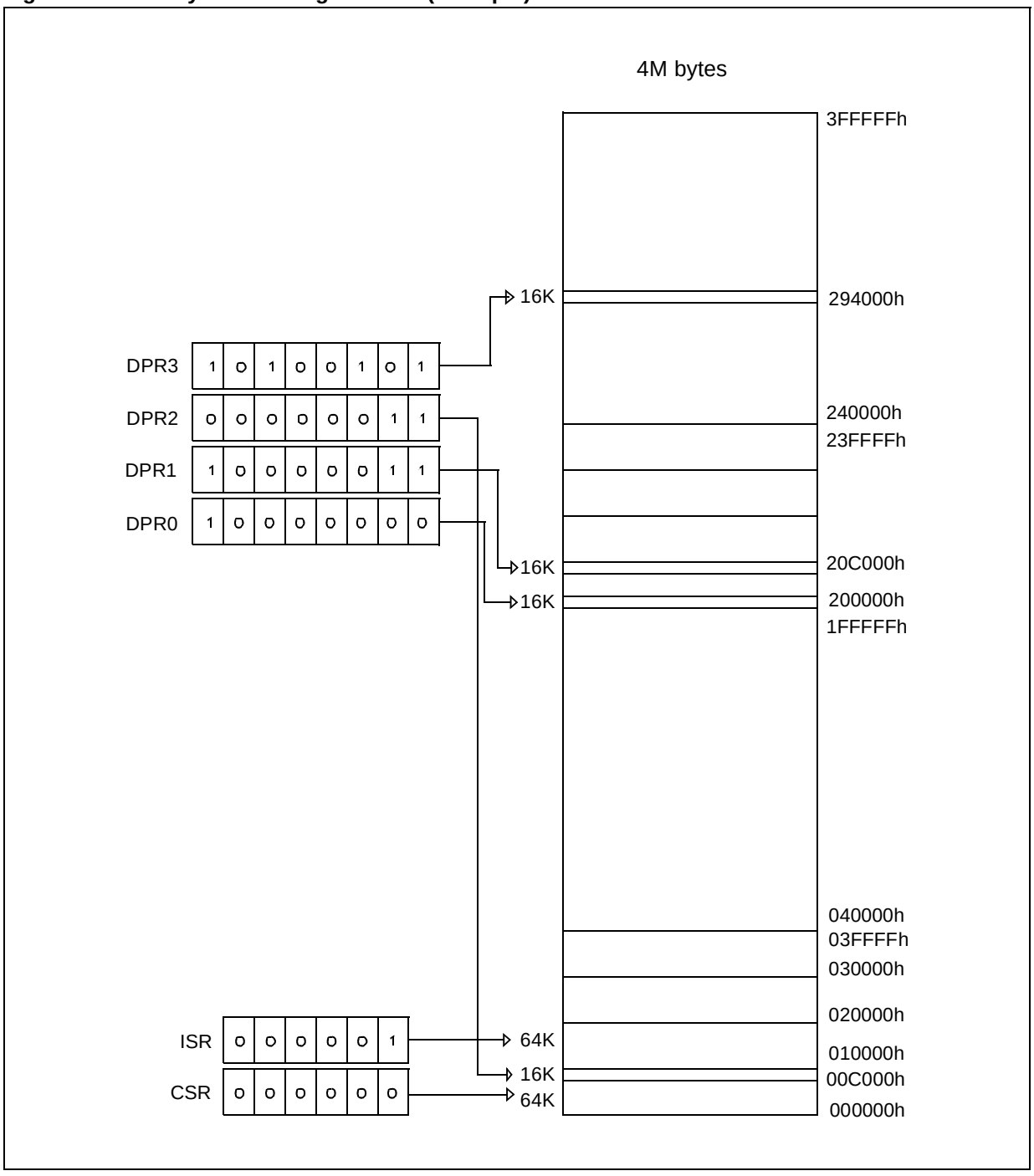
ISR and ENCSR bit (EMR2 register) are also described in the chapter relating to Interrupts, please refer to this description for further details.

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

Bits 5:0 = **ISR_[5:0]**: These bits define the 64-Kbyte memory segment (among 64) which contains the interrupt vector table and the code for interrupt service routines. These bits are used as the most significant address bits (A21-16). The ISR is used to extend the address space whenever an interrupt occurs: ISR points to the 64-Kbyte memory segment containing the interrupt vector table and the interrupt service routine code. See also the Interrupts chapter.

MMU REGISTERS (Cont'd)

Figure 17. Memory Addressing Scheme (example)



2.8 MMU USAGE

2.8.1 Normal Program Execution

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

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

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

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

2.8.2 Interrupts

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

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

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

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

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

3 INTERRUPTS

3.1 INTRODUCTION

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

The ST9 CPU can receive requests from the following sources:

– On-chip peripherals

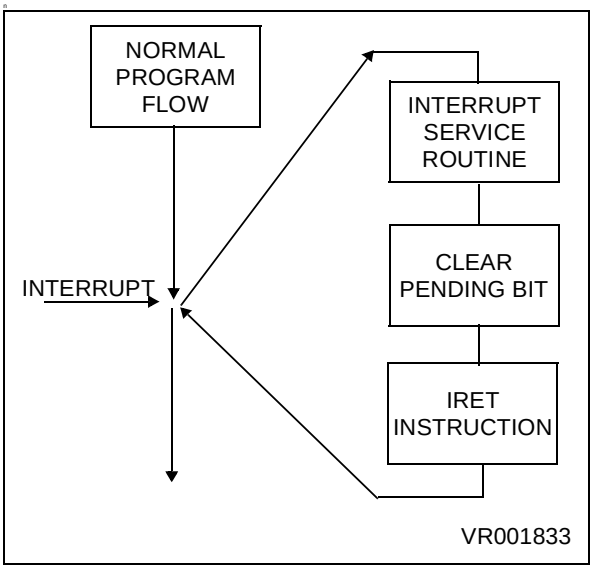
– External pins

– Top-Level Pseudo-non-maskable interrupt

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

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

Figure 18. Interrupt Response



3.2 INTERRUPT VECTORING

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

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

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

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

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

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

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

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

3.2.1 Divide by Zero trap

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

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

INTERRUPT VECTORING (Cont'd)

3.2.2 Segment Paging During Interrupt Routines

The ENCSR bit in the EMR2 register can be used to select whether the CSR is saved or not when an interrupt occurs.

For a description of the EMR2 register, see page 55.

ENCSR = 0

If ENCSR is reset, for the duration of the interrupt service routine, ISR is used instead of CSR and 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. This mode ensures compatibiliy with the original ST9.

ENCSR = 1

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

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

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

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

3.3 INTERRUPT PRIORITY LEVELS

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

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

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

3.4 PRIORITY LEVEL ARBITRATION

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

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

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

3.4.1 Priority level 7 (Lowest)

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

3.4.2 Maximum depth of nesting

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

PRIORITY LEVEL ARBITRATION (Cont'd)

3.4.3 Simultaneous Interrupts

If two or more requests occur at the same time and at the same priority level, an on-chip daisy chain, specific to every ST9 version, selects the channel with the highest position in the chain, as shown in Table 10

Table 10. Daisy Chain Priority

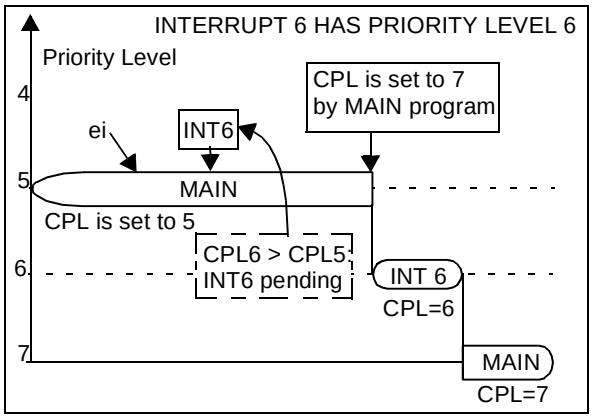
Highest Position	INTA0	INT0/WDT
	INTA1	INT1/STIM
	INTB0	INT2
	INTB1	INT3 [1)]
	INTC0	INT4 /OSD
	INTC1	INT5/ADC
	INTD0	INT6 [1)]
Lowest Position	INTD1	INT7/IR

Note 1: available on SDIP42 only

3.4.4 Dynamic Priority Level Modification

The main program and routines can be specifically prioritized. Since the CPL is represented by 3 bits in a read/write register, it is possible to modify dynamically the current priority value during program execution. This means that a critical section can have a higher priority with respect to other interrupt requests. Furthermore it is possible to prioritize even the Main Program execution by modifying the CPL during its execution. See Figure 19

Figure 19. Example of Dynamic priority level modification in Nested Mode



3.5 ARBITRATION MODES

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

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

3.5.1 Concurrent Mode

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

Start of Interrupt Routine

The interrupt cycle performs the following steps:

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

– The PC low byte is pushed onto system stack.

– The PC high byte is pushed onto system stack.

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

– The Flag register is pushed onto system stack.

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

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

End of Interrupt Routine

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

– The Flag register is popped from system stack.

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

– The PC high byte is popped from system stack.

– The PC low byte is popped from system stack.

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

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

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

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

ARBITRATION MODES (Cont'd)

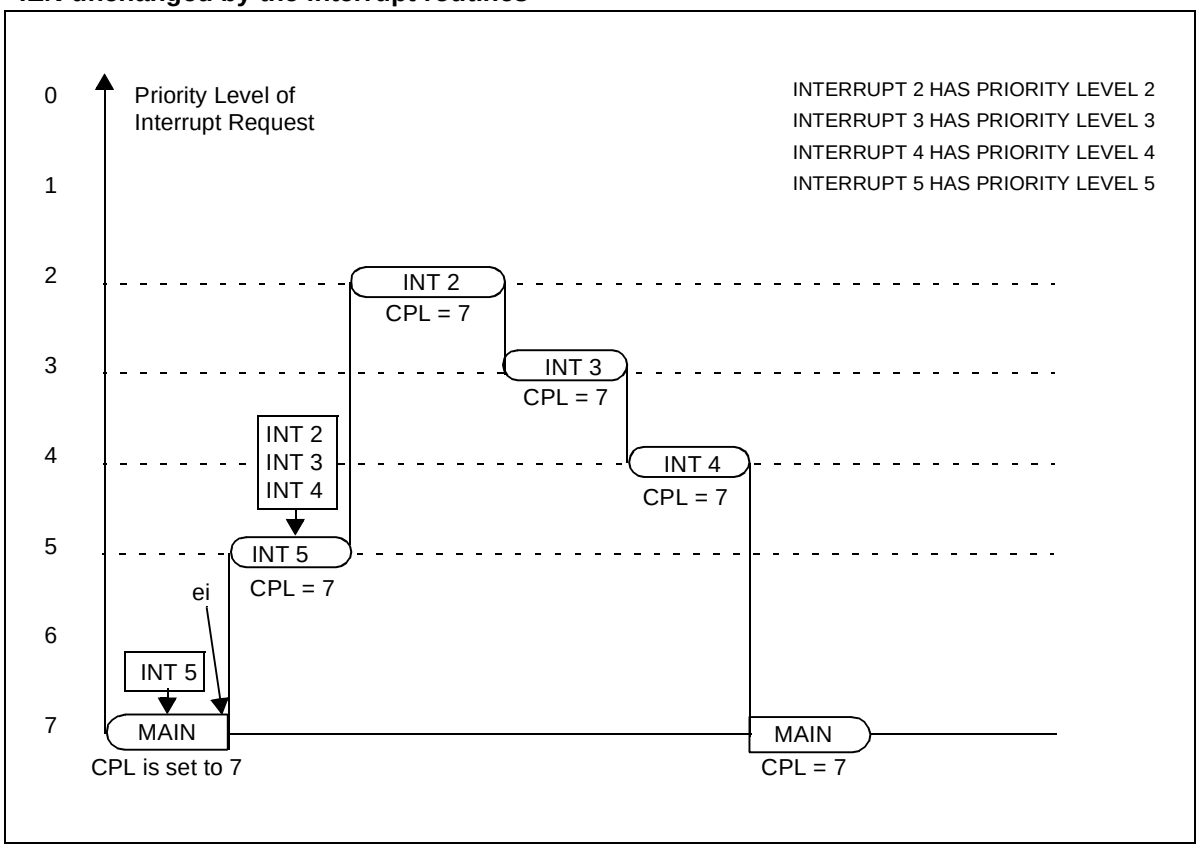
Examples

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

Example 1

In the first example, (simplest case, Figure 20) the `ei` instruction is not used within the interrupt service routines. This means that no new interrupt can be serviced in the middle of the current one. The interrupt routines will thus be serviced one after another, in the order of their priority, until the main program eventually resumes.

Figure 20. Simple Example of a Sequence of Interrupt Requests with:
- Concurrent mode selected and
- IEN unchanged by the interrupt routines



ARBITRATION MODES (Cont'd)

Example 2

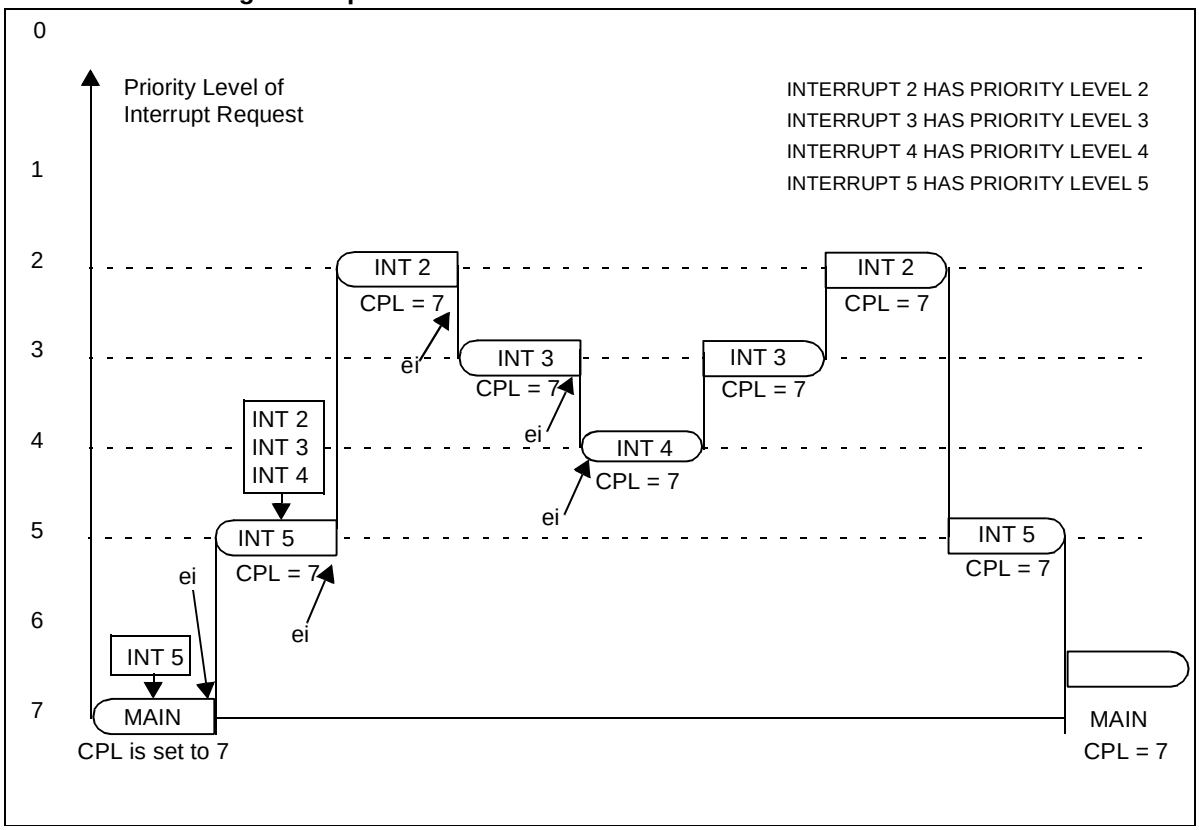
In the second example, (more complex, Figure 21), each interrupt service routine sets Interrupt Enable with the `ei` instruction at the beginning of the routine. Placed here, it minimizes response time for requests with a higher priority than the one being serviced.

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

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

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

Figure 21. Complex Example of a Sequence of Interrupt Requests with:
- Concurrent mode selected
- IEN set to 1 during interrupt service routine execution



ARBITRATION MODES (Cont'd)

3.5.2 Nested Mode

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

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

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

Start of Interrupt Routine

The interrupt cycle performs the following steps:

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

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

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

– The PC low byte is pushed onto system stack.

– The PC high byte is pushed onto system stack.

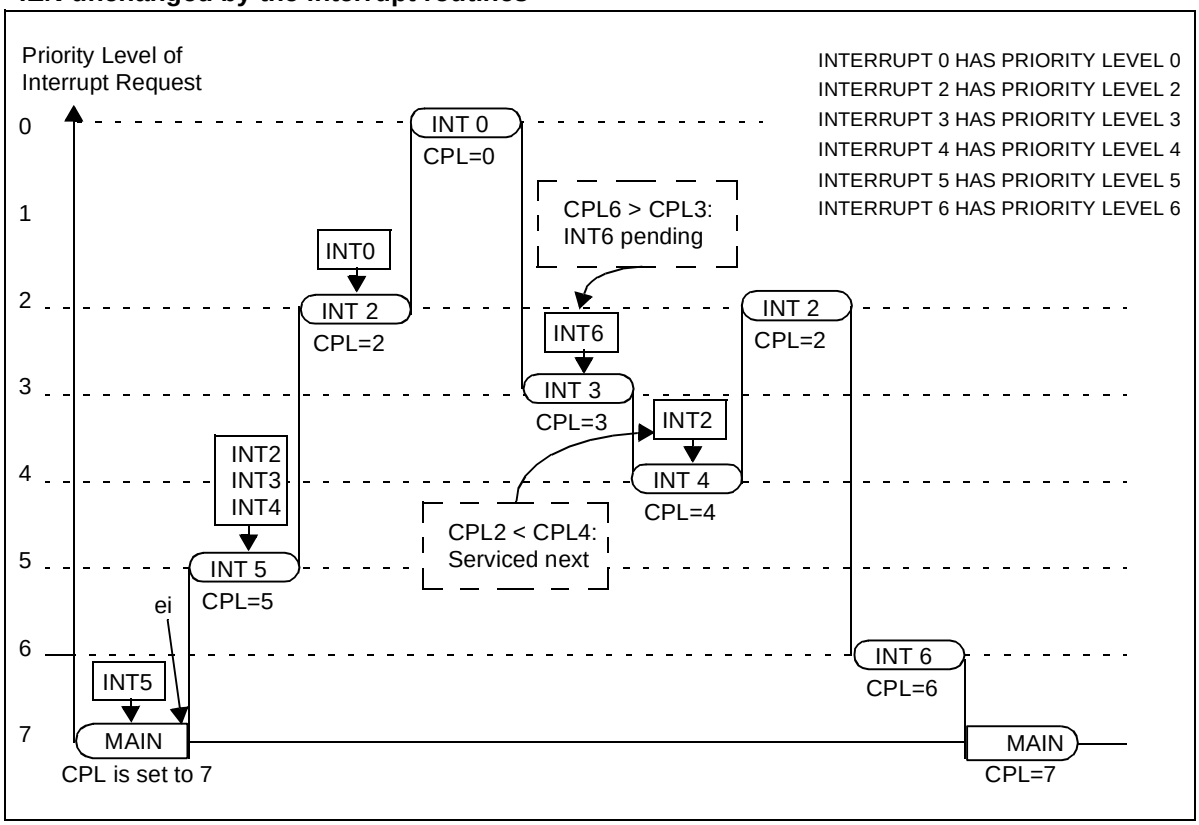
– If ENCSR is set, CSR is pushed onto system stack.

– The Flag register is pushed onto system stack.

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

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

Figure 22. Simple Example of a Sequence of Interrupt Requests with:
- Nested mode
- IEN unchanged by the interrupt routines



ARBITRATION MODES (Cont'd)

End of Interrupt Routine

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

– The Flag register is popped from system stack.

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

– The PC high byte is popped from system stack.

– The PC low byte is popped from system stack.

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

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

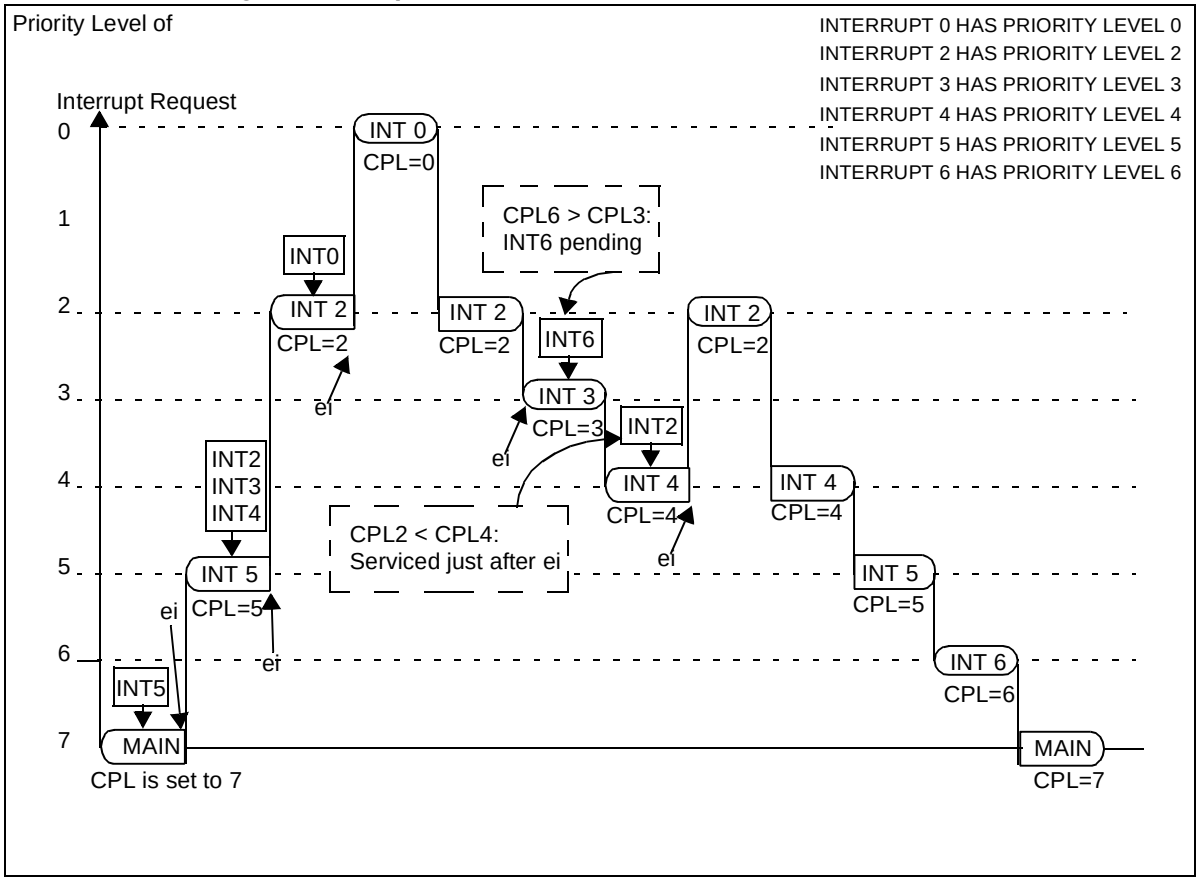
– If ENCSR is reset, CSR is used instead of ISR, unless the program returns to another nested routine.

The suspended routine thus resumes at the interrupted instruction.

Figure 22 contains a simple example, showing that if the `ei` instruction is not used in the interrupt service routines, nested and concurrent modes are equivalent.

Figure 23 contains a more complex example showing how nested mode allows nested interrupt processing (enabled inside the interrupt service routinesi using the `ei` instruction) according to their priority level.

Figure 23. Complex Example of a Sequence of Interrupt Requests with:
- Nested mode
- IEN set to 1 during the interrupt routine execution



3.6 EXTERNAL INTERRUPTS

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

Table 11. External Interrupt Channel Grouping

External Interrupt	Channel
INT7 INT6 [1)]	INTD1 INTD0 [1)]
INT5 INT4	INTC1 INTC0
INT3 [1)] INT2	INTB1 [1)] INTB0
INT1 INT0	INTA1 INTA0

Note 1: available on SDIP42 only

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 25.

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 24. Priority Level Examples

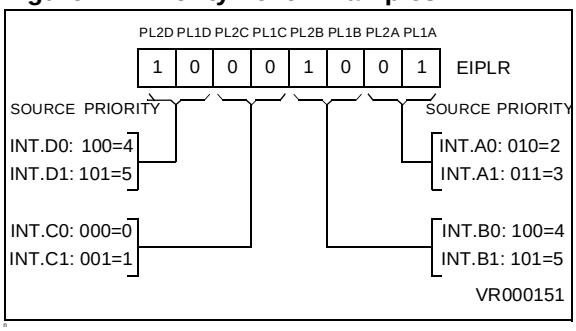


Figure 24 shows an example of priority levels.

Figure 25 gives an overview of the External interrupt control bits and vectors.

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

– The source of the interrupt channel A1 can be selected between the external pin INT4 (when INTS = “1”) or the on-chip Standard Timer.

– The source of the interrupt channel INTC0 can be selected between the INT4 external pin (DION=OSDE=0) or the Display Controller interrupt (all other cases) by programming the DION, OSDE bits in the OSDER register.

– The source of the interrupt channel C1 can be selected between the external pin INT5 (when the AD_INT bit in the AD-INT register=0) or the on-chip ADC (when AD-INT=1).

– The source of the interrupt channel D1 can be selected between the external pin INT7 (when the IRWDIS bit in the IRSC register = 1) or the on-chip IR (when IRWDIS=0).

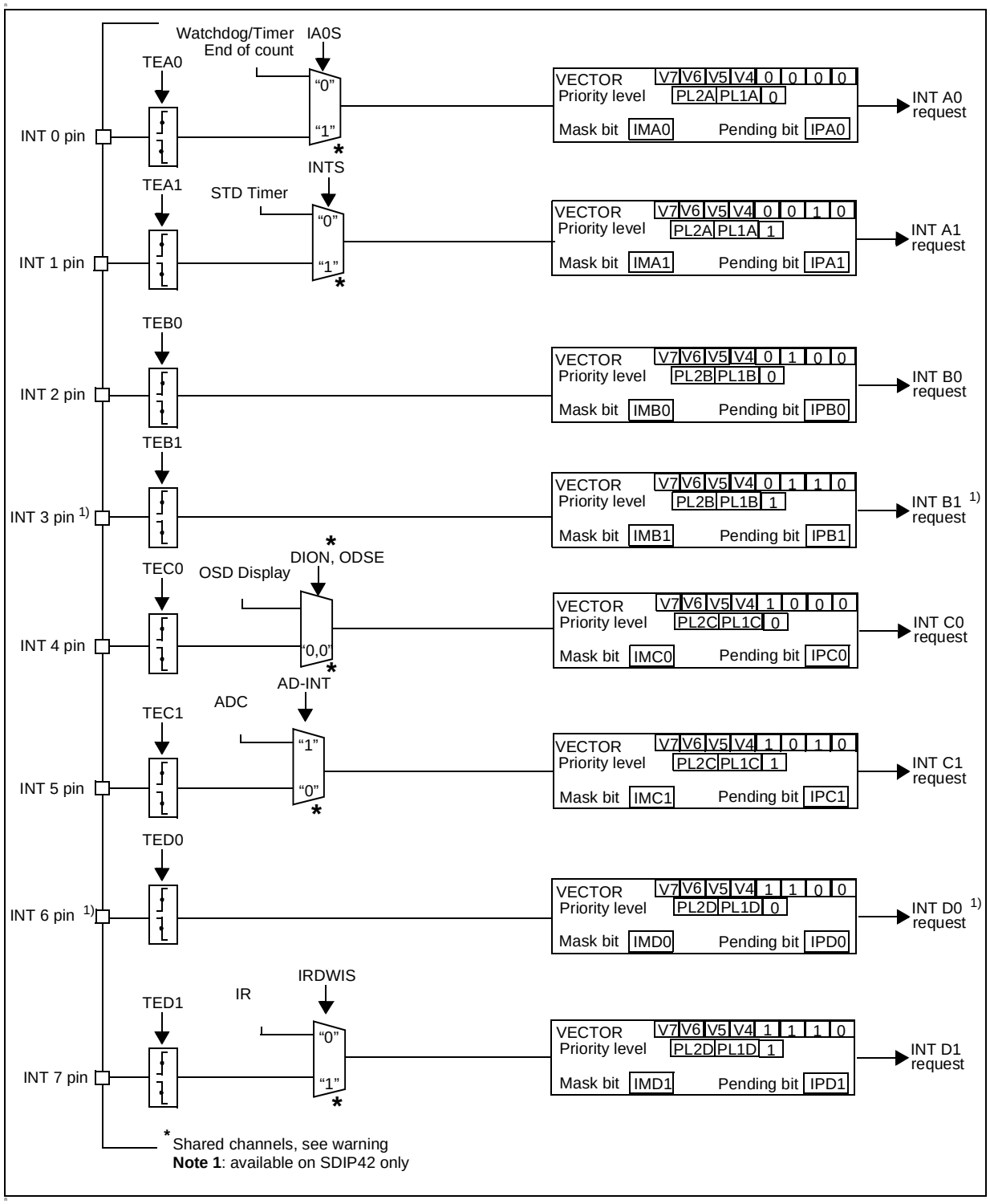
Warning: When using channels shared by both external interrupts and peripherals, special care must be taken to configure their control registers for both peripherals and interrupts.

Table 12. Multiplexed Interrupt Sources

Channel	Internal Interrupt Source	External Interrupt Source	Related Pin	
			SDIP42	SDIP32
INTA0	Timer/ Watchdog	INT0	P3.2	
INTA1	STIM Timer	INT1	P3.4	
INTB0		INT2	P2.4	
INTB1		INT3	P2.2	-
INTC0	OSD	INT4	P2.3	
INTC1	ADC	INT5	P2.7	
INTD0		INT6	P2.1	-
INTD1	IR	INT7	P2.0	

EXTERNAL INTERRUPTS (Cont'd)

Figure 25. External Interrupts Control Bits and Vectors



3.7 TOP LEVEL INTERRUPT

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

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

The Top Level Interrupt Service Routine cannot be interrupted by any other interrupt, 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.

3.8 ON-CHIP PERIPHERAL INTERRUPTS

The general structure of the peripheral interrupt unit is described here, however each on-chip peripheral has its own specific interrupt unit containing one or more interrupt channels.

Please refer to the specific peripheral chapter for the description of its interrupt features and control registers.

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

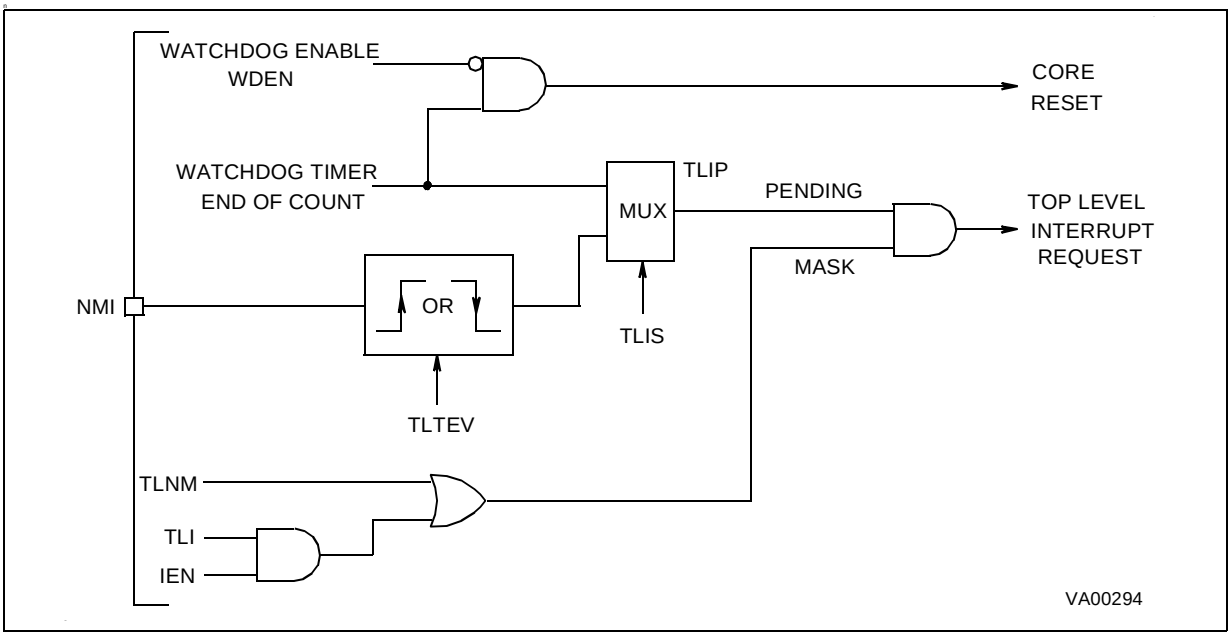
– **Interrupt Pending bit** (IP). Set by hardware when the Trigger Event occurs. Can be set/ cleared by software to generate/cancel pending interrupts and give the status for Interrupt polling.

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

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

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

Figure 26. Top Level Interrupt Structure



3.9 INTERRUPT RESPONSE TIME

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

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

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

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

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

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

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

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

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

3.10 INTERRUPT REGISTERS

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

7							0
-	TLIP	TLI	IEN	IAM	CPL2	CPL1	CPL0

Bit 7 = Reserved
This bit must be kept at 1.

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

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

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

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

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

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

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

7							0
TED1	TED0	TEC1	TEC0	TEB1	TEB0	TEA1	TEA0

Bit 7 = **TED1**: *INTD1 Trigger Event*
Bit 6 = **TED0**: *INTD0 Trigger Event*
Bit 5 = **TEC1**: *INTC1 Trigger Event*
Bit 4 = **TEC0**: *INTC0 Trigger Event*
Bit 3 = **TEB1**: *INTB1 Trigger Event*
Bit 2 = **TEB0**: *INTB0 Trigger Event*
Bit 1 = **TEA1**: *INTA1 Trigger Event*
Bit 0 = **TEA0**: *INTA0 Trigger Event*

These bits are set and cleared by software.
0: Select falling edge as interrupt trigger event
1: Select rising edge as interrupt trigger event

INTERRUPT REGISTERS (Cont'd)

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

7							0
IPD1	IPD0	IPC1	IPC0	IPB1	IPB0	IPA1	IPA0

Bit 7 = **IPD1**: *INTD1 Interrupt Pending bit*

Bit 6 = **IPD0**: *INTD0 Interrupt Pending bit*

Bit 5 = **IPC1**: *INTC1 Interrupt Pending bit*

Bit 4 = **IPC0**: *INTC0 Interrupt Pending bit*

Bit 3 = **IPB1**: *INTB1 Interrupt Pending bit*

Bit 2 = **IPB0**: *INTB0 Interrupt Pending bit*

Bit 1 = **IPA1**: *INTA1 Interrupt Pending bit*

Bit 0 = **IPA0**: *INTA0 Interrupt Pending bit*

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

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

7							0
IMD1	IMD0	IMC1	IMC0	IMB1	IMB0	IMA1	IMA0

Bit 7 = **IMD1**: *INTD1 Interrupt Mask*

Bit 6 = **IMD0**: *INTD0 Interrupt Mask*

Bit 5 = **IMC1**: *INTC1 Interrupt Mask*

Bit 4 = **IMC0**: *INTC0 Interrupt Mask*

Bit 3 = **IMB1**: *INTB1 Interrupt Mask*

Bit 2 = **IMB0**: *INTB0 Interrupt Mask*

Bit 1 = **IMA1**: *INTA1 Interrupt Mask*

Bit 0 = **IMA0**: *INTA0 Interrupt Mask*

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

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

7							0
PL2D	PL1D	PL2C	PL1C	PL2B	PL1B	PL2A	PL1A

Bits 7:6 = **PL2D, PL1D:** *INTD0, D1 Priority Level.*

Bits 5:4 = **PL2C, PL1C**: *INTC0, C1 Priority Level.*

Bits 3:2 = **PL2B, PL1B**: *INTB0, B1 Priority Level.*

Bits 1:0 = **PL2A, PL1A**: *INTA0, A1 Priority Level.*

These bits are set and cleared by software.

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

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

INTERRUPT REGISTERS (Cont'd)

EXTERNAL INTERRUPT VECTOR REGISTER (EIVR)

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

7							0
V7	V6	V5	V4	TLTEV	TLIS	IAOS	EWEN

Bits 7:4 = **V[7:4]**: *Most significant nibble of External Interrupt Vector*.
These bits are not initialized by reset. For a representation of how the full vector is generated from V[7:4] and the selected external interrupt channel, refer to Figure 25.

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

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

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

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

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

NESTED INTERRUPT CONTROL (NICR)

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

7							0
TLNM	HL6	HL5	HL4	HL3	HL2	HL1	HL0

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

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

INTERRUPT REGISTERS (Cont'd)

EXTERNAL MEMORY REGISTER 2 (EMR2)

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

7							0
0	ENCSR	0	0	1	1	1	1

Bits 7, 5:0 = Reserved, keep in reset state. Refer to the external Memory Interface Chapter.

Bit 6 = **ENCSR**: *Enable Code Segment Register.* This bit is set and cleared by software. It affects the ST9 CPU behaviour whenever an interrupt request is issued.

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

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

4 RESET AND CLOCK CONTROL UNIT (RCCU)

4.1 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.

4.2 CLOCK CONTROL REGISTERS

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

7							0
-	-	DIV2	PRS2	PRS1	PRS0	-	-

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

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

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

CLOCK CONTROL REGISTER (CLKCTL)
R240 - Read Write
Register Page: 55
Reset Value: 0000 0000 (00h)

7							0
-	-	-	-	SRESEN	-	-	-

Bits 7:4 = **Reserved.**
Must be kept reset for normal operation.

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

Bits 2:0 = **Reserved.**
Must be kept reset for normal operation.

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

7							0
-	WDG RES	SOFT RES	-	-	-	-	-

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

Bit 7 = **Reserved.**
Must be kept reset for normal operation.

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

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

Bits 4:0 = **Reserved.**
Must be kept reset for normal operation.

4.3 OSCILLATOR CHARACTERISTICS

Because of the real time need of the application, it is assumed the ST92186B will be used with a 4 MHz crystal fed to the Core by the frequency multiplier output after it is started and stabilized.

4.3.1 HALT State

When a HALT instruction is processed, it stops the main crystal oscillator preventing any derived clock into the chip. Exit from the HALT state can be obtained through a main system reset.

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

Figure 27. Crystal Oscillator

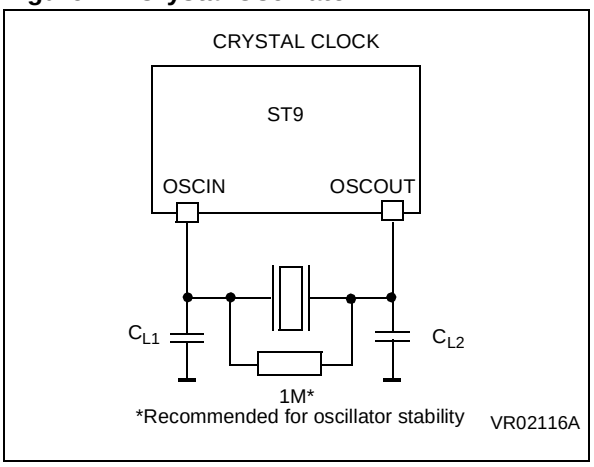


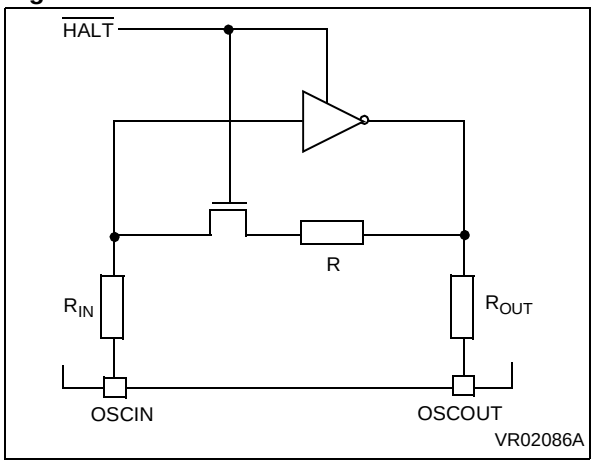
Table 13. Crystal Specification

	C_1=C_2=56pF	C_1=C_2=47pF
Rs max (ohm)	200	260

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

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

Figure 28. Internal Oscillator Schematic



4.4 RESET/STOP MANAGER

The RESET/STOP Manager resets the device when one of the three following triggering events occurs:

– A hardware reset, consequence of a falling edge on the **RESET** pin.

– A software reset, consequence of an HALT instruction when enabled.

– A Watchdog end of count.

The **RESET** input is schmitt triggered.

Note: The memorized Internal Reset (called RESETI) will be maintained active for a duration of 32768 Oscin periods (about 8 ms for a 4 MHz crystal) after the external input is released (set high).

This RESETI internal Reset signal is output on the I/O port bit P5.0 (active low) during the whole reset phase until the P5.0 configuration is changed by software. The true internal reset (to all macrocells) will only be released 511 Reference clock periods after the Memorized Internal reset is released.

It is possible to know which was the last **RESET** triggering event, by reading bits 5 and 6 of register CLK_FLAG.

Figure 29. Reset Overview

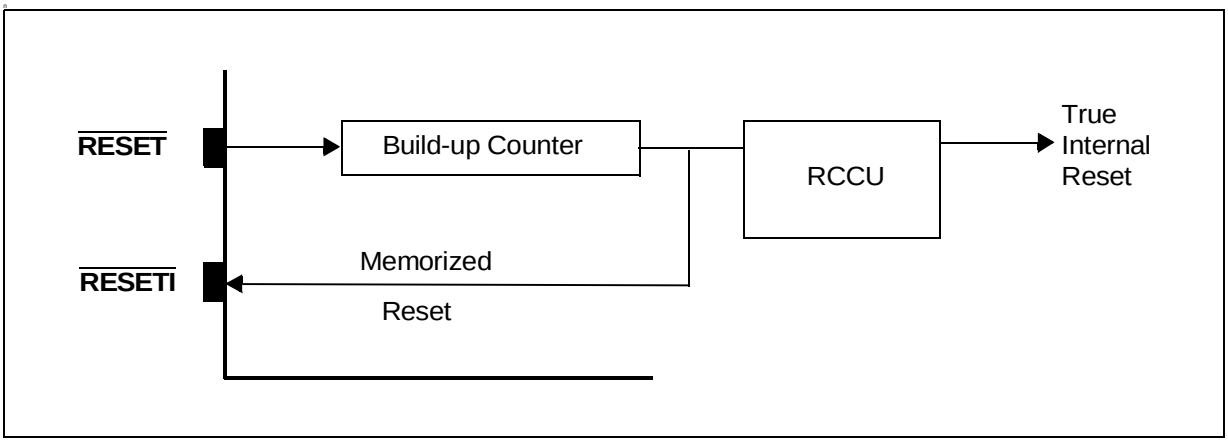
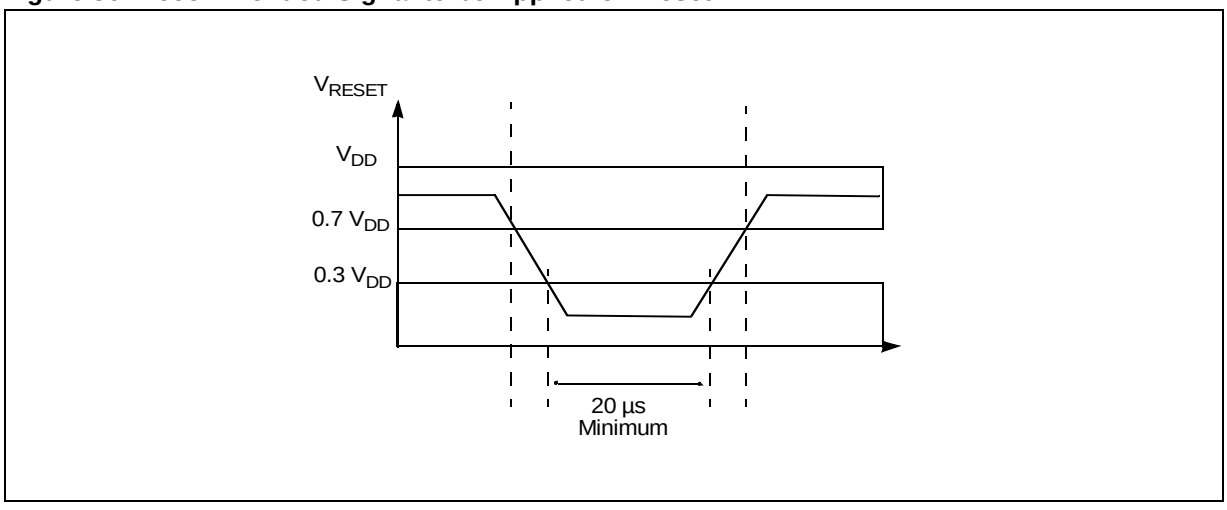


Figure 30. Recommended Signal to be Applied on Reset Pin



5 TIMING AND CLOCK CONTROLLER (TCC)

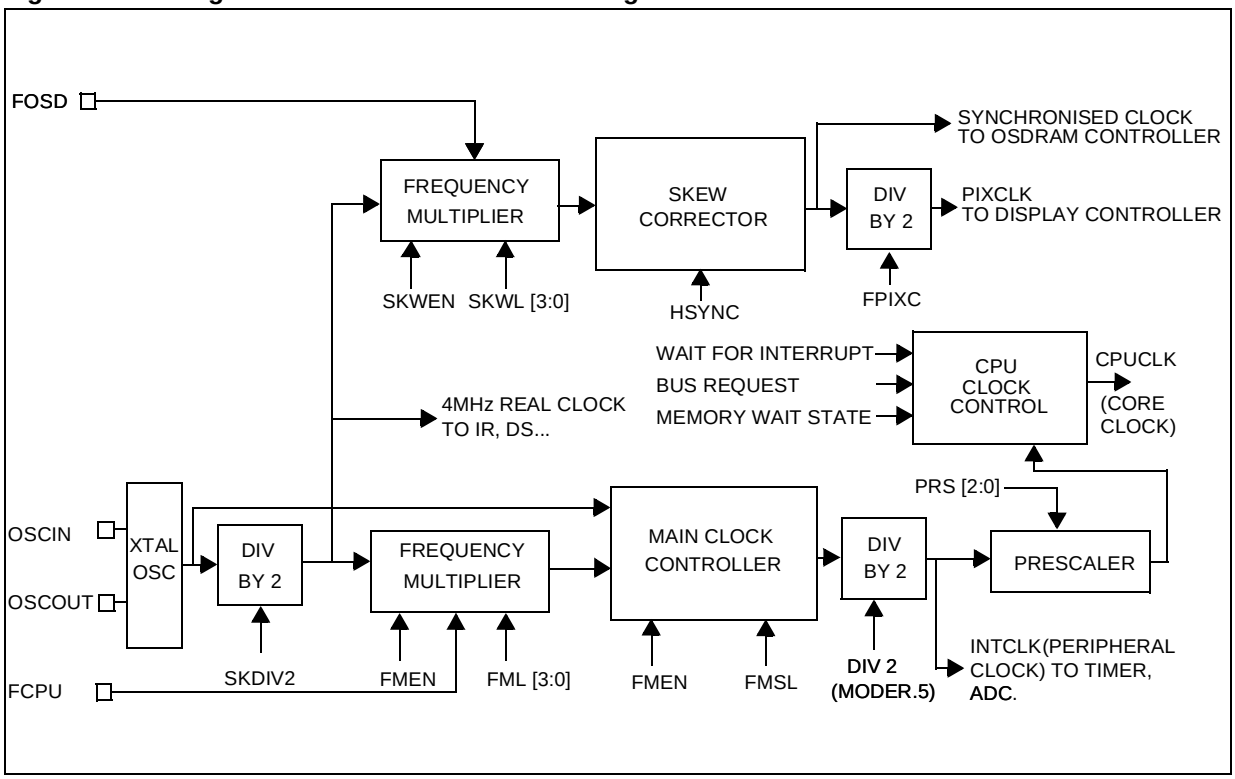
5.1 FREQUENCY MULTIPLIERS

Two on-chip frequency multipliers generate the proper frequencies for: the Core/Real time Peripherals and the Display related time base.

They follow the same basic scheme based on an integrated VCO driven by a three state phase comparator and a charge-pump (1 pin used for off-chip filtering components; a resistor in series with a capacitor tied to ground).

For both the Core and the Display frequency multipliers, a 4 bit programmable feed-back counter allows the adjustment of the multiplying factor to the application needs (a 4 MHz crystal is assumed).

Figure 31. Timing and Clock Controller Block Diagram



FREQUENCY MULTIPLIERS (Cont'd)

Off-chip filter components (to be confirmed)

– Core frequency multiplier (FCPU pin): 1.2K ohms; 47 nF plus 100 pF between the FCPU pin and GND.

– Skew frequency multiplier (FOSD pin): 1.2K ohms; 47 nF plus 100 pF between the FOSD pin and GND.

The frequency multipliers are off during and upon exiting from the reset phase. The user must program the desired multiplying factor, start the multiplier and then wait for its stability (refer to the Electrical Characteristics chapter for the specified delay).

Once the Core/Peripherals multiplier is stabilized, the Main Clock controller can be re-programmed through the FMSL bit in the MCCR register to provide the final frequency (CPUCLK) to the CPU.

The frequency multipliers are automatically switched off when the microprocessor enters HALT mode (the HALT mode forces the control register to its reset status).

Table 14. Examples of CPU Speed Choices

Crystal Frequency SKDIV2=0	FML (3:0)	CPUCLK
4 MHz	4	10 MHz
	5	12 MHz
	6	14 MHz
	7	16 MHz
	8	18 MHz
	9	20 MHz
	10	22 MHz
	11	24 MHz

Note: 24 MHz is the max. authorized frequency.
Caution: The values indicated in this table are the only authorized values.

Table 15. PIXCLK Frequency Choices

Crystal Frequency	SKW (3:0)	FPIXC	PIXCLK
SKDIV2=0			
4 MHz	6	0	14 MHz
	7	0	16 MHz
	8	0	18 MHz
	9	0	20 MHz
	10	0	22 MHz
	11	0	24 MHz
	6	1	28 MHz
	7	1	32 MHz
	8	1	36 MHz
	9	1	40 MHz

5.2 REGISTER DESCRIPTION

SKEW CLOCK CONTROL REGISTER (SKCCR)
R254 - Read/ Write
Register Page: 43
Reset value: 0000 0000 (00h)

7	6	5	4	3	2	1	0
SKWEN	SKDIV2	0	0	SKW3	SKW2	SKW1	SKW0

The HALT mode forces the register to its initialization state.

Bit 7= **SKWEN:** *Frequency Multiplier Enable bit.*
0: FM disabled (reset state), low-power consumption mode.
1: FM is enabled providing clock to the Skew corrector. The SKWEN bit must be set only after programming the SKW(3-0) bits.

Bit 6 = **SKDIV2:** *Skew Divide-by-2 Enable bit.*
0: Divide-by-2 disabled.
1: Divide-by-2 enabled.
This bit must be kept in reset state.

Bits 5:4 = Reserved. These bits are forced to 0 by hardware.

Bits 3:0 = **SKW:** *Skew Counter.*
These 4 bits program the down-counter inserted in the feedback loop of the Frequency Multiplier which generates the internal multiplied frequency PIXCLK. The PIXCLK value is calculated as follows :

If FPIXC=0 :

F(PIXCLK)=Crystal frequency * [(SKW(3:0)+1)]/2

If FPIXC=1 :

F(PIXCLK)=Crystal frequency * [(SKW(3:0)+1)]

Note: To program the FPIXC bit, refer to the description of the OSDER register in the OSD chapter.

MAIN CLOCK CONTROL REGISTER (MCCR)
R253 - Read/ Write
Register Page: 43
Reset value: 0000 0000 (00h)

7	6	5	4	3	2	1	0
FMEN	FMSL	0	0	FML3	FML2	FML1	FML0

The HALT mode forces the register to its initialization state.

Bit 7 = **FMEN:** *Frequency Multiplier Enable bit.*
0: FM disabled (reset state), low-power consumption mode.
1: FM is enabled, providing clock to the CPU. The FMEN bit must be set only after programming the FML(3:0) bits.

Bit 6 = **FMSL:** *Frequency Multiplier Select bit.*
This bit controls the choice of the ST9 core internal frequency between the external crystal frequency and the Main Clock issued by the frequency multiplier.

In order to secure the application, the ST9 core internal frequency is automatically switched back to the external crystal frequency if the frequency multiplier is switched off (FMEN =0) regardless of the value of the FMSL bit. Care must be taken to reset the FMSL bit before any frequency multiplier can restart (FMEN set back to 1).

After reset, the external crystal frequency is always sent to the ST9 Core.

Bits 5:4 = Reserved. These bits are forced to 0 by hardware.

Bits 3:0 = **FML:** *FM Counter.*
These 4 bits program the down-counter inserted in the feed-back loop of the Frequency Multiplier which generates the internal multiplied frequency Fimf. The Fimf value is calculated as follows :

Fimf = Crystal frequency * [(FML(3:0) + 1)] /2

6 I/O PORTS

6.1 INTRODUCTION

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

6.2 SPECIFIC PORT CONFIGURATIONS

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

6.3 PORT CONTROL REGISTERS

Each port is associated with a Data register (PxDR) and three Control registers (PxC0, PxC1, PxC2). These define the port configuration and allow dynamic configuration changes during program execution. Port Data and Control registers are mapped into the Register File as shown in Figure 32. 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 32. I/O Register Map

	GROUP E			GROUP F PAGE 2		GROUP F PAGE 3	
	System Registers		FFh	Reserved		Reserved	R255
			FEh	P3C2			R254
			FDh	P3C1			R253
			FCh	P3C0			R252
			FBh	Reserved			R251
			FAh	P2C2			R250
			F9h	P2C1			R249
			F8h	P2C0			R248
			F7h				R247
			F6h			P5C2	R246
E5h	P5DR	R229	F5h	Reserved		P5C1	R245
E4h	P4DR	R228	F4h			P5C0	R244
E3h	P3DR	R227	F3h			Reserved	R243
E2h	P2DR	R226	F2h	P0C2		P4C2	R242
E1h	P1DR	R225	F1h	P0C1		P4C1	R241
E0h	P0DR	R224	F0h	P0C0		P4C0	R240

PORT CONTROL REGISTERS (Cont'd)

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

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

Other reset conditions may apply in specific ST9 devices.

6.4 INPUT/OUTPUT BIT CONFIGURATION

By programming the control bits PxC0.n and PxC1.n (see Figure 33) 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).

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

The basic structure of the bit Px.n of a general purpose port Px is shown in Figure 34.

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 35).

– 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 33. Control Bits

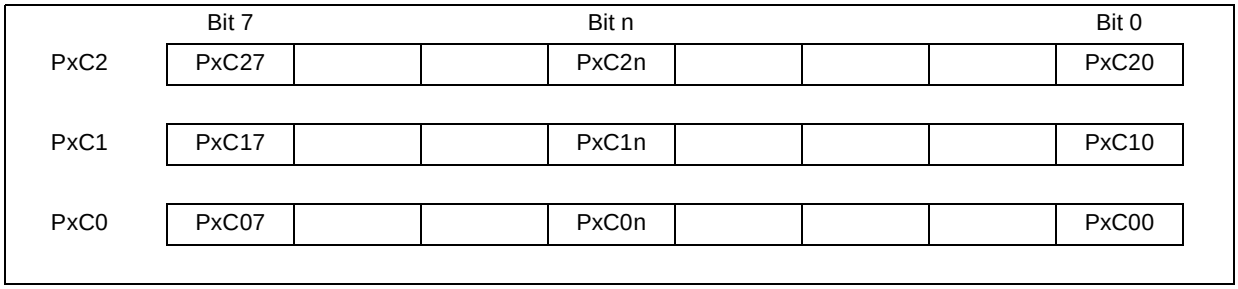


Table 16. Port Bit Configuration Table (n = 0, 1... 7; X = port number)

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

(1) For A/D Converter inputs.

Legend:

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

INPUT/OUTPUT BIT CONFIGURATION (Cont'd)

Figure 34. Basic Structure of an I/O Port Pin

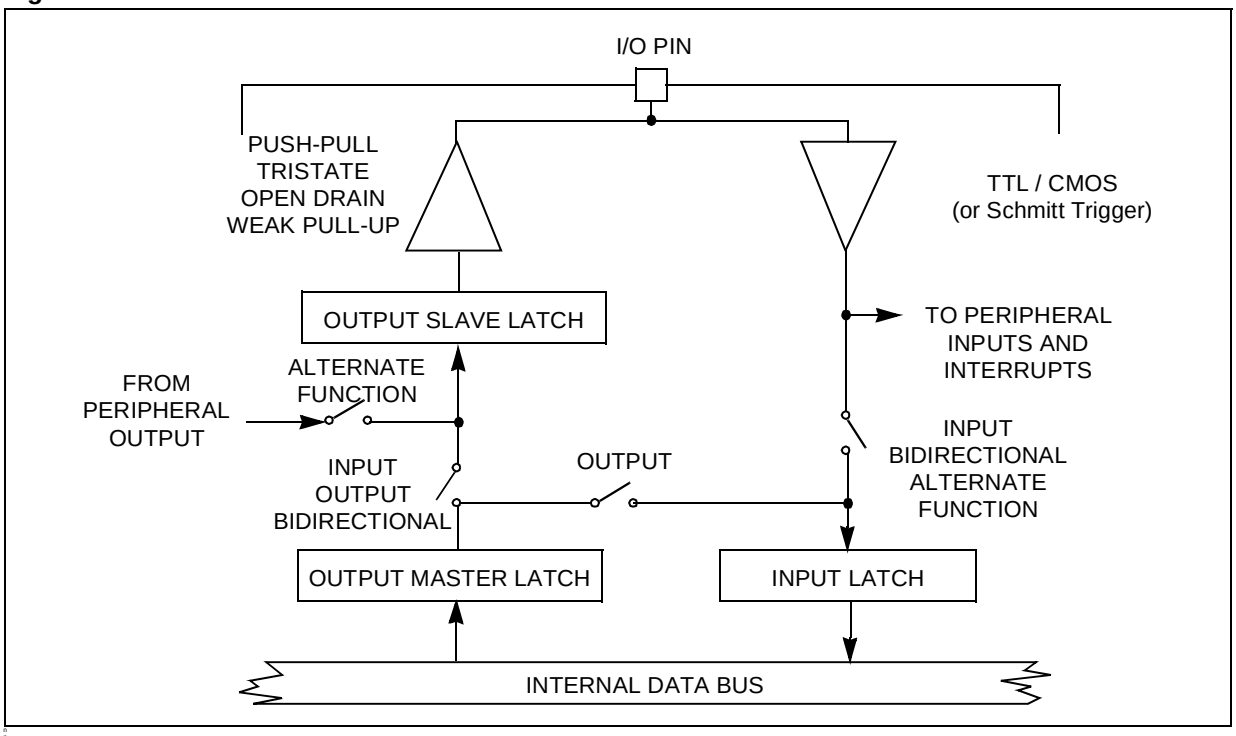


Figure 35. Input Configuration

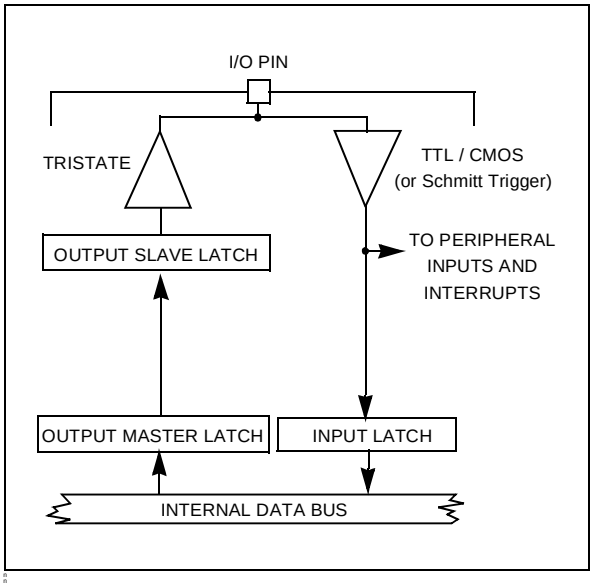
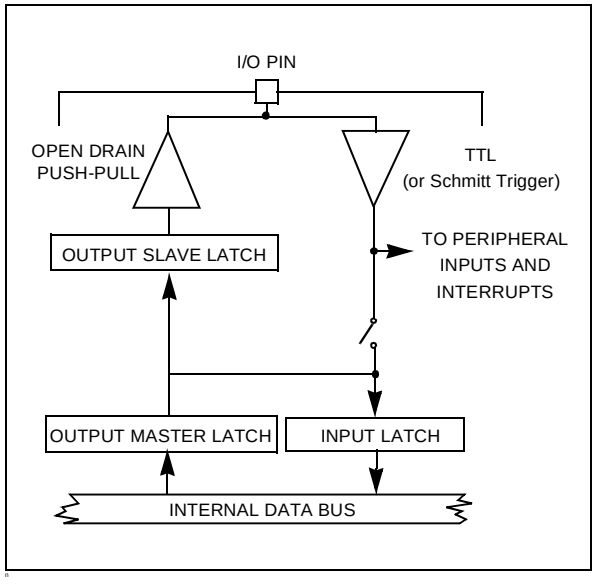


Figure 36. Output Configuration



INPUT/OUTPUT BIT CONFIGURATION (Cont'd)

When Px.n is programmed as an Output:
(Figure 36)

– 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 37)

– 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 38)

– 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 37. Bidirectional Configuration

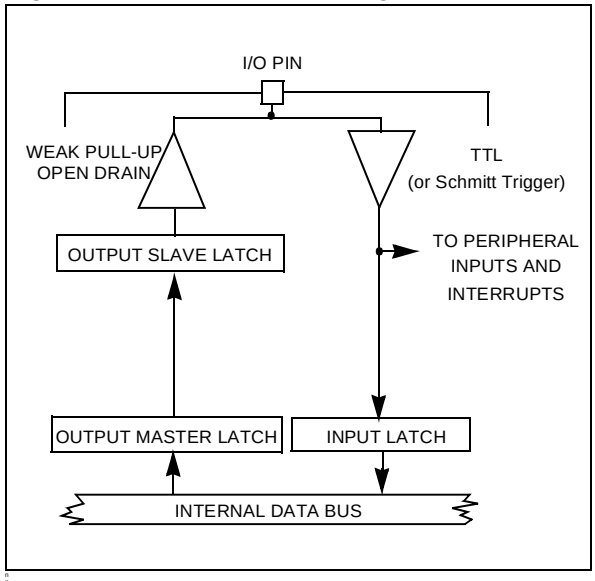
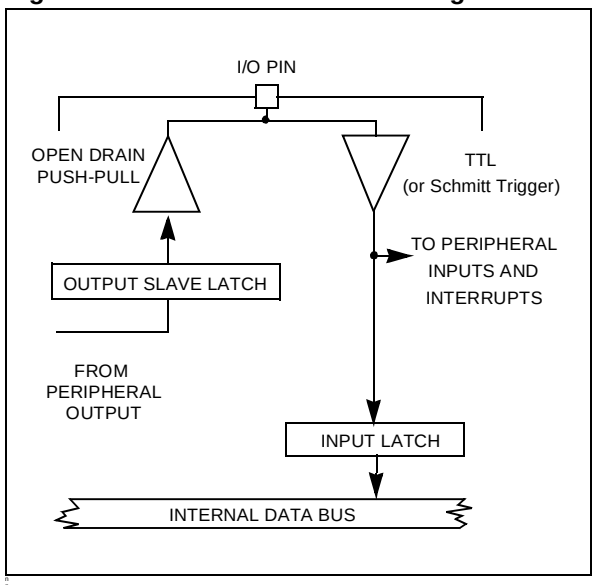


Figure 38. Alternate Function Configuration



6.5 ALTERNATE FUNCTION ARCHITECTURE

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

– Data bus Input/Output

– Alternate Function Input

– Alternate Function Output

6.5.1 Pin Declared as I/O

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

6.5.2 Pin Declared as an Alternate Function Input

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

6.5.3 Pin Declared as an Alternate Function Output

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

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

6.6 I/O STATUS AFTER WFI, HALT AND RESET

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

Mode	I/O Ports
WFI	Not Affected (clock outputs running)
HALT	Not Affected (clock outputs stopped)
RESET	Bidirectional Weak Pull-up (High impedance when disabled in hardware).

6.7 CONFIGURATION OF UNBONDED I/OS

Some I/Os are not bonded in the SDIP32 package. You must configure these unbonded I/Os (i.e. P2.1, P2.2, P3.1, P3.5, P3.7, P5.1, P5.2, P5.5 and P5.6) as output push-pull at the very first beginning of your software and never change this configuration after initialization. This will avoid unpredictable software behavior.

7 ON-CHIP PERIPHERALS

7.1 TIMER/WATCHDOG (WDT)

7.1.1 Introduction

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

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

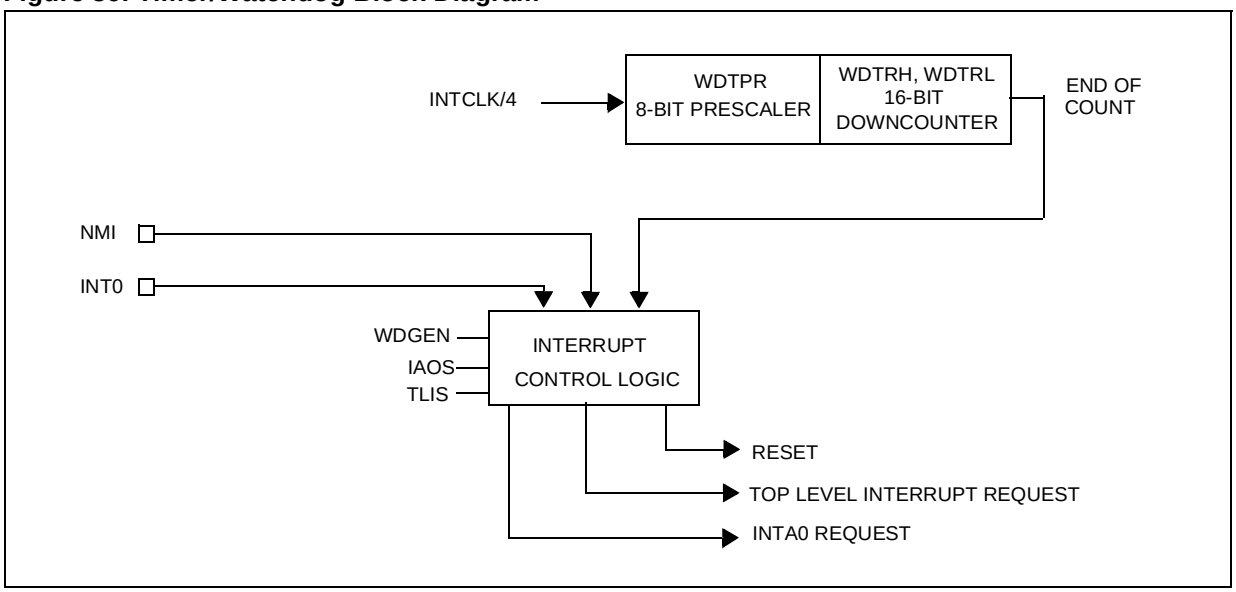
The main WDT registers are:

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

The hardware interface consists of up to five signals:

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

Figure 39. Timer/Watchdog Block Diagram



TIMER/WATCHDOG (Cont'd)

7.1.2 Functional Description

7.1.2.1 External Signals

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 is driven by an internal clock equal to INTCLK divided by 4.

7.1.2.2 Initialisation

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

7.1.2.3 Start/Stop

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

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

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

End of Count is when the counter is 0.

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

7.1.2.4 Single/Continuous Mode

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

Single Mode

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

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

Continuous Mode

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

7.1.3 Watchdog Timer Operation

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

7.1.3.1 Starting the Watchdog

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

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

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

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

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

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

TIMER/WATCHDOG (Cont'd)

7.1.3.2 Preventing Watchdog System Reset

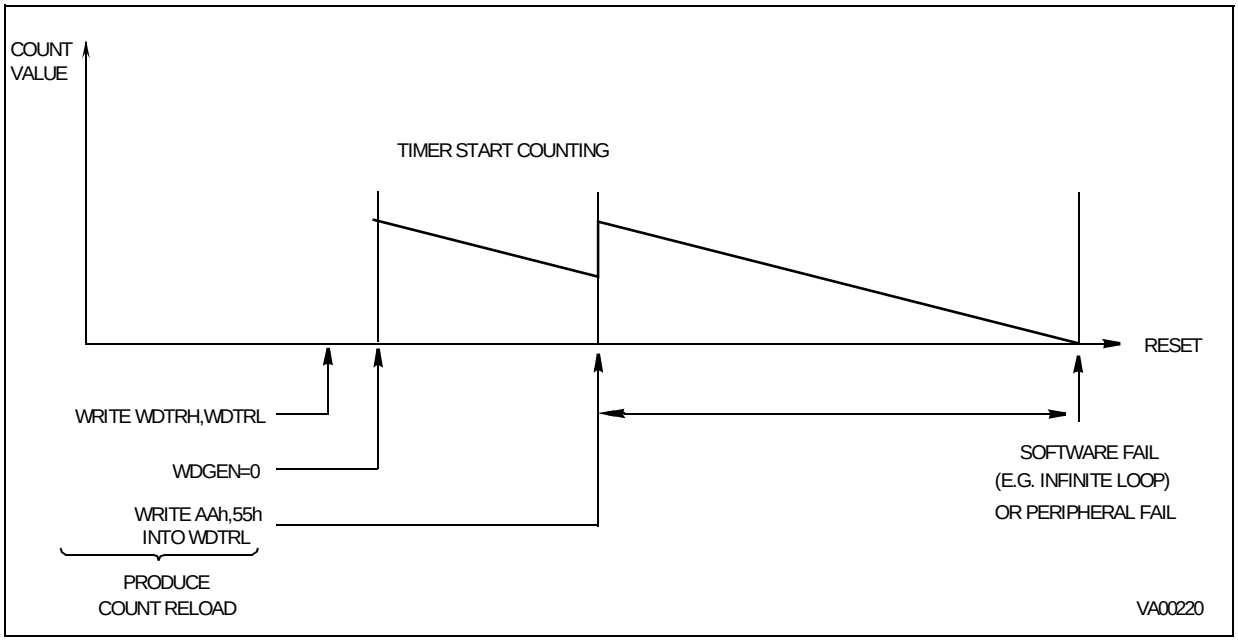
In order to prevent a system reset, the sequence AAh, 55h must be written to WDTLR (Watchdog Timer Low Register). Once 55h has been written, the Timer reloads the constant and counting restarts from the preset value.

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.

7.1.3.3 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 and S_C bits are ignored. Hence, regardless of their status, the counter always runs in Continuous Mode, driven by the internal clock.

Figure 40. Watchdog Timer Mode



TIMER/WATCHDOG (Cont'd)

7.1.4 WDT Interrupts

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

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

A block diagram of the interrupt logic is given in Figure 41.

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

Table 17 below, shows all the possible configurations of interrupt/reset sources which relate to the Timer/Watchdog.

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

Figure 41. Interrupt Sources

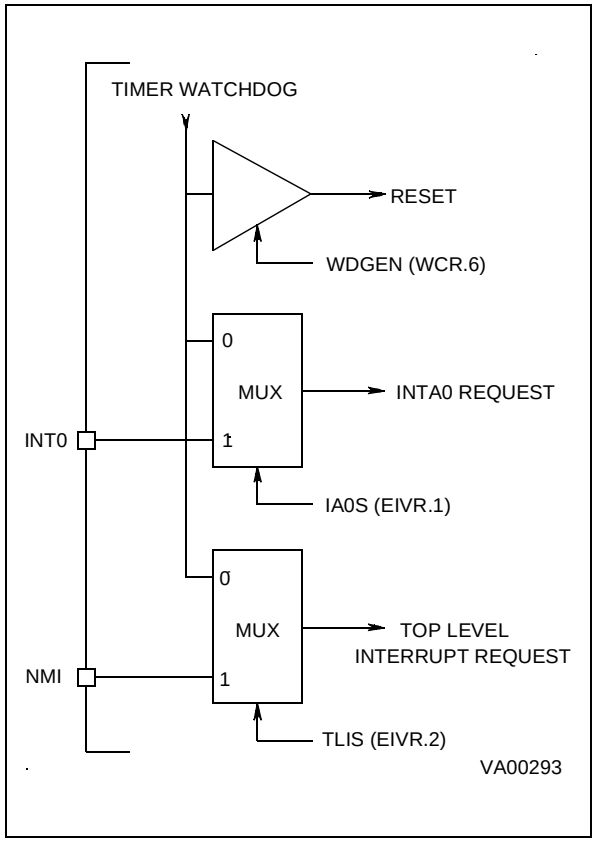


Table 17. Interrupt Configuration

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

Legend:
WDG = Watchdog function
SW TRAP = Software Trap

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

TIMER/WATCHDOG (Cont'd)

7.1.5 Register Description

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

WDTHR: Timer/Watchdog High Register

WDTLR: Timer/Watchdog Low Register

WDTPR: Timer/Watchdog Prescaler Register

WDTCR: Timer/Watchdog Control Register

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

Watchdog Mode Enable, (WCR.6)

Top Level Interrupt Selection, (EIVR.2)

Interrupt A0 Channel Selection, (EIVR.1)

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

Counter Register

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

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

7							0
R15	R14	R13	R12	R11	R10	R9	R8

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

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

7							0
R7	R6	R5	R4	R3	R2	R1	R0

TIMER/WATCHDOG (Cont'd)

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

7							0
PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0

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

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

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

7							0
ST_SP	S_C	0	1	0	0	1	0

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

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

Bits 5:0 = Reserved.
Must be kept in reset state.

TIMER/WATCHDOG (Cont'd)

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

7							0
0	WDGEN	1	1	1	1	1	1

Bit 7 = Reserved.
Must be kept in reset state.

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

Bits 5:0 = Reserved.
Must be kept in reset state.

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

7							0
V7	V6	V5	V4	TLTEV	TLIS	IAOS	EWEN

Bits 7:4 = **V[7:4]**: *Most significant nibble of External Interrupt Vector.*
These bits are described in the Interrupt section on page 54.

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

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

Bit 1 = **IA0S**: *Interrupt Channel A0 Selection.*
This bit is described in the Interrupt section on page 54.

Bit 0 = **EWEN**: *External Wait Enable.*
This bit is described in the Interrupt section on page 54.

7.2 STANDARD TIMER (STIM)

7.2.1 Introduction

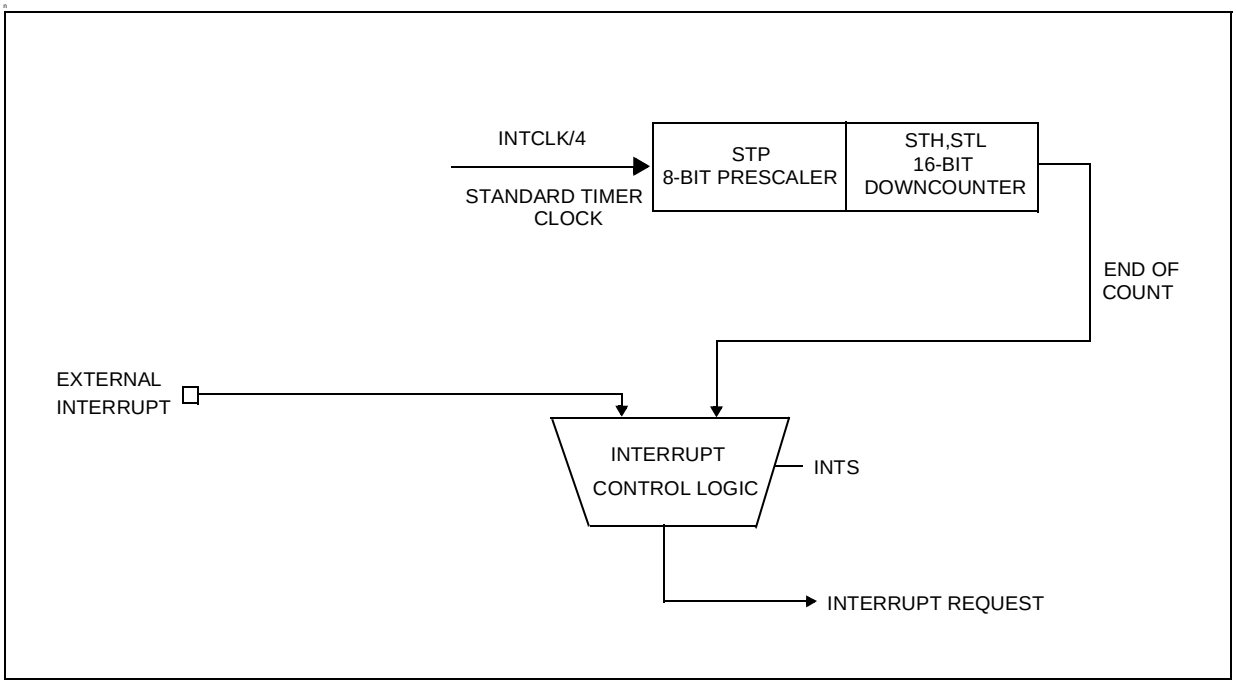
The Standard Timer includes a programmable 16-bit down counter and an associated 8-bit prescaler with Single and Continuous counting modes capability.

The Standard Timer is composed of a 16-bit down counter with an 8-bit prescaler. The input clock to the prescaler is driven by an internal clock equal to INTCLK divided by 4.

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

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

Figure 42. Standard Timer Block Diagram



STANDARD TIMER (Cont'd)

7.2.2 Functional Description

7.2.2.1 Timer/Counter control

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

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

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

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

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

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

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

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

7.2.3 Interrupt Selection

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

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

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

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

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

7.2.4 Register Mapping

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

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

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

STD Timer	Register	Register Address
STIM	STH0	R240 (F0h)
	STL0	R241 (F1h)
	STP0	R242 (F2h)
	STC0	R243 (F3h)

STANDARD TIMER (Cont'd)

7.2.5 Register Description

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

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

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

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

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

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

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

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

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

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

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

7							0
ST-SP	S-C	0	1	0	INTS	0	0

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

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

Bits 5:3 = Reserved.
Must be kept in reset state.

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

Bits 1:0 = Reserved.
Must be kept in reset state.

7.3 OSDRAM CONTROLLER

7.3.1 Introduction

The OSDRAM Controller handles the interface between the Display Controller, the CPU and the OSDRAM.

The time slots are allocated to each unit in order to optimize the response time.

The main features of the OSDRAM Controller are the following:

- Memory mapped in Memory Space (segment 22h of the MMU)
- DMA access for Display control
- Direct CPU access

7.3.2 Functional Description

The OSDRAM controller manages the data flows between the different sub-units (display controller, CPU) and the OSDRAM. A specific set of buses (16-bit data, 9-bit addresses) is dedicated to these data flows. The OSDRAM controller accesses these buses in real time. The OSDRAM controller has registers mapped in the ST9 register file.

As this OSDRAM controller has also to deal with TV real time signals (On-Screen-Display), a specific controller manages all exchanges:

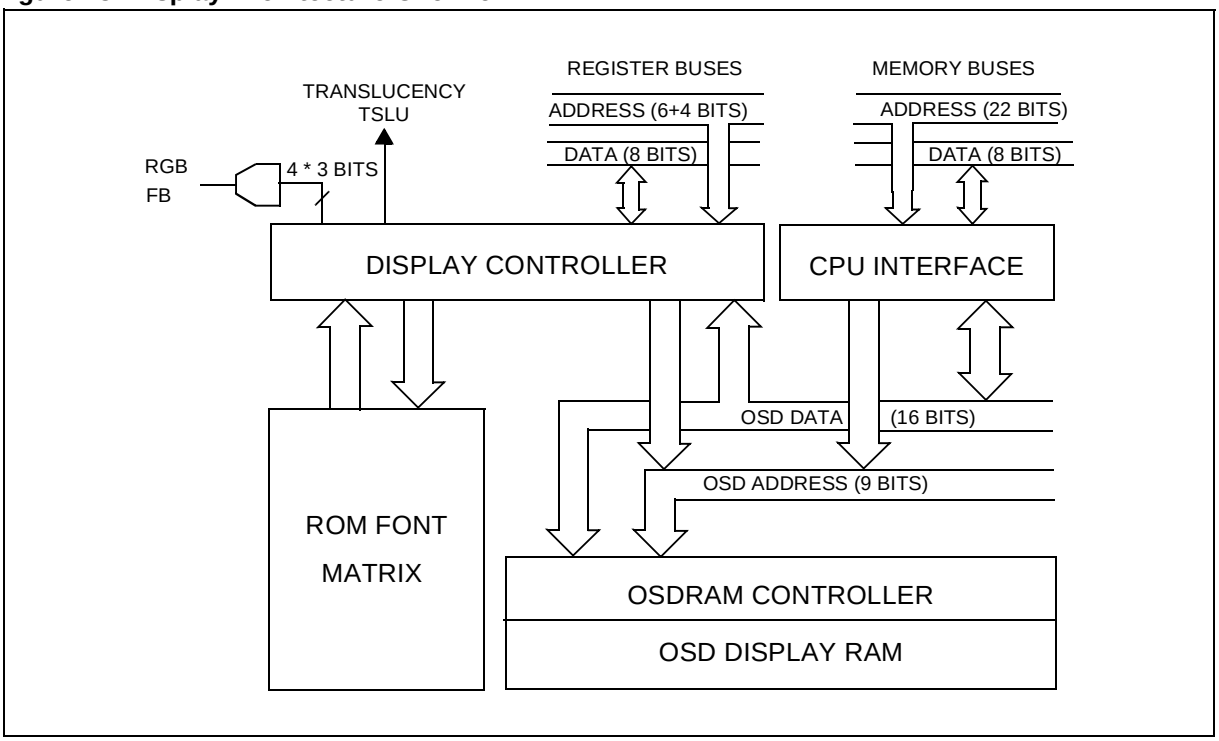
– Its timing generator uses the same frequency generator as the Display (Pixel frequency multiplier),

– Its controller can work in two TV modes:
  – **Single mode**: all time slots are dedicated to the CPU.
  – **Shared mode**: time slots are shared between the CPU and the Display. The shared mode is controlled by the Display controller.

– Its architecture gives priority to the TV real time constraints: whenever there is access contention between the CPU and the Display (shared mode), the CPU is automatically forced in a “wait” configuration until its request is served.

– Its controller enables a third operating mode (stand-alone mode) which allows the application to access the OSDRAM while the Display is turned off. In this case, the OSDRAM controller uses the CPU main clock.

Figure 43. Display Architecture Overview



OSDRAM CONTROLLER (Cont'd)

7.3.2.1 Time Sharing during Display

The time necessary to display a character on the screen defines the basic repetitive cycle of the OSDRAM controller. This whole cycle represents therefore 18 clock periods. This cycle is divided in 9 sub-cycles called "slots". Each slot is allocated in real-time either to the CPU or the Display:

– In single mode, this 9-slot cycle is repeated continuously providing only CPU slots (single cycle), until the OSDRAM controller is switched off by the main program execution.

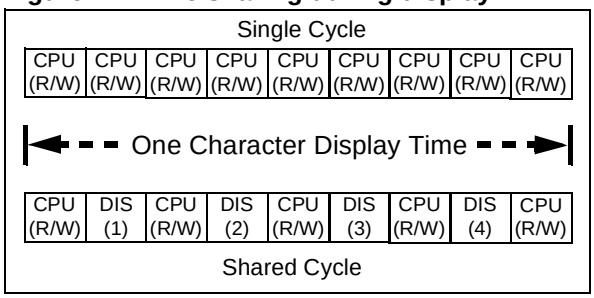
– In shared mode, this 9-slot cycle provides Display slots followed by CPU slots.

Each slot represents a two-byte exchange (read or write) between the OSDRAM memory and the other units:

Display Reading slot (DIS): 16 bits are read from the OSDRAM and sent to the display unit, the address being defined by the display address generator.

Direct CPU Access slot (CPU): 16 bits are exchanged (read or write) between the OSDRAM and its controller but only 8 bits are exchanged with the CPU, the address being defined by the CPU memory address bus.

Figure 44. Time sharing during display



Display reading is handled as follows:

– DIS(1) & DIS(2) are dedicated to reading the character code, its parallel attributes & associated palette pointer.

– DIS(3) provides the foreground palette.

– DIS(4) provides the background palette. In case of Underline activation (refer to the OSD controller paragraph for more details), the DIS(4) slot is no longer provides the background palette content (useless information) but recovers the Underline color set data.

The CPU write accesses are handled as follows:

Because of the 16-bit word width inside the OSDRAM matrix, it is obviously necessary to perform a CPU write access in 2 steps:

– Reading the OSDRAM word

– Rewriting it with the same values except for the 8 modified bits.

Each time a CPU write operation is started, the next following CPU slot will be used as a read slot, the effective write to the OSDRAM being completed at the next CPU slot.

7.3.2.2 Time sharing within the TV line

At the beginning of each TV line, the OSDRAM is accessed (read) by the Display controller in order to get all the row attributes. When the TV line is recognized as the one where data have to be displayed, the Shared cycle is activated at the time the data has to be processed for display.

OSDRAM CONTROLLER (Cont'd)

7.3.3 OSDRAM Controller Reset Configuration

During and after a reset, the OSDRAM access is disabled.

When the OSDRAM controller is software disabled, it will:

1. Complete the current slot.
2. Complete any pending write operation (a few slots may elapse).
3. Switch off any OSDRAM interface activity.

7.3.3.1 OSDRAM Controller Running Modes

2 control bits called "OSDE" (OSD Enable) and "DION" (Display ON) are used to enable the OSDRAM controller. Both are also shared by the Display controller.

These 2 bits are located in the OSDER register. This register is described in the On Screen Display Controller Chapter.

7.3.3.2 CPU Slowdown on OSDRAM access

As described above, the OSDRAM controller puts priority on TV real time constraints and may slowdown the CPU (through "wait" cycle insertion) when any OSDRAM access is requested. The effective duration of the CPU slowdown is a complex function of the OSDRAM controller working mode and of the respective PIXCLK frequency (OSDRAM frequency) and the Core INTCLK frequency.

7.3.3.3 OSDRAM Mapping

The OSDRAM is mapped in the memory space, segment 22h, starting from address 0000h to address 00FFh (256 bytes).

The OSDRAM mapping is described in the On Screen Display Controller Chapter.

7.4 ON SCREEN DISPLAY CONTROLLER (OSD)

7.4.1 Introduction

The OSD displays character data and menus on a TV screen.

Each row can be defined through three different Display configurations:

– **Serial mode:** each character is defined by an 8-bit word which provides the character address into the Font ROM memory. Some codes are reserved for color controls and do not address any character description. They are displayed as spaces and as a direct consequence are active on a “word” basis.

– **Basic parallel mode:** each character is defined by a 16-bit word which provides the character address in the Font ROM memory and its color attribute. This mode is called “parallel” as the colors are definable on a “per character” basis.

– **Extended parallel mode:** each character is defined by a 24-bit word which provides the character address in the Font ROM memory and its color and shape attributes. This mode is called “parallel” as the attributes are definable on a “per character” basis.

7.4.2 General Features

- 50/60 Hz and 100/120 Hz operation
- 525/625 lines operation, 4/3 or 16/9 format
- Interlaced or progressive scanning
- 18x26 or 9x13 character matrix user definable in ROM. Both matrixes can be mixed.
- Up to 63 characters per row
- 7 character sizes in 18x26, 4 in 9x13
- 512 possible colors in 4x16-entry palettes
- 8 levels of translucency on Fast Blanking
- Basic Parallel Mode for character based color definition
- Extended parallel mode for character based color and shape definition
- Rounding, fringing, shadowing, flashing, scrolling, italics, and various underlining modes

Definition of Terms used in this Chapter

– **Pixel**: minimum displayed element that the Display Controller can handle. Its vertical physical size is always one TV line. Its horizontal physical size is directly linked to the basic clock frequency (called Pixel clock) which synchronizes the OSD and is therefore independent of any magnification factor which may be applied to the displayed element.

– **Dot**: a dot defines the displayed element which corresponds to a single bit read from the Font ROM memory. A dot is represented on the screen by a "matrix" of pixels. The matrix size depends on the magnification factor applied.

7.4.3 Functional Description

All characters are user definable by masking the Font ROM content (except the one corresponding to code 00h which is reserved for test). Two different matrixes can be used and mixed:

– 18x26 character matrix

– 9x13 character matrix

The hardware display system has the capability of displaying one character row and requires the CPU to update the next display buffer prior to displaying the next row. Using a real time routine, the On Screen Display supports the display of as many character rows as the TV screen can physically handle.

The OSD can display up to 63 characters per row, depending on the row RAM buffer size (user definable, see Section 7.4.5.1).

A smart pixel processing unit provides extended features such as rounding, fringe, or shadowing for better picture quality. Other smart functions such as flashing, scrolling, italics, underlining allow the designing of a high quality display application.

The screen insertion of the displayed characters is fully synchronized by the vertical and horizontal TV synchronizing signals. The OSD controller generates the Red, Green, Blue and Fast Blanking video signals through 8-level DAC outputs. The Fast Blanking video signal can also be generated as a digital signal if needed.

OSD CONTROLLER (Cont'd)

7.4.3.1 Display Attributes

- Global screen attributes:
  - Border color
  - Border translucency
  - Turn all background color into border color
- Row parameters and attributes:
  - Row mode control (serial, basic parallel, extended parallel)
  - Row character count
  - Horizontal and Vertical shift
  - Active Range (used for vertical scrolling)
  - Font matrix selection
  - Size control (dot height and width definition)
  - Flashing control
  - Rounding control
  - Fringe control
- Serial character attributes:
  - Background color (16 user definable colors)
  - Foreground color (16 user definable colors)
  - Flashing control
  - Italics control
- Palette parallel character attributes:
  - Background color (16 user definable colors)
  - Foreground color (16 user definable colors)
- Shape parallel character attributes:
  - Character code extension
  - Double height
  - Double width
  - Foreground palette extension (32 user definable colors)
  - Background palette extension (32 user definable colors)
  - Flashing control
  - Shadowing control
  - Fringe control

7.4.3.2 OSD Area

When the Display controller is turned on, the TV screen will show a specific color prior to any data display which is called the "Border Color". The effective border color is fully software programmable from a palette of 512 colors through 2 control registers.

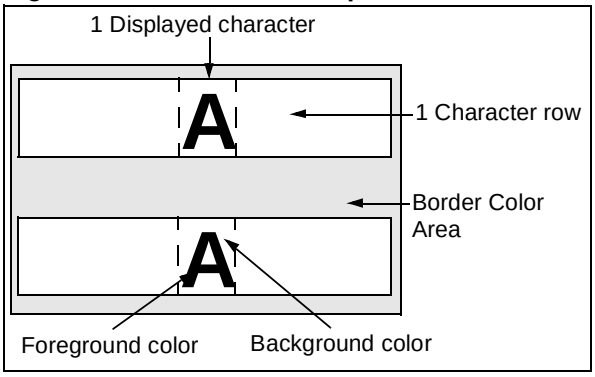
The border area translucency can be chosen from 8 different levels, from fully transparent to fully opaque. The 3 translucency control bits are accessible through the border color control registers.

When data are displayed by the OSD, they form rows of characters. All characters of one row are horizontally aligned.

For each displayed character, two kinds of colors must be programmed which are defined as:

- Background color
- Foreground color

Figure 45. OSD Area description



7.4.3.3 Color processing

Further color elements may be generated by the Display controller as a result of real time pixel calculations (they are not stored in the Font ROM memory). These are:

- Rounding pixels: they must be considered as "calculated" foreground pixels.
- Fringe pixels: they are always displayed with a black color and are never translucent.
- Underline pixels: they must be considered as "calculated" pixels. Their colors are defined through independent underline color values. Translucency levels are also programmable for underline pixels.

OSD CONTROLLER (Cont'd)

All colors are taken from a double Palette set (Background and Foreground) which are both OSD-RAM mapped and thus definable in real-time.

The priority of all color layers is, from highest to lowest:

Underline, foreground & rounding, fringe, background and border.

Character code 00h is a test character and has no user customized content. It is always displayed with a border color and will appear as a "non-displayed character".

7.4.3.4 Character Matrix Definition

A character is described by a matrix of dots stored in the Font ROM memory. Two matrix sizes are can be used to define each character pattern: either a 9x13 matrix or a 18x26 matrix. The matrix size can be redefined for each row Display buffer; mixing the 2 matrix sizes on a same screen is therefore possible.

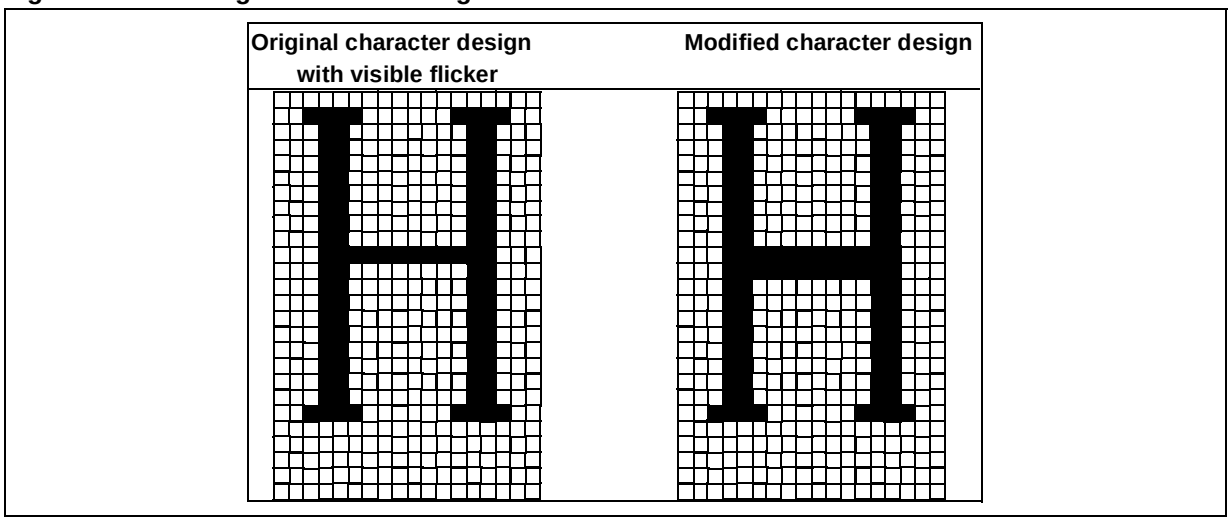
Refer to Figure 54, for an example of the Font ROM content.

The 9x13 matrix is a compressed format where each bit represents a 2x2 pixel dot (with no magnification applied). In interlaced mode, for each dot, 2 pixels are generated on a field, the 2 others on the other field; each row of the matrix is used for both fields.

The 18x26 matrix is an "expanded" format where each bit represents a 1x1 pixel dot (if no magnification). In interlaced mode, if no magnification is applied, each odd row of the matrix is displayed on a field, each even row on the other field.

Warning: As a result, when displaying the 18x26 matrix with no vertical magnification and in interlaced mode, flicker may be visible on characters containing long single-pixel rows. To avoid this effect, either use double pixel horizontal lines, if possible, when designing the font (see Figure 46). Alternatively, use the double height attribute or choose the foreground and background colors for reduced contrast.

Figure 46. Avoiding flicker on unmagnified 18x26 characters



OSD CONTROLLER (Cont'd)

7.4.3.5 Cursor & Flash

A cursor facility may be emulated under software control, using the "flash" attribute. This allows to have a "flash-on-word" in serial mode or a "flash-on-character" in extended parallel mode.

The cursor facility first requires activating the "flash on" row control bit (refer to Section 7.4.7.3), which acts as a general flash enable.

Then program the characters with Flashing Character attribute(s) (serial or extended parallel) at the required locations.

The flash effect is obtained by toggling the general flash enable bit.

Several flashing words or characters per screen can easily be implemented.

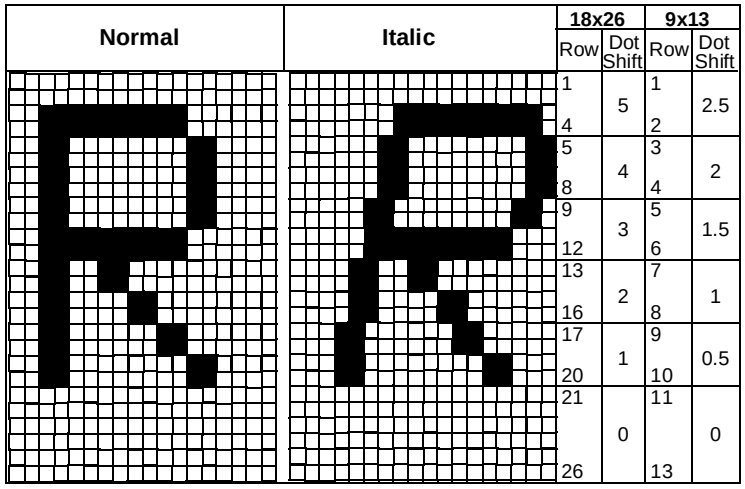
7.4.3.6 Italic mode

The italics attribute is a serial attribute; this means that Italic mode is available in serial mode only.

In the matrix description that follows, line 1 is at the top of the character, line 13 (or line 26) is at the bottom.

The codes (seen as spaces) needed to activate and deactivate the Italic attribute provide a convenient method for solving border problems between italic and non-italic characters.

Figure 47. Italic character mode



OSD CONTROLLER (Cont'd)

If the italic attribute is still active at the end of the row, the last character code to be displayed is truncated to the vertical position where it would have finished without the italic attribute.

If the background color is changed while italics are enabled, then the color attribute (seen as a space) is not slanted.

The right edge background corresponding to a control character is never slanted.

When a background code follows an italic character, then the background color of the italic character extends half way into the displayed background code location, regardless of the background Palette M bit (refer to Section 7.4.6.4 for a description of the M bit).

In the case where a background code is followed by a second background code while italics are on, the first background color will extend half way into the second background location.

The edges of the 00h code (test character seen as a border color) are never slanted.

The right edge of the 00h code is never slanted.

When a 00h code follows an italic character, then the background color of the italic character extends half way into the 00h code location.

In case a 00h code is followed by a background code while italics are on, the 00h code is never extended into the following code location.

7.4.3.7 Rounding and Fringe

Rounding can be enabled or disabled row by row (see Section 7.4.7.3).

For a 18x26 matrix size, there is no rounding facility when the character size is (1X,1Y).

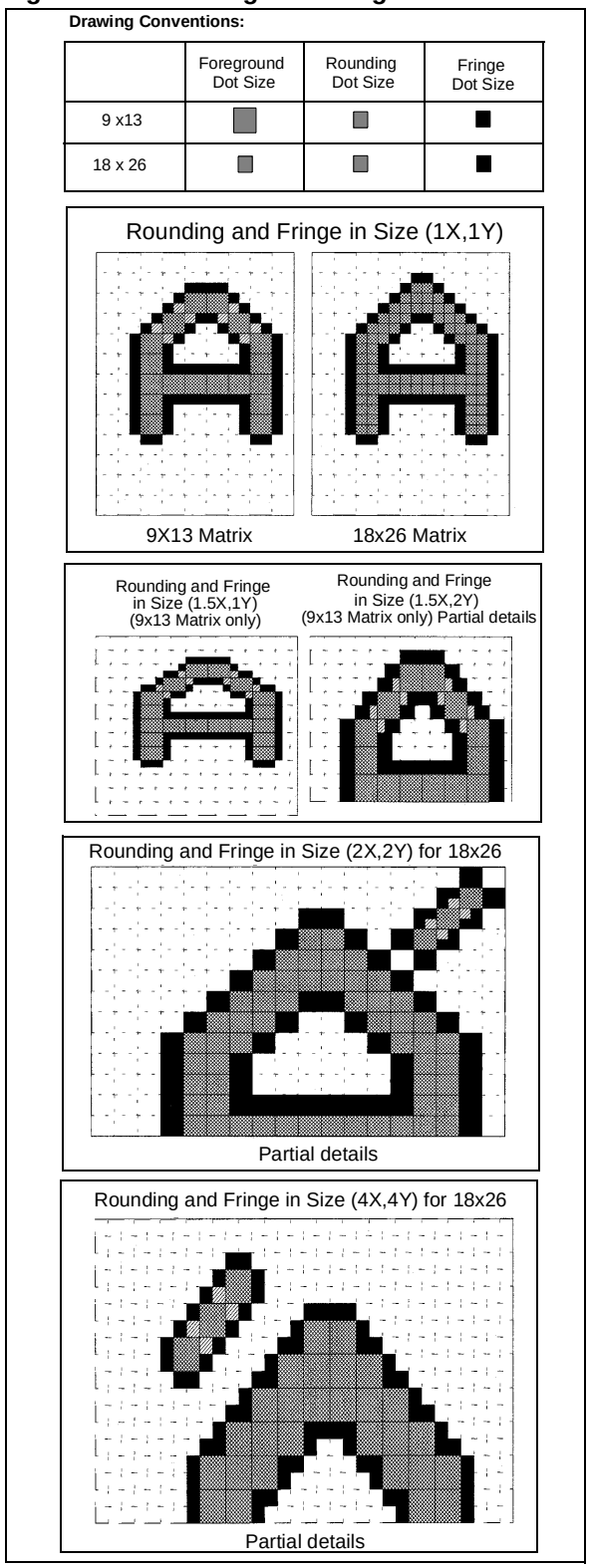
For any other (X,Y) size combinations, the rounding facility is allowed for the 18x26 font matrix, and rounding is available in any of the 3 row modes (serial, basic parallel, extended parallel).

The fringe mechanism can be enabled or disabled row by row (see row attributes description), but it can also be defined on a character basis in extended parallel mode (see shape parallel attributes for more details).

The fringe mechanism can be activated for any size and both matrix formats.

Please note that, for both matrixes, in the case of fringe usage in 1Y vertical size and interlaced mode, a flicker may appear on the screen as the fringe information is built on a field basis.

Figure 48. Rounding and Fringe



OSD CONTROLLER (Cont'd)

7.4.3.8 Scrolling

The row RAM buffer architecture of the Display allows all scrolling operations to be performed very easily by software: scroll up, scroll down, scroll left, scroll right and any horizontal/vertical mix.

In addition to the character row scrolling, a vertical "smooth" scrolling has been implemented. It requires defining the "active range" for each displayed row (refer to Section 7.4.7.4).

7.4.3.9 Color Palettes

The Display controller provides 4 user-definable palettes: two 16-color foreground palettes (one basic and one extended), and two 16-color background palettes (one basic and one extended). Each color is defined using 16 bits:

– **Foreground palette:** 3 bits for red level, 3 bits for green level, 3 bits for blue level, 3 bits for translucency, and 3 bits for the underline mode control.

– **Background palette:** 3 bits for red level, 3 bits for green level, 3 bits for blue level, 3 bits for translucency, and 1 bit for immediate color change control.

Two palettes are always available (one foreground palette and one background palette). The 2 other palettes are only accessible in extended parallel mode.

The palettes available in serial and basic parallel modes depend on the value of the PASW bit in the OSDDR register.

7.4.3.10 Underline mode control

The OSD is able to underline any character of the 9x13 or 18x26 matrix with 3 different colors selected from a palette, and 2 different dot lines.

Using 3 bits (see the Color Palettes paragraph above), it is possible to define the underline mode associated with each foreground palette entry.

In 9x13 mode, single or double underline can be set on lines 12 and 13 with the foreground color or two specific colors defined in underline color sets 1 & 2.

In 18x26 mode, single or double underline can be set on lines 23-24 and 25-26 with the foreground color or two specific colors defined in underline color sets 1 & 2.

For more details please refer to Section 7.4.6.

OSD CONTROLLER (Cont'd)

7.4.3.11 Translucency Function

The translucency feature is designed to provide a better OSD quality while displaying rows in mixed mode.

Instead of forcing the background color of any character to any full intensity color, (which will prevent the viewer from seeing a significant amount of the video picture), or having a fully transparent background (i.e. no background) which makes the OSD more difficult to read for the viewer, the translucency provides a real time mix between both the video and the OSD background information. This feature will appear as a "boxing" effect around all the displayed characters.

The translucency can be handled in two different ways at application level, depending on the video processor used in the application.

1. When the video processor accepts an analog control of the fast switch OSD signal (called "FB"), the translucency can be handled directly through the real time amplitude of the FB signal (refer to Color attribute control for Border, Background palette and Underline color settings).
2. When the video processor accepts only a digital FB signal, the translucency function may be implemented on the chassis with the help of an additional digital output of the MCU, which is provided as an I/O pin alternate function.

This digital output called "TSLU" is active (set to "1") when the OSD displays the Background and Foreground or when the mute is active (refer to the description of the LSM[2:0] bits in the OSDMR register) and is inactive the rest of the time (during foreground or if no display), including character 00h.

This TSLU signal is controlled by the TSLE bit in the OSDER register. When not used, (TSLE=0), the TSLU signal is held at "1" by hardware.

Application Note

In order to enable the Translucency Function (See Example No. 2, above), the following procedure must be performed:

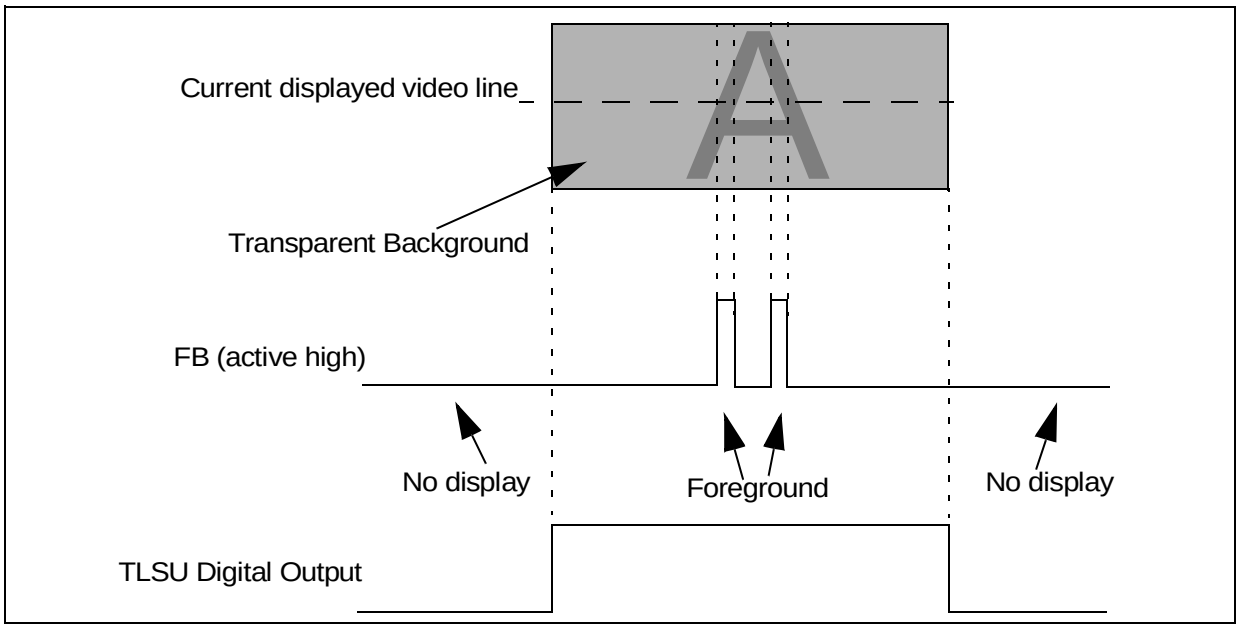
– Fast Blanking must be set to Digital mode. Set bit DIFB (bit 7) of register OSDBCR1 (R247, page 42) to 1. Refer to Section 7.4.8 Register Description.

– Initialize port 3.0 as a push-pull type Alternate Function output. Refer to Section 6.5.3 Pin Declared as an Alternate Function Output.

– Set bit TSLE (bit 2) of register OSDER (R248, page 42) to 1. Refer to Section 7.4.8 Register Description.

– For the selected colors (i.e. those which will appear as transparent with contrast reduction), set bits BT[2:0] to 0. Refer to Section 7.4.6.4 Background Palettes.

Figure 49. Digital Translucency Output Pin Example



OSD CONTROLLER (Cont'd)

Figure 50. Digital Translucency Display Scheme using STV2238D in PQFP64 Package

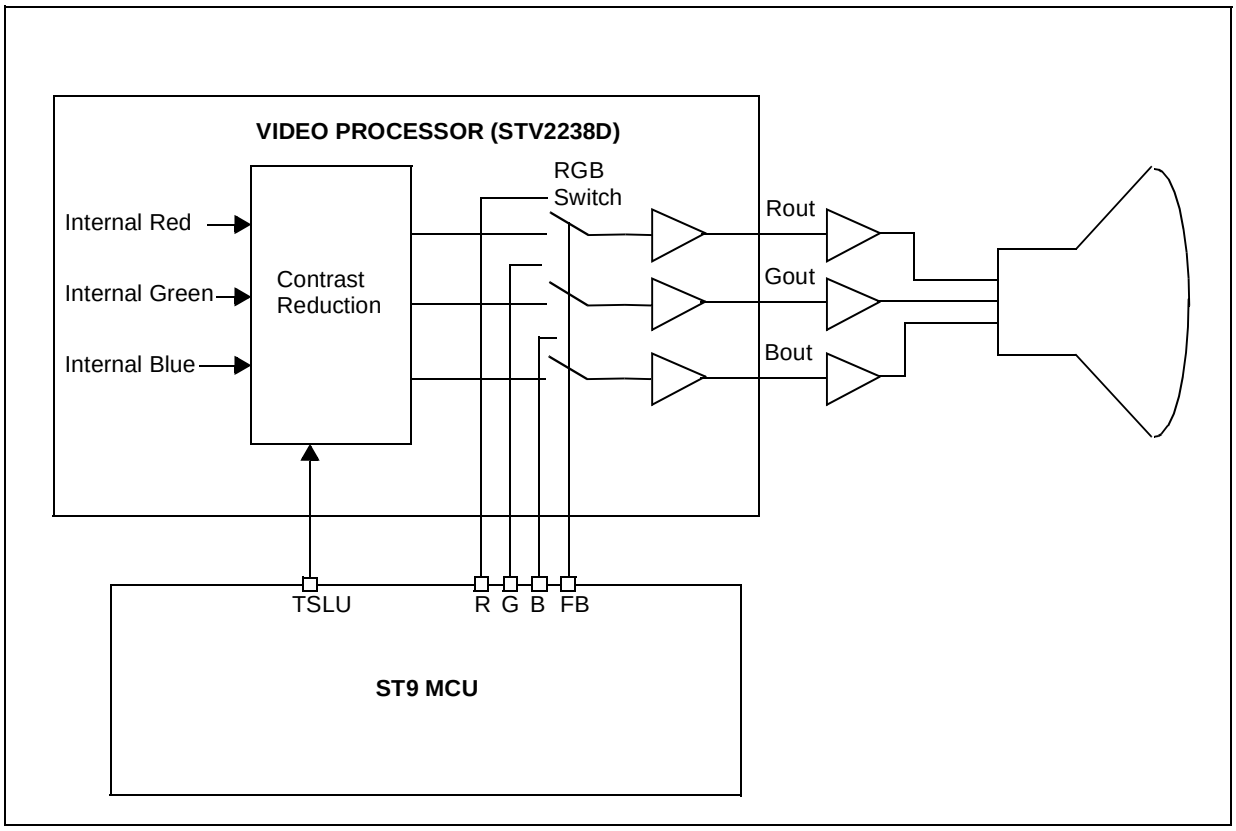
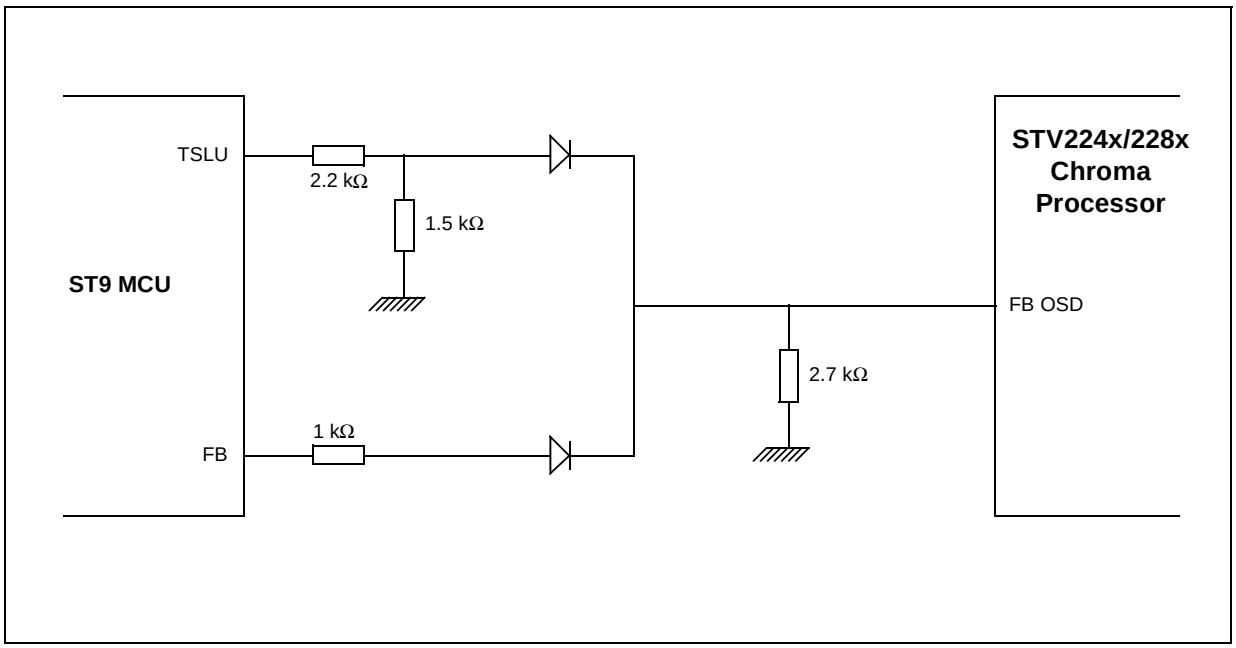


Figure 51. Application Example using STV224x/228x in SDIP56 Package



7.4.4 Horizontal and Vertical Sync

7.4.4.1 Pixel Clock Control

The Pixel clock is issued from a frequency multiplier which is locked to the main crystal frequency. The synthesized frequency is software programmable (4-bit value defining the multiplying factor) which provides flexibility for supporting various application conditions, from a basic 4/3 screen format and a “1H” horizontal sweep to a 16/9 format with a “2H” sweep, interlaced or progressive scanning. For more information, refer to the Timing and Clock Controller chapter.

Note: It is recommended to wait for a stable clock (approx. 35 ms) from the frequency multiplier before enabling the OSDRAM controller .

OSD CONTROLLER (Cont'd)

Vertical & Horizontal Sync Pulse Inputs

A spike filter has been implemented on the vertical Sync input. This circuit is inserted after internal polarity compensation of the VSYNC input signal (see VPOL bit of the Delay Register, OSDDR). It masks any spike on the vertical sync pulse with a duration smaller than 3µs. The leading edge of the VSYNC pin is affected by the vertical Sync pulse cleaner. The VSYNC edges are internally delayed by 3µs.

A Schmitt trigger provides noise immunity on the horizontal Sync pulse input and will add a delay between the deflection pulse and the effective count start of the OSD line processing.

7.4.4.2 Field Detection in interlaced mode

For TV sets working in interlaced mode, the Display controller has to retrieve the field information (some pixel information, like 18x26 matrix characters, rounding or fringe, is field based).

The Display is synchronized to HSYNC and VSYNC inputs. The phase relationship of these signals may be different from one chassis type to another. Therefore, in order to prevent vertical OSD jitter, some circuitry is implemented to provide a stable and secure field detection (OSD jitter may appear if the rising edge of an external vertical sync pulse coincides with that of an external horizontal sync pulse). This circuitry delays the vertical sync leading edge internally. The delay applied is software programmable through a 4-bit value (refer to bit DBLS in the OSDDR register).

The field information is then extracted by appropriate hardware logic.

7.4.4.3 Display Behaviour in 2H modes

The "2H" mode corresponds to a double scan display mode: the line frequency is doubled (to 31.5 kHz) compared to the traditional 60 Hz field, 262.5 lines per field. This mode requires doubling the pixel frequency and also adjusting some timing operations (refer to Section 7.4.4.1 and Section 7.4.4.2).

This feature is controlled by the DBLS bit in the OSDER register.

The double scan may be used in interlaced mode (100/120 Hz field frequency) or in progressive scanning ("non-interlaced" mode, 50/60 Hz field frequency).

This feature is enabled by the NIDS bit in the OSDER register.

RGB-FB Line Start Mute

The R, G, B & FB outputs are muted after each horizontal Sync pulse received on the HSYNC pin. The mute duration is controlled by software through a 3-bit value; these bits are called "LSM(2:0)" and are located in the Mute register OSDMR.

When the Display works in 1H mode (bit DBLS reset), the mute duration can be adjusted in 2µs steps from 2 to 14 µs. When the Display works in 2H mode (bit DBLS set), the mute duration can be adjusted in 1µs steps from 1 to 7 µs.

When the 3-bit mute value is "zero", the R, G, B & FB display outputs are muted during the duration of the horizontal Sync pulse received on the HSYNC pin.

For more details, refer to the DBLS bit in the OSDER and the LSM bits in the OSDMR register.

The HSY bit in the OSDFBR register provides an image of the mute.

OSD CONTROLLER (Cont'd)

7.4.5 Programming the Display

The row-wise RAM buffer contains the description of the characters to display:

– Row and character attributes (color, shape etc.)

– Horizontal shift code

– Character codes (addressing the Font ROM)

While one row buffer is displayed on the screen, the CPU has time to prepare the content of the next character row by filling up the second row buffer. At the time the next row must be displayed, the Display controller will point to the second row buffer, allowing the CPU to start loading data into the first row buffer for the following row. An interrupt request is generated each time the buffer pointer toggles.

The Vertical location of the next character row on the screen is programmed by software through the Event Line value (Refer to Figure 57). The vertical position of the beam is memorized by the Line counter which counts the TV horizontal synchronization pulses (called here "Scan Line"). When the Scan Line counter matches the Event Line value the buffer toggle mechanism is activated.

7.4.5.1 OSDRAM Mapping

The OSDRAM is mapped in segment 22h of the memory space.

In addition to row buffers, it is used to store color palettes.

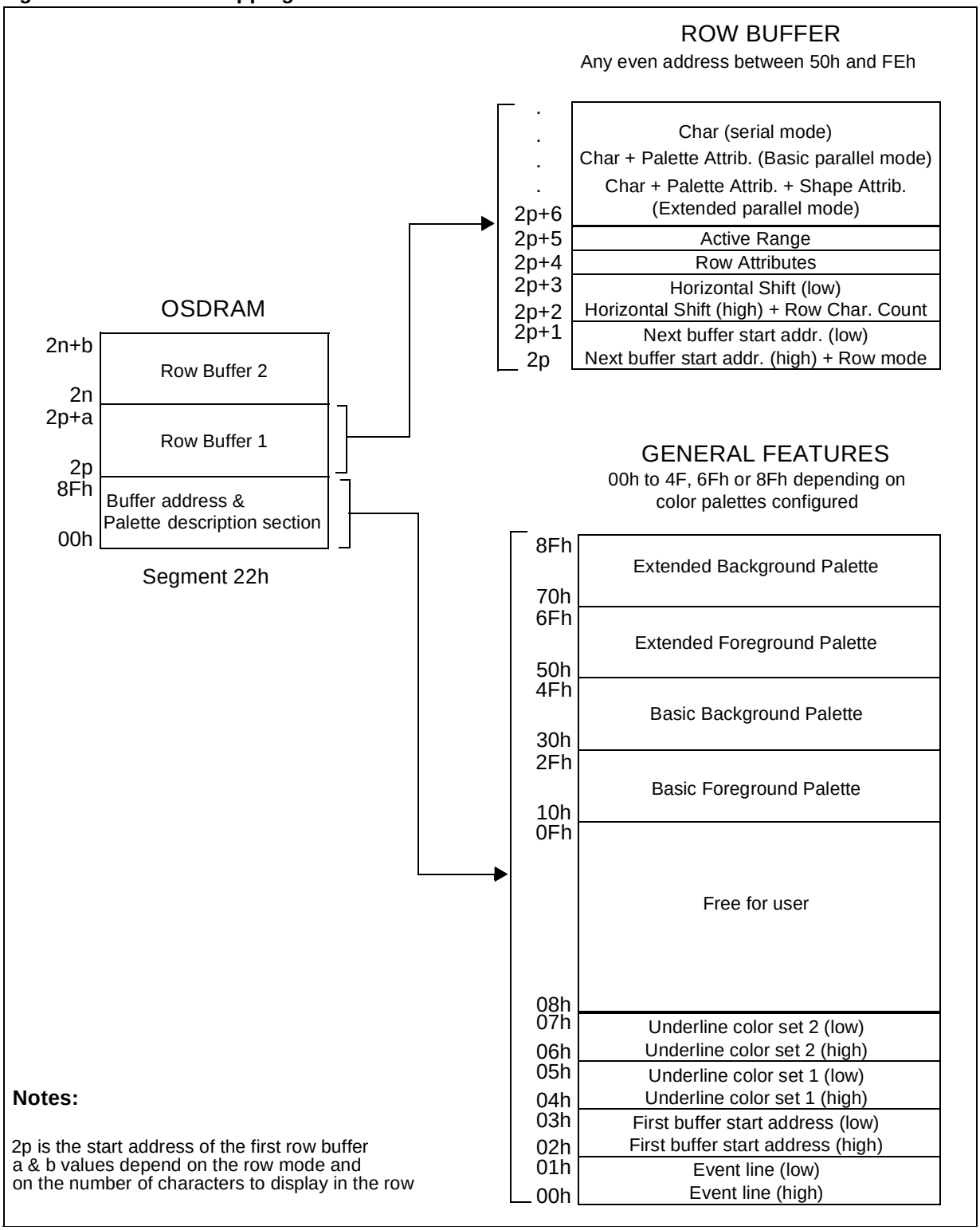
Note: The reset value of the OSDRAM contents is undefined.

General Overview

The Figure 52 gives a general overview of the OSDRAM mapping.

OSD CONTROLLER (Cont'd)

Figure 52. OSDRAM Mapping



OSD CONTROLLER (Cont'd)

7.4.5.2 Row Buffer Description

The start address for each buffer must be even.

Starting from address 2p+6, write a string corresponding to the character codes and the possible attributes as in the example below:

Byte	Serial Mode	Basic Parallel Mode	Extended Parallel Mode
1	Char or Attrib1	Char1	Char1
2	Char or Attrib2	PaletteAttrib1	PaletteAttrib1
3	Char or Attrib3	Char2	ShapeAttrib1
4	Char or Attrib4	PaletteAttrib2	Char2
5	Char or Attrib5	Char3	PaletteAttrib2
6	Char or Attrib6	PaletteAttrib3	ShapeAttrib2
...	...	...	...

7.4.5.3 OSDRAM Mapping Example 1

The left column of Figure 53 gives an example of OSDRAM mapping for the following configuration:

– The display uses basic parallel mode

– Only the two basic color palettes are used

– 36 characters are displayed per row

– The First Buffer Start Address (to be stored in 0002h and 0003h) is 0050h

– The Next Buffer Start Address to be stored in buffer 1 (address 0050h and 0051h) is 409Eh (009Eh is the address of buffer 2, and 40h is the code for basic parallel mode. Refer to Section 7.4.7.1).

– The Next Buffer Start Address to be stored in buffer 2 (address 009Eh and 009Fh) is 4050h (0050h is the address of buffer 1, and 40h is the code for basic parallel mode. Refer to Section 7.4.7.1 for more details).

7.4.5.4 OSDRAM Mapping Example 2

The right column of Figure 53 gives an example of OSDRAM mapping for the following configuration:

– The display uses extended parallel mode

– The four color palettes are used

– 16 characters are displayed per row

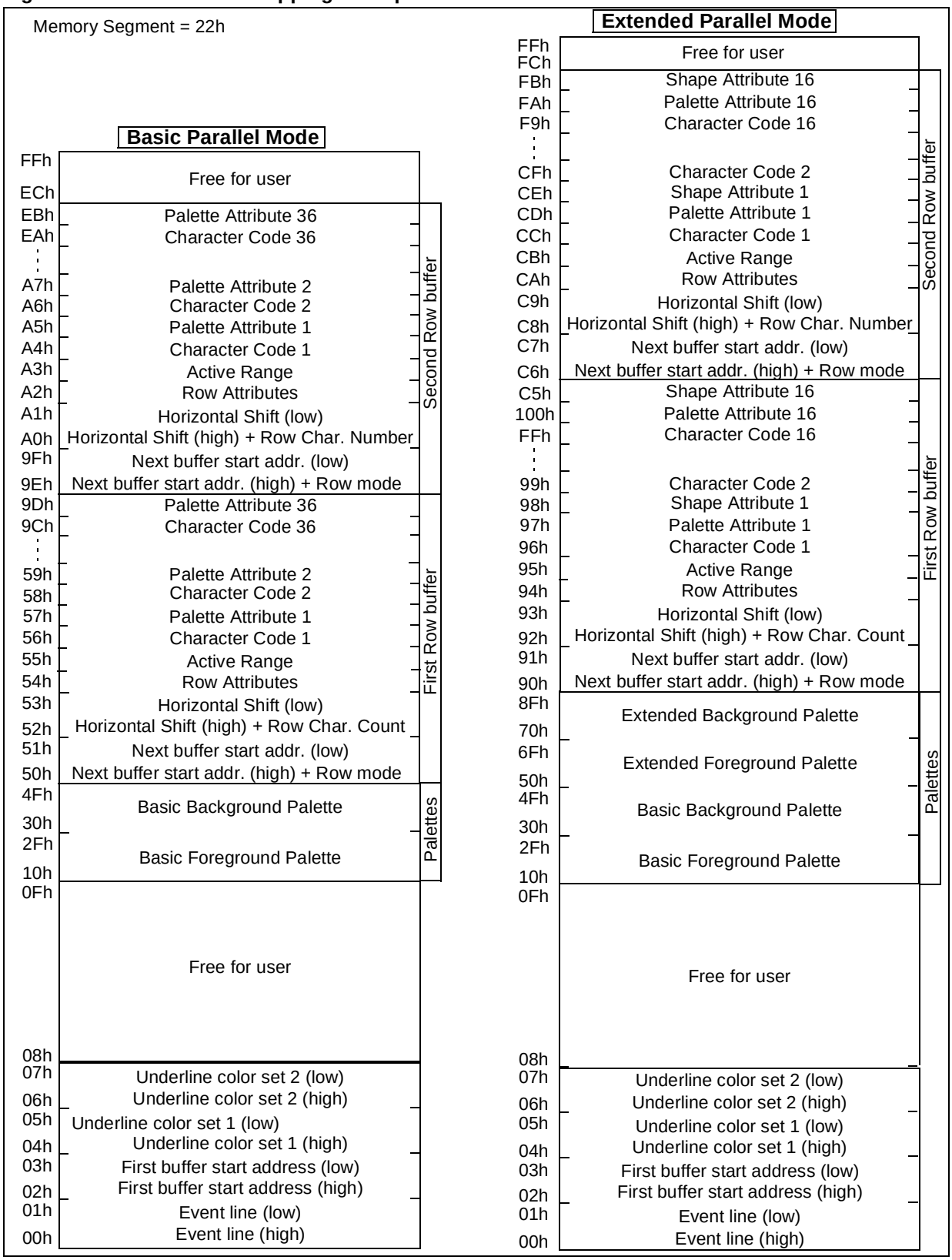
– The First Buffer Start Address (to be stored in 0002h and 0003h) is 0090h

– The Next Buffer Start Address to be stored in buffer 1 (address 0090h and 0091h) is 80C6h (00C6h is the address of buffer 2, and 80h is the code for extended parallel mode. Refer to Section 7.4.7.1 for more details).

– The Next Buffer Start Address to be stored in buffer 2 (address 00C6h and 00C7h) is 8090h (0090h is the address of buffer 1, and 80h is the code for extended parallel mode. Refer to Section 7.4.7.1 for more details).

Note: To keep some OSDRAM locations free, configure only those features that you really use (color palettes, underline palettes), as shown in the two examples.

Figure 53. Parallel Mode Mapping Examples



OSD CONTROLLER (Cont'd)

7.4.5.5 Font ROM

To address the characters in Font ROM, refer to Figure 54. To obtain the character code, add the line code to the column code.

Example 1: The code for the 'A' character is:

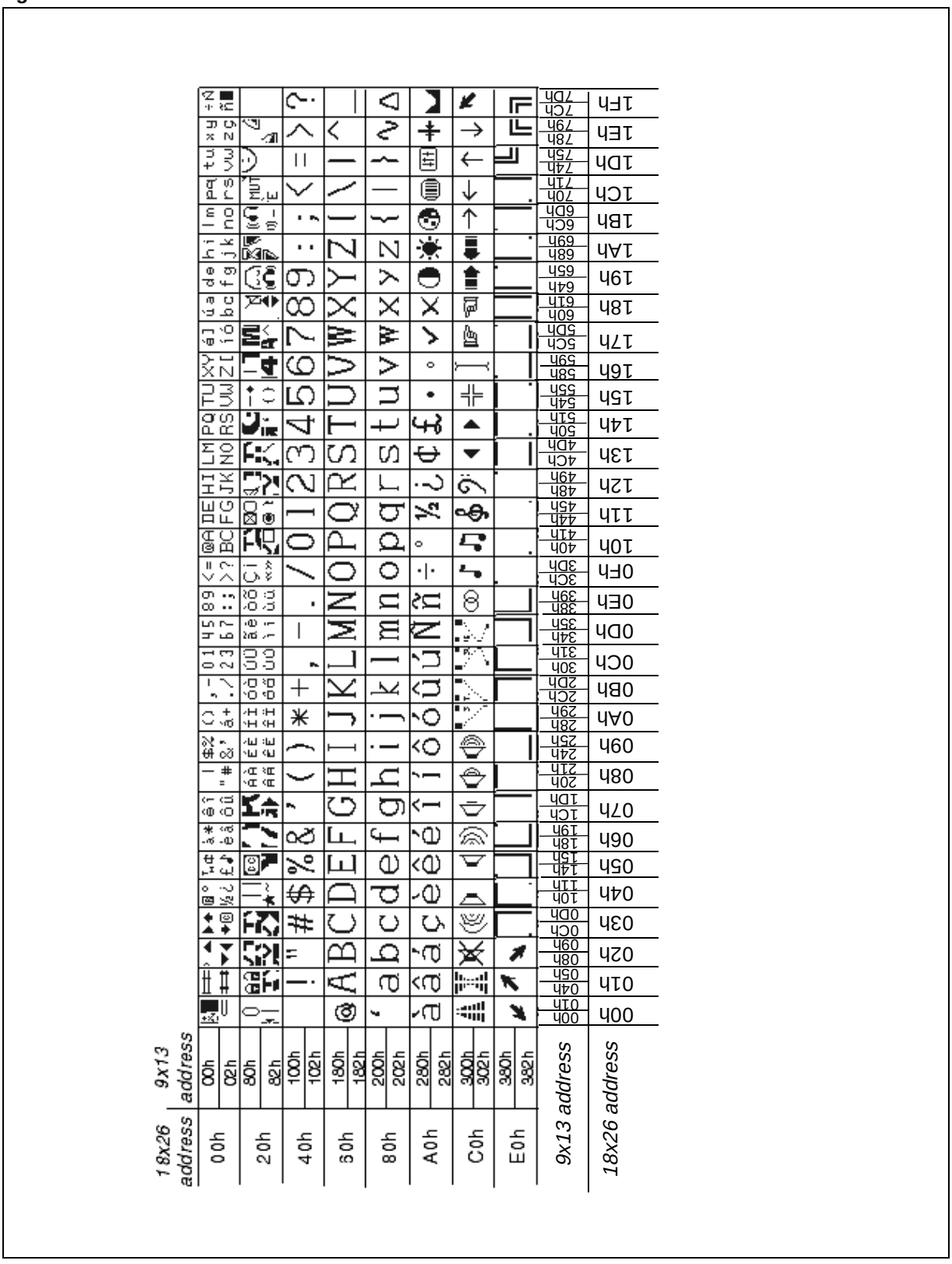
Matrix	Line Code	+	Column Code	=	Character Code
9x13	00h		41h		41h
18x26	60h		01h		61h

Example 2: The code for the '{' character is:

Matrix	Line Code	+	Column Code	=	Character Code
9x13	82h		54h		D6h
18x26	80h		1Bh		9Bh

Note: The two first 9X13 characters (address 00h and 01h) are the ST92186B control characters. They cannot be modified by the user.

Figure 54. ST92186 Font ROM Contents



OSD CONTROLLER (Cont'd)

7.4.5.6 Event Line

Address in Segment 22h: 00h (Bits 15:8), 01h (Bits 7:0)

15							8	7							0
-	-	-	-	-	-	-	EL8	EL7	EL6	EL5	EL4	EL3	EL2	EL1	EL0

EL[8:0] is a 9-bit number specifying at which TV line number the character row display should start.

For more details refer to Section 7.4.7.8

Bits 15:8: are at address 00h in Segment 22h.

Bits 7:0 are at address 01h

Bits 15:9 are reserved.

7.4.5.7 First Buffer Start Address

Address in Segment 22h: 02h (Bits 15:8), 03h (Bits 7:0)

15							8	7							0
-	-	-	-	-	-	-	FSA8	FSA7	FSA6	FSA5	FSA4	FSA3	FSA2	FSA1	0

To handle the display properly, the user must store the start address of the first OSDRAM row buffer (the row buffer containing the first row to be displayed when the Display controller is turned on). Two bytes are reserved for this in the OSDRAM. (See Figure 52).

Bits 15:8 are at address 02h in Segment 22h

Bits 7:0 are at address 03h

Bits 15:9 are reserved.

As row buffer start addresses are always even addresses, Bit 0 is forced to 0 by hardware.

OSD CONTROLLER (Cont'd)

7.4.6 Programming the Color Palettes

The Palette attributes are coded inside the two Palettes (Basic background and foreground, and Extended background and foreground); they are therefore accessible in all modes, parallel as well as serial.

A palette contains 16 colors each defined with a 16-bit word. For each color, you can define the red level (1 of 8 values), the green level (1 value among 8), the blue level (1 value among 8), and the translucency level (1 value among 8).

The color palettes also bring improvements in underline control to allow for "Windows-like" buttons.

Once programmed, color palettes can be changed in real-time when the OSD is running, that is to say when the software is filling one row buffer while the other one is displayed (but take care that the row currently displayed may be using some of the colors which you want to modify).

OSD CONTROLLER (Cont'd)

7.4.6.1 Underline Color Set 1 (USC1)

Address in Segment 22h: 04h (Bits 15:8), 05h (Bits 7:0)

15							8	7							0
				U1T2	U1T1	U1T0	U1R2	U1R1	U1R0	U1G2	U1G1	U1G0	U1B2	U1B1	U1B0

To support "windows-like" button effects, the color of the 2 (or 4 in 18x26 matrix) bottom dot lines of a character row may be defined using the Underline attributes. Two dedicated 2-byte words define 2 color sets, "Underline color set 1" called "UCS1" and "Underline color set 2" called "UCS2". They are used by the Underline mode in addition to the current foreground color.

This provides a four color choice for both rows 12 and 13 (9x13 character matrix) or pair of rows 23-24 and 25-26 (18x26 character matrix): none (background), foreground, UCS1 or UCS2, as shown in Table 18.

The UCS1 data is mapped in a fixed OSDRAM location (See Figure 52).

Bits 15:12 = Free for the user

Bits 11:9 = **U1T[2:0]** *Underline color set 1 Translucency*
These bits configure the background translucency level applied to the color (refer to Section 7.4.3.11 for more details).
U1T[2:0] = 7 means that this color will be fully opaque (no video mixed in it on the display)
U1T[2:0] = 0 means that this color will be fully transparent (the video is displayed instead of this color)

Bits 8:6 = **U1R[2:0]** *Underline color set 1 Red color*
These bits configure the background red intensity for the color.
U1R[2:0] = 0 means that no red is used to define the color.
U1R[2:0] = 7 means that the maximum red intensity is used in the color.

Bits 5:3 = **U1G[2:0]** *Underline color set 1 Green color*
These bits configure the background green intensity for the color.
U1G[2:0] = 0 means that no green is used to define the color.
U1G[2:0] = 7 means that the maximum green intensity is used in the color.

Bits 2:0 = **U1B[2:0]** *Underline color set 1 Blue color*
These bits configure the background blue intensity for the color.
U1B[2:0] = 0 means that no blue is used to define the color.
U1B[2:0] = 7 means that the maximum blue intensity is used in the color.

OSD CONTROLLER (Cont'd)

7.4.6.2 Underline Color Set 2 (UCS2)

Address in Segment 22h: 06h (Bits 15:8), 07h (Bits 7:0)

15								8	7						0
-	-	-	-	U2T2	U2T1	U2T0	U2R2	U2R1	U2R0	U2G2	U2G1	U2G0	U2B2	U2B1	U2B0

Bits 15:12 = Free for the user

Bits 11:9 = **U2T[2:0]** *Underline color set 1 Translucency*
These bits configure the background translucency level applied to the color (refer to Section 7.4.3.11 for more details).
U2T[2:0] = 7 means that this color will be fully opaque (no video mixed in it on the display)
U2T[2:0] = 0 means that this color will be fully transparent (the video is displayed instead of this color)

Bits 8:6 = **U2R[2:0]** *Underline color set 1 Red color*
These bits configure the background red intensity for the color.
U2R[2:0] = 0 means that no red is used to define the color.
U2R[2:0] = 7 means that the maximum red intensity is used in the color.

Bits 5:3 = **U2G[2:0]** *Underline color set 1 Green color*
These bits configure the background green intensity for the color.
U2G[2:0] = 0 means that no green is used to define the color.
U2G[2:0] = 7 means that the maximum green intensity is used in the color.

Bits 2:0 = **U2B[2:0]** *Underline color set 1 Blue color*
These bits configure the background blue intensity for the color.
U2B[2:0] = 0 means that no blue is used to define the color.
U2B[2:0] = 7 means that the maximum blue intensity is used in the color.

Warning: The UCS1 and UCS2 data may be used as "row attributes" providing more than 3 underline colors on screen. This requires taking some care when their contents are modified.

The UCS1 and UCS2 contents are fetched for display only when dot lines 12/13 of the character are being processed, but at that time (while dot lines 12 and 13 are processed) any change to UCS1 or UCS2 is forbidden.

Software should change the UCS1 or UCS2 while the display processes character dot lines 1 to 11. It is recommended to associate the UCS1 or UCS2 management of the currently displayed buffer with the routine which handles the next buffer preparation. Refer to Section 7.4.7.9.

OSD CONTROLLER (Cont'd)

7.4.6.3 Foreground Palettes

15							8	7							0
	U2	U1	U0	FT2	FT1	FT0	FR2	FR1	FR0	FG2	FG1	FG0	FB2	FB1	FB0

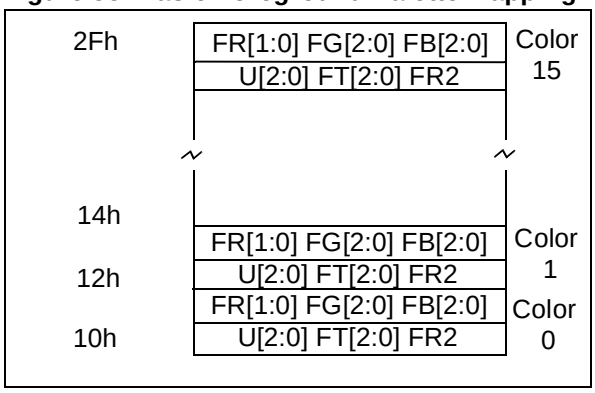
The foreground palettes (Basic and Extended) both use the same principle:

– The basic foreground palette is stored in OS-DRAM (segment 22h) starting from 10h to 2Fh (see Figure 52).

– The extended foreground palette is stored in OS-DRAM (segment 22h) starting from 50h to 6Fh (see Figure 52).

A 16-bit word is used to define each color in the palette, located at even addresses between 10h and 2Eh (even value) for the basic foreground palette, and between 50h and 6Eh (even value) for the extended foreground one.

Figure 55. Basic Foreground Palette Mapping



Bit 15 = Free for the user

Bits 14:12 = **U[2:0]** *Underline mode control bits.*
These bits configure the underline mode for the dot line 12 (lines 23 and 24 if using the 18x26 matrix) and line 13 (lines 25 and 26 if using the 18x26 matrix).
See Table 18.

Bits 11:9 = **FT[2:0]** *Foreground Translucency*
These bits configure the foreground translucency level applied to the color (refer to Section 7.4.3.11 for more details).
FT[2:0] = 7 means that this color will be fully opaque (no video mixed in it on the display)
FT[2:0] = 0 means that this color will be fully transparent (the video is displayed instead of this color)

Bits 8:6 = **FR[2:0]** *Foreground Red color*
These bits configure the foreground red intensity for the color.
FR[2:0] = 0 means that no red is used to define the color.
FR[2:0] = 7 means that the maximum red intensity is used in the color.

Bits 5:3 = **FG[2:0]** *Foreground Green color*
These bits configure the foreground green intensity for the color.
FG[2:0] = 0 means that no green is used to define the color.
FG[2:0] = 7 means that the maximum green intensity is used in the color.

Bits 2:0 = **FB[2:0]** *Foreground Blue color*
These bits configure the foreground blue intensity for the color.
FB[2:0] = 0 means that no blue is used to define the color.
FB[2:0] = 7 means that the maximum blue intensity is used in the color.

OSD CONTROLLER (Cont'd)

Table 18. Underline Mode Control Bits Description

U2	U1	U0	Underline color for a 9x13 dot matrix	Underline color for a 18x26 dot matrix
0	0	0	No underline	No underline
0	0	1	Line 12: foreground color	Lines 23-24: foreground color
0	1	0	Line 13: underline color set 1	Lines 25-26: underline color set 1
0	1	1	Line 13: underline color set 2	Lines 25-26: underline color set 2
1	0	0	Line 12-13: underline color set 1	Lines 23-24-25-26: underline color set 1
1	0	1	Line 12-13: underline color set 2	Lines 23-24-25-26: underline color set 2
1	1	0	Line 12: underline color set 1; line 13: underline color set 2	Lines 23-24: underline color set 1; lines 25-26: underline color set 2
1	1	1	Line 12: underline color set 2; line 13: underline color set 1	Lines 23-24: underline color set 2; lines 25-26: underline color set 1

Note: Take care when changing (or stopping) underline mode for a character, when the background color is displayed with "underline mode change in the center of the character" (M bit in Background palette).

In this case, the underline doesn't stop at the end of the character but stops in the middle of the following character. The underline color used is the last background color displayed in the current pixel line.

This always happens, when changing underline mode in the following ways:

– From THICK to THIN or no underline

– From one THIN color to another when the underlining is not on the same line.

(THICK underline, in a 9 x 13 matrix, is underlining on 2 lines and, in a 18 x 26 matrix, on 4 lines.
THIN underline, in a 9 x 13 matrix, is underlining on 1 line and, in a 18 x 26 matrix, on 2 lines.)

OSD CONTROLLER (Cont'd)

7.4.6.4 Background Palettes

15							8	7							0
M				BT2	BT1	BT0	BR2	BR1	BR0	BG2	BG1	BG0	BB2	BB1	BB0

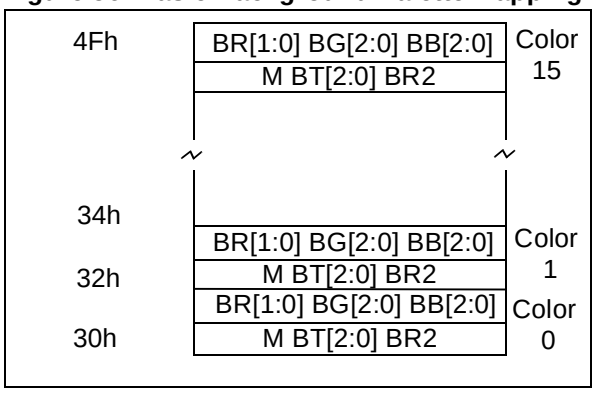
The background palettes (Basic and Extended) both use the same principle:

The basic background palette is stored in OSDRAM (segment 22h) starting from 30h to 4Fh (see Figure 52).

The extended background palette is stored in OSDRAM (segment 22h) starting from 70h to 8Fh (see Figure 52).

A 16-bit word is used to define each color in the palette, located at even addresses between 30h and 4Eh (even value) for the basic background palette, and between 70h and 8Eh (even value) for the extended background one.

Figure 56. Basic Background Palette Mapping



Bit 15 = **M** *Background color change bit.*
This bit determines where the background color change occurs.
0: The background color change takes effect immediately.
1: The background color change occurs in the center of the character.

Notes:

– When the preceding character is slanted (italics on, only available in serial mode), a background color change only occurs in the center of the character regardless of the M bit value.

– If the M bit is used in parallel mode at the beginning of a row, it is strongly recommended to insert a space or null character before setting the M bit in order to control the first half background color.

Bits 14:12 = free for the user

Bits 11:9 = **BT[2:0]** *Background Translucency*
These bits configure the background translucency level applied to the color (refer to Section 7.4.3.11 for more details).
BT[2:0] = 7 means that this color will be fully opaque (no video mixed in it on the display)
BT[2:0] = 0 means that this color will be fully transparent (the video is displayed instead of this color)

Bits 8:6 = **BR[2:0]** *Background Red color*
These bits configure the background red intensity for the color.
BR[2:0] = 0 means that no red is used to define the color.
BR[2:0] = 7 means that the maximum red intensity is used in the color.

Bits 5:3 = **BG[2:0]** *Background Green color*
These bits configure the background green intensity for the color.
BG[2:0] = 0 means that no green is used to define the color.
BG[2:0] = 7 means that the maximum green intensity is used in the color.

Bits 2:0 = **BB[2:0]** *Background Blue color*
These bits configure the background blue intensity for the color.
BB[2:0] = 0 means that no blue is used to define the color.
BB[2:0] = 7 means that the maximum blue intensity is used in the color.

OSD CONTROLLER (Cont'd)

7.4.7 Programming the Row Buffers

The 2 row buffers are based on the same structure (Figure 52):

– Next Buffer Start Address

– Row Mode

– Horizontal Shift

– Row Character Count

7.4.7.1 Next Buffer Start Address And Row Mode

Address in Segment 22h: 2p (Bits 15:8), 2p + 1 (Bits 7:0). See Figure 52.

15							8	7							0
WM2	WM1						NSA8	NSA7	NSA6	NSA5	NSA4	NSA3	NSA2	NSA1	0

To display more than 1 character row on the screen, you must specify the start address of the next OSDRAM row buffer (the row buffer containing the next row to be displayed when the current row is completely processed). Two bytes are reserved for this in the OSDRAM. (See Figure 52).

As it is possible to display each row using different modes (serial, basic parallel, and extended parallel), the row mode has to be specified for the current row buffer.

Bits 15:14 = **WM[2:1]** *Serial/parallel Row Mode control*
These bits define the row mode for the buffer defined by the Next Buffer Start Address (NSA[8:0] bits). Refer to Table 19 for details.

Table 19. Serial/Parallel mode control

WM2	WM1	Mode
1	1	(reserved)
1	0	Extended Parallel
0	1	Basic Parallel
0	0	Serial

Bits 13:9 = Free for the user

Bits 8:0 = **NSA[8:0]** *Next buffer Start Address*
These bits define the start address of the next row buffer.
As row buffer start addresses are always even addresses, the NSA0 is not implemented, and Bit 0 is forced to 0 by hardware.

OSD CONTROLLER (Cont'd)

7.4.7.2 Horizontal Shift and Row Character Count

Address in Segment 22h: 2p + 2 (Bits 15:8), 2p + 3 (Bits 7:0). See Figure 52.

15							8	7							0
RCN5	RCN4	RCN3	RCN2	RCN1	RCN0	HS9	HS8	HS7	HS6	HS5	HS4	HS3	HS2	HS1	HS0

For each row to be displayed, the number of characters in the row, and the horizontal position of the row on the screen need to be specified.

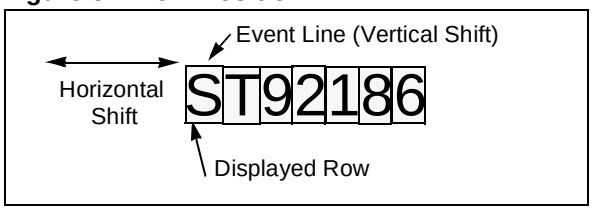
Let's assume that the start address of the current row buffer is 2p (even address).

Bits 15:10 = **RCN[5:0]** *Row Character Count*
These bits define the number of characters to display in the current row.
The Display controller allows to display from 1 to 63 characters (in parallel mode) or 61 (in serial mode, as the first 2 bytes are taken as attributes).. For example, a 36 character row requires programming RCN[5:0] = 24h.

Bits 9:0 = **HS[9:0]** *Horizontal Shift*
These bits define the horizontal shift value. They specify, in terms of number of Pixel clock periods, the horizontal shift applied from the leading edge of the Hsync pulse to the beginning of the first displayed character (1st character in parallel modes, 3rd character in serial mode). Refer to Figure 57. Loading any value smaller than 01h is forbidden.

The result is given by the formula:
Horizontal shift = [HS[9:0]+ 47] * (2*PIXCLK)
(where PIXCLK is the clock issued from the Skew Corrector). Refer to the Timing and Clock Control chapter for programming information.

Figure 57. Row Position



OSD CONTROLLER (Cont'd)

7.4.7.3 Row Attributes

Address in Segment 22h: 2p + 4. See Figure 52.

7							0
FM	UH		SY	SX	FON	ROU	FR

For each row to be displayed, specify the font matrix used (9x13 or 18x26), the size of the characters, the rounding, the fringe and the flash mode.

Let's assume that the start address of the current row buffer is 2p (even address).

Bit 7 = **FM** *Font Matrix*
This bit selects the 9x13 or the 18x26 font matrix for the current row.
The FM bit is not an address extension bit but it affects how the Font ROM content is addressed.
0: the characters use a 9x13 font matrix.
1: the characters use an 18x26 font matrix

Bit 6 = **UH** *Upper Half*
When 18x26 characters are displayed in double height (i.e. if the DBLY parallel attribute is set), the UH bit defines if the current row displays the lower or upper half of the character.
This bit has no effect when a 9x13 matrix is used or when the character has normal height (when DBLY= 0).
0: the lower half of the double-height character is displayed.
1: the upper half of the double-height character is displayed.

Bit 5 = Free for the user

Bit 4 = **SY** *Vertical Size control bit*
This bit defines the character Dot height
Refer to Table 20, Table 21, and Table 22 for complete details.

Note: The dot height is also affected by DBLY, defined in the parallel attribute section.

Bit 3 = **SX** *Horizontal Size control bit*
This bit defines the character Dot width
Refer to Table 20, Table 21, and Table 22 for complete details.

Note: The dot width is also affected by DBLX, defined in the parallel attribute section.

Table 20. 9x13 Font Matrix

SY	SX	Dot Height	Dot Width	Matrix size
0	0	2 lines	2 pixels	1Y-1X
0	1	2 lines	3 pixels	1Y-1.5X
1	0	4 lines	3 pixels	2Y-1.5X
1	1	4 lines	4 pixels	2Y-2X

Table 21. 18x26 Font Matrix (1)

SX	Character width DBLX=0	Dot width		Character width DBLX=1	Dot width
0	1X	1 pixel		2X	2 pixels
1	2X	2 pixels		4X	4 pixels

Table 22. 18x26 Font Matrix (2)

SY	Vertical Dot Size DBLY = 0	Vertical Dot Size DBLY = 1
0	1 line	2 lines
1	2 lines	4 lines

Note: In 18x26 matrix mode, this mechanism provides 7 different sizes which are: 1Y-1X, 2Y-2X, 4Y-4X but also 2Y-1X, 1Y-2X, 4Y-2X, 2Y-4X.

Note: When the Display controller works in basic parallel mode, the DBLX and DBLY bits are not accessible, and are assumed to be always programmed to "0".

Bit 2 = **FON** *Flash ON control bit*
This bit is used to control the flashing period by software. It is only available in serial and extended parallel modes. This bit has no effect in basic parallel mode.
0: the flashing mechanism is disabled for the whole row in all modes.
1: the flashing mechanism is enabled for the whole row. All characters described in the row with a serial or a parallel flash attribute are displayed as space with background color. Underline also flashes.

OSD CONTROLLER (Cont'd)

Bit 1 = **ROU** *Rounding control bit*
This bit enables or disables the rounding for the whole row.
0: the rounding is disabled
1: the rounding is enabled

Note: For a 18x26 matrix size, there is no rounding when the character size is (1X,1Y). For any other (X,Y) size combinations, the rounding is possible for the 18x26 font matrix.

Bit 0 = **FR** *Fringe control bit*
This bit enables or disables the fringe for the whole row. The fringe mechanism can be activated for any size and matrix format.
0: the character fringe is disabled
1: the character fringe is enabled.

Note: In case of fringe usage in 1Y vertical size and interlaced mode, a flicker may appear on screen as the fringe information is built on a field basis.

7.4.7.4 Active Range

Address in Segment 22h: 2p + 5. See Figure 52.

7							0
RS3	RS2	RS1	RS0	RE3	RE2	RE1	RE0

The active range feature is useful for software controlled smooth vertical scrolling (up or down).

For each row to be displayed, the first line (RS) and the last line (RE) to be displayed for the current row have to be specified.

The two values (RS[3:0] and RE[3:0]) are compared to the row line counter value. If the value of the counter is outside the active range (less than RS[3:0] or greater than or equal to RE[3:0]), the border color is displayed as defined by its attributes. Otherwise, if the counter value is inside the active range, the normal character pixel processing and display is done (See Figure 58).

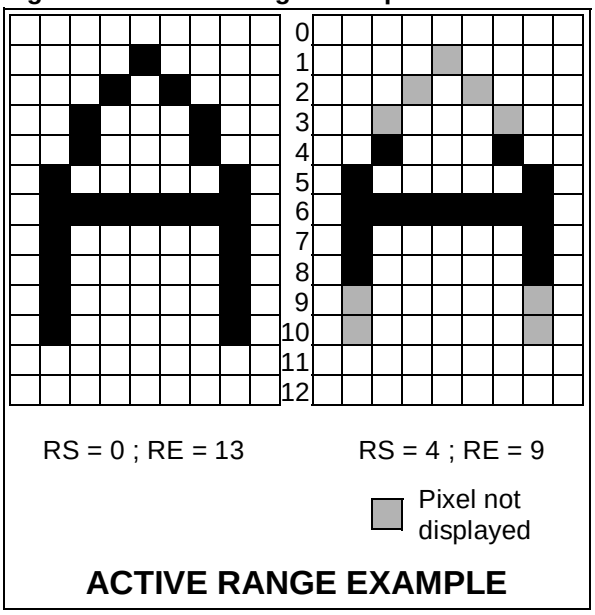
Let's assume that the start address of the current row buffer is 2p (even address).

Bits 7:4 = **RS[3:0]** *Active Range Start value*
The RS[3:0] value range is 0h-Ch (0-12) in all cases (9x13 or 18x26 character matrix).

Bits 3:0 = **RE[3:0]** *Active Range End value*
The RE[3:0] value range is 1h-Dh (1-13) in all cases (9x13 or 18x26 character matrix).

Note: For 18x26 matrix characters, the active range is calculated by pair of TV lines, i.e. the active range always starts on the first field and finishes on the second field (in interlaced mode).
In case of non-interlaced display mode, the active range is calculated by pair of TV lines.

Figure 58. Active Range Example



OSD CONTROLLER (Cont'd)

7.4.7.5 Serial mode

In serial mode, only 1 byte is used to describe the character code or the attribute (Figure 52):

– If the most significant bit (Bit 7) of this byte is 0, the byte represents a character code.

– If the most significant bit (Bit 7) of this byte is 1, the byte represents a serial attribute. Then the display controller uses Bit 6 to determine whether the attribute is a foreground serial attribute (Bit 6 = 0) or a background serial attribute (Bit 6 = 1).

A consequence of this structure is that in serial mode, only the first 128 characters stored in the font ROM can be accessed (the value of the 7 least significant bits of the character code = the character number in font ROM).

The first two bytes of the row buffer describing the row are displayed as border color. The first character to be displayed in serial mode is in fact the 3rd of the row buffer.

All the attributes (background and foreground) are displayed as space with background color.

Let's assume that the start address of the current row buffer is 2p (even address).

In this case the character codes and attributes are stored in the OSDRAM at the address 2p+5+z, where z value is 1 to RCN (RCN is the row character count defined in Section 7.4.7.2).

Character Code in Serial Mode

~~A~~ddress in Segment 22h: 2p + 5 + z. **See** Figure 52**.**

7							0
0	CHC6	CHC5	CHC4	CHC3	CHC2	CHC1	CHC0

Bits 6:0 = **CHC[6:0]** *Character Code in serial mode*
The CHC[6:0] value points to one of the first 128 characters stored in the font ROM.

BACKGROUND SERIAL ATTRIBUTE
Address in Segment 22h: 2p + 5 + z. See Figure 52.

7							0
1	1	x	x	BP3	BP2	BP1	BP0

Bits 5:4 Reserved

Bits 3:0 = **BP[3:0]** *Background Palette pointer*
The BP[3:0] value points to one of the 16 predefined entries of the background palette for the background color and the translucency.

For example, BP[3:0] = 0 points to the first background palette entry, 30h & 31h if the palette swap bit PASW of the OSDDR register is reset (see registers description for more details).

Note: the display of the background serial attribute is affected by the use of the Italics (see the foreground serial attribute) and also by the "M" bit located in the Background Palette (see Section 7.4.6.4 for further details).

OSD CONTROLLER (Cont'd)

Foreground Serial Attribute

Address in Segment 22h: 2p + 5 + z. See Figure 52

7							0
1	0	FLA	IT	FP3	FP2	FP1	FP0

Bit 5 = **FLA** *Flash control bit*
This bit controls the flashing feature (see Section 0.2.4.2).
0: All the following characters in the row are not affected by the flashing mechanism.
1: All the following characters in the row follow the flashing mechanism.

Note: Flashing characters are alternatively displayed as spaces and in normal mode (non-flashing), depending on the value of the FON bit in the Row Attribute byte (see Section 7.4.7.3).
The flash rate is controlled by software by toggling the "FON" bit.

Bit 4 = **IT** *Italics control bit*
This bit enables the italic feature for the row. (See Section 7.4.3.6)
0: Italics are disabled.
1: All the following characters, until the end of the row, or the next foreground serial attribute are displayed in italics.

Note: Italics mode is only available in serial mode.

Bits 3:0 = **FP[3:0]** *Foreground Palette pointer*
The FP[3:0] value points to one of the 16 predefined entries of the foreground palette for foreground color, translucency and underline style of all the following characters (see Section 7.4.6.3).

For example, FP[3:0] = 0 points to the first foreground palette entry, 10h & 11h if the palette swap bit PASW of the OSDDR register is reset.

7.4.7.6 Basic parallel mode

In basic parallel mode, each character code (1 byte) is followed by a palette attribute (1 byte). See Figure 52.

Let's assume that the start address of the current row buffer is 2p (even address).

In this case the character codes are stored in OSDRAM at the address 2p+4+2z, and the palette attributes are stored in the OSDRAM at the address 2p+5+2z, where z ranges from 1 to RCN (RCN is the row character count defined in Section 7.4.7.2).

The character code structure allows pointing to the first 256 characters of the font ROM (the character code value = the character number in font ROM).

CHARACTER CODE IN BASIC PARALLEL MODE
Address in Segment 22h: 2p+4+2z. See Figure 52.

7							0
CHC7	CHC6	CHC5	CHC4	CHC3	CHC2	CHC1	CHC0

Bits 7:0 = **CHC[7:0]** *Character Code in basic parallel mode*
The CHC[7:0] value points to one of the first 256 characters stored in the font ROM.

PALETTE ATTRIBUTE
Address in segment 22h: 2p+5+2z. See Figure 52.

7							0
FP3	FP2	FP1	FP0	BP3	BP2	BP1	BP0

Bits 7:4 = **FP[3:0]** *Foreground Palette pointer*
The FP[3:0] value points to one of the 16 predefined entries of the foreground palette for the foreground color, the translucency and the underline style of all the following characters (see Section 7.4.6.3 for further details).

For example, FP[3:0] = 0 points to the first foreground palette entry, 10h & 11h if the palette swap bit PASW of the OSDDR register is reset (see registers description for more details).

Bits 3:0 = **BP[3:0]** *Background Palette pointer*
The BP[3:0] value points to one of the 16 predefined entries of the background palette for the background color and the translucency.

For example, BP[3:0] = 0 points to the first background palette entry, 30h & 31h if the palette swap bit PASW of the OSDDR register is reset (see registers description for more details).

Note: the background color of the character is affected by the use of the "M" bit located in the Background Palette (see Section 7.4.6.4).

OSD CONTROLLER (Cont'd)

7.4.7.7 Extended parallel mode

In extended parallel mode, each character code (1 byte) is followed by a palette attribute (1 byte) and a shape attribute (1 byte). Refer to Figure 52.

Let's assume that the start address of the current row buffer is 2p (even address).

The character codes are stored in the OSDRAM at the address 2p+3+3z, the palette attributes are stored at the address 2p+4+3z, and the shape attributes are stored at the address 2p+5+3z, where z range value is 1 to RCN (RCN is the row character count defined in Section 7.4.7.2).

The character code structure, using the shape attribute, allows you to point to any of the font ROM characters.

The shape attribute definition depends on the font matrix used for the row (it depends on the FM bit, see Section 7.4.7.3).

CHARACTER CODE IN EXTENDED PARALLEL MODE

Address in segment 22h: 2p+3+3z. See Figure 52.

7							0
CHC7	CHC6	CHC5	CHC4	CHC3	CHC2	CHC1	CHC0

Bits 7:0 = **CHC[7:0]** *Character Code in extended parallel mode*
The CHC[7:0] bits are used, combined with 3 or 1 bits of the shape attribute (for a 9x13 or 18x26 matrix), to point to any of the characters stored in the font ROM (refer to the shape attribute description for more details).

PALETTE ATTRIBUTE

Address in segment 22h: 2p+4+3z. See Figure 52.

7							0
FP3	FP2	FP1	FP0	BP3	BP2	BP1	BP0

Refer to Section 7.4.7.6 for the bit description.

OSD CONTROLLER (Cont'd)

SHAPE ATTRIBUTE - 9x13 MATRIX

Address in segment 22h: 2p+5+3z. See Figure 52.

7							0
CODX	SPL1	SPL0	FXP	BXP	FOC	SHA	FRC

Bit 7 = **CODX** *Character Code Extension*
This bit is used with SPL[1:0] as the character address extension. Thus it is possible to address any of the 9x13 characters of the font ROM. (See the SPL[1:0] bit description for more details)

Bits 6:5 = **SPL[1:0]** *Character Split control bits*
These "character split" bits are used with the CODX bit as the character address extension. It is then possible to address any of the 9x13 characters of the font ROM.
If the character code is ChC[7:0] (1 byte), then the character addressed with this structure is: CODX.SPL1.SPL0.ChC[7:0]

Bit 4 = **FXP** *Foreground Extended Palette addressing bit*
This bit is combined with the PASW bit in the OSDDR register to give the MSB bit of the foreground palette address. It allows the foreground palette, (basic or extended), to be selected on a per-character basis. See Table 23 for details.

Table 23. Foreground palette selection

PASW	FXP	Foreground palette used
0	0	basic
0	1	extended
1	0	extended
1	1	basic

Bit 3 = **BXP** *Background Extended Palette addressing bit*
This bit is combined with the PASW bit in the OSDDR register to give the MSB bit of the background palette address. It allows the background palette, (basic or extended), to be selected on a per-character basis. See Table 24 for details.

PASW	BXP	Background palette used
0	0	basic
0	1	extended
1	0	extended
1	1	basic

Bit 2 = **FOC** *Flash On Character control bit*
This bit enables the flash mechanism for the current character.
0: The flash mechanism is disabled. The current character is displayed, whatever the flash row attribute value (FON) is (see Section 7.4.7.3).
1: The flash mechanism is enabled. If the flash row attribute (FON) is "on" (see Section 7.4.7.3), the character is displayed as space using the background color.

Bit 1 = **SHA** *shadow mode control bit*
This bit enables or disables a black shadow shape on the right and bottom edges of the current character foreground.
0: No shadow is added
1: A black shadow is added on the current character.

Note: This shadow shape follows the same algorithm as the fringe (see Section 7.4.3.7 for further details).

Bit 0 = **FRC** *Fringe on Character control bit*
This bit enables or disables a fringe on the current character foreground.
0: No fringe is added.
1: A fringe is added on the current character if the fringe row attribute bit FR is set (see Section 7.4.7.3 for more details).

Note: The fringe follows the algorithm described in Section 7.4.3.7.

OSD CONTROLLER (Cont'd)

SHAPE ATTRIBUTE - 18x26 MATRIX

Address in segment 22h: 2p+5+3z. See Figure 52.

7							0
CODX	DBLY	DBLX	FXP	BXP	FOC	SHA	FRC

Bit 7 = **CODX** *Character Code Extension*
This bit is used as the character address extension. Then it is possible to address any of the 18x26 characters of the font ROM.
Let's assume that the character code is ChC[7:0] (1 byte), then the character addressed with this structure is: CODX.ChC[7:0]

Bit 6 = **DBLY** *Double height control bit*
This bit controls the double height feature applied on the current row height.
0: The character is displayed with the current row height, as defined by SY (Section 7.4.7.3).
1: The current character is displayed with a double height than defined by SY. The display of the lower or upper half of the character is controlled by the UH row attribute bit (refer to Section 7.4.7.3).

Bit 5 = **DBLX** *Double width control bit*
This bit controls the double width feature applied to the current row character width.
0: The character is displayed with the current width as defined by SX (Section 7.4.7.3).
1: The character is displayed in double width, according to the following rules:
- It covers the next character location
- The next character location is read and decoded but not processed,
- If the character is the last one in the row, it will be truncated to its left half.

Bit 4:0 = Please refer to the bit descriptions in the 9x13 matrix shape attributes.

OSD CONTROLLER (Cont'd)

7.4.7.8 Row buffer Management

To Start the Display:

0. Write the (DION, OSDE) bits to (1,0) to access the OSDRAM with the CPU clock,
1. Initialize the color palettes,
2. Initialize the "first buffer start address" with the address of the first byte of the above Row Buffer (address 0002h & 0003h of the segment 22h),
3. Fill up one of the row buffers with the data to display the desired row (only in case the TE bit in the OSDER register has been set),
4. Initialize the Event Line value to the desired one (address 0000h & 0001h of the segment 22h),
5. Start the Display controller by programming the mode control bits (DION, OSDE) and the transfer enable bit (TE) to the desired working mode. It is mandatory to start the display following the algorithm below:

```
1   unsigned char tmp;

    spp(OSD_PG);              /* select the OSD register page */
    OSDFBR &= ~0x06;          /* reset DINT & MOIT bits */
5
    while (OSDFBR & OSDm_Vsy);          /* wait a Low to High transition on VSYNC */
    while (!(OSDFBR & OSDm_Vsy));

    tmp= OSDMR;               /* save LSM bits */
10  OSDMR &= ~0x07;           /* reset the LSM bits so that the Hsy bit will be an image of the HSYNC pulse */

    OSDER = 0x40;             /* OSDRAM interface enabled with PIXEL clock */

    di();                     /* disable all interrupts */
15
    while (OSDFBR & OSDm_Hsy);          /* wait a HSYNC pulse : Low -> High -> Low transition */
    while (!(OSDFBR & OSDm_Hsy));
    while (OSDFBR & OSDm_Hsy);

20  OSDMR = tmp;              /* recover old LSM bit value */

    spp(OSD_PG);              /* select the OSD register page */
    OSDER |= 0xE0;            /* start the display by setting the appropriate bits,
                              set at least OSDE, DION, and TE bits. Then set the other bits as required
25                            for your application. */

    ei();                     /* enable interrupts again*/
```

OSD CONTROLLER (Cont'd)

The real time control provides:

– A continuous search of matching values between Scan line and Event Line (this condition being evaluated at each TV line start).

– A display interrupt generation when the match condition is detected.

– In "full OSD mode", if the TE bit in the OSDER register is set, the switch from one row buffer to the second row buffer when the match occurs. If the TE bit in the OSDER register is reset, when a matching condition occurs, the previous row buffer will be kept.

7.4.7.9 Handling the Row Buffers in Continuous Mode:

– When the line match condition is detected, an interrupt is sent to the CPU.

Let us then assume the TE bit in the OSDER register is set. When the interrupt is executed:

– The Event Line value must be programmed to the next desired value.

– The next Row Buffer content must be filled up by the data of the next row to display. The next row buffer is easily identified using the BUFL bit in the OSDFBR register.

Let us then assume the TE bit in the OSDER register is reset. When the interrupt is fetched:

– The Event Line value must be programmed to the next desired value.

– The Next Row Buffer content might be filled up by the data of the next row to display, if desired.

– The content of the current Row Buffer is NOT displayed but simply ignored as it should have already been displayed in a previous cycle.

Note: In case the TE bit in the OSDER register is kept reset, there is no need to manage the second row buffer as it will never be used.

OSD CONTROLLER (Cont'd)

7.4.8 Register Description

To run the Display controller properly, you need to program the 7 registers that configure the display

BORDER COLOR REGISTER 2 (OSDBCR2)
R246 - Read/Write
Register Page: 42
Reset Value: 0x00 0000

7							0
B2BC		BOS2	BOS1	BOS0	BOR2	BOR1	BOR0

Bit 7 = **B2BC** *Background to Border Color control bit.*
This bit allows to force the background color of all the characters to the border color.
0: All characters backgrounds are normally displayed
1: All character backgrounds are forced to the current border color and translucency level.

Bit 6 = Reserved

Bits 5:3 = **BOS[2:0]** *Border color translucency*
These bits control the Border color translucency.
BOS[2:0] = 7 means that the border color will be fully opaque (no video mixed in it on the display)
BOS[2:0] = 0 means that the border color will be fully transparent (the video is displayed instead of this color)

Bits 2:0 = **BOR[2:0]** *Red Border color*
These bits configure the red intensity for the border color.
BOR[2:0] = 0 means that no red is used to define the border color.
BOR[2:0] = 7 means that the maximum red intensity is used in the border color.

BORDER COLOR REGISTER 1 (OSDBCR1)
R247 - Read/Write
Register Page: 42
Reset Value: 0x00 0000

7							0
DIFB	-	BOG2	BOG1	BOG0	BOB2	BOB1	BOB0

Bit 7 = **DIFB** *Digital FB control bit*
This bit selects the Fast Blanking (FB) output as analog or digital.
0: The FB DAC works as an 8-level DAC output from 0V up to 1V (with a 500ohms internal impedance to ground).
1: The FB DAC works as a 2-level DAC output, the high level providing an amplitude higher than 2.7 volts. All translucency control bits are managed as follows:
- the code (0,0,0) generates a "0" output (0 volt),
- all other codes generate a "1" output (>2.7 V).

Note: This applies to BT[2:0], FT[2:0], U2T[2:0], U2T[2:0] and BOS[2:0] (refer to Section 7.4.6.3, Section 7.4.6.4 and Section 7.4.6.1, and to the OSDBCR2 register).

Bit 6 = Reserved

Bits 5:3 = **BOG[2:0]** *Green Border color*
These bits configure the green intensity for the border color.
BOG[2:0] = 0 means that no green is used to define the border color.
BOG[2:0] = 7 means that the maximum green intensity is used in the border color.

Bits 2:0 = **BOB[2:0]** *Blue Border color*
These bits configure the blue intensity for the border color.
BOB[2:0] = 0 means that no blue is used to define the border color.
BOB[2:0] = 7 means that the maximum blue intensity is used in the border color.

OSD CONTROLLER (Cont'd)

ENABLE REGISTER (OSDER)
R248 - Read/Write
Register Page: 42
Reset Value: 0000 0000 (00h)

7							0
DION	OSDE	TE	DBLS	NIDS	TSLE	-	FPIXC

Bit 7 = **DION** *Display ON*
This bit is used in combination with the OSDE bit to control the display working mode. See Table 25.

Warning: after a reset, a valid HSYNC signal is required to write to the OSDRAM, whatever the clock rate (CPU or Pixel clock rate).

Bit 6 = **OSDE** *OSD Enable*
This bit is used in combination with the DION bit to control the display working mode. See Table 18.

Note 1: When the (DION,OSDE) bits switch from any other value to (1,1), i.e. when the controller is switched to a full OSD function, the "first buffer start address" content is used to locate the first Row buffer to process.
While the full Display function is running, the DION & OSDE bits remain set and the first buffer start address is not used again, even if both bits are rewritten to "1".

Note 2: It is strongly recommended to use state 3 **only** if the OSDRAM has been initialized using state 2.

Warning 1: States 3 and 4 (refer to Table 25) can only be used if HSYNC and VSYNC are applied on the external pins.

Warning 2: After a reset, a valid HSYNC signal is required to write to the OSDRAM, regardless of the clock rate (CPU or Pixel clock rate).

Warning 3: When the OSD is displayed, it is advised not to write to the OSDRAM when a VSYNC pulse occurs. In Normal operating mode, this configration will never happen.

Bit 5 = **TE** *Transfer Enable bit*
This bit controls the "swap to next row buffer" function whenever the Scan Line counter content matches the Event Line parameter value.
An interrupt request pulse is generated and forwarded to the core each time the match occurs regardless of the value of TE.
0: Row buffer swap disabled. The current row buffer content is simply ignored and the screen will display the border color, as if the current buffer content was already processed.
1: A Row buffer swap enabled

Note: Refer to Section 7.4.7.8 for more details about using the TE bit.

Bit 4 = **DBLS** *Double Scan bit*
This bit defines if the display works in 1H or 2H mode.
0: The display works in "single scan" or "1H" mode.
1: The display works in "double scan" or "2H" mode. The 2H mode is used in progressive scan display (60Hz field, 525 lines).

Note: the DBLS bit acts on the display vertical delay for field determination (refer to the VD[3:0] bits of the Delay register OSDDR).
The DBLS bit also acts on the Line Start Mute (refer to the LSM[2:0] bits of the Mute register OSDMR) and the HSY flag bit.

Bit 3 = **NIDS** *Non Interlaced Display control bit*
This bit selects Interlaced or Non-Interlaced mode.
0: The display works in interlaced mode (line counting, fringe and rounding algorithms are 2-field based)
1: The display works in non-interlaced mode (line counting, fringe and rounding algorithms are 1-field based).

Bit 2 = **TSLE** *Translucency Enable bit*
This bit enables or disables the digital translucency signal (TSLU) generation. (Refer to Section 7.4.3.11).
0: The TSLU signal built by the Display controller remains continuously idle regardless of OSD activity.
1: The TSLU signal carries the real time background information and can be output through the I/O pin alternate function.

Bit 1 = Reserved. Must be kept at reset.

Bit 0 = **FPIXC** *Fast Pixel Clock control bit*
This bit handles the divide-by-2 prescaler inserted between the Skew Corrector output and the Display Pixel clock input.
0: The Skew corrector clock output is divided by 2 to provide the Pixel clock.
1: The Skew corrector clock output is directly taken as the Pixel Clock.

OSD CONTROLLER (Cont'd)

Note: For further information, refer to the Timing and Clock Control chapter.

Table 25. OSDRAM Interface Configuration

DION	OSDE	OSDRAM Interface clock	OSD Function	Detailed Configuration	State
0	0	off, no RAM access	off	The OSDRAM controller and the Display are both disabled. The CPU has no access to the OSDRAM.	1
1	0	on, CPU clock	off	The Display is disabled. The OSDRAM controller is running using the CPU clock, allowing for CPU accesses.	2
0	1	on, Pixel clock	no OSD, time control on	The Display is partially enabled. The OSDRAM controller is running with the Pixel clock, allowing CPU accesses. The OSD pixel processing is disabled (RGB & FB outputs are turned off), the TV oriented time control is still running, such as event line control, interrupt generation, flag bits and field calculation. The border control is inactive.	3
1	1	on, Pixel clock	fully on	The OSDRAM controller and the Display are both enabled. The OSDRAM controller is running with the Pixel clock, allowing CPU accesses. The RGB & FB outputs are turned on. The border control is activated.	4

OSD CONTROLLER (Cont'd)

DELAY REGISTER (OSDDR)
R249 - Read/Write
Register Page: 42
Reset Value: 0xxx xxxx

7							0
PASW	HPOL	VPOL	FBPOL	VD3	VD2	VD1	VD0

Note: The display may flicker if you write to the Delay Register while OSD is fully on.

Bit 7 = **PASW** *Palette Swap bit*
The PASW bit is used in serial or basic parallel modes to provide access to the extended palettes. It is still active when you work in extended parallel mode, however it not needed as the FXP and BXP bits are available (refer to Section 7.4.7.7).
0: The basic palette sets are used.
1: The extended palette sets are used.

Bit 6 = **HPOL** *Hsync signal Polarity selection bit*
This bit has to be configured according to the polarity of the Hsync input signal.
0: Hsync input pulses are of positive polarity
1: Hsync input pulses are of negative polarity

Bit 5 = **VPOL** *Vsync signal POLarity selection bit*
This bit has to be configured according to the polarity of the Vsync input signal.
0: Vsync input pulses are of positive polarity
1: Vsync input pulses are of negative polarity

Bit 4 = **FBPOL** *Fast Blanking signal Polarity selection bit*
This bit selects the polarity of the Fast Blanking (FB) output signal.
0: FB output pulses are of positive polarity (FB active high)
1: FB output pulses are of negative polarity (FB active low)

Note: the FB signal is kept active during the VSYNC vertical retrace.

Bits 3:0 = **VD[3:0]** *Vertical Delay control bits*
This 4-bit value is used to program an internal delay on vertical sync pulses applied to VSYNC input pin.

The purpose of the programmable delay is to prevent vertical OSD jitter in case the rising edge of external vertical sync pulse coincides with thatofan external horizontal sync pulse.
The delay applied is expressed by the following equations (4 MHz is the frequency issued directly from the crystal oscillator):

for 2H display mode:
[VD[3:0]+1] * 8*(1/4MHz) =< d =< [VD[3:0]+2] * 8*(1/4MHz)

for 1H display mode:
[VD[3:0]+1] *16*(1/4MHz) =< d =< [VD[3:0]+2] *16*(1/4MHz)

Note: programming the Vertical Delay to Fh will freeze the scan line counter disabling any further RGB output.

Note: It is mandatory for the CPU to initialize the Vertical Delay Register to avoid any problems.

SCAN LINE REGISTER (OSDSLR)
R251 - Read Only
Register Page: 42
Reset Value: xxxx xxxx (xxh)

7							0
SL7	SL6	SL5	SL4	SL3	SL2	SL1	SL0

Bits 7:0 = **SL[7:0]** *Scan Line Counter Value*
These bits indicate the current vertical position of the TV beam.
The most significant bit SL8 of this counter is located in the Flag Bit register OSDFBR (see below).
This counter starts from 0 at the top of the screen (i.e. after the Vsync pulse) and is incremented by HSYNC.

OSD CONTROLLER (Cont'd)

FLAG BIT REGISTER (OSDFBR)
R250 - Read/Write
Register Page: 42
Reset Value: xxxx xxxx (xxh)

7							0
BUFL	VSY	HSY	VSDL	FIELD	DINT	-	SL8

Bit 7 = **BUFL** *Buffer Flag bit*
This bit indicates which Row Buffer of the OSD RAM is being used by the Display.
The BUFL flag is automatically re-evaluated each time the Scan & Event line matching condition is fulfilled. In case the "TE" bit is reset (see the OSDER register), the "BUFL" flag remains unchanged as NO row buffer change occurs.
0: The OSD displays the content of the second row buffer (the one NOT pointed by the "first buffer start address" value).
1: The OSD displays the content of the Row Buffer location pointed by the "first buffer start address" value.

Note: The BUFL flag is automatically reset when the (DION,OSDE) bits are switching from any other value to (1,1); it will be set at the first Buffer transfer (scan & event match and TE=1).

Bit 6 = **VSY** *Vsync status bit*
This bit gives the status of the VSYNC input signal.
0: the VSYNC input signal is inactive.
1: the VSYNC input signal is active.

Note: The VSYNC signal polarity is compensated to always provide VSY=1 during the vertical pulse.

Bit 5 = **HSY** *Hsync status bit*
This bit gives the status of the internal HS signal generated by the skew corrector and locked to the external HSYNC signal.
0: The HS signal is inactive.
1: The HS signal is active.

Note: the HSYNC signal polarity is compensated to provide HSY=1 during the horizontal pulse.

Note: HSY remains active during the whole "Line Start Mute" timing which is software controlled through both the DBLS bit and the LSM[2:0] value (see OSDER and OSDMR registers).

Bit 4 = **VSDL** *Delayed Vertical Pulse status bit*
This bit indicates the status of the VDPLS internal signal (it is the delayed vertical pulse issued from the programmable vertical delay unit as described by the OSDDR register bits VD[3:0])
0: The VDPLS internal signal is inactive
1: The VDPLS internal signal is active

Note: the VDPLS signal polarity is compensated to provide VSDL =1 during the vertical pulse.

Bit 3 = **FIELD** *Field status bit*
This bit indicates the current TV field.
0: TV beam is in field 2 (even field)
1: TV beam is in field 1 (odd field)

Bit 2 = **DINT** *Display Interrupt flag bit*
This bit is set by hardware when an OSD interrupt occurs. This bit must be reset by Software.

Bit 1 = Reserved.

Bit 0 = **SL8** *Most Significant Bit of the Scan Line counter*
Refer to the description of the OSDSLR register.

OSD CONTROLLER (Cont'd)

MUTE REGISTER (OSDMR)
R252 - Read/Write
Register Page: 42
Reset Value: 00xx x000

7							0
-	-	-	-	-	LSM2	LSM1	LSM0

Bits 7:3 = Reserved

Bits 2:0 = **LSM[2:0]** *Line Start Mute value*
These bits are used to program the mute duration after the beginning of each TV line.
When the Display works in 1H mode the mute duration can be adjusted in 2µs steps from 2 to 14 µs.
When the Display works in 2H mode the mute duration can be adjusted in 1µs steps from 1 to 7 µs.
The LSM bits also define the HSY flag duration.
The Mute duration is expressed by the following equation (the "1µs" is a frequency issued from the crystal oscillator):
for 2H display mode: T_{mute} = LSM[2:0] * (1 µs)
for 1H display mode: T_{mute} = LSM[2:0] * 2*(1 µs)
in both 1H/2H modes, if LSM[2:0] = 0 then T_{mute} = Hsync width.

OSD CONTROLLER (Cont'd)

Table 26. OSD Register Map

Register Number Page 42	Register Name	7	6	5	4	3	2	1	0
246	**OSDBCR2** Reset Value	B2BC 0	- X	BOS2 0	BOS1 0	BOS0 0	BOR2 0	BOR1 0	BOR0 0
247	**OSDBCR1** Reset Value	DIFB 0	- X	BOG2 0	BOG1 0	BOG0 0	BOB2 0	BOB1 0	BOB0 0
248	**OSDER** Reset Value	DION 0	OSDE 0	TE 0	DBLS 0	NIDS 0	TSLE 0	- 0	FPIXC 0
249	**OSDDR** Reset Value	PASW 0	HPOL X	VPOL X	FBPOL X	VD3 X	VD2 X	VD1 X	VD0 X
250	**OSDFBR** Reset Value	BUFL X	VSY X	HSY X	VSDL X	FIELD X	DINT X	- X	SL8 X
251	**OSDSLR** Reset Value	SL7 X	SL6 X	SL5 X	SL4 X	SL3 X	SL2 X	SL1 X	SL0 X
252	**OSDMR** Reset Value	- 0	- 0	- X	- X	- X	LSM2 0	LSM1 0	LSM0 0

7.5 IR PREPROCESSOR (IR)

7.5.1 Functional Description

The IR Preprocessor measures the interval between adjacent edges of the demodulated output signal from the IR amplifier/detector. You can specify the polarity using the POSED and NEGED bit in the IRSCR register The measurement is represented in terms of a count obtained with a 12.5KHz clock and stored in the IRPR register.

Whenever an edge of specified polarity is detected, the count accumulated since the previously detected edge is latched into an 8-bit register and an interrupt request IRQ is generated if the IRWDIS bit is reset in the IRSCR register.

Note: Any count less than 255 stored in the latch register is over-written in case the µP fails to execute the read before the next edge occurs.

In case an edge is not detected in about 20ms (the count reaches its maximum value of 255) the count is latched immediately and the IRQ flag is set. An overflow flag (not accessable) is also set internally.

Each time an interrupt is received, it must be acknowleged by writing any value in the IRPR register. Otherwise no further interrupts will be generated.

Warning: The content of the latch cannot be changed as long as the overflow flag remains set. To clear the IRQ and internal overflow flags, just write any value in the IRPR register. As long as the internal overflow flag is set, no interrupt is generated.

The IR input signal is preprocessed by a spike filter. The FLSEL bit of the IRSCR register determines the width of filtered pulse.

7.5.2 Register Description

IR PULSE REGISTER (IRPR)

R248 - Read only
Register Page: 43
Reset Value: 0000 0000 (00h)

7							0
IR7	IR6	IR5	IR4	IR3	IR2	IR1	IR0

Bits 7:0 = **IR[7:0]**: IR pulse width in terms of number of 12.5KHz clock cycles.

IR/SYNC CONTROL REGISTER (IRSCR)

R250 - Read/Write
Register Page: 43
Reset Value: 0000 0000 (00h)

7							0
0	0	-	IRWDIS	FLSEL	POSED	NEGED	-

Bits 7:6 = Reserved.
Forced by hardware to 0.

Bit 5 = Reserved.

Bit 4 = **IRWDIS**: *External Interrupt Source.*

This bit is set and cleared by software. It selects the source of the interrupt assigned to the external interrupt channel. Refer to the Interrupt Chapter.
0: The interrupt request from the IR preprocessor is forwarded to the CPU
1: The interrupt from the external interrupt pin is forwarded to the CPU

Bit 3 = **FLSEL**: *Spike filter pulse width selection*

This bit is set and cleared by software. It selects the spike filter width.
0: Filter pulses narrower than 2µs
1: Filter pulses narrower than 160µs

Bits 2:1 = **POSED**, **NEGED** *Edge selection for the duration measurement*

NEGED	POSED	Count latch at ...
1	1	Positive or negative transition of IR or when count reaches 255
1	0	Negative transition of IR or when count reaches 255
0	1	Positive transition of IR or when count reaches 255
0	0	Only when count reaches 255

Bit 0 = Reserved.

7.6 VOLTAGE SYNTHESIS TUNING CONVERTER (VS)

7.6.1 Description

The on-chip Voltage Synthesis (VS) converter allows the generation of a tuning reference voltage in a TV set application. The peripheral is composed of a 14-bit counter that allows the conversion of the digital content in a tuning voltage, available at the VS output pin, by using PWM (Pulse Width Modulation) and BRM (Bit Rate Modulation) techniques. The 14-bit counter gives 16384 steps which allow a resolution of approximately 2 mV over a tuning voltage of 32 V. This corresponds to a tuning resolution of about 40 KHz per step in UHF band (the actual value will depend on the characteristics of the tuner).

The tuning word consists of a 14-bit word contained in the registers VSDR1 (R254) and VSDR2 (R255) both located in page 59.

Coarse tuning (PWM) is performed using the seven most significant bits. Fine tuning (BRM) is performed using the the seven least significant bits. With all “0”s loaded, the output is 0. As the tuning voltage increases from all “0”s, the number of pulses in one period increases to 128 with all pulses being the same width. For values larger than 128, the PWM takes over and the number of pulses in one period remains constant at 128, but the width changes. At the other end of the scale, when almost all “1”s are loaded, the pulses will start to link together and the number of pulses will decrease.

When all “1”s are loaded, the output will be almost 100% high but will have a low pulse (1/16384 of the high pulse).

7.6.2 Output Waveforms

Included inside the VS are the register latches, a reference counter, PWM and BRM control circuitry. The clock for the 14-bit reference counter is derived from the main system clock (referred to as INTCLK) after a division by 4. For example, using an internal 12 MHz on-chip clock (see Timing & Clock Controller chapter) leads to a 3 MHz input for the VS counter.

From the point of view of the circuit, the seven most significant bits control the coarse tuning, while the seven least significant bits control the fine tuning. From the application and software point of view, the 14 bits can be considered as one binary number.

As already mentioned the coarse tuning consists of a PWM signal with 128 steps: we can consider the fine tuning to cover 128 coarse tuning cycles.

The VS Tuning Converter is implemented with 2 separate outputs (VSO1 and VSO2) that can drive 2 separate Alternate Function outputs of 2 standard I/O port bits. A control bit allows you to choose which output is activated (only one output can be activated at a time).

Note: On SDIP32, only VSO2 is available.

When a VS output is not selected because the VS is disabled or because the second output is selected, it stays at a logical “one” level, allowing you to use the corresponding I/O port bit either as a normal I/O port bit or for a possible second Alternate Function output.

A second control bit allows the VS function to be started (or stopped) by software.

VOLTAGE SYNTHESIS (Cont'd)

PWM Generation

The counter increments continuously, clocked at INTCLK divided by 4. Whenever the 7 least significant bits of the counter overflow, the VS output is set.

The state of the PWM counter is continuously compared to the value programmed in the 7 most significant bits of the tuning word. When a match occurs, the output is reset thus generating the PWM output signal on the VS pin.

This Pulse Width modulated signal must be filtered, using an external RC network placed as close as possible to the associated pin. This provides an analog voltage proportional to the average charge passed to the external capacitor. Thus for a higher mark/space ratio (High time much greater than Low time) the average output voltage is higher. The external components of the RC network should be selected for the filtering level required for control of the system variable.

Figure 59. Typical PWM Output Filter

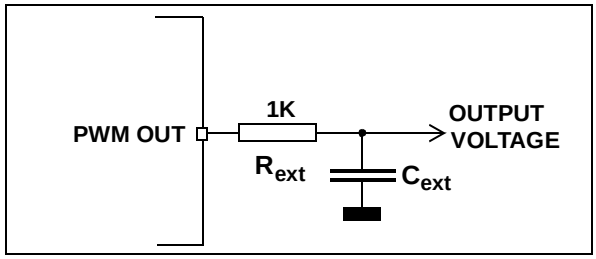
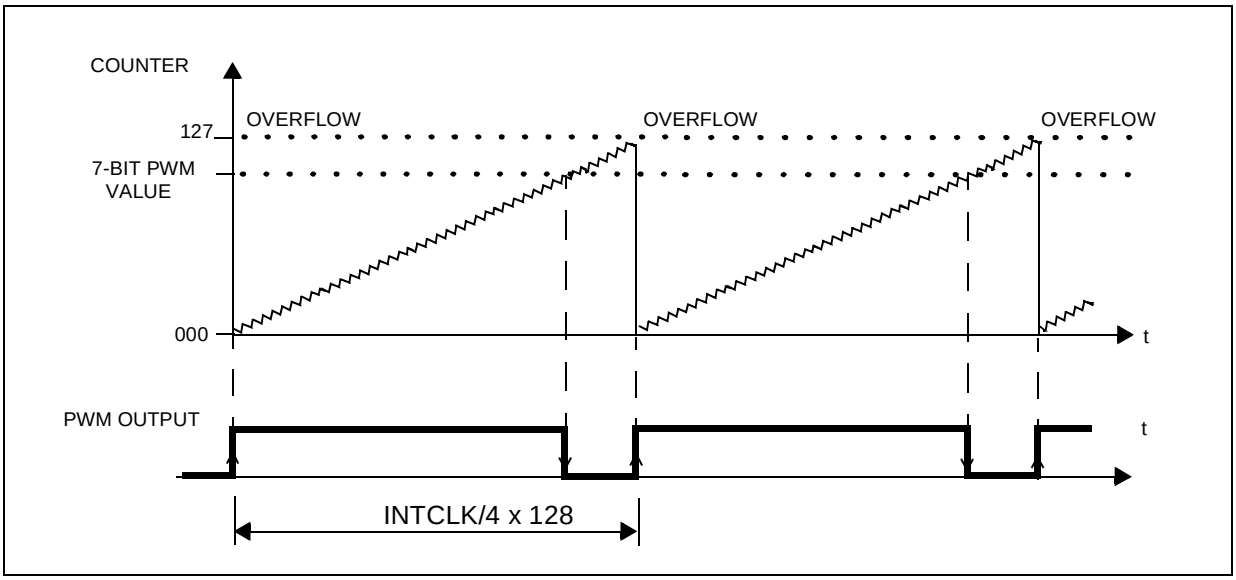
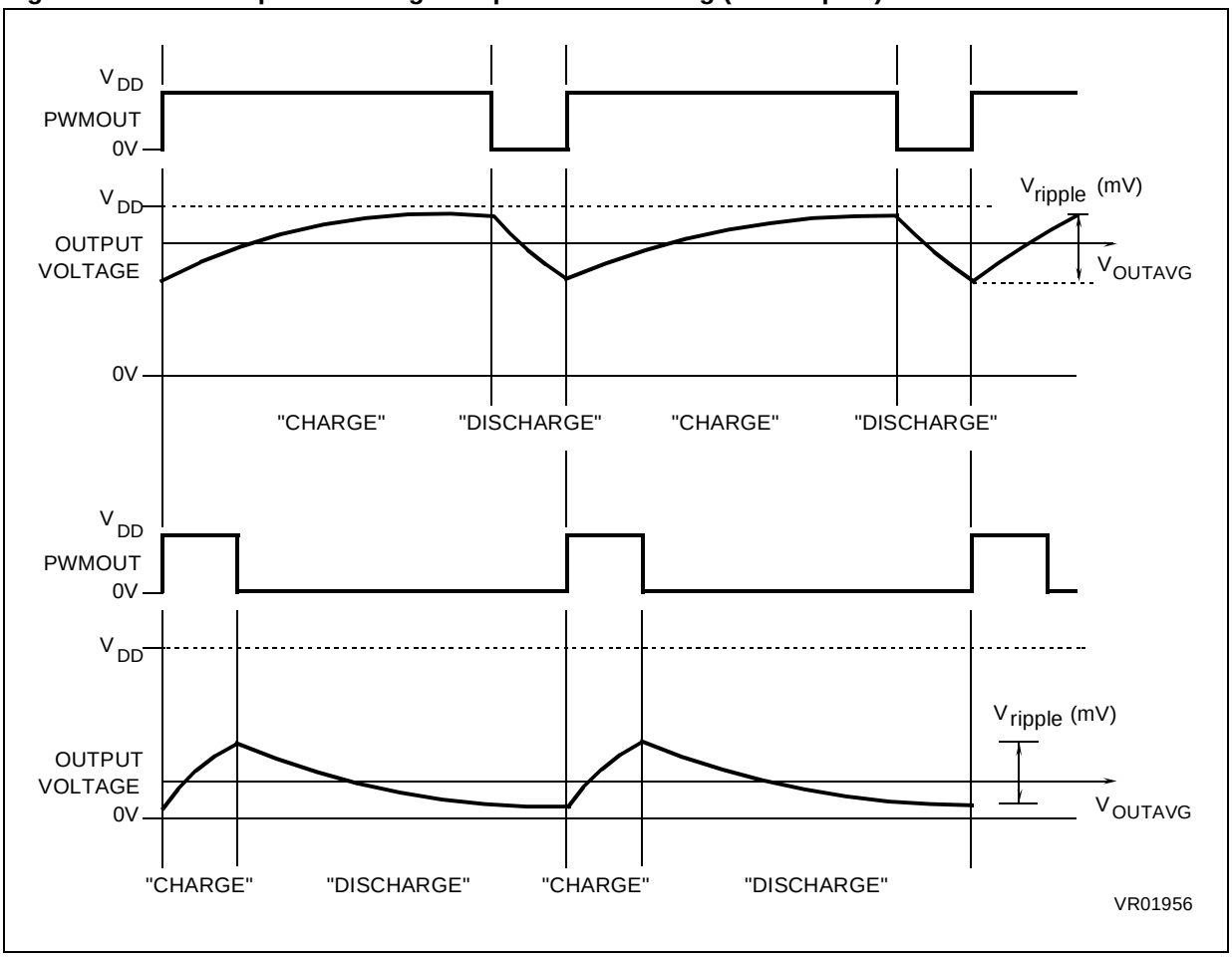


Figure 60. PWM Generation



VOLTAGE SYNTHESIS (Cont'd)

Figure 61. PWM Simplified Voltage Output After Filtering (2 examples)



VOLTAGE SYNTHESIS (Cont'd)

BRM Generation

The BRM bits allow the addition of a pulse to widen a standard PWM pulse for specific PWM cycles. This has the effect of "fine-tuning" the PWM Duty cycle (without modifying the base duty cycle), thus, with the external filtering, providing additional fine voltage steps.

The incremental pulses (with duration of T_{INTCLK}/4) are added to the beginning of the original PWM pulse and thus cause the PWM high time to be extended by this time with a corresponding reduction in the low time. The PWM intervals which are added to are specified in the lower 7 bits of the tuning word and are encoded as shown in the following table.

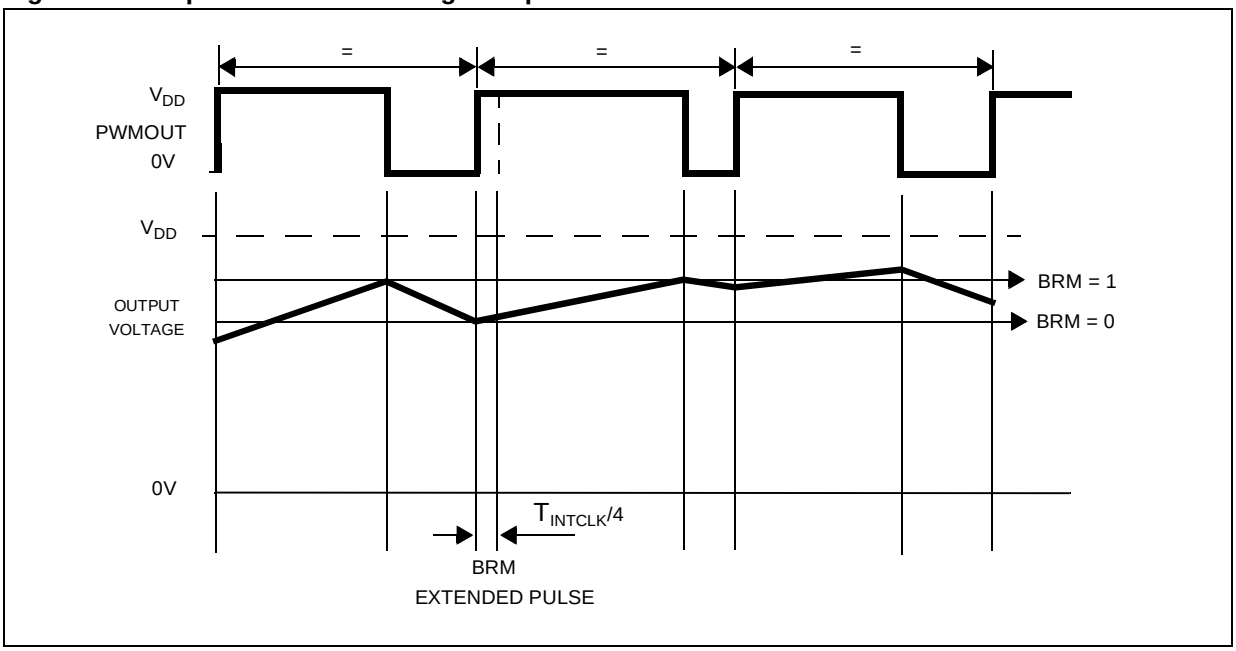
Table 27. 7-Bit BRM Pulse Addition Positions

Fine Tuning	No. of Pulses added at the following Cycles
0000001	64
0000010	32, 96
0000100	16, 48, 80, 112
0001000	8, 24,... 104, 120
0010000	4, 12,... 116, 124
0100000	2, 6,... 122, 126
1000000	1, 3,... 125, 127

The BRM values shown may be combined together to provide a summation of the incremental pulse intervals specified.

The pulse increment corresponds to the PWM resolution.

Figure 62. Simplified Filtered Voltage Output Schematic with BRM added



VOLTAGE SYNTHESIS (Cont'd)

7.6.3 Register Description

VS DATA AND CONTROL REGISTER 1 (VSDR1)
R254 - Read/Write
Register Page: 59
Reset Value: 0000 0000 (00h)

7	6	5	4	3	2	1	0
VSE	VSWP	VD13	VD12	VD11	VD10	VD9	VD8

Bit 7 = **VSE**: *VS enable bit.*
0: VS Tuning Converter disabled (i.e. the clock is not forwarded to the VS counter and the 2 outputs are set to 1 (idle state)
1: VS Tuning Converter enabled.

Bit 6 = **VSWP**: *VS Output Select*
This bit controls which VS output is enabled to output the VS signal.
0: VSO1 output selected
1: VSO2 output selected

Note: This bit must be set to 1 on SDIP32.

Bits 5:0 = **VD[13:8]** *Tuning word bits.*
These bits are the 6 most significant bits of the Tuning word forming the PWM selection. The VD13 bit is the MSB.

VS DATA AND CONTROL REGISTER 2 (VSDR2)
R255 - Read/Write
Register Page: 59
Reset Value: 0000 0000 (00h)

7							0
VD7	VD6	VD5	VD4	VD3	VD2	VD1	VD0

Bits 7:0 = **VD[7:0]** *Tuning word bits.*
These bits are the 8 least significant data bits of the VS Tuning word. All bits are accessible. Bits VD6 - VD0 form the BRM pulse selection. VD7 is the LSB of the 7 bits forming the PWM selection.

7.7 PWM GENERATOR

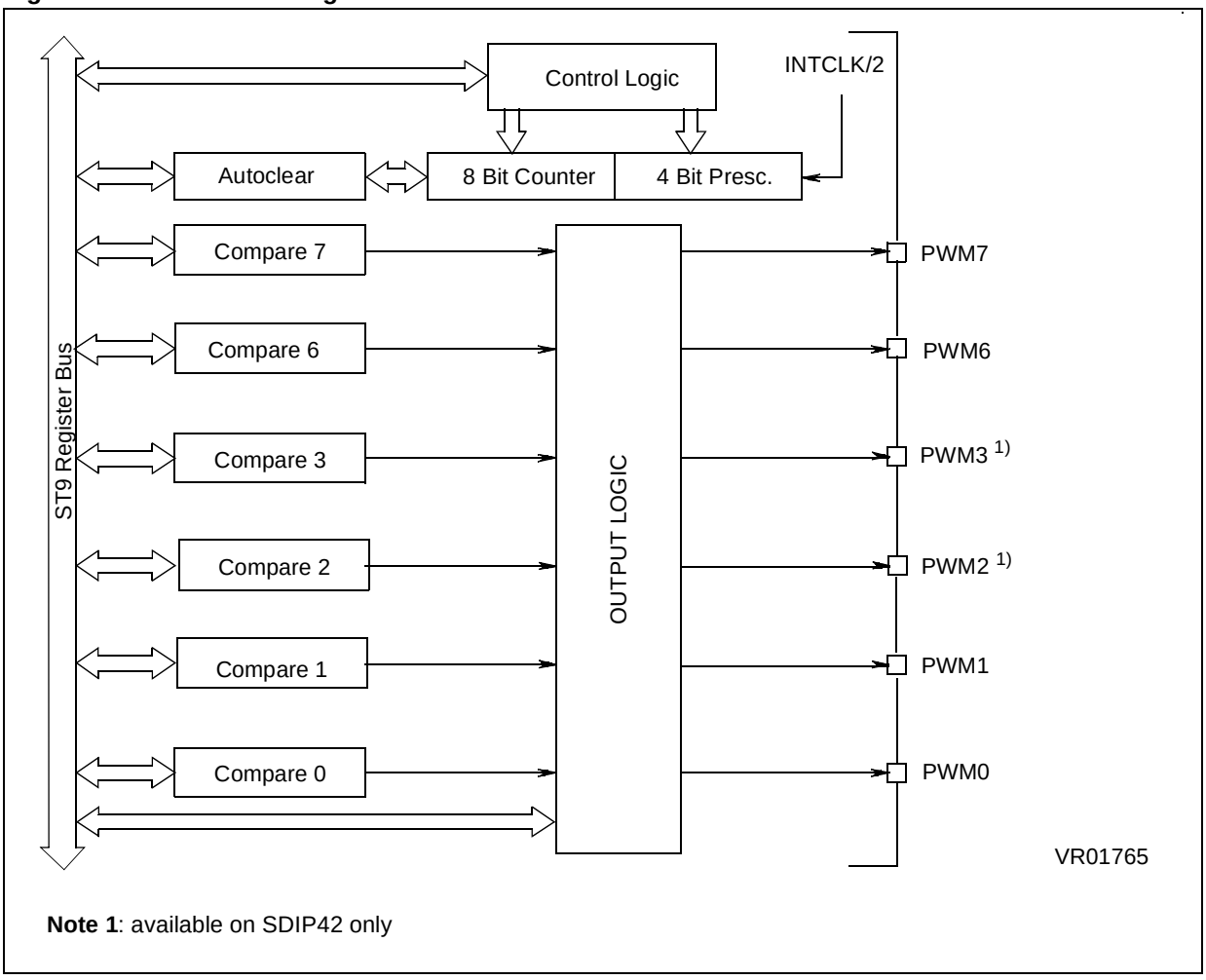
7.7.1 Introduction

The PWM (Pulse Width Modulated) signal generator allows the digital generation of up to 6 (SDIP42) or 4 (SDIP32) analog outputs when used with an external filtering network.

The unit is based around an 8-bit counter which is driven by a programmable 4-bit prescaler, with an input clock signal equal to the internal clock INTCLK divided by 2. For example, with a 12 MHz Internal clock, using the full 8-bit resolution, a frequency range from 1465 Hz up to 23437 Hz can be achieved.

Higher frequencies, with lower resolution, can be achieved by using the autoclear register. As an example, with a 12 MHz Internal clock, a maximum PWM repetition rate of 93750 Hz can be reached with 6-bit resolution.

Figure 63. PWM Block Diagram.



Note 1: available on SDIP42 only

PWM GENERATOR (Cont'd)

Up to 6 (SDIP42) or 4 (SDIP32) PWM outputs can be selected as Alternate Functions of an I/O port. Each output bit is independently controlled by a separate Compare Register. When the value programmed into the Compare Register and the counter value are equal, the corresponding output bit is set. The output bit is reset by a counter clear (by overflow or autoclear), generating the variable PWM signal.

Each output bit can also be complemented or disabled under software control.

7.7.2 Register Mapping

The registers of the PWM Generator are mapped in page 59.

Register Address	Register	Function
R240	CM0	Ch. 0 Compare Register
R241	CM1	Ch. 1 Compare Register
R242	CM2	Ch. 2 Compare Register 1)
R243	CM3	Ch. 3 Compare Register 1)
R244	Reserved	
R245	Reserved	
R246	CM6	Ch. 6 Compare Register
R247	CM7	Ch. 7 Compare Register
R248	ACR	Autoclear Register
R249	CRR	Counter Read Register
R250	PCTLR	Prescaler/ Reload Reg.
R251	OCPLR	Output Complement Reg.
R252	OER	Output Enable Register
R253- R255	—	Reserved

Note 1: available on SDIP42 only

Figure 64. PWM Action When Compare Register = 0 (no complement)

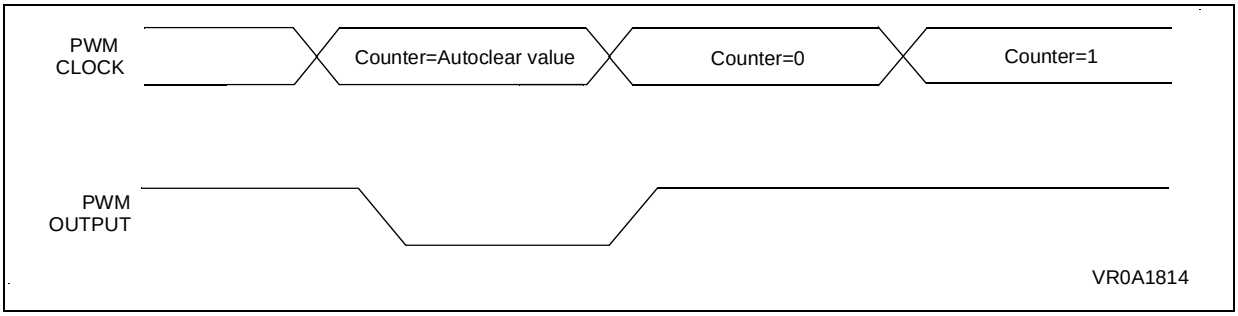
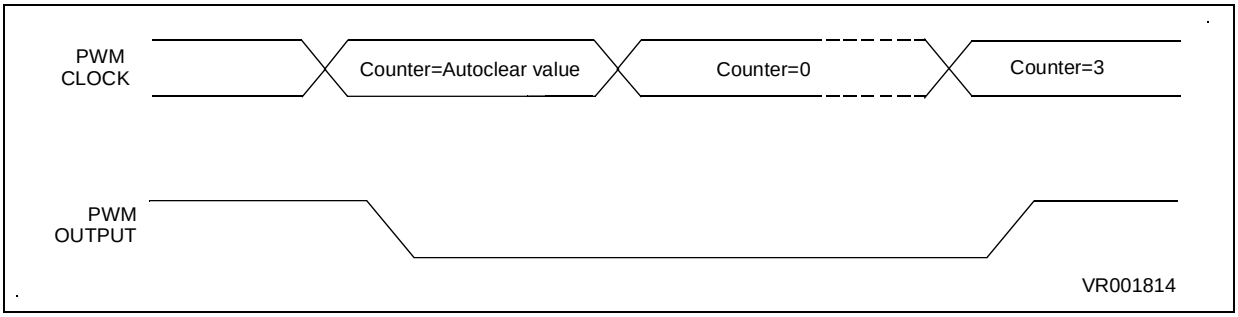


Figure 65. PWM Action When Compare Register = 3 (no complement)



PWM GENERATOR (Cont'd)

7.7.2.1 Register Description

COMPARE REGISTER 0 (CM0)
R240 - Read/Write
Register Page: 59
Reset Value: 0000 0000 (00h)

7							0
CM0.7	CM0.6	CM0.5	CM0.4	CM0.3	CM0.2	CM0.1	CM0.0

This is the compare register controlling PWM output 0. When the programmed content is equal to the counter content, a SET operation is performed on PWM output 0 (if the output has not been complemented or disabled).

Bits 7:0 = **CM0.[7:0]**: PWM Compare value Channel 0.

COMPARE REGISTER 1 (CM1)
R241 - Read/Write
Register Page: 59
Reset Value: 0000 0000 (00h)

7							0
CM1.7	CM1.6	CM1.5	CM1.4	CM1.3	CM1.2	CM1.1	CM1.0

This is the compare register controlling PWM output 1.

COMPARE REGISTER 2 (CM2) (SDIP42 only)
R242 - Read/Write
Register Page: 59
Reset Value: 0000 0000 (00h)

7							0
CM2.7	CM2.6	CM2.5	CM2.4	CM2.3	CM2.2	CM2.1	CM2.0

This is the compare register controlling PWM output 2.

COMPARE REGISTER 3 (CM3) (SDIP42 only)
R243 - Read/Write
Register Page: 59
Reset Value: 0000 0000 (00h)

7							0
CM3.7	CM3.6	CM3.5	CM3.4	CM3.3	CM3.2	CM3.1	CM3.0

This is the compare register controlling PWM output 3.

COMPARE REGISTER 6 (CM6)
R246 - Read/Write
Register Page: 59
Reset Value: 0000 0000 (00h)

7							0
CM6.7	CM6.6	CM6.5	CM6.4	CM6.3	CM6.2	CM6.1	CM6.0

This is the compare register controlling PWM output 6.

COMPARE REGISTER 7 (CM7)
R247 - Read/Write
Register Page: 59
Reset Value: 0000 0000 (00h)

7							0
CM7.7	CM7.6	CM7.5	CM7.4	CM7.3	CM7.2	CM7.1	CM7.0

This is the compare register controlling PWM output 7.

PWM GENERATOR (Cont'd)

AUTOCLEAR REGISTER (ACR)
R248 - Read/Write
Register Page: 59
Reset Value: 1111 1111 (FFh)

7							0
AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0

This register behaves exactly as a 9th compare Register, but its effect is to clear the CRR counter register, so causing the desired PWM repetition rate.

The reset condition generates the free running mode. So, FFh means count by 256.

Bits 7:0 = **AC[7:0]**: *Autoclear Count Value.*

When 00 is written to the Compare Register, if the ACR register = FFh, the PWM output bit is always set except for the last clock count (255 set and 1 reset; the converse when the output is complemented). If the ACR content is less than FFh, the PWM output bit is set for a number of clock counts equal to that content (see Figure 2).

Writing the Compare register constant equal to the ACR register value causes the output bit to be always reset (or set if complemented).

Example: If 03h is written to the Compare Register, the output bit is reset when the CRR counter reaches the ACR register value and set when it reaches the Compare register value (after 4 clock counts, see Figure 65). The action will be reversed if the output is complemented. The PWM mark/space ratio will remain constant until changed by software writing a new value in the ACR register.

COUNTER REGISTER (CRR)
R249 - Read Only
Register Page: 59
Reset Value: 0000 0000 (00h)

7							0
CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0

This read-only register returns the current counter value when read.

The 8 bit Counter is initialized to 00h at reset, and is a free running UP counter.

Bits 7:0 = **CR[7:0]**: *Current Counter Value.*

PRESCALER AND CONTROL REGISTER (PCTL)
R250 - Read/Write
Register Page: 59
Reset Value: 0000 1100 (0Ch)

7							0
PR3	PR2	PR1	PR0	1	1	CLR	CE

Bits 7:4 = **PR[3:0]** *PWM Prescaler value*.
These bits hold the Prescaler preset value. This is reloaded into the 4-bit prescaler whenever the prescaler (DOWN Counter) reaches the value 0, so determining the 8-bit Counter count frequency. The value 0 corresponds to the maximum counter frequency which is INTCLK/2. The value Fh corresponds to the maximum frequency divided by 16 (INTCLK/32).

The reset condition initializes the Prescaler to the Maximum Counter frequency.

PR[3:0]	Divider Factor	Frequency
0	1	INTCLK/2 (Max.)
1	2	INTCLK/4
2	3	INTCLK/6
..	..	..
Fh	16	INTCLK/32 (Min.)

Bits 3:2 = Reserved.
Forced by hardware to "1"

Bit 1 = **CLR**: *Counter Clear.*
This bit when set, allows both to clear the counter, and to reload the prescaler. The effect is also to clear the PWM output. It returns "0" if read.

Bit 0 = **CE**: *Counter Enable.*
This bit enables the counter and the prescaler when set to "1". It stops both when reset without affecting their current value, allowing the count to be suspended and then restarted by software "on fly".

PWM GENERATOR (Cont'd)

OUTPUT COMPLEMENT REGISTER (OCPL)
R251- Read/Write
Register Page 59
Reset Value: 0000 0000 (00h)

7							0
OCPL.7	OCPL.6	-	-	OCPL.3	OCPL.2	OCPL.1	OCPL.0

This register allows the PWM output level to be complemented on an individual bit basis.

In default mode (reset configuration), each comparison true between a Compare register and the counter has the effect of setting the corresponding output.

At counter clear (either by autoclear comparison true, software clear or overflow when in free running mode), all the outputs are cleared.

By setting each individual bit (OCPL.x) in this register, the logic value of the corresponding output will be inverted (i.e. reset on comparison true and set on counter clear).

Example: When set to "1", the OCPL.1 bit complements the PWM output 1.

Bit 7 = **OCPL.7**: *Complement PWM Output 7.*

Bit 6 = **OCPL.6**: *Complement PWM Output 6.*

Bits 5:4 = Reserved.
Must be kept in reset state.

Bit 3 = **OCPL.3**[1)]: *Complement PWM Output 3.*

Bit 2 = **OCPL.2**[1)]: *Complement PWM Output 2.*

Bit 1 = **OCPL.1**: *Complement PWM Output 1.*

Bit 0 = **OCPL.0**: *Complement PWM Output 0.*

Note 1: available on SDIP42 only.
On SDIP32, these bits are reserved and must be kept reset.

OUTPUT ENABLE REGISTER (OER)
R252 - Read/Write
Register Page: 59
Reset Value: 0000 0000 (00h)

7							0
OE.7	OE.6	-	-	OE.3	OE.2	OE.1	OE.0

These bits are set and cleared by software.
0: Force the corresponding PWM output to logic level 1. This allows the port pins to be used for normal I/O functions or other alternate functions (if available).
1: Enable the corresponding PWM output.

Example: Writing 03h into the OE Register will enable only PWM outputs 0 and 1, while outputs 2, 3, 4, 5, 6 and 7 will be forced to logic level "1".

Bit 7 = **OE.7**: *Output Enable PWM Output 7.*

Bit 6 = **OE.6**: *Output Enable PWM Output 6.*

Bits 5:4 = Reserved.
Must be kept in reset state.

Bit 3 = **OE.3**[1)]: *Output Enable PWM Output 3.*

Bit 2 = **OE.2**[1)]: *Output Enable PWM Output 2.*

Bit 1 = **OE.1**: *Output Enable PWM Output 1.*

Bit 0 = **OE.0**: *Output Enable PWM Output 0.*

Note 1: available on SDIP42 only.
On SDIP32, these bits are reserved and must be kept reset.

7.8 A/D CONVERTER (A/D)

7.8.1 Introduction

The 8-bit Analog to Digital Converter uses a fully differential analog configuration for the best noise immunity and precision performance. The analog voltage references of the converter are connected to the internal AV_{DD} & AV_{SS} analog supply pins of the chip if they are available, otherwise to the ordinary V_{DD} and V_{SS} supply pins of the chip. The guaranteed accuracy depends on the device (see Electrical Characteristics). A fast Sample/Hold allows quick signal sampling for minimum warping effect and conversion error.

7.8.2 Main Features

- 8-bit resolution A/D Converter
- Single Conversion Time (including Sampling Time):
  - 138 internal system clock periods in slow mode (~5.6 µs @25Mhz internal system clock);
  - 78 INTCLK periods in fast mode (~6.5 µs @ 12MHZ internal system clock)
- Sample/Hold: Tsample=
  - 84 INTCLK periods in slow mode (~3.4 µs @25Mhz internal system clock)
  - 48 INTCLK periods in fast mode (~4 µs @12Mhz internal system clock)
- Up to 5 (SDIP42) or 3 (SDIP32) Analog Inputs
- Single/Continuous Conversion Mode
- External source Trigger (Alternate synchronization)
- Power Down mode (Zero Power Consumption)
- 1 Control Logic Register
- 1 Data Register

7.8.3 General Description

Depending on device, up to 5 (SDIP42) or 3 (SDIP32) analog inputs can be selected by software.

Different conversion modes are provided: single, continuous, or triggered. The continuous mode performs a continuous conversion flow of the selected channel, while in the single mode the selected channel is converted once and then the logic waits for a new hardware or software restart.

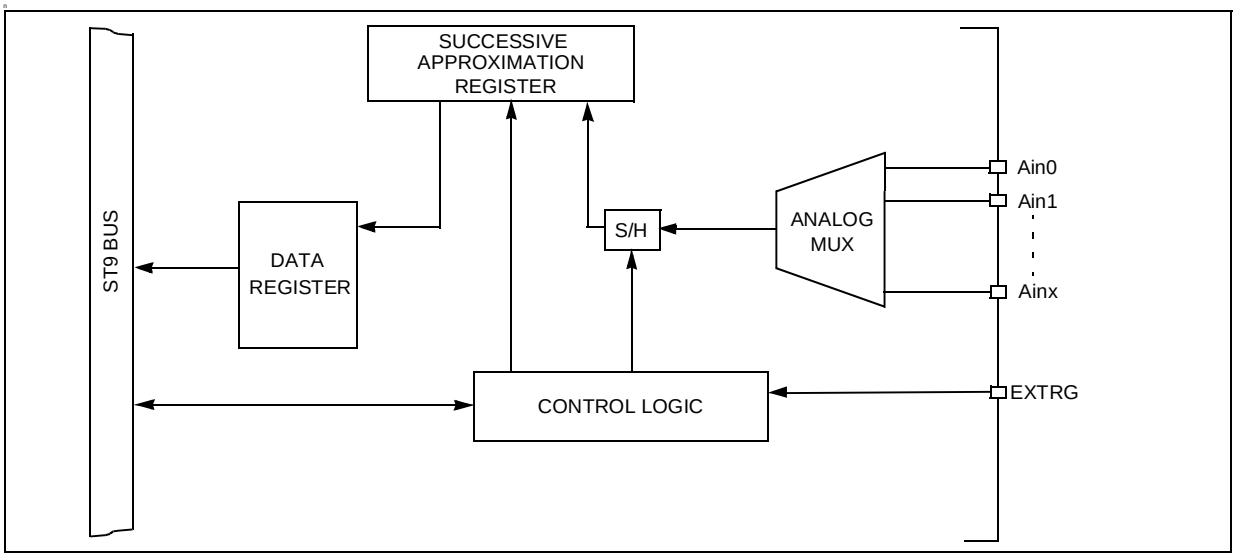
A data register (ADDTR) is available, mapped in page 62, allowing data storage (in single or continuous mode).

The start conversion event can be managed either:

– by software, writing the START/STOP bit of the Control Logic Register

– or by hardware using an external signal on the EXTRG triggered input (negative edge sensitive) connected as an Alternate Function to an I/O port bit.

Figure 66. A/D Converter Block Diagram



A/D CONVERTER (Cont'd)

The conversion technique used is successive approximation, with AC coupled analog fully differential comparators blocks plus a Sample and Hold logic and a reference generator.

The internal reference (DAC) is based on the use of a binary-ratioed capacitor array. This technique allows the specified monotonicity (using the same ratioed capacitors as sampling capacitor). A Power Down programmable bit sets the A/D converter analog section to a zero consumption idle status.

7.8.3.1 Operating Modes

The two main operating modes, single and continuous, can be selected by writing 0 (reset value) or 1 into the CONT bit of the Control Logic Register.

Single Mode

In single mode (CONT=0 in ADCLR) the STR bit is forced to '0' after the end of channel i-th conversion; then the A/D waits for a new start event. This mode is useful when a set of signals must be sampled at a fixed frequency imposed by a Timer unit or an external generator (through the alternate synchronization feature). A simple software routine monitoring the STR bit can be used to save the current value before a new conversion ends (so to create a signal samples table within the internal memory or the Register File). Furthermore, if the R242.0 bit (register AD-INT, bit 0) is set, at the end of conversion, a negative edge on the connected external interrupt channel (see Interrupts Chapter) is generated to allow the reading of the converted data by means of an interrupt routine.

Continuous Mode

In continuous mode (CONT=1 in ADCLR) a continuous conversion flow is entered by a start event on the selected channel until the STR bit is reset by software.

At the end of each conversion, the Data Register (ADCDR) content is updated with the last conversion result, while the former value is lost. When the conversion flow is stopped, an interrupt request is generated with the same modality previously described.

7.8.3.2 Alternate Synchronization

This feature is available in both single/continuous modes. The negative edge of external EXTRG signal can be used to synchronize the conversion start with a trigger pulse. This event can be enabled or masked by programming the TRG bit in the ADCLR Register.

The effect of alternate synchronization is to set the STR bit, which is cleared by hardware at the end of each conversion in single mode. In continuous mode any trigger pulse following the first one will be ignored. The synchronization source must provide a pulse (1.5 internal system clock, 125ns @ 12 MHz internal clock) of minimum width, and a period greater (in single mode) than the conversion time (~6.5us @ 12 MHz internal clock). If a trigger occurs when the STR bit is still '1' (conversions still in progress), it is ignored (see Electrical Characteristics).

Warning: If the EXTRG signal is already active when TRG bit is set, the conversion starts immediately.

7.8.3.3 Power-Up Operations

Before enabling any A/D operation mode, set the POW bit of the ADCLR Register at least 60 µs before the first conversion starts to enable the biasing circuits inside the analog section of the converter. Clearing the POW bit is useful when the A/D is not used so reducing the total chip power consumption. This state is also the reset configuration and it is forced by hardware when the core is in HALT state (after a HALT instruction execution).

A/D CONVERTER (Cont'd)

7.8.4 Register Description

A/D CONTROL LOGIC REGISTER (ADCLR)
R241 - Read/Write
Register Page: 62
Reset value: 0000 0000 (00h)

7							0
C2	C1	C0	FS	TRG	POW	CONT	STR

This 8-bit register manages the A/D logic operations. Any write operation to it will cause the current conversion to be aborted and the logic to be re-initialized to the starting configuration.

Bits 7:5 = **C[2:0]**: *Channel Address.*
These bits are set and cleared by software. They select channel `i` conversion as follows:

C2	C1	C0	Channel Enabled
0	0	0	Channel 0
0	0	1	Channel 1 [1)]
0	1	0	Channel 2 [1)]
0	1	1	Channel 3
1	0	0	Channel 4
1	0	1	Reserved
1	1	0	Reserved
1	1	1	Reserved

Note 1: Available on SDIP42 only

Bit 4 = **FS**: *Fast/Slow*.
This bit is set and cleared by software.
0: Fast mode. Single conversion time: 78 x INTCLK (5.75µs at INTCLK = 12 MHz)
1: Slow mode. Single conversion time: 138 x INTCLK (11.5µs at INTCLK = 12 MHz)

Note: Fast conversion mode is only allowed for internal speeds which do not exceed 12 MHz.

Bit 3 = **TRG**: *External Trigger Enable*.
This bit is set and cleared by software.
0: External Trigger disabled.
1: A negative (falling) edge on the EXTRG pin writes a "1" into the STR bit, enabling start of conversion.

Bit 2 = **POW**: *Power Enable*.
This bit is set and cleared by software.
0: Disables all power consuming logic.
1: Enables the A/D logic and analog circuitry.

Bit 1 = **CONT**: *Continuous/Single Mode Select*.
This bit it set and cleared by software.
0: Single mode: after the current conversion ends, the STR bit is reset by hardware and the converter logic is put in a wait status. To start another conversion, the STR bit has to be set by software or hardware.
1: Select Continuous Mode, a continuous flow of A/D conversions on the selected channel, starting when the STR bit is set.

Bit 0 = **STR**: *Start/Stop*.
This bit is set and cleared by software. It is also set by hardware when the A/D is synchronized with an external trigger.
0: Stop conversion on channel i. An interrupt is generated if the STR was previously set and the AD-INT bit is set.
1: Start conversion on channel i

Warning: When accessing this register, it is recommended to keep the related A/D interrupt channel masked or disabled to avoid spurious interrupt requests.

A/D CHANNEL `i` DATA REGISTER (ADDTR)
R240 - Read/Write
Register Page: 62
Reset value: undefined

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

The result of the conversion of the selected channel is stored in the 8-bit ADDTR, which is reloaded with a new value every time a conversion ends.

Bit 7:0 = **R[7:0]**: *Channel `i` conversion result.*

A/D CONVERTER (Cont'd)

A/D INTERRUPT REGISTER (ADINT)
Register Page: 62
R242 - Read/write
Reset value: 0000 0001 (01h)

7							0
-	-	-	-	-	-	-	AD-INT

Bits 7:1 = Reserved.

Bit 0 = **AD-INT**: *AD Converter Interrupt Enable*.

This bit is set and cleared by software. It allows the interrupt source to be switched between the A/D Converter and an external interrupt pin (See Interrupts chapter).
0: A/D Interrupt disabled. External pin selected as interrupt source.
1: A/D Interrupt enabled.

8 ELECTRICAL CHARACTERISTICS

The ST92186B device contains circuitry to protect the inputs against damage due to high static voltage or electric field. Nevertheless it is advised to take normal precautions and to avoid applying to this high impedance voltage circuit any voltage higher than the maximum rated voltages. It is recommended for proper operation that V_{IN} and V_{OUT} be constrained to the range

$V_{SS} \le (V_{IN}$ or $V_{OUT}) \le V_{DD}$

To enhance reliability of operation, it is recommended to connect unused inputs to an appropriate logic voltage level such as V_{SS} or V_{DD}.
All the voltages in the following table, are referenced to V_{SS}.

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
$V_{DD}1$	Supply Voltage	V_{SS} – 0.3 to V_{SS} + 7.0	V
V_{DDA}	Analog Supply Voltage	V_{SS} – 0.3 to V_{DD} +0.3	V
V_I	Input Voltage	V_{SS} – 0.3 to V_{DD} +0.3	V
V_I	Input Voltage	V_{SS} – 0.7 to V_{DD} +0.7 *	V
V_O	Output Voltage	V_{SS} – 0.3 to V_{DD} +0.3	V
V_O	Output Voltage	V_{SS} – 0.7 to V_{DD} +0.7 *	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

*Current is limited to |<200µA| into the pin

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.

RECOMMENDED OPERATING CONDITIONS

Symbol	Parameter	Value		Unit
		Min.	Max.	
T_A	Operating Temperature	-10	70	°C
V_{DD}	Operating Supply Voltage	4.5	5.5	V
V_{DDa}	Analog Supply Voltage	4.5	5.5	V
f_{OSCE}	External Oscillator Frequency		4.0	MHz
f_{INTCLK}	Internal Clock Frequency	0[1]	24	MHz

Note 1. 1MHz when A/D is used

DC ELECTRICAL CHARACTERISTICS

(V_{DD} = 5V ± 10% T_A = 0°C + 70°C, unless otherwise specified)

Symbol	Parameter	Test Conditions	Value		Unit
			Min.	Max.	
V_{IH}	Input High Level	TTL	2		V
		CMOS	0.7 V_{DD}		V
V_{IL}	Input Low Level	TTL		0.8	V
		CMOS		0.3 V_{DD}	V
V_{IHRS}	$\overline{\text{RESET}}$ Input High Level		0.8 V_{DD}		V
V_{ILRS}	$\overline{\text{RESET}}$ Input Low Level			0.3 V_{DD}	V
V_{HYRS}	$\overline{\text{RESET}}$ Input Hysteresis		0.5		V
V_{IH}	P4[1:0], P3[6:4], P3[1:0], P2[5:4] and P2.0 Input High Level		0.8 V_{DD}		V
V_{IL}	P4[1:0], P3[6:4], P3[1:0], P2[5:4] and P2.0 Input Low Level			0.3 V_{DD}	V
V_{IHY}	P4[1:0], P3[6:4], P3[1:0], P2[5:4] and P2.0 Input Hysteresis		0.5		V
V_{IHVH}	HSYNC/VSYNC Input High Level		0.8 V_{DD}		V
V_{ILVH}	HSYNC/VSYNC Input Low Level			0.3 V_{DD}	V
V_{HYHV}	HSYNC/VSYNC Input Hysteresis		1.2		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_{LKIO}	I/O Pin Input Leakage Current	Hi-Z Input, 0V < V_{IN} < V_{DD}		±1	µA
I_{LKRS}	$\overline{\text{RESET}}$ Pin Input Leakage Current	0V < V_{IN} < V_{DD}		±1	µA
$I_{LKA/D}$	A/D Pin Input Leakage Current	Alternate function open drain		±1	µA
I_{LKOS}	OSCIN Pin Input Leakage Current	0V < V_{IN} < V_{DD}		±1	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 ELECTRICAL CHARACTERISTICS

PIN CAPACITANCE

(V_{DD} = 5V ± 10%; T_A = -10°C + 70°C, unless otherwise specified)

Symbol	Parameter	Conditions	Value		Unit
			Typ.	Max	
C_{IO}	Pin Capacitance Digital Input/Output		8	10	pF

CURRENT CONSUMPTION

(V_{DD} = 5V ± 10%; T_A = -10°C + 70°C, unless otherwise specified)

Symbol	Parameter	Conditions	Value		Unit
			Typ.	Max	
I_{CC1}	Run Mode Current	(Notes 1, 2) INTCLK=16MHz	80	100	mA
I_{CC2}	Run Mode Current	(Notes 1, 2) INTCLK=24MHz	100	120	mA
I_{CC3}	Run Mode Current	(Notes 1, 2) INTCLK=4MHz	25	30	mA
I_{CC4}	Run Mode Current	(Notes 1, 5) INTCLK=16MHz	65	78	mA
I_{CCA}	Analog Current (V_{DDA} pin)	Freq. Multipliers , A/D, OSD & DACs Off.	1	10	µA
I_{ILPR}	Reset Mode Current	(Note 3)	10	100	µA
I_{HALT}	HALT Mode Current	(Note 4)	10	100	µA

Notes:

1. All ports are configured in push-pull output mode (output is high). VSYNC and HSYNC are tied to V_{SS}. The internal clock prescaler is in divide-by-1 mode. The external CLOCK pin (OSCIN) is driven by a square wave external clock at 4 MHz.

2. The CPU is fed by a frequency issued by the on-chip Frequency Multiplier. The Skew Corrector Frequency Multiplier provides a 28 MHz clock. All peripherals are working.

3. All ports are configured in push-pull output mode (output is high). VSYNC and HSYNC are tied to V_{SS}. External CLOCK pin (OSCIN) and Reset pins are held low. All peripherals are disabled.

4. All ports are configured in push-pull output mode (output is high). VSYNC and HSYNC are tied to V_{SS}. All peripherals are disabled.

5. The CPU is fed by a frequency issued by the on-chip Frequency Multiplier. The Skew Corrector Frequency Multiplier provides a 14Mhz clock. OSD, A/D, PWM, Std Timer and WDG Timer peripherals are running.

AC ELECTRICAL CHARACTERISTICS (Cont'd)

CLOCK TIMING

(V_{DD} = 5V ± 10% T_A = -10°C + 70°C, unless otherwise specified)

Symbol	Parameter	Conditions	Value		Unit
			Min	Max	
TpC	OSCIN Clock Period	intern. div. by 2	41.7		ns
		intern. div. by 1	83.3		ns
TrC	OSCIN rise time			12	ns
TfC	OSCIN fall time			12	ns
TwCL	OSCIN low width	intern. div. by 2	17		ns
		intern. div. by 1	38		ns
TwCH	OSCIN high width	intern. div. by 2	17		ns
		intern. div. by 1	38		ns

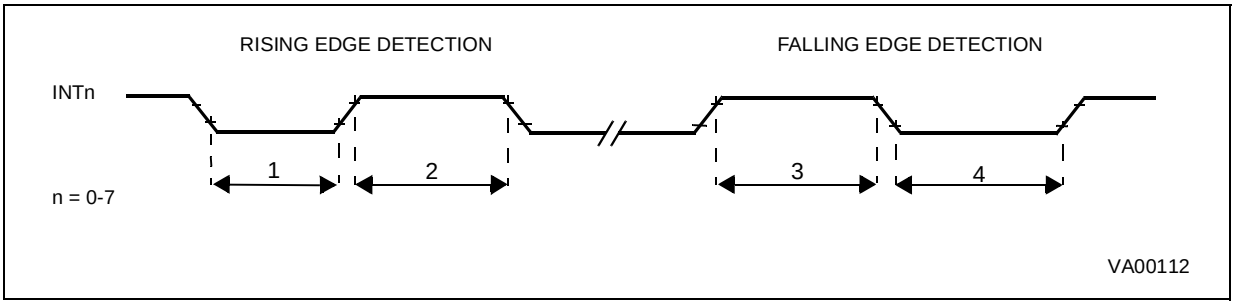
EXTERNAL INTERRUPT TIMING

(Rising or falling edge mode; V_{DD} = 5V ± 10%; T_A = -10°C + 70°C, unless otherwise specified)

N°	Symbol	Parameter	Conditions				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	TwLF	Low Level Minimum Pulse width in Falling Edge Mode	2TpC + 12	TpC + 12	95		ns
4	TwHF	High Level Minimum Pulse width in Falling Edge Mode	2TpC + 12	TpC + 12	95		ns

Note: The value in the left hand two columns shows the formula used to calculate the minimum or maximum timing from the oscillator clock period, prescale value and number of wait cycles inserted. The value in the rignt hand two columns shows the minimum and maximum for an external clock at 24 MHz divided by 2, prescale value of zero and zero wait status.

EXTERNAL INTERRUPT TIMING



AC ELECTRICAL CHARACTERISTICS (Cont'd)

SKEW CORRECTOR TIMING TABLE
(V_{DD} = 5V ± 10%; T_A = -10°C + 70°C, unless otherwise specified)

Symbol	Parameter	Conditions	Value Max	Unit
Tjskw	Jitter on RGB output	28 MHz Skew corrector clock frequency	<12*	ns

The OSD jitter is measured from leading edge to leading edge of a single character row on consecutive TV lines. The value is an envelope of 100 fields
*Max. value at all CPU operating frequencies

AC ELECTRICAL CHARACTERISTICS (Cont'd)

OSD DAC CHARACTERISTICS

(V_{DD} = 5V ± 10%; T_A = -10°C + 70°C, unless otherwise specified)

Symbol	Parameter	Conditions	Value			Unit
			Min	Typ	Max	
	Output impedance FB,R,G,B				100	Ohm
	Output voltage FB,R,G,B	Cload = 20 pF RL=100K				
	code = 111		0.976	1.170	1.364	V
	code = 110		0.863	1.034	1.205	
	code = 101		0.751	0.899	1.046	
	code = 100		0.638	0.763	0.887	
	code = 011		0.525	0.627	0.729	
	code = 010		0.412	0.491	0.570	
	code = 001		0.300	0.356	0.411	
	code = 000		0.157	0.220	0.252	
	FB = 1 FB = 0					V
	Relative voltage accuracy	(*)			±5	%
	R/G/B to FB 50% point matching	FB DAC mode (**)			5	ns
	R/G/B to FB 50% point matching	FB digital mode (***)			5	ns
	Pixel Frequency	Cload = 20 pF			20 (****)	MHz.
		Cload = 10 pF			40 (****)	MHz.

(*) Output voltage matching of the R,G and B levels on a single device for each of the 8 levels
(**) Phase matching (50% point on both rise & fall time) on R, G, B, FB lines (FB in DAC mode)
(***) Phase matching (50% point on both rise & fall time) on R, G, B, FB lines (FB in digital mode)
(****) 95% of the signal amplitude is reached within the specified clock period

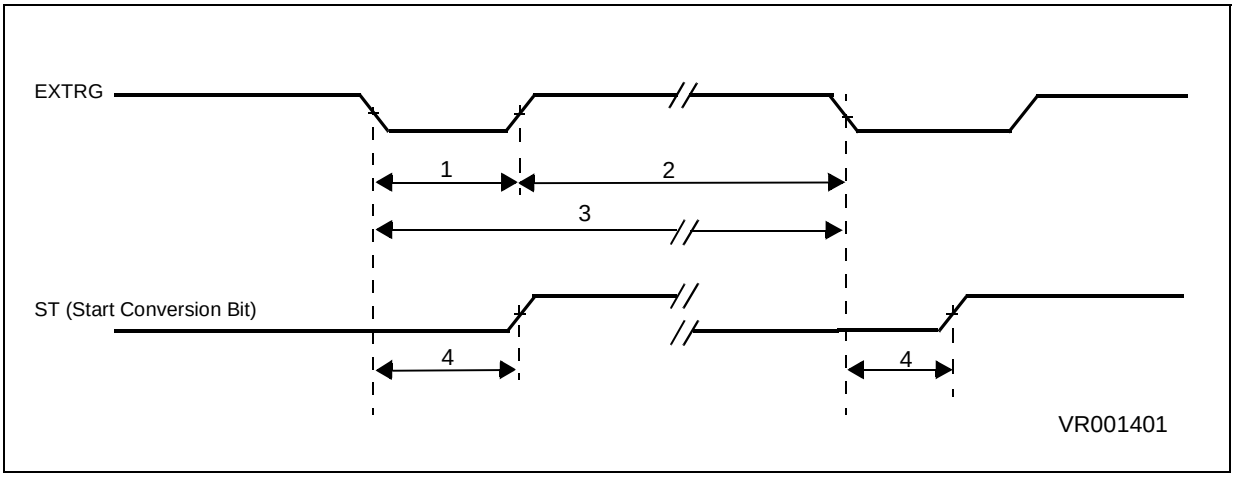
AC ELECTRICAL CHARACTERISTICS (Cont'd)

A/D CONVERTER, EXTERNAL TRIGGER TIMING TABLE

(V_{DD}= 5V +/-10%; T_A= 0 to 70°C, unless otherwise specified)

N°	Symbol	Parameter	Conditions	Value		Unit
				min	max	
1	T_{low}	Pulse Width		1.5		INTCLK
2	T_{high}	Pulse Distance		1.5		INTCLK
3	T_{ext}	Period/fast Mode		79		INTCLK
4	T_{str}	Start Conversion Delay		0.5	1.5	INTCLK

A/D CONVERTER, EXTERNAL TRIGGER TIMING TABLE



A/D CONVERTER. ANALOG PARAMETERS TABLE

(V_{DD}= 5V +/-10% ; T_A= 0 to 70°C, unless otherwise specified))

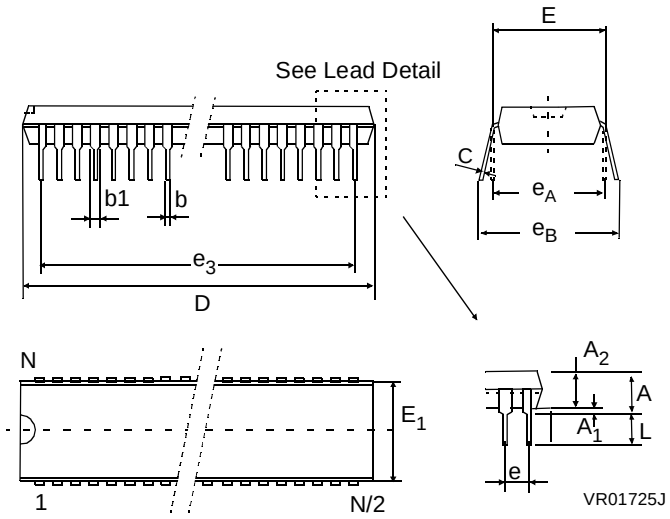
Parameter	Value			Unit	Note
	typ (*)	min	max	(**)	
Analog Input Range		V_{SS}	V_{DD}	V	
Conversion Time		138		INTCLK	(1,2)
Sample Time		87.51		INTCLK	(1)
Power-up Time		60		µs	
Resolution	8			bits	
Differential Non Linearity	0.5	0.3	1.5	LSBs	(4)
Integral Non Linearity			2	LSBs	(4)
Absolute Accuracy			2	LSBs	(4)
Input Resistance			1.5	Kohm	(3)
Hold Capacitance			1.92	pF	

Notes:
(*) The values are expected at 25 Celsius degrees with V_{DD}= 5V
(**)'LSBs' , as used here, as a value of V_{DD}/256
(1) @ 24 MHz external clock
(2) including Sample time
(3) it must be considered as the on-chip series resistance before the sampling capacitor
(4) DNL ERROR= max {[V(i) -V(i-1)] / LSB-1}INL ERROR= max {[V(i) -V(0)] / LSB-i}
ABSOLUTE ACCURACY= overall max conversion error

9 GENERAL INFORMATION

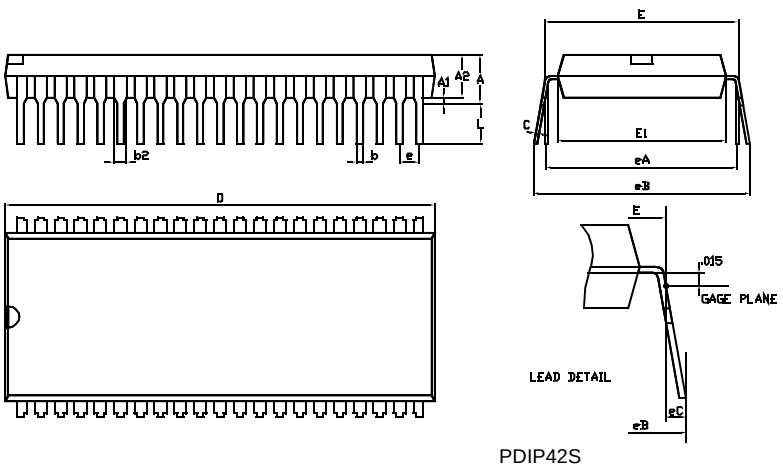
9.1 PACKAGE MECHANICAL DATA

Figure 67. 32-Pin Shrink Plastic Dual In Line Package



Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A	3.56	3.76	5.08	0.140	0.148	0.200
A1	0.51			0.020		
A2	3.05	3.56	4.57	0.120	0.140	0.180
b	0.36	0.46	0.58	0.014	0.018	0.023
b1	0.76	1.02	1.40	0.030	0.040	0.055
C	0.20	0.25	0.36	0.008	0.010	0.014
D	27.43	27.94	28.45	1.080	1.100	1.120
E	9.91	10.41	11.05	0.390	0.410	0.435
E1	7.62	8.89	9.40	0.300	0.350	0.370
e		1.78			0.070	
eA		10.16			0.400	
eB			12.70			0.500
L	2.54	3.05	3.81	0.100	0.120	0.150
	Number of Pins					
N	32					

Figure 68. 42-Pin Shrink Plastic Dual In-Line Package



PDIP42S

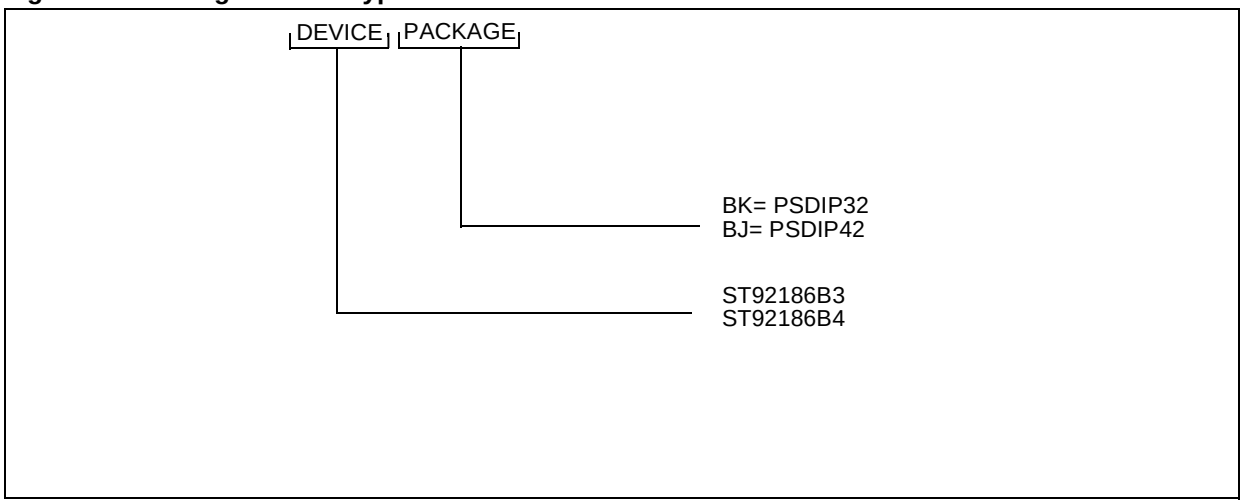
Dim.	mm			inches		
	Min	Typ	Max	Min	Typ	Max
A			5.08			0.200
A1	0.51			0.020		
A2	3.05	3.81	4.57	0.120	0.150	0.180
b		0.46	0.56		0.018	0.022
b2		1.02	1.14		0.040	0.045
C	0.23	0.25	0.38	0.009	0.010	0.015
D	36.58	36.83	37.08	1.440	1.450	1.460
E	15.24		16.00	0.600		0.630
E1	12.70	13.72	14.48	0.500	0.540	0.570
e		1.78			0.070	
eA		15.24			0.600	
eB			18.54			0.730
eC	0.00		1.52	0.000		0.060
L	2.54	3.30	3.56	0.100	0.130	0.140
	Number of Pins					
N	42					

9.2 ORDERING INFORMATION

Device	ROM (Kbytes)	RAM (bytes)
ST92186Bx3	24	640
ST92186Bx4	32	

Note: The EPROM and OTP versions are supported by the ST92196A family.

Figure 69. Package Device Types



ST92186B OPTION LIST

Customer: . . . . . .
Address: . . . . . .
. . . . . .
Contact: . . . . . .
Phone No: . . . . . .
Reference/ROM Code* : . . . . . .
*The ROM code name is assigned by STMicroelectronics.

Please confirm characteristics of device:

Device:	[ ] ST92186B3 24K ROM	[ ] ST92186B4 32K ROM
Package:	[ ] PSDIP32 [ ] PSDIP42	
Temperature Range:	-10°C to 70 °C	
OSD Code:	[ ] OSD filename _ _ _ _ _ _ _ _ _.OSD	
Special Marking:	[ ] No	[ ] Yes "_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _"

For marking, one line is possible with maximum 15 characters (SDIP32) or 16 characters (SDIP42).
Authorized characters are letters, digits, '.', '-', '/' and spaces only.

Quantity forecast: [_ _ _ _ _ _ _] K units per year for a period of [_ _] years.
Preferred production start date: [_ _ _ _ _ _ _] (YYYY/MM/DD)

Date . . . . . .

Customer Signature . . . . . .

10 SUMMARY OF CHANGES

Rev.	Main Changes	Date
2.2	Modification of VIH levels in DC Electrical Characteristics table on page 139	9 March 2001