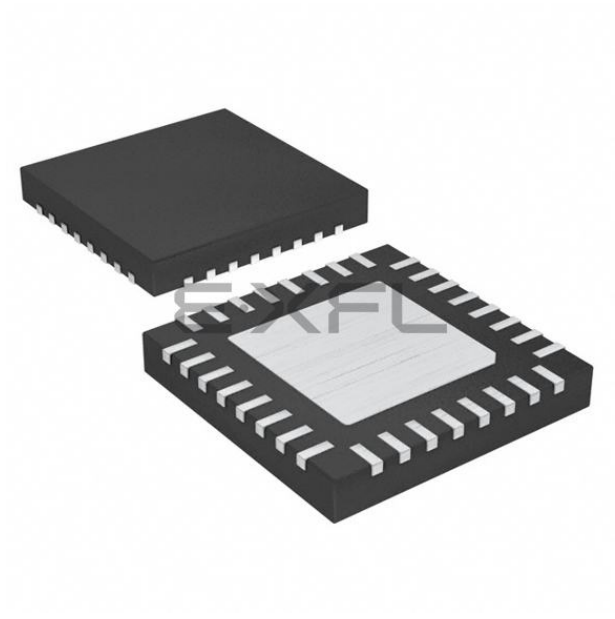


Welcome to E-XFL.COM

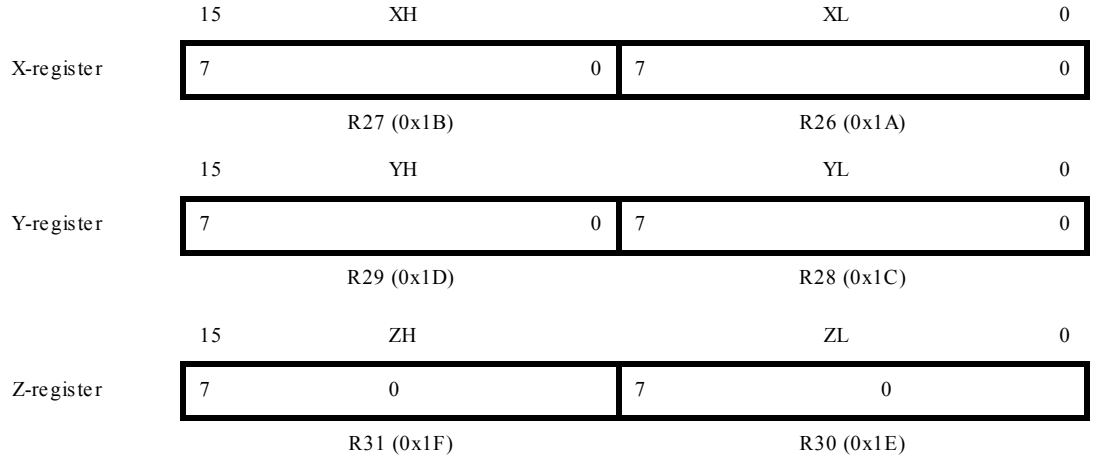
What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	AVR
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	27
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	32-VFQFN Exposed Pad
Supplier Device Package	32-VFQFN (5x5)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/atmega168pb-mu

Figure 12-3. The X-, Y-, and Z-registers



In the different addressing modes these address registers have functions as fixed displacement, automatic increment, and automatic decrement. See *Instruction Set Summary* for details.

12.5. Stack Pointer

The Stack is mainly used for storing temporary data, for storing local variables and for storing return addresses after interrupts and subroutine calls. The Stack is implemented as growing from higher to lower memory locations. The Stack Pointer Register always points to the top of the Stack.

The Stack Pointer points to the data SRAM Stack area where the Subroutine and Interrupt Stacks are located. A Stack PUSH command will decrease the Stack Pointer. The Stack in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. Initial Stack Pointer value equals the last address of the internal SRAM and the Stack Pointer must be set to point above start of the SRAM. See the table for Stack Pointer details.

Table 12-1. Stack Pointer Instructions

Instruction	Stack pointer	Description
PUSH	Decremented by 1	Data is pushed onto the stack
ICALL RCALL	Decremented by 2	Return address is pushed onto the stack with a subroutine call or interrupt
POP	Incremented by 1	Data is popped from the stack
RET RETI	Incremented by 2	Return address is popped from the stack with return from subroutine or return from interrupt

The AVR Stack Pointer is implemented as two 8-bit registers in the I/O space. The number of bits actually used is implementation dependent. Note that the data space in some implementations of the AVR architecture is so small that only SPL is needed. In this case, the SPH Register will not be present.

12.5.2. Stack Pointer Low Register

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Reset value of SPL is RAMEND.

Name: SPL
Offset: 0x5D
Reset: 0xXX
Property: When addressing as I/O Register: address offset is 0x3D

Bit	7	6	5	4	3	2	1	0
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bit 7 – SP7: Stack Pointer Address 7

Bit 6 – SP6: Stack Pointer Address 6

Bit 5 – SP5: Stack Pointer Address 5

Bit 4 – SP4: Stack Pointer Address 4

Bit 3 – SP3: Stack Pointer Address 3

Bit 2 – SP2: Stack Pointer Address 2

Bit 1 – SP1: Stack Pointer Address 1

Bit 0 – SP0: Stack Pointer Address 0

12.6. Instruction Execution Timing

This section describes the general access timing concepts for instruction execution. The AVR CPU is driven by the CPU clock clk_{CPU}, directly generated from the selected clock source for the chip. No internal clock division is used. The Figure below shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access Register File concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

Flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared. If an interrupt condition occurs while the corresponding interrupt enable bit is cleared, the Interrupt Flag will be set and remembered until the interrupt is enabled, or the flag is cleared by software. Similarly, if one or more interrupt conditions occur while the Global Interrupt Enable bit is cleared, the corresponding Interrupt Flag(s) will be set and remembered until the Global Interrupt Enable bit is set, and will then be executed by order of priority.

The second type of interrupts will trigger as long as the interrupt condition is present. These interrupts do not necessarily have Interrupt Flags. If the interrupt condition disappears before the interrupt is enabled, the interrupt will not be triggered. When the AVR exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

The Status Register is not automatically stored when entering an interrupt routine, nor restored when returning from an interrupt routine. This must be handled by software.

When using the CLI instruction to disable interrupts, the interrupts will be immediately disabled. No interrupt will be executed after the CLI instruction, even if it occurs simultaneously with the CLI instruction. The following example shows how this can be used to avoid interrupts during the timed EEPROM write sequence.

Assembly Code Example

```
in r16, SREG ; store SREG value
cli ; disable interrupts during timed sequence
sbi EECR, EEMPE ; start EEPROM write
sbi EECR, EEPE
out SREG, r16 ; restore SREG value (I-bit)
```

C Code Example

```
char cSREG;
cSREG = SREG; /* store SREG value */
/* disable interrupts during timed sequence */
_CLI();
EECR |= (1<<EEMPE); /* start EEPROM write */
EECR |= (1<<EEPE);
SREG = cSREG; /* restore SREG value (I-bit) */
```

When using the SEI instruction to enable interrupts, the instruction following SEI will be executed before any pending interrupts, as shown in this example.

Assembly Code Example

```
sei ; set Global Interrupt Enable
sleep ; enter sleep, waiting for interrupt
; note: will enter sleep before any pending interrupt(s)
```

C Code Example

```
__enable_interrupt(); /* set Global Interrupt Enable */
__sleep(); /* enter sleep, waiting for interrupt */
/* note: will enter sleep before any pending interrupt(s) */
```

Related Links

Interrupts on page 80

Memory Programming on page 374

Boot Loader Support – Read-While-Write Self-Programming on page 356

- Brown-out Reset
- 2-wire Serial Interface address match
- Timer/Counter2 interrupt
- SPM/EEPROM ready interrupt
- External level interrupt on INT0 or INT1
- Pin change interrupt

Note: 1. Timer/Counter2 will only keep running in asynchronous mode.

Related Links

8-bit Timer/Counter2 with PWM and Asynchronous Operation on page 205

15.5. Power-Down Mode

When the SM[2:0] bits are written to '010', the SLEEP instruction makes the MCU enter Power-Down mode. In this mode, the external Oscillator is stopped, while the external interrupts, the 2-wire Serial Interface address watch, and the Watchdog continue operating (if enabled).

Only one of these events can wake up the MCU:

- External Reset
- Watchdog System Reset
- Watchdog Interrupt
- Brown-out Reset
- 2-wire Serial Interface address match
- External level interrupt on INT0 or INT1
- Pin change interrupt

This sleep mode basically halts all generated clocks, allowing operation of asynchronous modules only.

Note: If a level triggered interrupt is used for wake-up from Power-Down, the required level must be held long enough for the MCU to complete the wake-up to trigger the level interrupt. If the level disappears before the end of the Start-up Time, the MCU will still wake up, but no interrupt will be generated. The start-up time is defined by the SUT and CKSEL Fuses.

When waking up from Power-Down mode, there is a delay from the wake-up condition occurs until the wake-up becomes effective. This allows the clock to restart and become stable after having been stopped. The wake-up period is defined by the same CKSEL Fuses that define the Reset Time-out period.

Related Links

Clock Sources on page 49
External Interrupts on page 95
System Clock and Clock Options on page 48

15.6. Power-save Mode

When the SM[2:0] bits are written to 011, the SLEEP instruction makes the MCU enter Power-save mode. This mode is identical to Power-down, with one exception:

If Timer/Counter2 is enabled, it will keep running during sleep. The device can wake up from either Timer Overflow or Output Compare event from Timer/Counter2 if the corresponding Timer/Counter2 interrupt enable bits are set in TIMSK2, and the Global Interrupt Enable bit in SREG is set.

If Timer/Counter2 is not running, Power-down mode is recommended instead of Power-save mode.

The Timer/Counter2 can be clocked both synchronously and asynchronously in Power-save mode. If Timer/Counter2 is not using the asynchronous clock, the Timer/Counter Oscillator is stopped during sleep. If Timer/Counter2 is not using the synchronous clock, the clock source is stopped during sleep. Even if the synchronous clock is running in Power-save, this clock is only available for Timer/Counter2.

15.7. Standby Mode

When the SM[2:0] bits are written to '110' and an external crystal/resonator clock option is selected, the SLEEP instruction makes the MCU enter Standby mode. This mode is identical to Power-Down with the exception that the Oscillator is kept running. From Standby mode, the device wakes up in six clock cycles.

15.8. Extended Standby Mode

When the SM[2:0] bits are written to '111' and an external crystal/resonator clock option is selected, the SLEEP instruction makes the MCU enter Extended Standby mode. This mode is identical to Power-Save mode with the exception that the Oscillator is kept running. From Extended Standby mode, the device wakes up in six clock cycles.

15.9. Power Reduction Register

The Power Reduction Register (PRR) provides a method to stop the clock to individual peripherals to reduce power consumption. The current state of the peripheral is frozen and the I/O registers can not be read or written. Resources used by the peripheral when stopping the clock will remain occupied, hence the peripheral should in most cases be disabled before stopping the clock. Waking up a module, which is done by clearing the bit in PRR, puts the module in the same state as before shutdown.

Module shutdown can be used in Idle mode and Active mode to significantly reduce the overall power consumption. In all other sleep modes, the clock is already stopped.

Related Links

PRR on page 69

15.10. Minimizing Power Consumption

There are several possibilities to consider when trying to minimize the power consumption in an AVR controlled system. In general, sleep modes should be used as much as possible, and the sleep mode should be selected so that as few as possible of the device's functions are operating. All functions not needed should be disabled. In particular, the following modules may need special consideration when trying to achieve the lowest possible power consumption.

15.10.1. Analog to Digital Converter

If enabled, the ADC will be enabled in all sleep modes. To save power, the ADC should be disabled before entering any sleep mode. When the ADC is turned off and on again, the next conversion will be an extended conversion.

Related Links

Analog-to-Digital Converter on page 322

Address	Labels		Code Comments
0x016	rjmp	EE_RDY	; EEPROM Ready Handler
0x017	rjmp	ANA_COMP	; Analog Comparator Handler
0x018	rjmp	TWI	; 2-wire Serial Interface Handler
0x019	rjmp	SPM_RDY	; Store Program Memory Ready Handler
;			
0x01A	RESET: ldi	r16, high(RAMEND)	; Main program start
0x01B	out	SPH,r16	; Set Stack Pointer to top of RAM
0x01C	ldi	r16, low(RAMEND)	
0x01D	out	SPL,r16	
0x01E	sei		; Enable interrupts
0x01F	<instr>	xxx	

When the BOOTRST Fuse is unprogrammed, the Boot section size set to 2Kbytes and the IVSEL bit in the MCUCR Register (MCUCR.IVSEL) is set before any interrupts are enabled, the most typical and general program setup for the Reset and Interrupt Vector Addresses in ATmega88PB is:

Address	Labels		Code Comments
0x000	RESET: ldi	r16,high(RAMEND)	; Main program start
0x001	out	SPH,r16	; Set Stack Pointer to top of RAM
0x002	ldi	r16,low(RAMEND)	
0x003	out	SPL,r16	
0x004	sei		; Enable interrupts
0x005	<instr>	xxx	
;			
.org	0xC01		
0xC01	rjmp	EXT_INT0	; IRQ0 Handler
0xC02	rjmp	EXT_INT1	; IRQ1 Handler
...	...		...
;			
0xC19	rjmp	SPM_RDY	; Store Program Memory Ready Handler

Boot Loader Support – Read-While-Write Self-Programming on page 356
ATmega168PB Boot Loader Parameters on page 370

17.4. Register Description

17.4.1. Moving Interrupts Between Application and Boot Space

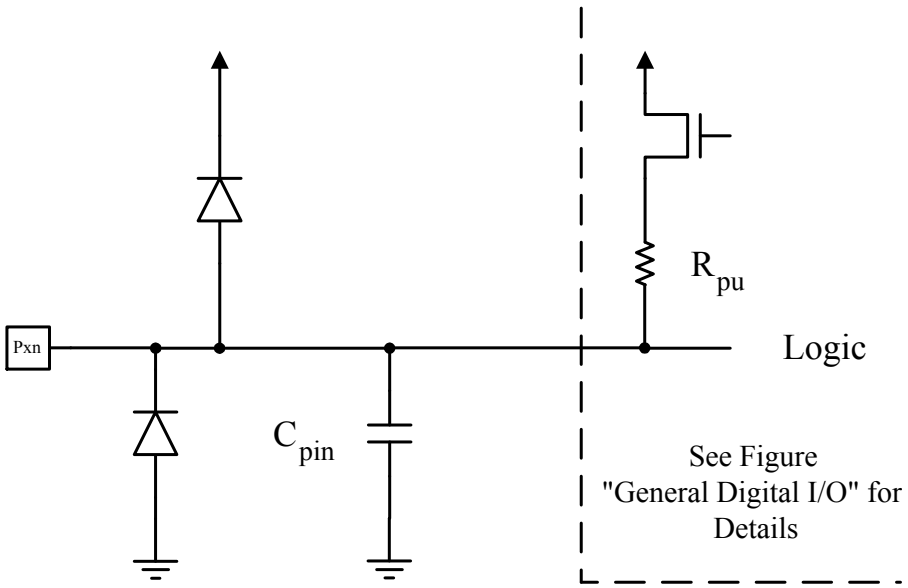
The MCU Control Register controls the placement of the Interrupt Vector table.

19. I/O-Ports

19.1. Overview

All AVR ports have true Read-Modify-Write functionality when used as general digital I/O ports. This means that the direction of one port pin can be changed without unintentionally changing the direction of any other pin with the SBI and CBI instructions. The same applies when changing drive value (if configured as output) or enabling/disabling of pull-up resistors (if configured as input). Each output buffer has symmetrical drive characteristics with both high sink and source capability. The pin driver is strong enough to drive LED displays directly. All port pins have individually selectable pull-up resistors with a supply-voltage invariant resistance. All I/O pins have protection diodes to both V_{CC} and Ground as indicated in the following figure.

Figure 19-1. I/O Pin Equivalent Schematic



All registers and bit references in this section are written in general form. A lower case “x” represents the numbering letter for the port, and a lower case “n” represents the bit number. However, when using the register or bit defines in a program, the precise form must be used. For example, PORTB3 for bit no. 3 in Port B, here documented generally as PORTxn.

Three I/O memory address locations are allocated for each port, one each for the Data Register – PORTx, Data Direction Register – DDRx, and the Port Input Pins – PINx. The Port Input Pins I/O location is read only, while the Data Register and the Data Direction Register are read/write. However, writing '1' to a bit in the PINx Register will result in a toggle in the corresponding bit in the Data Register. In addition, the Pull-up Disable – PUD bit in MCUCR disables the pull-up function for all pins in all ports when set.

Using the I/O port as General Digital I/O is described in next section. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in *Alternate Port Functions* section in this chapter. Refer to the individual module sections for a full description of the alternate functions.

Enabling the alternate function of some of the port pins does not affect the use of the other pins in the port as general digital I/O.

19.2.2. Toggling the Pin

Writing a '1' to PINxn toggles the value of PORTxn, independent on the value of DDRxn. The SBI instruction can be used to toggle one single bit in a port.

19.2.3. Switching Between Input and Output

When switching between tri-state ({DDxn, PORTxn} = 0b00) and output high ({DDxn, PORTxn} = 0b11), an intermediate state with either pull-up enabled {DDxn, PORTxn} = 0b01) or output low ({DDxn, PORTxn} = 0b10) must occur. Normally, the pull-up enabled state is fully acceptable, as a high-impedance environment will not notice the difference between a strong high driver and a pull-up. If this is not the case, the PUD bit in the MCUCR Register can be set to disable all pull-ups in all ports.

Switching between input with pull-up and output low generates the same problem. The user must use either the tri-state ({DDxn, PORTxn} = 0b00) or the output high state ({DDxn, PORTxn} = 0b11) as an intermediate step.

The following table summarizes the control signals for the pin value.

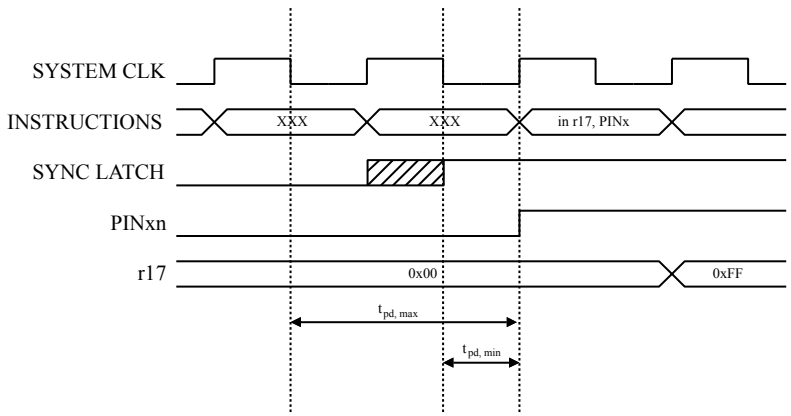
Table 19-1. Port Pin Configurations

DDxn	PORTxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

19.2.4. Reading the Pin Value

Independent of the setting of Data Direction bit DDxn, the port pin can be read through the PINxn Register bit. As shown in Ports as General Digital I/O, the PINxn Register bit and the preceding latch constitute a synchronizer. This is needed to avoid metastability if the physical pin changes value near the edge of the internal clock, but it also introduces a delay. The following figure shows a timing diagram of the synchronization when reading an externally applied pin value. The maximum and minimum propagation delays are denoted $t_{pd,max}$ and $t_{pd,min}$ respectively.

Figure 19-3. Synchronization when Reading an Externally Applied Pin value



Consider the clock period starting shortly after the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded

- PCINT4: Pin Change Interrupt source 4. The PB4 pin can serve as an external interrupt source.

- MOSI0/TXD1/OC2A/PCINT3 – Port B, Bit 3
  - MOSI0: SPI0 Master Data output, Slave Data input for SPI0 channel. When the SPI0 is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB3. When the SPI0 is enabled as a Master, the data direction of this pin is controlled by DDB3. When the pin is forced by the SPI0 to be an input, the pull-up can still be controlled by the PORTB3 bit.
  - TXD1: Transmit Data (Data output pin for the USART1). When the USART1 Transmitter is enabled, this pin is configured as an output regardless of the value of DDB3.
  - OC2A: Output Compare Match output. The PB3 pin can serve as an external output for the Timer/Counter2 Compare Match A. The PB3 pin has to be configured as an output (DDB3 set '1') to serve this function. The OC2A pin is also the output pin for the PWM mode timer function.
  - PCINT3: Pin Change Interrupt source 3. The PB3 pin can serve as an external interrupt source.

- SS0/OC1B/PCINT2 – Port B, Bit 2
  - SS0: Slave0 Select input. When the SPI0 is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB2. As a Slave, the SPI0 is activated when this pin is driven low. When the SPI0 is enabled as a Master, the data direction of this pin is controlled by DDB2. When the pin is forced by the SPI0 to be an input, the pull-up can still be controlled by the PORTB2 bit.
  - OC1B: Output Compare Match output. The PB2 pin can serve as an external output for the Timer/Counter1 Compare Match B. The PB2 pin has to be configured as an output (DDB2 set (one)) to serve this function. The OC1B pin is also the output pin for the PWM mode timer function.
  - PCINT2: Pin Change Interrupt source 2. The PB2 pin can serve as an external interrupt source.

- OC1A/PCINT1 – Port B, Bit 1
  - OC1A: Output Compare Match output. The PB1 pin can serve as an external output for the Timer/Counter1 Compare Match A. The PB1 pin has to be configured as an output (DDB1 set (one)) to serve this function. The OC1A pin is also the output pin for the PWM mode timer function.
  - PCINT1: Pin Change Interrupt source 1. The PB1 pin can serve as an external interrupt source.

- ICP1/CLKO/PCINT0 – Port B, Bit 0
  - ICP1: Input Capture Pin. The PB0 pin can act as an Input Capture Pin for Timer/Counter1.
  - CLKO: Divided System Clock. The divided system clock can be output on the PB0 pin. The divided system clock will be output if the CKOUT Fuse is programmed, regardless of the PORTB0 and DDB0 settings. It will also be output during reset.
  - PCINT0: Pin Change Interrupt source 0. The PB0 pin can serve as an external interrupt source.

Table 19-3 Port B Pins Alternate Functions and Table 19-5 Overriding Signals for Alternate Functions in PB3...PB0 relate the alternate functions of Port B to the overriding signals shown in Figure 19-5 Alternate Port Functions(1). SPI MSTR INPUT and SPI SLAVE OUTPUT constitute the MISO signal, while MOSI is divided into SPI MSTR OUTPUT and SPI SLAVE INPUT.

- SDA0: 2-wire Serial Interface0 Data. When the TWEN bit in TWCR0 is set (one) to enable the 2-wire Serial Interface, pin PC4 is disconnected from the port and becomes the Serial Data I/O pin for the 2-wire Serial Interface0. In this mode, there is a spike filter on the pin to suppress spikes shorter than 50 ns on the input signal, and the pin is driven by an open drain driver with slew-rate limitation.
- PCINT12: Pin Change Interrupt source 12. The PC4 pin can serve as an external interrupt source.
- PC4 can also be used as ADC input Channel 4. The ADC input channel 4 uses digital power.

- ADC3/PCINT11 – Port C, Bit 3
  - PC3 can also be used as ADC input Channel 3. The ADC input channel 3 uses analog power.
  - PCINT11: Pin Change Interrupt source 11. The PC3 pin can serve as an external interrupt source.
- ADC2/PCINT10 – Port C, Bit 2
  - PC2 can also be used as ADC input Channel 2. The ADC input channel 2 uses analog power.
  - PCINT10: Pin Change Interrupt source 10. The PC2 pin can serve as an external interrupt source.
- SCK1/ADC1/PCINT9 – Port C, Bit 1
  - PC1 can also be used as ADC input Channel 1. The ADC input channel 1 uses analog power.
  - SCK1: Master Clock output, Slave Clock input pin for SPI1 channel. When the SPI1 is enabled as a Slave, this pin is configured as an input regardless of the setting of DDB5. When the SPI1 is enabled as a Master, the data direction of this pin is controlled by DDC1. When the pin is forced by the SPI1 to be an input, the pull-up can still be controlled by the PORTC1 bit.
  - PCINT9: Pin Change Interrupt source 9. The PC1 pin can serve as an external interrupt source.
- ADC0/MISO1/PCINT8 – Port C, Bit 0
  - PC0 can also be used as ADC input Channel 0. The ADC input channel 0 uses analog power.
  - MISO1: Master1 Data input, Slave Data output pin for SPI1 channel. When the SPI1 is enabled as a Master, this pin is configured as an input regardless of the setting of DDC0. When the SPI1 is enabled as a Slave, the data direction of this pin is controlled by DDC0. When the pin is forced by the SPI1 to be an input, the pull-up can still be controlled by the PORTC0 bit.
  - PCINT8: Pin Change Interrupt source 8. The PC0 pin can serve as an external interrupt source.

The tables below relate the alternate functions of Port C to the overriding signals shown in Figure 19-5 Alternate Port Functions(1).

Table 19-7. Overriding Signals for Alternate Functions in PC6...PC4[1]

Signal Name	PC6/$\overline{\text{RESET}}$/PCINT14	PC5/SCL0/ADC5/PCINT13	PC4/SDA0/ADC4/PCINT12
PUOE	RSTDISBL	TWEN0	TWEN0
PUOV	1	PORTC5 • $\overline{\text{PUD}}$	PORTC4 • $\overline{\text{PUD}}$
DDOE	RSTDISBL	TWEN0	TWEN0
DDOV	0	SCL_OUT0	SDA_OUT0

Table 19-9. Port D Pins Alternate Functions

Port Pin	Alternate Function
PD7	AIN1 (Analog Comparator Negative Input) PCINT23 (Pin Change Interrupt 23)
PD6	AIN0 (Analog Comparator Positive Input) OC0A (Timer/Counter0 Output Compare Match A Output) PCINT22 (Pin Change Interrupt 22)
PD5	T1 (Timer/Counter 1 External Counter Input) OC0B (Timer/Counter0 Output Compare Match B Output) PCINT21 (Pin Change Interrupt 21)
PD4	XCK0 (USART0 External Clock Input/Output) T0 (Timer/Counter 0 External Counter Input) PCINT20 (Pin Change Interrupt 20)
PD3	INT1 (External Interrupt 1 Input) OC2B (Timer/Counter2 Output Compare Match B Output) PCINT19 (Pin Change Interrupt 19)
PD2	INT0 (External Interrupt 0 Input) PCINT18 (Pin Change Interrupt 18)
PD1	TXD0 (USART0 Output Pin) PCINT17 (Pin Change Interrupt 17)
PD0	RXD1 (USART1 Input Pin) PCINT16 (Pin Change Interrupt 16)

The alternate pin configuration is as follows:

- AIN1/OC2B/PCINT23 – Port D, Bit 7

21.2.2. Registers

The Timer/Counter (TCNT1), Output Compare Registers (OCRA/B), and Input Capture Register (ICR1) are all 16-bit registers. Special procedures must be followed when accessing the 16-bit registers. These procedures are described in section Accessing 16-bit Registers.

The Timer/Counter Control Registers (TCCR1A/B/C) are 8-bit registers and have no CPU access restrictions. Interrupt requests (abbreviated to Int.Req. in the block diagram) signals are all visible in the Timer Interrupt Flag Register (TIFR1). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK1). TIFR1 and TIMSK1 are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler, or by an external clock source on the T1 pin. The Clock Select logic block controls which clock source and edge the Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the Clock Select logic is referred to as the timer clock (clk_{T1}).

The double buffered Output Compare Registers (OCR1A/B) are compared with the Timer/Counter value at all time. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pin (OC1A/B). See Output Compare Units. The compare match event will also set the Compare Match Flag (OCF1A/B) which can be used to generate an Output Compare interrupt request.

The Input Capture Register can capture the Timer/Counter value at a given external (edge triggered) event on either the Input Capture pin (ICP1) or on the Analog Comparator pins. The Input Capture unit includes a digital filtering unit (Noise Canceler) for reducing the chance of capturing noise spikes.

The TOP value, or maximum Timer/Counter value, can in some modes of operation be defined by either the OCR1A Register, the ICR1 Register, or by a set of fixed values. When using OCR1A as TOP value in a PWM mode, the OCR1A Register can not be used for generating a PWM output. However, the TOP value will in this case be double buffered allowing the TOP value to be changed in run time. If a fixed TOP value is required, the ICR1 Register can be used as an alternative, freeing the OCR1A to be used as PWM output.

Related Links

Analog Comparator on page 315

21.3. Block Diagram

The Power Reduction TC1 bit in the Power Reduction Register (PRR.PRTIM1) must be written to zero to enable the TC1 module.

Table 23-1. Definitions

Constant	Description
BOTTOM	The counter reaches the BOTTOM when it becomes zero (0x00).
MAX	The counter reaches its maximum when it becomes 0xFF (decimal 255).
TOP	The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR2A Register. The assignment is dependent on the mode of operation.

23.2.2. Registers

The Timer/Counter (TCNT2) and Output Compare Register (OCR2A and OCR2B) are 8-bit registers. Interrupt request (shorten as Int.Req.) signals are all visible in the Timer Interrupt Flag Register (TIFR2). All interrupts are individually masked with the Timer Interrupt Mask Register (TIMSK2). TIFR2 and TIMSK2 are not shown in the figure.

The Timer/Counter can be clocked internally, via the prescaler, or asynchronously clocked from the TOSC1/2 pins, as detailed later in this section. The asynchronous operation is controlled by the Asynchronous Status Register (ASSR). The Clock Select logic block controls which clock source he Timer/Counter uses to increment (or decrement) its value. The Timer/Counter is inactive when no clock source is selected. The output from the Clock Select logic is referred to as the timer clock (clk_{T2}).

The double buffered Output Compare Register (OCR2A and OCR2B) are compared with the Timer/Counter value at all times. The result of the compare can be used by the Waveform Generator to generate a PWM or variable frequency output on the Output Compare pins (OC2A and OC2B). See Output Compare Unit for details. The compare match event will also set the Compare Flag (OCF2A or OCF2B) which can be used to generate an Output Compare interrupt request.

23.3. Timer/Counter Clock Sources

The Timer/Counter can be clocked by an internal synchronous or an external asynchronous clock source:

The clock source clk_{T2} is by default equal/synchronous to the MCU clock, $clk_{I/O}$.

When the Asynchronous TC2 bit in the Asynchronous Status Register (ASSR.AS2) is written to '1', the clock source is taken from the Timer/Counter Oscillator connected to TOSC1 and TOSC2.

For details on asynchronous operation, see the description of the ASSR. For details on clock sources and prescaler, see Timer/Counter Prescaler.

23.4. Counter Unit

The main part of the 8-bit Timer/Counter is the programmable bi-directional counter unit. Below is the block diagram of the counter and its surroundings.

23.11.2. TC2 Control Register B

Name: TCCR2B
Offset: 0xB1
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	FOC2A	FOC2B			WGM22	CS22	CS21	CS20
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

Bit 7 – FOC2A: Force Output Compare A
The FOC2A bit is only active when the WGM bits specify a non-PWM mode.

To ensure compatibility with future devices, this bit must be set to zero when TCCR2B is written when operating in PWM mode. When writing a logical one to the FOC2A bit, an immediate Compare Match is forced on the Waveform Generation unit. The OC2A output is changed according to its COM2A1:0 bits setting. Note that the FOC2A bit is implemented as a strobe. Therefore it is the value present in the COM2A1:0 bits that determines the effect of the forced compare.

A FOC2A strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR2A as TOP.

The FOC2A bit is always read as zero.

Bit 6 – FOC2B: Force Output Compare B
The FOC2B bit is only active when the WGM bits specify a non-PWM mode.

To ensure compatibility with future devices, this bit must be set to zero when TCCR2B is written when operating in PWM mode. When writing a logical one to the FOC2B bit, an immediate Compare Match is forced on the Waveform Generation unit. The OC2B output is changed according to its COM2B1:0 bits setting. Note that the FOC2B bit is implemented as a strobe. Therefore it is the value present in the COM2B1:0 bits that determines the effect of the forced compare.

A FOC2B strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR2B as TOP.

The FOC2B bit is always read as zero.

Bit 3 – WGM22: Waveform Generation Mode
Refer to TCCR2A.

Bits 2:0 – CS2n: Clock Select [n = 0..2]
The three Clock Select bits select the clock source to be used by the Timer/Counter.

Table 23-10. Clock Select Bit Description

CA22	CA21	CS20	Description
0	0	0	No clock source (Timer/Counter stopped).
0		1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)

24.5.3. SPI Data Register 0

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: SPDR
Offset: 0x4E
Reset: 0xXX
Property: When addressing as I/O Register: address offset is 0x2E

Bit	7	6	5	4	3	2	1	0
	SPID[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bits 7:0 – SPID[7:0]: SPI Data

The SPI Data Register is a read/write register used for data transfer between the Register File and the SPI Shift Register. Writing to the register initiates data transmission. Reading the register causes the Shift Register Receive buffer to be read.

25.12.2. USART Control and Status Register 0 A

Name: UCSR0A
Offset: 0xC0
Reset: 0x20
Property: -

Bit	7	6	5	4	3	2	1	0
	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
Access	R	R/W	R	R	R	R	R/W	R/W
Reset	0	0	1	0	0	0	0	0

Bit 7 – RXC0: USART Receive Complete
This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled, the receive buffer will be flushed and consequently the RXC0 bit will become zero. The RXC0 Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE0 bit).

Bit 6 – TXC0: USART Transmit Complete
This flag bit is set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR0). The TXC0 Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC0 Flag can generate a Transmit Complete interrupt (see description of the TXCIE0 bit).

Bit 5 – UDRE0: USART Data Register Empty
The UDRE0 Flag indicates if the transmit buffer (UDR0) is ready to receive new data. If UDRE0 is one, the buffer is empty, and therefore ready to be written. The UDRE0 Flag can generate a Data Register Empty interrupt (see description of the UDRIE0 bit). UDRE0 is set after a reset to indicate that the Transmitter is ready.

Bit 4 – FE0: Frame Error
This bit is set if the next character in the receive buffer had a Frame Error when received. I.e., when the first stop bit of the next character in the receive buffer is zero. This bit is valid until the receive buffer (UDR0) is read. The FEn bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSR0A.

This bit is reserved in Master SPI Mode (MSPIM).

Bit 3 – DOR0: Data OverRun
This bit is set if a Data OverRun condition is detected. A Data OverRun occurs when the receive buffer is full (two characters), it is a new character waiting in the Receive Shift Register, and a new start bit is detected. This bit is valid until the receive buffer (UDR0) is read. Always set this bit to zero when writing to UCSR0A.

This bit is reserved in Master SPI Mode (MSPIM).

Bit 2 – UPE0: USART Parity Error
This bit is set if the next character in the receive buffer had a Parity Error when received and the Parity Checking was enabled at that point (UPM01 = 1). This bit is valid until the receive buffer (UDR0) is read. Always set this bit to zero when writing to UCSR0A.

This bit is reserved in Master SPI Mode (MSPIM).

The UDORDn bit in UCSRnC sets the frame format used by the USART in MSPIM mode. The Receiver and Transmitter use the same setting. Note that changing the setting of any of these bits will corrupt all ongoing communication for both the Receiver and Transmitter.

16-bit data transfer can be achieved by writing two data bytes to UDRn. A UART transmit complete interrupt will then signal that the 16-bit value has been shifted out.

26.5.1. USART MSPIM Initialization

The USART in MSPIM mode has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting master mode of operation (by setting DDR_XCKn to one), setting frame format and enabling the Transmitter and the Receiver. Only the transmitter can operate independently. For interrupt driven USART operation, the Global Interrupt Flag should be cleared (and thus interrupts globally disabled) when doing the initialization.

Note: To ensure immediate initialization of the XCKn output the baud-rate register (UBRRn) must be zero at the time the transmitter is enabled. Contrary to the normal mode USART operation the UBRRn must then be written to the desired value after the transmitter is enabled, but before the first transmission is started. Setting UBRRn to zero before enabling the transmitter is not necessary if the initialization is done immediately after a reset since UBRRn is reset to zero.

Before doing a re-initialization with changed baud rate, data mode, or frame format, be sure that there is no ongoing transmissions during the period the registers are changed. The TXCn Flag can be used to check that the Transmitter has completed all transfers, and the RXCn Flag can be used to check that there are no unread data in the receive buffer. Note that the TXCn Flag must be cleared before each transmission (before UDRn is written) if it is used for this purpose.

The following simple USART initialization code examples show one assembly and one C function that are equal in functionality. The examples assume polling (no interrupts enabled). The baud rate is given as a function parameter. For the assembly code, the baud rate parameter is assumed to be stored in the r17:r16 registers.

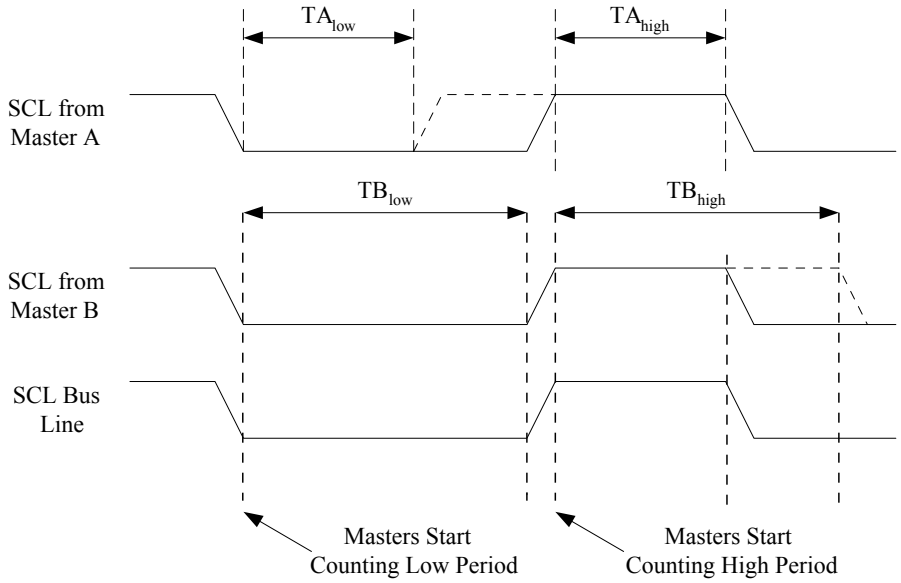
Assembly Code Example

```
clr r18
out UBRRnH,r18
out UBRRnL,r18
; Setting the XCKn port pin as output, enables master mode.
sbi XCKn_DDR, XCKn
; Set MSPI mode of operation and SPI data mode 0.
ldi r18, (1<<UMSELn1)|(1<<UMSELn0)|(0<<UCPHAn)|(0<<UCPOLn)
out UCSRnC,r18
; Enable receiver and transmitter.
ldi r18, (1<<RXENn)|(1<<TXENn)
out UCSRnB,r18
; Set baud rate.
; IMPORTANT: The Baud Rate must be set after the transmitter is enabled!
out UBRRnH, r17
out UBRRnL, r18
ret
```

C Code Example

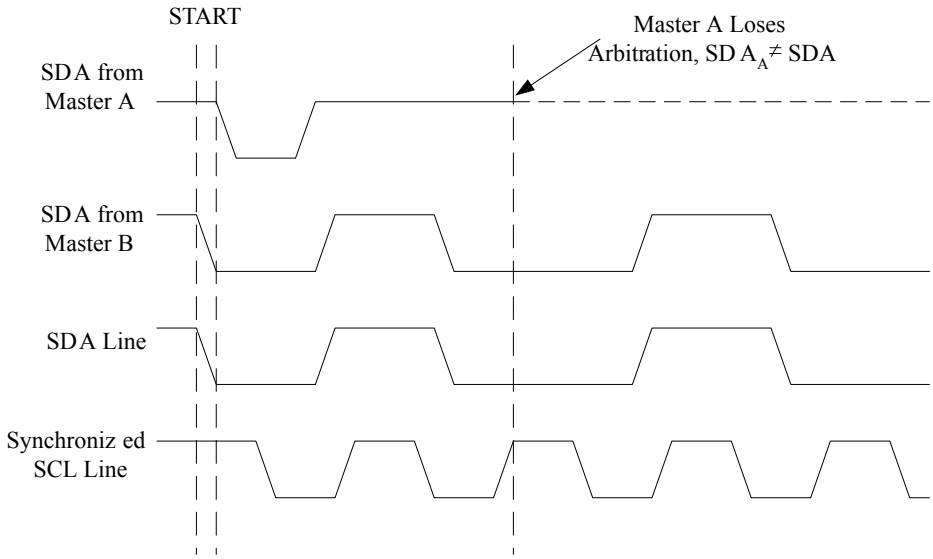
```
{
 UBRRn = 0;
 /* Setting the XCKn port pin as output, enables master mode. */
 XCKn_DDR |= (1<<XCKn);
 /* Set MSPI mode of operation and SPI data mode 0. */
 UCSRnC = (1<<UMSELn1)|(1<<UMSELn0)|(0<<UCPHAn)|(0<<UCPOLn);
 /* Enable receiver and transmitter. */
 UCSRnB = (1<<RXENn)|(1<<TXENn);
 /* Set baud rate. */
```

Figure 27-7. SCL Synchronization Between Multiple Masters



Arbitration is carried out by all masters continuously monitoring the SDA line after outputting data. If the value read from the SDA line does not match the value the Master had output, it has lost the arbitration. Note that a Master can only lose arbitration when it outputs a high SDA value while another Master outputs a low value. The losing Master should immediately go to Slave mode, checking if it is being addressed by the winning Master. The SDA line should be left high, but losing masters are allowed to generate a clock signal until the end of the current data or address packet. Arbitration will continue until only one Master remains, and this may take many bits. If several masters are trying to address the same Slave, arbitration will continue into the data packet.

Figure 27-8. Arbitration Between Two Masters



Note that arbitration is not allowed between:

- A REPEATED START condition and a data bit
- A STOP condition and a data bit
- A REPEATED START and a STOP condition

It is the user software's responsibility to ensure that these illegal arbitration conditions never occur. This implies that in multi-master systems, all data transfers must use the same composition of SLA+R/W and

Note: The EEPRPOM memory is preserved during Chip Erase if the EESAVE Fuse is programmed.

Load Command “Chip Erase”:

1. Set XA1, XA0 to “10”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “1000 0000”. This is the command for Chip Erase.
4. Give XTAL1 a positive pulse. This loads the command.
5. Give $\overline{WR}$ a negative pulse. This starts the Chip Erase. RDY/$\overline{BSY}$ goes low.
6. Wait until RDY/$\overline{BSY}$ goes high before loading a new command.

33.7.4. Programming the Flash

The Flash is organized in pages as number of Words in a Page and number of Pages in the Flash. When programming the Flash, the program data is latched into a page buffer. This allows one page of program data to be programmed simultaneously. The following procedure describes how to program the entire Flash memory:

Step A. Load Command “Write Flash”

1. Set XA1, XA0 to “10”. This enables command loading.
2. Set BS1 to “0”.
3. Set DATA to “0001 0000”. This is the command for Write Flash.
4. Give XTAL1 a positive pulse. This loads the command.

Step B. Load Address Low Byte

1. Set XA1, XA0 to “00”. This enables address loading.
2. Set BS1 to “0”. This selects low address.
3. Set DATA = Address low byte (0x00 - 0xFF).
4. Give XTAL1 a positive pulse. This loads the address low byte.

Step C. Load Data Low Byte

1. Set XA1, XA0 to “01”. This enables data loading.
2. Set DATA = Data low byte (0x00 - 0xFF).
3. Give XTAL1 a positive pulse. This loads the data byte.

Step D. Load Data High Byte

1. Set BS1 to “1”. This selects high data byte.
2. Set XA1, XA0 to “01”. This enables data loading.
3. Set DATA = Data high byte (0x00 - 0xFF).
4. Give XTAL1 a positive pulse. This loads the data byte.

Step E. Latch Data

1. Set BS1 to “1”. This selects high data byte.
2. Give PAGEL a positive pulse. This latches the data bytes. (Please refer to the figure, Programming the Flash Waveforms, in this section for signal waveforms)