8-bit AVR Microcontrollers

ATmega324P/V

DATASHEET COMPLETE

Introduction

The Atmel® picoPower® ATmega324P is a low-power CMOS 8-bit microcontroller based on the AVR® enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega324P achieves throughputs close to 1MIPS per MHz. This empowers system designer to optimize the device for power consumption versus processing speed.

Feature

High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller Family

- Advanced RISC Architecture
  - 131 Powerful Instructions
  - Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 32KBytes of In-System Self-Programmable Flash Program Memory
  - 1KBytes EEPROM
  - 2KBytes Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data Retention: 20 Years at 85°C/100 Years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Atmel QTouch® Library Support
  - Capacitive Touch Buttons, Sliders and Wheels
  - QTouch and QMatrix acquisition
  - Up to 64 Sense Channels
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC
    - Differential Mode with Selectable Gain at 1×, 10× or 200×
  - One Byte-oriented 2-wire Serial Interface (Philips I^2C compatible)
  - Two Programmable Serial USART
  - One Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP
  - 44-lead TQFP
  - 44-pad VQFN/QFN
- Operating Voltage:
  - 1.8 - 5.5V for ATmega324PV
  - 2.7 - 5.5V for ATmega324P
- Speed Grades
  - ATmega324PV:
    - 0 - 4MHz @ 1.8V - 5.5V
    - 0 - 10MHz @ 2.7V - 5.5V
  - ATmega324P:
    - 0 - 10MHz @ 2.7V - 5.5V
    - 0 - 20MHz @ 4.5 - 5.5V
- Power Consumption at 1MHz, 1.8V, 25°C
  - Active Mode: 0.4mA
  - Power-down Mode: 0.1µA
  - Power-save Mode: 0.6µA (Including 32kHz RTC)

1. Refer to *Data Retention*.

Related Links

Data Retention on page 20

Table of Contents

1. Description

The Atmel® ATmega324P is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega324P achieves throughputs close to 1MIPS per MHz. This empowers system designer to optimize the device for power consumption versus processing speed.

The Atmel AVR® core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in a single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega324P provides the following features: 32Kbytes of In-System Programmable Flash with Read-While-Write capabilities, 1Kbytes EEPROM, 2Kbytes SRAM, 32 general purpose I/O lines, 32 general purpose working registers, Real Time Counter (RTC), three flexible Timer/Counters with compare modes and PWM, two serial programmable USARTs , one byte-oriented 2-wire Serial Interface (I2C), a 8-channel 10-bit ADC with optional differential input stage with programmable gain, a programmable Watchdog Timer with internal Oscillator, an SPI serial port, IEEE std. 1149.1 compliant JTAG test interface, also used for accessing the On-chip Debug system and programming and six software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or hardware reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption. In Extended Standby mode, both the main oscillator and the asynchronous timer continue to run.

Atmel offers the QTouch® library for embedding capacitive touch buttons, sliders and wheels functionality into AVR microcontrollers. The patented charge-transfer signal acquisition offers robust sensing and includes fully debounced reporting of touch keys and includes Adjacent Key Suppression® (AKS™) technology for unambiguous detection of key events. The easy-to-use QTouch Suite toolchain allows you to explore, develop and debug your own touch applications.

The device is manufactured using Atmel's high density non-volatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-System through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The Boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega324P is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega324P is supported with a full suite of program and system development tools including: C Compilers, Macro Assemblers, Program Debugger/Simulators, In-Circuit Emulators, and Evaluation kits.

2. Configuration Summary

The table below compares the device series of feature and pin compatible devices, providing a seamless migration path.

Table 2-1. Configuration Summary and Device Comparison

Features	ATmega164/V	ATmega324/V	ATmega644/V
Pin Count	40/44	40/44	40/44
Flash (Bytes)	16K	32K	64K
SRAM (Bytes)	1K	2K	4K
EEPROM (Bytes)	512	1K	2K
General Purpose I/O Lines	32	32	32
SPI	1	1	1
TWI (I^2C)	1	1	1
USART	2	2	2
ADC	10-bit 15ksps	10-bit 15ksps	10-bit 15ksps
ADC Channels	8	8	8
Analog Comparator	1	1	1
8-bit Timer/Counters	2	2	2
16-bit Timer/Counters	1	1	1
PWM channels	6	6	6
Packages	PDIP TQFP VQFN/QFN	PDIP TQFP VQFN/QFN	PDIP TQFP VQFN/QFN

3. Ordering Information

Speed [MHz](3)	Power Supply [V]	Ordering Code(2)	Package(1)	Operational Range
10	1.8 - 5.5	ATmega324PV-10AU ATmega324PV-10AUR(4) ATmega324PV-10PU ATmega324PV-10MU ATmega324PV-10MUR(4)	44A 44A 40P6 44M1 44M1	Industrial (-40°C to 85°C)
20	2.7 - 5.5	ATmega324P-20AU ATmega324P-20AUR(4) ATmega324P-20PU ATmega324P-20MU ATmega324P-20MUR(4)	44A 44A 40P6 44M1 44M1	Industrial (-40°C to 85°C)
10	1.8 - 5.5	ATmega324PV-10AN ATmega324PV-10ANR(4) ATmega324PV-10PN ATmega324PV-10MN ATmega324PV-10MNR(4)	44A 44A 40P6 44M1 44M1	Industrial (-40°C to 105°C)
20	2.7 - 5.5	ATmega324P-20AN ATmega324P-20ANR(4) ATmega324P-20PN ATmega324P-20MN ATmega324P-20MNR(4)	44A 44A 40P6 44M1 44M1	Industrial (-40°C to 105°C)

Note:

1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.
2. Pb-free packaging, complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.
3. Refer to *Speed Grades* for Speed vs. V_{CC}
4. Tape & Reel.

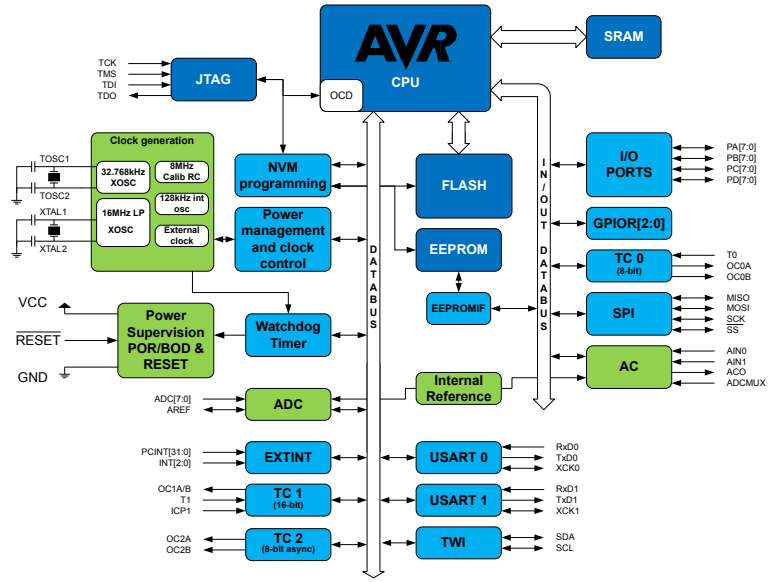
Package Type	
40P6	40-pin, 0.600” Wide, Plastic Dual Inline Package (PDIP)
44A	44-lead, Thin (1.0mm) Plastic Quad Flat Package (TQFP)
44M1	44-pad, 7 × 7 × 1.0mm body, lead pitch 0.50mm, Thermally Enhanced Plastic Very Thin Quad Flat No-Lead (VQFN)

Related Links

Speed Grades on page 399

4. Block Diagram

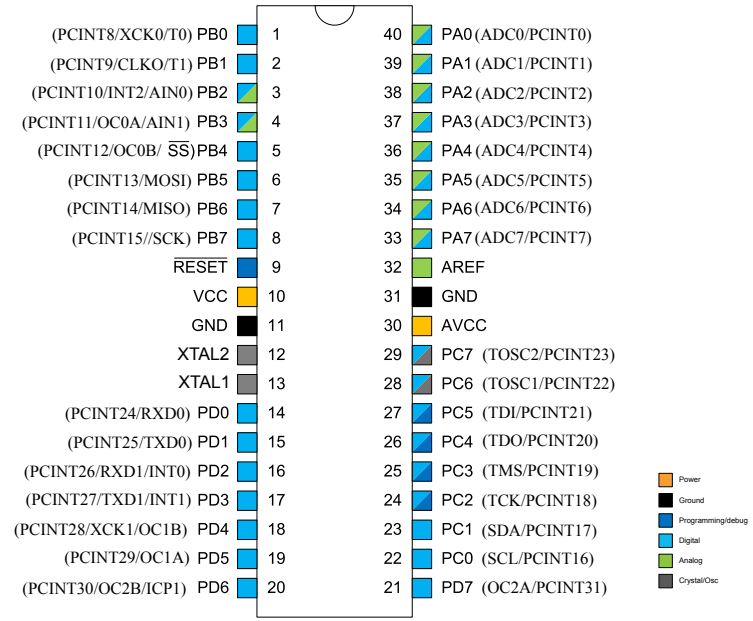
Figure 4-1. Block Diagram



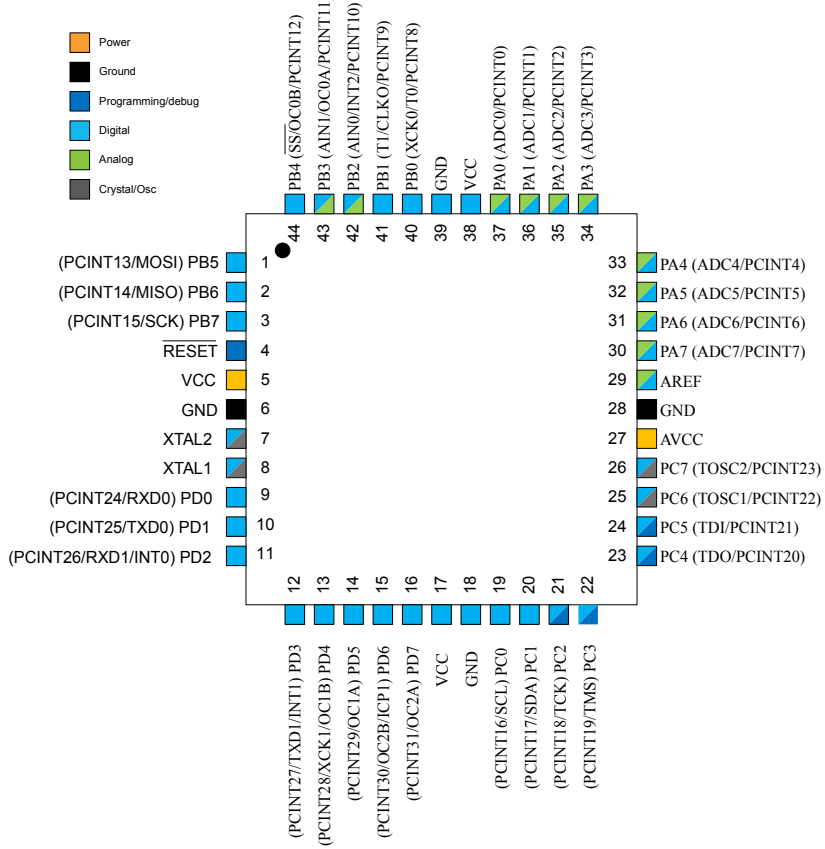
5. Pin Configurations

5.1. Pinout

5.1.1. PDIP



5.1.2. TQFN and QFN



5.2. Pin Descriptions

5.2.1. VCC

Digital supply voltage.

5.2.2. GND

Ground.

5.2.3. Port A (PA[7:0])

This port serves as analog inputs to the Analog-to-digital Converter.

This is an 8-bit, bi-directional I/O port with internal pull-up resistors, individually selectable for each bit. The output buffers have symmetrical drive characteristics, with both high sink and source capability. As inputs, the port pins that are externally pulled low will source current if pull-up resistors are activated. Port pins are tri-stated when a reset condition becomes active, even if the clock is not running.

5.2.4. Port B (PB[7:0])

This is an 8-bit, bi-directional I/O port with internal pull-up resistors, individually selectable for each bit. The output buffers have symmetrical drive characteristics, with both high sink and source capability. As inputs, the port pins that are externally pulled low will source current if pull-up resistors are activated. Port pins are tri-stated when a reset condition becomes active, even if the clock is not running.

This port also serves the functions of various special features.

5.2.5. Port C (PC[7:0])

This is an 8-bit, bi-directional I/O port with internal pull-up resistors, individually selectable for each bit. The output buffers have symmetrical drive characteristics, with both high sink and source capability. As inputs, the port pins that are externally pulled low will source current if pull-up resistors are activated. Port pins are tri-stated when a reset condition becomes active, even if the clock is not running.

This port also serves the functions of the JTAG interface, along with special features.

5.2.6. Port D (PD[7:0])

This is an 8-bit, bi-directional I/O port with internal pull-up resistors, individually selectable for each bit. The output buffers have symmetrical drive characteristics, with both high sink and source capability. As inputs, the port pins that are externally pulled low will source current if pull-up resistors are activated. Port pins are tri-stated when a reset condition becomes active, even if the clock is not running.

This port also serves the functions of various special features.

5.2.7. $\overline{\text{RESET}}$

Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. Shorter pulses are not guaranteed to generate a reset.

5.2.8. XTAL1

Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

5.2.9. XTAL2

Output from the inverting Oscillator amplifier.

5.2.10. AVCC

AVCC is the supply voltage pin for Port A and the Analog-to-digital Converter. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.

5.2.11. AREF

This is the analog reference pin for the Analog-to-digital Converter.

6. I/O Multiplexing

Each pin is by default controlled by the PORT as a general purpose I/O and alternatively it can be assigned to one of the peripheral functions.

The following table describes the peripheral signals multiplexed to the PORT I/O pins.

Table 6-1. PORT Function Multiplexing

32-pin TQFP/ QFN/ MLF Pin #	40-pin PDIP Pin #	PAD	EXTINT	PCINT	ADC/AC	OSC	T/C # 0	T/C # 1	USART	I2C	SPI	JTAG
1	6	PB[5]		PCINT13							MOSI	
2	7	PB[6]		PCINT14							MISO	
3	8	PB[7]		PCINT15							SCK	
4	9	$\overline{\text{RESET}}$										
5	10	VCC										
6	11	GND										
7	12	XTAL2										
8	13	XTAL1										
9	14	PD[0]		PCINT24					RxD0			
10	15	PD[1]		PCINT25					TxD0			
11	16	PD[2]	INT0	PCINT26					RxD1			
12	17	PD[3]	INT1	PCINT27					TXD1			
13	18	PD[4]		PCINT28				OC1B	XCK1			
14	19	PD[5]		PCINT29				OC1A				
15	20	PD[6]		PCINT30			OC2B	ICP1				
16	21	PD[7]		PCINT31			OC2A					
17	-	VCC							RxD2		MISO1	
18	-	GND							TxD2		MOSI1	
19	22	PC[0]		PCINT16						SCL		
20	23	PC[1]		PCINT17						SDA		
21	24	PC[2]		PCINT18								TCK
22	25	PC[3]		PCINT19								TMS
23	26	PC[4]		PCINT20								TDO
24	27	PC[5]		PCINT21								TDI
25	28	PC[6]		PCINT22		TOSC1						
26	29	PC[7]		PCINT23		TOSC2						
27	30	AVCC										
28	31	GND										
29	32	AREF			AREF							
30	33	PA[7]		PCINT7	ADC7							
31	34	PA[6]		PCINT6	ADC6							
32	35	PA[5]		PCINT5	ADC5							
33	36	PA[4]		PCINT4	ADC4							
34	37	PA[3]		PCINT3	ADC3							

32-pin TQFP/ QFN/ MLF Pin #	40-pin PDIP Pin #	PAD	EXTINT	PCINT	ADC/AC	OSC	T/C # 0	T/C # 1	USART	I2C	SPI	JTAG
35	38	PA[2]		PCINT2	ADC2							
36	39	PA[1]		PCINT1	ADC1							
37	40	PA[0]		PCINT0	ADC0							
38	-	VCC								SDA1		
39	-	GND								SCL1		
40	1	PB[0]		PCINT8			T0		XCK0			
41	2	PB[1]		PCINT9		CLKO		T1				
42	3	PB[2]	INT2	PCINT10	AIN0							
43	4	PB[3]		PCINT11	AIN1		OC0A					
44	5	PB[4]		PCINT12			OC0B				$\overline{SS}$	
-	-	GND										
-	-	GND										
-	-	GND										
-	-	GND										
-	-	GND										

13. Interrupts

13.1. Overview

This section describes the specifics of the interrupt handling of the device. For a general explanation of the AVR interrupt handling, refer to the description of *Reset and Interrupt Handling*.

In general:

- Each Interrupt Vector occupies two instruction words.
- The Reset Vector is affected by the BOOTRST fuse, and the Interrupt Vector start address is affected by the IVSEL bit in MCUCR (MCUCR.IVSEL)

Related Links

Reset and Interrupt Handling on page 27

13.2. Interrupt Vectors in ATmega324P

Table 13-1. Reset and Interrupt Vectors in ATmega324P

Vector No	Program Address(2)	Source	Interrupts definition
1	0x0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 1
4	0x0006	INT2	External Interrupt Request 2
5	0x0008	PCINT0	Pin Change Interrupt Request 0
6	0x000A	PCINT1	Pin Change Interrupt Request 1
7	0x000C	PCINT2	Pin Change Interrupt Request 2
8	0x000E	PCINT3	Pin Change Interrupt Request 3
9	0x0010	WDT	Watchdog Time-out Interrupt
10	0x0012	TIMER2_COMPA	Timer/Counter2 Compare Match A
11	0x0014	TIMER2_COMPB	Timer/Coutner2 Compare Match B
12	0x0016	TIMER2_OVF	Timer/Counter2 Overflow
13	0x0018	TIMER1_CAPT	Timer/Counter1 Capture Event
14	0x001A	TIMER1_COMPA	Timer/Counter1 Compare Match A
15	0x001C	TIMER1_COMPB	Timer/Coutner1 Compare Match B
16	0x001E	TIMER1_OVF	Timer/Counter1 Overflow
17	0x0020	TIMER0_COMPA	Timer/Counter0 Compare Match A
18	0x0022	TIMER0_COMPB	Timer/Coutner0 Compare Match B
19	0x0024	TIMER0_OVF	Timer/Counter0 Overflow
20	0x0026	SPI_STC	SPI Serial Transfer Complete

Vector No	Program Address(2)	Source	Interrupts definition
21	0x0028	USART_RX	USART Rx Complete
22	0x002A	USART_UDRE	USART Data Register Empty
23	0x002C	USART_TX	USART Tx Complete
24	0x002E	ANALOG_COMP	Analog Comparator
25	0x0030	ADC	ADC Conversion Complete
26	0x0032	EE_READY	EEPROM Ready
27	0x0034	TWI	TWI Transfer complete
28	0x0036	SPM_READY	Store Program Memory Ready
29	0x0038	USART1_RX	USART1 Rx Complete
30	0x003A	USART1_UDRE	USART1, Data Register Empty
31	0x003C	USART1_TX	USART1, Tx Complete

Note:

1. When the BOOTRST Fuse is programmed, the device will jump to the Boot Loader address at reset, see *Memory programming*
2. When the IVSEL bit in MCUCR is set, Interrupt Vectors will be moved to the start of the Boot Flash Section. The address of each Interrupt Vector will then be the address in this table added to the start address of the Boot Flash Section.

The table below shows reset and Interrupt Vectors placement for the various combinations of BOOTRST and MCUCR.IVSEL settings. If the program never enables an interrupt source, the Interrupt Vectors are not used, and regular program code can be placed at these locations. This is also the case if the Reset Vector is in the Application section while the Interrupt Vectors are in the Boot section or vice versa.

Table 13-2. Reset and Interrupt Vectors placement

BOOTRST	IVSEL	Reset Address	Interrupt Vectors Start Address
1	0	0x0000	0x0002
1	1	0x0000	Boot Reset Address + 0x0002
0	0	Boot Reset Address	0x0002
0	1	Boot Reset Address	Boot Reset Address + 0x0002

Note: The Boot Reset Address is shown in Table *Boot size configuration* in Boot Loader Parameters. For the BOOTRST Fuse “1” means unprogrammed while “0” means programmed.

```
Address   Labels    Code                    Comments
0x0000              jmp     RESET            ; Reset
0x0002              jmp     INT0             ; IRQ0
0x0004              jmp     INT1             ; IRQ1
0x0006              jmp     INT2             ; IRQ2
0x0008              jmp     PCINT0           ; PCINT0
0x000A              jmp     PCINT1           ; PCINT1
0x000C              jmp     PCINT2           ; PCINT2
0x000E              jmp     PCINT3           ; PCINT3
0x0010              jmp     WDT              ; Watchdog Timeout
0x0012              jmp     TIM2_COMPA       ; Timer2 CompareA
0x0014              jmp     TIM2_COMPB       ; Timer2 CompareB
0x0016              jmp     TIM2_OVF         ; Timer2 Overflow
0x0018              jmp     TIM1_CAPT        ; Timer1 Capture
0x001A              jmp     TIM1_COMPA       ; Timer1 CompareA
0x001C              jmp     TIM1_COMPB       ; Timer1 CompareB
0x001E              jmp     TIM1_OVF         ; Timer1 Overflow
0x0020              jmp     TIM0_COMPA       ; Timer0 CompareA
0x0022              jmp     TIM0_COMPB       ; Timer0 CompareB
0x0024              jmp     TIM0_OVF         ; Timer0 Overflow
0x0026              jmp     SPI_STC          ; SPI Transfer Complete
0x0028              jmp     USART_RXC        ; USART RX Complete
0x002A              jmp     USART_UDRE       ; USART UDR Empty
0x002C              jmp     USART_TXC        ; USART TX Complete
0x002E              jmp     ANA_COMP         ; Analog Comparator
0x0030              jmp     ADC              ; ADC Conversion Complete
0x0032              jmp     EE_RDY           ; EEPROM Ready
0x0034              jmp     TWI              ; 2-wire Serial
0x0036              jmp     SPM_RDY          ; SPM Ready
0x0038              jmp     USART1_RXC       ; USART1 RX Complete
0x003A              jmp     USART1_UDRE      ; USART1 UDR Empty
0x003C              jmp     USART1_TXC       ; USART1 TX Complete
;
0x003E    RESET:    ldi     r16,high(RAMEND) ; Main program start
0x003F              out     SPH,r16          ; Set Stack Pointer to top of RAM
0x0040              ldi     r16,low(RAMEND)
0x0041              out     SPL,r16
0x0042              sei                      ; Enable interrupts
0x0043              <instr>  xxx
...       ...       ...                         ...
```


When the BOOTRST Fuse is unprogrammed, the Boot section size set to 8Kbytes and the MCUCR.IVSEL is set before any interrupts are enabled, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```
Address   Labels    Code                    Comments
0x00000   RESET:    ldi     r16,high(RAMEND) ; Main program start
0x00001             out     SPH,r16          ; Set Stack Pointer to top of RAM
0x00002             ldi     r16,low(RAMEND)
0x00003             out     SPL,r16
0x00004             sei                      ; Enable interrupts
0x00005             <instr> xxx
;
.org 0x1F002
0x1F002             jmp     EXT_INT0         ; IRQ0 Handler
0x1F004             jmp     EXT_INT1         ; IRQ1 Handler
...                 ...     ...              ;
0x1FO36             jmp     SPM_RDY          ; SPM Ready Handler
```

When the BOOTRST Fuse is programmed and the Boot section size set to 8Kbytes, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```
Address   Labels    Code                    Comments
.org 0x0002
0x00002             jmp     EXT_INT0         ; IRQ0 Handler
0x00004             jmp     EXT_INT1         ; IRQ1 Handler
...                 ...     ...              ;
0x00036             jmp     SPM_RDY          ; SPM Ready Handler
;
.org       0x1F000
0x1F000    RESET:   ldi     r16,high(RAMEND) ; Main program start
0x1F001             out     SPH,r16          ; Set Stack Pointer to top of RAM
0x1F002             ldi     r16,low(RAMEND)
0x1F003             out     SPL,r16
0x1F004             sei                      ; Enable interrupts
0x1F005             <instr>  xxx
```

When the BOOTRST Fuse is programmed, the Boot section size set to 8Kbytes and the MCUCR.IVSEL Register is set before any interrupts are enabled, the most typical and general program setup for the Reset and Interrupt Vector Addresses is:

```
Address    Labels    Code                    Comments
;
.org 0x1F000
0x1F000              jmp     RESET            ; Reset handler
0x1F002              jmp     EXT_INT0         ; IRQ0 Handler
```

```
0x1F004              jmp     EXT_INT1            ; IRQ1 Handler
...                  ...     ...                 ;
0x1F036              jmp     SPM_RDY             ; SPM Ready Handler
;
0x1F03E    RESET:    ldi     r16,high(RAMEND)    ; Main program start
0x1F03F              out     SPH,r16             ; Set Stack Pointer to top of RAM
0x1F040              ldi     r16,low(RAMEND)
0x1F041              out     SPL,r16
0x1F042              sei                         ; Enable interrupts
0x1F043              <instr> xxx
```

13.3. Register Description

13.3.1. Moving Interrupts Between Application and Boot Space

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

13.3.2. MCU Control 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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: MCUCR
Offset: 0x55
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x35

Bit	7	6	5	4	3	2	1	0
	JTD	BODS	BODSE	PUD			IVSEL	IVCE
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bit 7 – JTD
When this bit is zero, the JTAG interface is enabled if the JTAGEN Fuse is programmed. If this bit is one, the JTAG interface is disabled. In order to avoid unintentional disabling or enabling of the JTAG interface, a timed sequence must be followed when changing this bit: The application software must write this bit to the desired value twice within four cycles to change its value. Note that this bit must not be altered when using the On-chip Debug system.

Bit 6 – BODS: BOD Sleep
The BODS bit must be written to '1' in order to turn off BOD during sleep. Writing to the BODS bit is controlled by a timed sequence and the enable bit BODSE. To disable BOD in relevant sleep modes, both BODS and BODSE must first be written to '1'. Then, BODS must be written to '1' and BODSE must be written to zero within four clock cycles.

The BODS bit is active three clock cycles after it is set. A sleep instruction must be executed while BODS is active in order to turn off the BOD for the actual sleep mode. The BODS bit is automatically cleared after three clock cycles.

Bit 5 – BODSE: BOD Sleep Enable
BODSE enables setting of BODS control bit, as explained in BODS bit description. BOD disable is controlled by a timed sequence.

Bit 4 – PUD: Pull-up Disable
When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01).

Bit 1 – IVSEL: Interrupt Vector Select
When the IVSEL bit is cleared (zero), the Interrupt Vectors are placed at the start of the Flash memory. When this bit is set (one), the Interrupt Vectors are moved to the beginning of the Boot Loader section of the Flash. The actual address of the start of the Boot Flash Section is determined by the BOOTSZ Fuses. To avoid unintentional changes of Interrupt Vector tables, a special write procedure must be followed to change the IVSEL bit:

1. Write the Interrupt Vector Change Enable (IVCE) bit to one.
2. Within four cycles, write the desired value to IVSEL while writing a zero to IVCE.

Interrupts will automatically be disabled while this sequence is executed. Interrupts are disabled in the cycle IVCE is set, and they remain disabled until after the instruction following the write to IVSEL. If IVSEL is not written, interrupts remain disabled for four cycles. The I-bit in the Status Register is unaffected by the automatic disabling.

Note: If Interrupt Vectors are placed in the Boot Loader section and Boot Lock bit BLB02 is programmed, interrupts are disabled while executing from the Application section. If Interrupt Vectors are placed in the Application section and Boot Lock bit BLB12 is programed, interrupts are disabled while executing from the Boot Loader section.

Bit 0 – IVCE: Interrupt Vector Change Enable

The IVCE bit must be written to logic one to enable change of the IVSEL bit. IVCE is cleared by hardware four cycles after it is written or when IVSEL is written. Setting the IVCE bit will disable interrupts, as explained in the IVSEL description above. See Code Example below.

Assembly Code Example

```
Move_interrupts:
; Get MCUCR
in    r16, MCUCR
mov   r17, r16
; Enable change of Interrupt Vectors
ori   r16, (1<<IVCE)
out   MCUCR, r16
; Move interrupts to Boot Flash section
ori   r17, (1<<IVSEL)
out   MCUCR, r17
ret
```

C Code Example

```
void Move_interrupts(void)
{
uchar temp;
/* GET MCUCR*/
temp = MCUCR;
/* Enable change of Interrupt Vectors */
MCUCR = temp|(1<<IVCE);
/* Move interrupts to Boot Flash section */
MCUCR = temp|(1<<IVSEL);
}
```


14. External Interrupts

14.1. EXINT - External Interrupts

The External Interrupts are triggered by the INT pin or any of the PCINT pins. Observe that, if enabled, the interrupts will trigger even if the INT or PCINT pins are configured as outputs. This feature provides a way of generating a software interrupt.

The Pin Change Interrupt Request 3 (PCI3) will trigger if any enabled PCINT[31:24] pin toggles. The Pin Change Interrupt Request 2 (PCI2) will trigger if any enabled PCINT[23:16] pin toggles. The Pin Change Interrupt Request 1 (PCI1) will trigger if any enabled PCINT[15:8] pin toggles. The Pin Change Interrupt Request 0 (PCI0) will trigger if any enabled PCINT[7:0] pin toggles. The PCMSK3, PCMSK2, PCMSK1 and PCMSK0 Registers control which pins contribute to the pin change interrupts. Pin change interrupts on PCINT are detected asynchronously. This implies that these interrupts can be used for waking the part also from sleep modes other than Idle mode.

The external interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the External Interrupt Control Register A (EICRA). When the external interrupts are enabled and are configured as level triggered, the interrupts will trigger as long as the pin is held low. Note that recognition of falling or rising edge interrupts on INT requires the presence of an I/O clock. Low level interrupt on INT is detected asynchronously. This implies that this interrupt can be used for waking the part also from sleep modes other than Idle mode. The I/O clock is halted in all sleep modes except Idle mode.

Note: Note that 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.

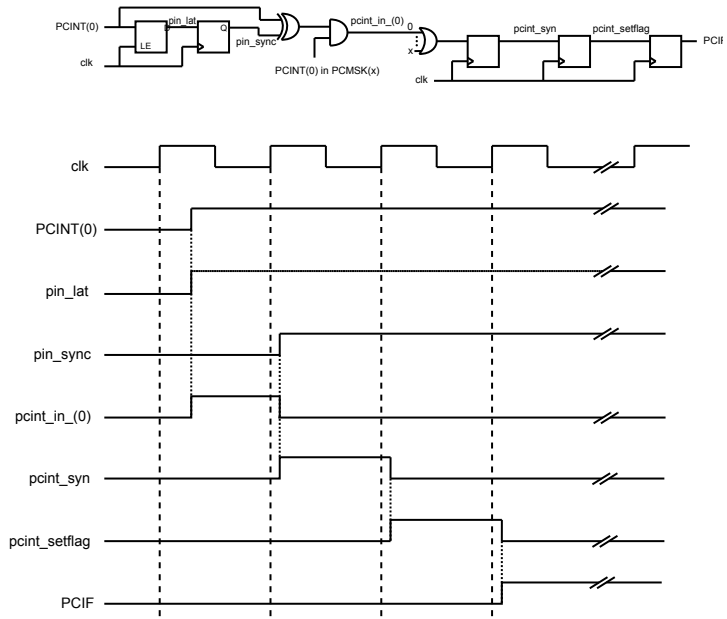
Related Links

System Clock and Clock Options on page 42

14.1.1. Pin Change Interrupt Timing

An example of timing of a pin change interrupt is shown in the following figure.

Figure 14-1. Timing of pin change interrupts



14.1.2. Register Description

14.1.2.1. External Interrupt Control Register A

The External Interrupt Control Register A contains control bits for interrupt sense control.

Name: EICRA
Offset: 0x69
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
			ISC21	ISC20	ISC11	ISC10	ISC01	ISC00
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bits 5:4 – ISC2n: Interrupt Sense Control 2 [n = 1:0]
The External Interrupt 2 is activated by the external pin INT2 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT2 pin that activate the interrupt are defined in table below. The value on the INT2 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Value	Description
00	The low level of INT2 generates an interrupt request.
01	Any logical change on INT2 generates an interrupt request.
10	The falling edge of INT2 generates an interrupt request.
11	The rising edge of INT2 generates an interrupt request.

Bits 3:2 – ISC1n: Interrupt Sense Control 1 [n = 1:0]
The External Interrupt 1 is activated by the external pin INT1 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT1 pin that activate the interrupt are defined in the table below. The value on the INT1 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Value	Description
00	The low level of INT1 generates an interrupt request.
01	Any logical change on INT1 generates an interrupt request.
10	The falling edge of INT1 generates an interrupt request.
11	The rising edge of INT1 generates an interrupt request.

Bits 1:0 – ISC0n: Interrupt Sense Control 0 [n = 1:0]
The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask are set. The level and edges on the external INT0 pin that activate the interrupt are defined in table below. The value on the INT0 pin is sampled before detecting edges. If edge or toggle interrupt is selected, pulses that last longer than one clock period will generate an interrupt. Shorter pulses are not

guaranteed to generate an interrupt. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt.

Value	Description
00	The low level of INT0 generates an interrupt request.
01	Any logical change on INT0 generates an interrupt request.
10	The falling edge of INT0 generates an interrupt request.
11	The rising edge of INT0 generates an interrupt request.

14.1.2.2. External Interrupt Mask 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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: EIMSK
Offset: 0x3D
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x1D

Bit	7	6	5	4	3	2	1	0
						INT2	INT1	INT0
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 2 – INT2: External Interrupt Request 2 Enable
When the INT2 bit is set and the I-bit in the Status Register (SREG) is set, the external pin interrupt is enabled. The Interrupt Sense Control2 bits 1/0 (ISC21 and ISC20) in the External Interrupt Control Register A (EICRA) define whether the external interrupt is activated on rising and/or falling edge of the INT2 pin or level sensed. Activity on the pin will cause an interrupt request even if INT2 is configured as an output. The corresponding interrupt of External Interrupt Request 2 is executed from the INT2 Interrupt Vector.

Bit 1 – INT1: External Interrupt Request 1 Enable
When the INT1 bit is set and the I-bit in the Status Register (SREG) is set, the external pin interrupt is enabled. The Interrupt Sense Control1 bits 1/0 (ISC11 and ISC10) in the External Interrupt Control Register A (EICRA) define whether the external interrupt is activated on rising and/or falling edge of the INT1 pin or level sensed. Activity on the pin will cause an interrupt request even if INT1 is configured as an output. The corresponding interrupt of External Interrupt Request 1 is executed from the INT1 Interrupt Vector.

Bit 0 – INT0: External Interrupt Request 0 Enable
When the INT0 bit is set and the I-bit in the Status Register (SREG) is set, the external pin interrupt is enabled. The Interrupt Sense Control0 bits 1/0 (ISC01 and ISC00) in the External Interrupt Control Register A (EICRA) define whether the external interrupt is activated on rising and/or falling edge of the INT0 pin or level sensed. Activity on the pin will cause an interrupt request even if INT0 is configured as an output. The corresponding interrupt of External Interrupt Request 0 is executed from the INT0 Interrupt Vector.

14.1.2.3. External Interrupt Flag 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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: EIFR
Offset: 0x3C
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x1C

Bit	7	6	5	4	3	2	1	0
						INTF2	INTF1	INTF0
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 2 – INTF2: External Interrupt Flag 2
When an edge or logic change on the INT2 pin triggers an interrupt request, INTF2 will be set. If the I-bit in SREG and the INT2 bit in EIMSK are set, the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing '1' to it. This flag is always cleared when INT2 is configured as a level interrupt.

Bit 1 – INTF1: External Interrupt Flag 1
When an edge or logic change on the INT1 pin triggers an interrupt request, INTF1 will be set. If the I-bit in SREG and the INT1 bit in EIMSK are set, the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing '1' to it. This flag is always cleared when INT1 is configured as a level interrupt.

Bit 0 – INTF0: External Interrupt Flag 0
When an edge or logic change on the INT0 pin triggers an interrupt request, INTF0 will be set. If the I-bit in SREG and the INT0 bit in EIMSK are set, the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing '1' to it. This flag is always cleared when INT0 is configured as a level interrupt.

14.1.2.4. Pin Change Interrupt Control Register

Name: PCICR
Offset: 0x68
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
					PCIE3	PCIE2	PCIE1	PCIE0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – PCIE3: Pin Change Interrupt Enable 3
When the PCIE3 bit is set and the I-bit in the Status Register (SREG) is set, pin change interrupt 3 is enabled. Any change on any enabled PCINT[31:24] pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI3 Interrupt Vector. PCINT[31:24] pins are enabled individually by the PCMSK3 Register.

Bit 2 – PCIE2: Pin Change Interrupt Enable 2
When the PCIE2 bit is set and the I-bit in the Status Register (SREG) is set, pin change interrupt 2 is enabled. Any change on any enabled PCINT[23:16] pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI2 Interrupt Vector. PCINT[23:16] pins are enabled individually by the PCMSK2 Register.

Bit 1 – PCIE1: Pin Change Interrupt Enable 1
When the PCIE1 bit is set and the I-bit in the Status Register (SREG) is set, pin change interrupt 1 is enabled. Any change on any enabled PCINT[14:8] pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI1 Interrupt Vector. PCINT[14:8] pins are enabled individually by the PCMSK1 Register.

Bit 0 – PCIE0: Pin Change Interrupt Enable 0
When the PCIE0 bit is set and the I-bit in the Status Register (SREG) is set, pin change interrupt 0 is enabled. Any change on any enabled PCINT[7:0] pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI0 Interrupt Vector. PCINT[7:0] pins are enabled individually by the PCMSK0 Register.

14.1.2.5. Pin Change Interrupt Flag 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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: PCIFR
Offset: 0x3B
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x1B

Bit	7	6	5	4	3	2	1	0
					PCIF3	PCIF2	PCIF1	PCIF0
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – PCIF3: Pin Change Interrupt Flag 3
When a logic change on any PCINT[31:24] pin triggers an interrupt request, PCIF3 will be set. If the I-bit in SREG and the PCIE3 bit in PCICR are set, the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing '1' to it.

Bit 2 – PCIF2: Pin Change Interrupt Flag 2
When a logic change on any PCINT[23:16] pin triggers an interrupt request, PCIF2 will be set. If the I-bit in SREG and the PCIE2 bit in PCICR are set, the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing '1' to it.

Bit 1 – PCIF1: Pin Change Interrupt Flag 1
When a logic change on any PCINT[15:8] pin triggers an interrupt request, PCIF1 will be set. If the I-bit in SREG and the PCIE1 bit in PCICR are set, the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing '1' to it.

Bit 0 – PCIF0: Pin Change Interrupt Flag 0
When a logic change on any PCINT[7:0] pin triggers an interrupt request, PCIF0 will be set. If the I-bit in SREG and the PCIE0 bit in PCICR are set, the MCU will jump to the corresponding Interrupt Vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing '1' to it.

14.1.2.6. Pin Change Mask Register 0

Name: PCMSK0
Offset: 0x6B
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – PCINTn: Pin Change Enable Mask [n = 7:0]
Each PCINT[7:0] bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT[7:0] is set and the PCIE0 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT[7:0] is cleared, pin change interrupt on the corresponding I/O pin is disabled.

14.1.2.7. Pin Change Mask Register 1

Name: PCMSK1
Offset: 0x6C
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PCINT8, PCINT9, PCINT10, PCINT11, PCINT12, PCINT13, PCINT14, PCINT15: Pin Change Enable Mask
Each PCINT[15:8]-bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT[15:8] is set and the PCIE1 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT[15:8] is cleared, pin change interrupt on the corresponding I/O pin is disabled.

14.1.2.8. Pin Change Mask Register 2

Name: PCMSK2
Offset: 0x6D
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – PCINT16, PCINT17, PCINT18, PCINT19, PCINT20, PCINT21, PCINT22, PCINT23: Pin Change Enable Mask
Each PCINT[23:16]-bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT[23:16] is set and the PCIE2 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT[23:16] is cleared, pin change interrupt on the corresponding I/O pin is disabled.

14.1.2.9. Pin Change Mask Register 3

Name: PCMSK3
Offset: 0x73
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	PCINT31	PCINT30	PCINT29	PCINT28	PCINT27	PCINT26	PCINT25	PCINT24
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

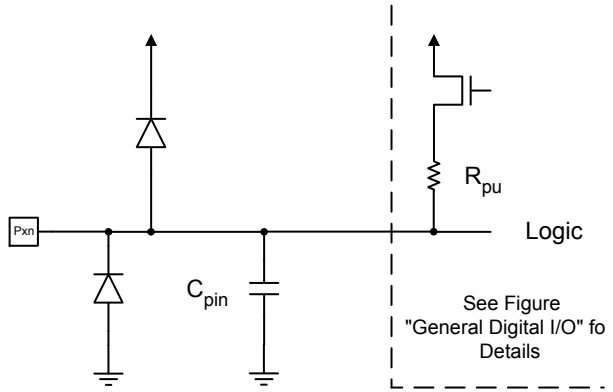
Bits 0, 1, 2, 3, 4, 5, 6, 7 – PCINT24, PCINT25, PCINT26, PCINT27, PCINT28, PCINT29, PCINT30, PCINT31: Pin Change Enable Mask
Each PCINT[31:24]-bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT[31:24] is set and the PCIE3 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT[31:24] is cleared, pin change interrupt on the corresponding I/O pin is disabled.

15. I/O-Ports

15.1. 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 15-1. 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.

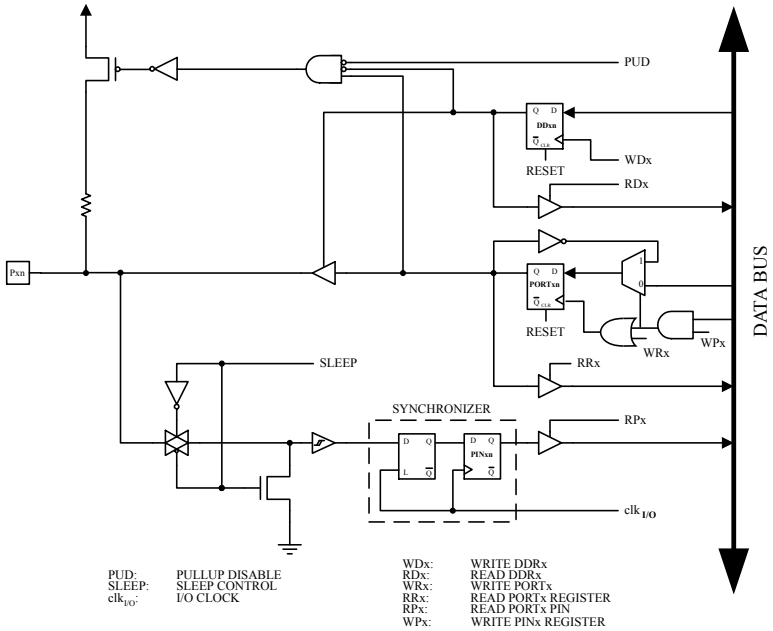
Using the I/O port as General Digital I/O is described in next section. Most port pins are multiplexed with alternate functions for the peripheral features on the device. How each alternate function interferes with the port pin is described in *Alternate Port Functions* section in this chapter. Refer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

15.2. Ports as General Digital I/O

The ports are bi-directional I/O ports with optional internal pull-ups. The following figure shows the functional description of one I/O-port pin, here generically called Pxn.

Figure 15-2. General Digital I/O(1)



Note: 1. WRx, WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, SLEEP, and PUD are common to all ports.

15.2.1. Configuring the Pin

Each port pin consists of three register bits: DDxn, PORTxn, and PINxn. As shown in the Register Description, the DDxn bits are accessed at the DDRx I/O address, the PORTxn bits at the PORTx I/O address, and the PINxn bits at the PINx I/O address.

The DDxn bit in the DDRx Register selects the direction of this pin. If DDxn is written to '1', Pxn is configured as an output pin. If DDxn is written to '0', Pxn is configured as an input pin.

If PORTxn is written to '1' when the pin is configured as an input pin, the pull-up resistor is activated. To switch the pull-up resistor off, PORTxn has to be written to '0' or the pin has to be configured as an output pin. The port pins are tri-stated when reset condition becomes active, even if no clocks are running.

If PORTxn is written to '1' when the pin is configured as an output pin, the port pin is driven high. If PORTxn is written logic zero when the pin is configured as an output pin, the port pin is driven low.

15.2.2. 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.

15.2.3. 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.

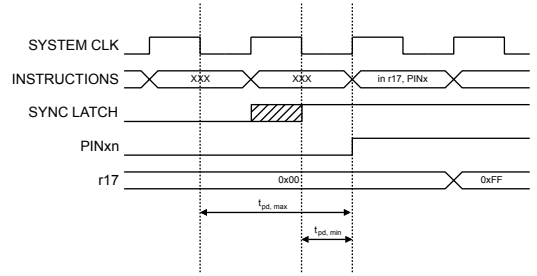
Table 15-1. Port Pin Configurations

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)

15.2.4. 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 15-3. Synchronization when Reading an Externally Applied Pin value

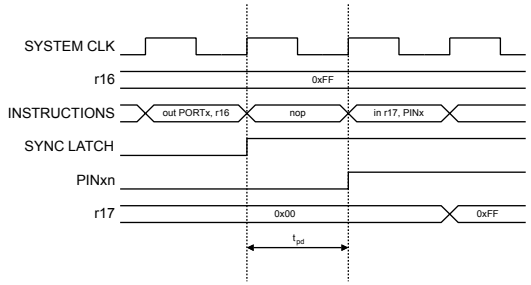


Consider the clock period starting shortly after the first falling edge of the system clock. The latch is closed when the clock is low, and goes transparent when the clock is high, as indicated by the shaded

region of the “SYNC LATCH” signal. The signal value is latched when the system clock goes low. It is clocked into the PINxn Register at the succeeding positive clock edge. As indicated by the two arrows tpd,max and tpd,min, a single signal transition on the pin will be delayed between ½ and 1½ system clock period depending upon the time of assertion.

When reading back a software assigned pin value, a nop instruction must be inserted as indicated in the following figure. The out instruction sets the “SYNC LATCH” signal at the positive edge of the clock. In this case, the delay tpd through the synchronizer is 1 system clock period.

Figure 15-4. Synchronization when Reading a Software Assigned Pin Value



The following code example shows how to set port B pins 0 and 1 high, 2 and 3 low, and define the port pins from 4 to 7 as input with pull-ups assigned to port pins 6 and 7. The resulting pin values are read back again, but as previously discussed, a nop instruction is included to be able to read back the value recently assigned to some of the pins.

Assembly Code Example(1)

```
...
 ; Define pull-ups and set outputs high
 ; Define directions for port pins
 ldi r16,(1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0)
 ldi r17,(1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0)
 out PORTB,r16
 out DDRB,r17
 ; Insert nop for synchronization
 nop
 ; Read port pins
 in r16,PINB
 ...
```

Note: 1. For the assembly program, two temporary registers are used to minimize the time from pull-ups are set on pins 0, 1, 6, and 7, until the direction bits are correctly set, defining bit 2 and 3 as low and redefining bits 0 and 1 as strong high drivers.

C Code Example

```
unsigned char i;
 ...
 /* Define pull-ups and set outputs high */
 /* Define directions for port pins */
 PORTB = (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0);
 DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
 /* Insert nop for synchronization*/
 __no_operation();
 /* Read port pins */
 i = PINB;
 ...
```


15.2.5. Digital Input Enable and Sleep Modes

As shown in the figure of General Digital I/O, the digital input signal can be clamped to ground at the input of the Schmitt Trigger. The signal denoted SLEEP in the figure, is set by the MCU Sleep Controller in Power-down mode and Standby mode to avoid high power consumption if some input signals are left floating, or have an analog signal level close to $V_{CC}/2$.

SLEEP is overridden for port pins enabled as external interrupt pins. If the external interrupt request is not enabled, SLEEP is active also for these pins. SLEEP is also overridden by various other alternate functions as described in *Alternate Port Functions* section in this chapter.

If a logic high level is present on an asynchronous external interrupt pin configured as “Interrupt on Rising Edge, Falling Edge, or Any Logic Change on Pin” while the external interrupt is not enabled, the corresponding External Interrupt Flag will be set when resuming from the above mentioned Sleep mode, as the clamping in these sleep mode produces the requested logic change.

15.2.6. Unconnected Pins

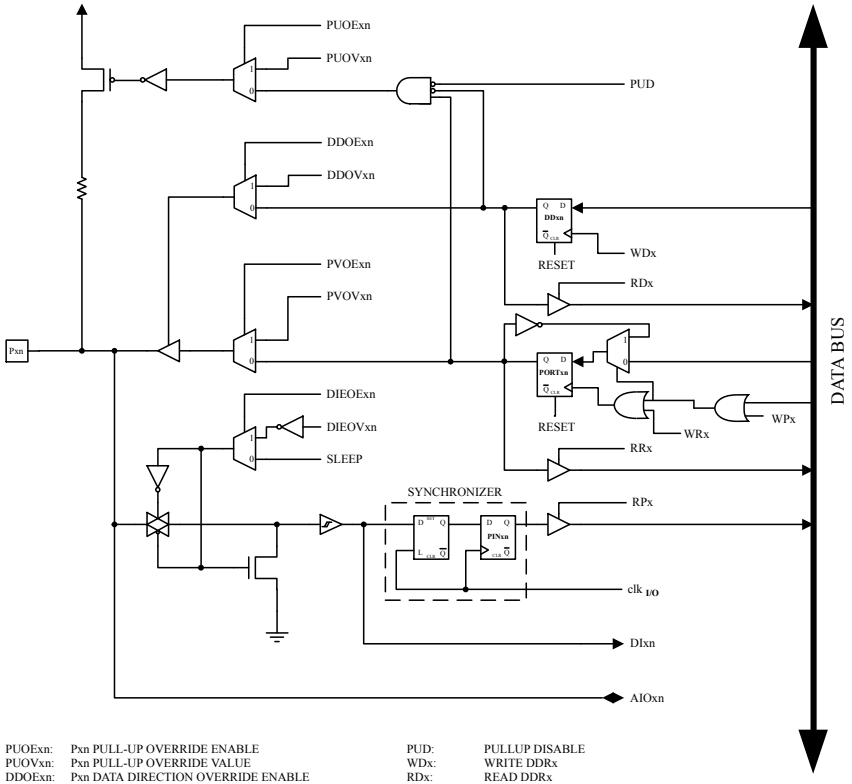
If some pins are unused, it is recommended to ensure that these pins have a defined level. Even though most of the digital inputs are disabled in the deep sleep modes as described above, floating inputs should be avoided to reduce current consumption in all other modes where the digital inputs are enabled (Reset, Active mode and Idle mode).

The simplest method to ensure a defined level of an unused pin, is to enable the internal pull-up. In this case, the pull-up will be disabled during reset. If low power consumption during reset is important, it is recommended to use an external pull-up or pull-down. Connecting unused pins directly to V_{CC} or GND is not recommended, since this may cause excessive currents if the pin is accidentally configured as an output.

15.3. Alternate Port Functions

Most port pins have alternate functions in addition to being general digital I/Os. The following figure shows how the port pin control signals from the simplified Figure 15-2 can be overridden by alternate functions. The overriding signals may not be present in all port pins, but the figure serves as a generic description applicable to all port pins in the AVR microcontroller family.

Figure 15-5. Alternate Port Functions[1]



PUOExn:	Pxn PULL-UP OVERRIDE ENABLE	PUD:	PULLUP DISABLE
PUOVxn:	Pxn PULL-UP OVERRIDE VALUE	WDx:	WRITE DDRx
DDOExn:	Pxn DATA DIRECTION OVERRIDE ENABLE	RDx:	READ DDRx
DDOVxn:	Pxn DATA DIRECTION OVERRIDE VALUE	RRx:	READ PORTx REGISTER
PVOExn:	Pxn PORT VALUE OVERRIDE ENABLE	WRx:	WRITE PORTx
PVOVxn:	Pxn PORT VALUE OVERRIDE VALUE	RPx:	READ PORTx PIN
DIEOExn:	Pxn DIGITAL INPUT-ENABLE OVERRIDE ENABLE	WPx:	WRITE PINx
DIEOVxn:	Pxn DIGITAL INPUT-ENABLE OVERRIDE VALUE	$clk_{I/O}$:	I/O CLOCK
SLEEP:	SLEEP CONTROL	DIxn:	DIGITAL INPUT PIN n ON PORTx
		AIOxn:	ANALOG INPUT/OUTPUT PIN n ON PORTx

Note: 1. WRx, WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, SLEEP, and PUD are common to all ports. All other signals are unique for each pin.

The following table summarizes the function of the overriding signals. The pin and port indexes from previous figure are not shown in the succeeding tables. The overriding signals are generated internally in the modules having the alternate function.

Table 15-2. Generic Description of Overriding Signals for Alternate Functions

Signal Name	Full Name	Description
PUOE	Pull-up Override Enable	If this signal is set, the pull-up enable is controlled by the PUOV signal. If this signal is cleared, the pull-up is enabled when {DDxn, PORTxn, PUD} = 0b010.
PUOV	Pull-up Override Value	If PUOE is set, the pull-up is enabled/disabled when PUOV is set/cleared, regardless of the setting of the DDxn, PORTxn, and PUD Register bits.
DDOE	Data Direction Override Enable	If this signal is set, the Output Driver Enable is controlled by the DDOV signal. If this signal is cleared, the Output driver is enabled by the DDxn Register bit.
DDOV	Data Direction Override Value	If DDOE is set, the Output Driver is enabled/disabled when DDOV is set/cleared, regardless of the setting of the DDxn Register bit.
PVOE	Port Value Override Enable	If this signal is set and the Output Driver is enabled, the port value is controlled by the PVOV signal. If PVOE is cleared, and the Output Driver is enabled, the port Value is controlled by the PORTxn Register bit.
PVOV	Port Value Override Value	If PVOE is set, the port value is set to PVOV, regardless of the setting of the PORTxn Register bit.
DIEOE	Digital Input Enable Override Enable	If this bit is set, the Digital Input Enable is controlled by the DIEOV signal. If this signal is cleared, the Digital Input Enable is determined by MCU state (Normal mode, sleep mode).
DIEOV	Digital Input Enable Override Value	If DIEOE is set, the Digital Input is enabled/disabled when DIEOV is set/cleared, regardless of the MCU state (Normal mode, sleep mode).
DI	Digital Input	This is the Digital Input to alternate functions. In the figure, the signal is connected to the output of the Schmitt Trigger but before the synchronizer. Unless the Digital Input is used as a clock source, the module with the alternate function will use its own synchronizer.
AIO	Analog Input/ Output	This is the Analog Input/output to/from alternate functions. The signal is connected directly to the pad, and can be used bi-directionally.

The following subsections shortly describe the alternate functions for each port, and relate the overriding signals to the alternate function. Refer to the alternate function description for further details.

15.3.1. Alternate Functions of Port A

The Port A pins with alternate functions are shown in the table below:

Table 15-3. Port A Pins Alternate Functions

Port Pin	Alternate Functions
PA7	ADC7 (ADC input channel 7) PCINT7 (Pin Change Interrupt 7)
PA6	ADC6 (ADC input channel 6) PCINT6 (Pin Change Interrupt 6)

Port Pin	Alternate Functions
PA5	ADC5 (ADC input channel 5) PCINT5 (Pin Change Interrupt 5)
PA4	ADC4 (ADC input channel 4) PCINT4 (Pin Change Interrupt 4)
PA3	ADC3 (ADC input channel 3) PCINT3 (Pin Change Interrupt 3)
PA2	ADC2 (ADC input channel 2) PCINT2 (Pin Change Interrupt 2)
PA1	ADC1 (ADC input channel 1) PCINT1 (Pin Change Interrupt 1)
PA0	ADC0 (ADC input channel 0) PCINT0 (Pin Change Interrupt 0)

The alternate pin configuration is as follows:

- ADC[7:0]/PCINT[7:0] – Port A, Bit [7:0]
  - ADC[7:0]: Analog to Digital Converter Channels [7:0].
  - PCINT[7:0]: Pin Change Interrupt source [7:0]. The PA[7:0] pins can serve as external interrupt sources.

Table 15-4. Overriding Signals for Alternate Functions in PA7...PA4

Signal Name	PA7/ADC7/ PCINT7	PA6/ADC6/ PCINT6	PA5/ADC5/ PCINT5	PA4/ADC4/ PCINT4
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
DIEOE	PCINT7 • PCIE0 + ADC7D	PCINT6 • PCIE0 + ADC6D	PCINT5 • PCIE0 + ADC5D	PCINT4 • PCIE0 + ADC4D
DIEOV	PCINT7 • PCIE0	PCINT6 • PCIE0	PCINT5 • PCIE0	PCINT4 • PCIE0
DI	PCINT7 INPUT	PCINT6 INPUT	PCINT5 INPUT	PCINT4 INPUT
AIO	ADC7 INPUT	ADC6 INPUT	ADC5 INPUT	ADC4 INPUT

Table 15-5. Overriding Signals for Alternate Functions in PA3...PA0

Signal Name	PA3/ADC3/ PCINT3	PA2/ADC2/ PCINT2	PA1/ADC1/ PCINT1	PA0/ADC0/ PCINT0
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
DIEOE	PCINT3 • PCIE0 + ADC3D	PCINT2 • PCIE0 + ADC2D	PCINT1 • PCIE0 + ADC1D	PCINT0 • PCIE0 + ADC0D
DIEOV	PCINT3 • PCIE0	PCINT2 • PCIE0	PCINT1 • PCIE0	PCINT0 • PCIE0
DI	PCINT3 INPUT	PCINT2 INPUT	PCINT1 INPUT	PCINT0 INPUT
AIO	ADC3 INPUT	ADC2 INPUT	ADC1 INPUT	ADC0 INPUT

15.3.2. Alternate Functions of Port B

The Port B pins with alternate functions are shown in the table below:

Table 15-6. Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB7	SCK (SPI Bus Master clock input) PCINT15 (Pin Change Interrupt 15)
PB6	MISO (SPI Bus Master Input/Slave Output) PCINT14 (Pin Change Interrupt 14)
PB5	MOSI (SPI Bus Master Output/Slave Input) PCINT13 (Pin Change Interrupt 13)
PB4	$\overline{SS}$ (SPI Slave Select input) OC0B (Timer/Counter 0 Output Compare Match B Output) PCINT12 (Pin Change Interrupt 12)
PB3	AIN1 (Analog Comparator Negative Input) OC0A (Timer/Counter 0 Output Compare Match A Output) PCINT11 (Pin Change Interrupt 11)
PB2	AIN0 (Analog Comparator Positive Input) INT2 (External Interrupt 2 Input) PCINT10 (Pin Change Interrupt 10)

Port Pin	Alternate Functions
PB1	T1 (Timer/Counter 1 External Counter Input) CLKO (Divided System Clock Output) PCINT9 (Pin Change Interrupt 9)
PB0	T0 (Timer/Counter 0 External Counter Input) XCK0 (USART0 External Clock Input/Output) PCINT8 (Pin Change Interrupt 8)

The alternate pin configuration is as follows:

- SCK/PCINT15 – Port B, Bit 7
  - SCK: Master Clock output, Slave Clock input pin for SPI0 channel. When the SPI0 is enabled as a slave, this pin is configured as an input regardless of the setting of DDB7. When the SPI0 is enabled as a master, the data direction of this pin is controlled by DDB7. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB7 bit.
  - PCINT15: Pin Change Interrupt source 15. The PB7 pin can serve as an external interrupt source.
- MISO/PCINT14 – Port B, Bit 6
  - MISO: Master Data input, Slave Data output pin for SPI channel. When the SPI0 is enabled as a master, this pin is configured as an input regardless of the setting of DDB6. When the SPI is enabled as a slave, the data direction of this pin is controlled by DDB6. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB6 bit.
  - PCINT14: Pin Change Interrupt source 14. The PB6 pin can serve as an external interrupt source.
- MOSI/PCINT13 – Port B, Bit 5
  - MOSI: SPI Master Data output, Slave Data input for SPI channel. When the SPI0 is enabled as a slave, this pin is configured as an input regardless of the setting of DDB5. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB5. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB5 bit.
  - PCINT13: Pin Change Interrupt source 13. The PB5 pin can serve as an external interrupt source.
- $\overline{SS}$/OC0B/PCINT12 – Port B, Bit 4
  - $\overline{SS}$: Slave Port Select input. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB4. As a slave, the SPI0 is activated when this pin is driven low. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB4. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB4 bit.
  - OC0B: Output Compare Match B output. The PB4 pin can serve as an external output for the Timer/Counter0 Output Compare. The pin has to be configured as an output (DDB4 set “1”) to serve this function. The OC0B pin is also the output pin for the PWM mode timer function.
  - PCINT12: Pin Change Interrupt source 12. The PB4 pin can serve as an external interrupt source.
- AIN1/OC0A/PCINT11– Port B, Bit 3
  - AIN1: Analog Comparator Negative input. This pin is directly connected to the negative input of the Analog Comparator.

  - OC0A: Output Compare Match A output. The PB3 pin can serve as an external output for the Timer/Counter0 Output Compare. The pin has to be configured as an output (DDB3 set “1”) to serve this function. The OC0A pin is also the output pin for the PWM mode timer function.
  - PCINT11: Pin Change Interrupt source 11. The PB3 pin can serve as an external interrupt source.
- AIN0/INT2/PCINT10 – Port B, Bit 2
  - AIN0: Analog Comparator Positive input. This pin is directly connected to the positive input of the Analog Comparator.
  - INT2: External Interrupt source 2. The PB2 pin can serve as an External Interrupt source to the MCU.
  - PCINT10: Pin Change Interrupt source 10. The PB2 pin can serve as an external interrupt source.
- T1/CLKO/PCINT9 – Port B, Bit 1
  - T1: Timer/Counter1 counter source.
  - CLKO: Divided System Clock: The divided system clock can be output on the PB1 pin. The divided system clock will be output if the CKOUT Fuse is programmed, regardless of the PORTB1 and DDB1 settings. It will also be output during reset.
  - PCINT9: Pin Change Interrupt source 9. The PB1 pin can serve as an external interrupt source.
- T0/XCK0/PCINT8 – Port B, Bit 0
  - T0: Timer/Counter0 counter source.
  - XCK0: USART0 External clock. The Data Direction Register (DDB0) controls whether the clock is output (DDB0 set “1”) or input (DDB0 cleared). The XCK0 pin is active only when the USART0 operates in Synchronous mode.
  - PCINT8: Pin Change Interrupt source 8. The PB0 pin can serve as an external interrupt source.

Table 15-7 and Table 15-8 relate the alternate functions of Port B to the overriding signals shown in Figure 15-5. SPI MSTR INPUT and SPI SLAVE OUTPUT constitute the MISO signal, while MOSI is divided into SPI MSTR OUTPUT and SPI SLAVE INPUT.

Table 15-7. Overriding Signals for Alternate Functions in PB[7:4]

Signal Name	PB7/SCK/PCINT15	PB6/MISO/PCINT14	PB5/MOSI/PCINT13	PB4/$\overline{SS}$/OC0B/PCINT12
PUOE	SPE • $\overline{MSTR}$	SPE • MSTR	SPE • $\overline{MSTR}$	SPE • $\overline{MSTR}$
PUOV	PORTB7 • $\overline{PUD}$	PORTB6 • $\overline{PUD}$	PORTB5 • $\overline{PUD}$	PORTB4 • $\overline{PUD}$
DDOE	SPE • $\overline{MSTR}$	SPE • MSTR	SPE • $\overline{MSTR}$	SPE • $\overline{MSTR}$
DDOV	0	0	0	0
PVOE	SPE • MSTR	SPE • $\overline{MSTR}$	SPE • MSTR	OC0B ENABLE
PVOV	SCK OUTPUT	SPI SLAVE OUTPUT	SPI MSTR OUTPUT	OC0B
DIEOE	PCINT15 • PCIE1	PCINT14 • PCIE1	PCINT13 • PCIE1	PCINT12 • PCIE1
DIEOV	1	1	1	1

Signal Name	PB7/SCK/PCINT15	PB6/MISO/PCINT14	PB5/MOSI/PCINT13	PB4/$\overline{SS}$/OC0B/PCINT12
DI	SCK INPUT PCINT15 INPUT	SPI MSTR INPUT PCINT14 INPUT	SPI SLAVE INPUT PCINT13 INPUT	SPI $\overline{SS}$ PCINT12 INPUT
AIO	-	-	–	–

Table 15-8. Overriding Signals for Alternate Functions in PB[3:0]

Signal Name	PB3/AIN1/OC0A/PCINT11	PB2/AIN0/INT2/PCINT10	PB1/T1/CLKO/PCINT9	PB0/T0/XCK0/PCINT8
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	CKOUT	0
DDOV	0	0	CKOUT	0
PVOE	OC0A ENABLE	0	CKOUT	0
PVOV	OC0A	0	CLK I/O	0
DIEOE	PCINT11 • PCIE1	INT2 ENABLE PCINT10 • PCIE1	PCINT9 • PCIE1	PCINT8 • PCIE1
DIEOV	1	1	1	1
DI	PCINT11 INPUT	INT2 INPUT PCINT10 INPUT	T1 INPUT PCINT9 INPUT	T0 INPUT PCINT8 INPUT
AIO	AIN1 INPUT	AIN0 INPUT	–	–

15.3.3. Alternate Functions of Port C

The Port C pins with alternate functions are shown in the table below:

Table 15-9. Port C Pins Alternate Functions

Port Pin	Alternate Function
PC7	TOSC2 (Timer Oscillator pin 2) PCINT23 (Pin Change Interrupt 23)
PC6	TOSC1 (Timer Oscillator pin 1) PCINT22 (Pin Change Interrupt 22)
PC5	TDI (JTAG Test Data Input) PCINT21 (Pin Change Interrupt 21)
PC4	TDO (JTAG Test Data Output) PCINT20 (Pin Change Interrupt 20)
PC3	TMS (JTAG Test Mode Select) PCINT19 (Pin Change Interrupt 19)
PC2	TCK (JTAG Test Clock) PCINT18 (Pin Change Interrupt 18)
PC1	SDA (two-wire Serial Bus Data Input/Output Line) PCINT17 (Pin Change Interrupt 17)
PC0	SCL (two-wire Serial Bus Clock Line) PCINT16 (Pin Change Interrupt 16)

The alternate pin configuration is as follows:

- TOSC2/PCINT23 – Port C, Bit 7
  - TOSC2: Timer Oscillator pin 2. The PC7 pin can serve as an external interrupt source to the MCU.
  - PCINT23: Pin Change Interrupt source 23. The PC7 pin can serve as an external interrupt source.
- TOSC1/PCINT22 – Port C, Bit 6
  - TOSC1: Timer Oscillator pin 1. The PC6 pin can serve as an external interrupt source to the MCU.
  - PCINT22: Pin Change Interrupt source 22. The PC6 pin can serve as an external interrupt source.
- TDI/PCINT21 – Port C, Bit 5
  - TDI: JTAG Test Data Input.
  - PCINT21: Pin Change Interrupt source 21. The PC5 pin can serve as an external interrupt source.
- TDO/PCINT20 – Port C, Bit 4
  - TDO: JTAG Test Data Output.
  - PCINT20: Pin Change Interrupt source 20. The PC4 pin can serve as an external interrupt source.
- TMS/PCINT19 – Port C, Bit 3
  - TMS: JTAG Test Mode Select.
  - PCINT19: Pin Change Interrupt source 19. The PC3 pin can serve as an external interrupt source.
- TCK/PCINT18 – Port C, Bit 2
  - TCK: JTAG Test Clock.
  - PCINT18: Pin Change Interrupt source 18. The PC2 pin can serve as an external interrupt source.
- SDA/PCINT17 – Port C, Bit 1
  - SDA: two-wire Serial Bus Data Input/Output Line
  - PCINT17: Pin Change Interrupt source 17. The PC1 pin can serve as an external interrupt source.

- SCL/PCINT16 – Port C, Bit 0
  - SCL: two-wire Serial Bus Clock Line.
  - PCINT16: Pin Change Interrupt source 16. 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 15-5.

Table 15-10. Overriding Signals for Alternate Functions in PC[7:4]

Signal Name	PC7/TOSC2/PCINT23	PC6/TOSC1/PCINT22	PC5/TDI/PCINT21	PC4/TDO/PCINT20
PUOE	AS2 • $\overline{\text{EXCLK}}$	AS2	JTAGEN	JTAGEN
PUOV	0	0	1	1
DDOE	AS2 • $\overline{\text{EXCLK}}$	AS2	JTAGEN	JTAGEN
DDOV	0	0	0	SHIFT_IR + SHIFT_DR
PVOE	0	0	0	JTAGEN
PVOV	0	0	0	TDO
DIEOE	AS2 • $\overline{\text{EXCLK}}$ + PCINT23 • PCIE2	AS2 + PCINT22 • PCIE2	JTAGEN + PCINT21 • PCIE2	JTAGEN + PCINT20 • PCIE2
DIEOV	$\overline{\text{AS2}}$	EXCLK + $\overline{\text{AS2}}$	$\overline{\text{JTAGEN}}$	$\overline{\text{JTAGEN}}$
DI	PCINT23 INPUT	PCINT22 INPUT	PCINT21 INPUT	PCINT20 INPUT
AIO	TC2 OSC OUTPUT	TC1 OSC INPUT	TDI INPUT	-

Table 15-11. Overriding Signals for Alternate Functions in PC[3:0]

Signal Name	PC3/TMS/PCINT19	PC2/TCK/PCINT18	PC1/SDA/PCINT17	PC0/SCL/PCINT16
PUOE	JTAGEN	JTAGEN	TWEN	TWEN
PUOV	1	1	PORTC1 • $\overline{\text{PUD}}$	PORTC0 • $\overline{\text{PUD}}$
DDOE	JTAGEN	JTAGEN	TWEN	TWEN
DDOV	0	0	0	0
PVOE	0	0	TWEN	TWEN
PVOV	0	0	SDA OUT	SCL OUT
DIEOE	JTAGEN + PCINT19 • PCIE2	JTAGEN + PCINT18 • PCIE2	PCINT17 • PCIE2	PCINT16 • PCIE2
DIEOV	$\overline{\text{JTAGEN}}$	$\overline{\text{JTAGEN}}$	1	1
DI	PCINT19 INPUT	PCINT18 INPUT	PCINT17 INPUT	PCINT16 INPUT
AIO	TMS INPUT	TCK INPUT	SDA INPUT	SCL INPUT

15.3.4. Alternate Functions of Port D

The Port D pins with alternate functions are shown in the table below:

Table 15-12. Port D Pins Alternate Functions

Port Pin	Alternate Function
PD7	OC2A (Timer/Counter2 Output Compare Match A Output) PCINT31 (Pin Change Interrupt 31)
PD6	ICP1 (Timer/Counter1 Input Capture Trigger) OC2B (Timer/Counter2 Output Compare Match B Output) PCINT30 (Pin Change Interrupt 30)
PD5	OC1A (Timer/Counter1 Output Compare Match A Output) PCINT29 (Pin Change Interrupt 29)
PD4	OC1B (Timer/Counter1 Output Compare Match B Output) XCK1 (USART1 External Clock Input/Output) PCINT28 (Pin Change Interrupt 28)
PD3	INT1 (External Interrupt1 Input) TXD1 (USART1 Transmit Pin) PCINT27 (Pin Change Interrupt 27)
PD2	INT0 (External Interrupt0 Input) RXD1 (USART1 Receive Pin) PCINT26 (Pin Change Interrupt 26)
PD1	TXD0 (USART0 Transmit Pin) PCINT25 (Pin Change Interrupt 25)
PD0	RXD0 (USART0 Receive Pin) PCINT24 (Pin Change Interrupt 24)

The alternate pin configuration is as follows:

- OC2A/PCINT31 – Port D, Bit 7
  - OC2A: Output Compare Match output. The PD7 pin can serve as an external output for the Timer/Counter2 Compare Match A. The PD7 pin has to be configured as an output (DDD7 set '1') to serve this function. The OC2A pin is also the output pin for the PWM mode timer function.
  - PCINT31: Pin Change Interrupt source 31. The PD7 pin can serve as an external interrupt source.
- ICP1/OC2B/PCINT30 – Port D, Bit 6
  - ICP1: Input Capture Pin 1. The PD6 pin can act as an input capture pin for Timer/Counter1.
  - OC2B: Output Compare Match B output. The PD6 pin can serve as an external output for the Timer/Counter2 Output Compare B. The pin has to be configured as an output (DDD6 set '1') to serve this function. The OC2B pin is also the output pin for the PWM mode timer function.

- PCINT30: Pin Change Interrupt source 30. The PD6 pin can serve as an external interrupt source.

- OC1A/PCINT29 – Port D, Bit 5
  - OC1A: Output Compare Match output. The PD5 pin can serve as an external output for the Timer/Counter1 Compare Match A. The PD5 pin has to be configured as an output (DDD5 set '1') to serve this function. The OC1A pin is also the output pin for the PWM mode timer function.
  - PCINT29: Pin Change Interrupt source 29. The PD5 pin can serve as an external interrupt source.
- OC1B/XCK1/PCINT28 – Port D, Bit 4
  - OC1B: Output Compare Match B output. The PD4 pin can serve as an external output for the Timer/Counter1 Output Compare B. The pin has to be configured as an output (DDD4 set '1') to serve this function. The OC1B pin is also the output pin for the PWM mode timer function.
  - XCK1: USART1 external clock.
  - PCINT28: Pin Change Interrupt source 28. The PD4 pin can serve as an external interrupt source.
- INT1/TXD1/PCINT27 – Port D, Bit 3
  - INT1: External Interrupt source 1. The PD3 pin can serve as an external interrupt source.
  - TXD1: Transmit Data (Data output pin for the USART). When the USART Transmitter is enabled, this pin is configured as an output regardless of the value of DDD3.
  - PCINT27: Pin Change Interrupt source 27. The PD3 pin can serve as an external interrupt source.
- INT0/RXD1/PCINT26 – Port D, Bit 2
  - INT0: External Interrupt source 0. The PD2 pin can serve as an external interrupt source.
  - RXD1: Receive Data (Data input pin for the USART1). When the USART1 Receiver is enabled this pin is configured as an input regardless of the value of DDD2. When the USART forces this pin to be an input, the pull-up can still be controlled by the PORTD2 bit.
  - PCINT26: Pin Change Interrupt source 26. The PD2 pin can serve as an external interrupt source.
- TXD0/PCINT25 – Port D, Bit 1
  - TXD0: Transmit Data (Data output pin for the USART0). When the USART0 Transmitter is enabled, this pin is configured as an output regardless of the value of DDD1.
  - PCINT25: Pin Change Interrupt source 25. The PD1 pin can serve as an external interrupt source.
- RXD0/PCINT24 – Port D, Bit 0
  - RXD0: Receive Data (Data input pin for the USART0). When the USART0 Receiver is enabled this pin is configured as an input regardless of the value of DDD0. When the USART forces this pin to be an input, the pull-up can still be controlled by the PORTD0 bit.
  - PCINT24: Pin Change Interrupt source 24. The PD0 pin can serve as an external interrupt source.

The tables below relate the alternate functions of Port D to the overriding signals shown in Figure 15-5.

Table 15-13. Overriding Signals for Alternate Functions PD[7:4]

Signal Name	PD7/OC2A/PCINT31	PD6/ICP1/OC2B/PCINT30	PD5/OC1A/PCINT29	PD4/OC1B/XCK1/PCINT28
PUOE	0	0	0	0
PUO	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	OC2A ENABLE	OC2B ENABLE	OC1A ENABLE	OC1B ENABLE
PVOV	OC2A	OC2B	OC1A	OC1B
DIEOE	PCINT31 • PCIE3	PCINT30 • PCIE3	PCINT29 • PCIE3	PCINT28 • PCIE3
DIEOV	1	1	1	1
DI	PCINT31 INPUT	ICP1 INPUT PCINT30 INPUT	PCINT29 INPUT	PCINT28 INPUT
AIO	-	-	–	–

Table 15-14. Overriding Signals for Alternate Functions in PD[3:0](1)

Signal Name	PD3/INT1/TXD1/PCINT27	PD2/INT0/RXD1/PCINT26	PD1/TXD0/PCINT25	PD0/RXD0/PCINT24
PUOE	TXEN1	RXEN1	TXEN0	RXEN0
PUO	0	PORTD2 • $\overline{PUD}$	0	PORTD0 • $\overline{PUD}$
DDOE	TXEN1	RXEN1	TXEN0	RXEN0
DDOV	1	0	1	0
PVOE	TXEN1	0	TXEN0	0
PVOV	TXD1	0	TXD0	0
DIEOE	INT1 ENABLE PCINT27 • PCIE3	INT2 ENABLE PCINT26 • PCIE3	PCINT25 • PCIE3	PCINT24 • PCIE3
DIEOV	1	1	1	1
DI	INT1 INPUT PCINT27 INPUT	INT0 INPUT RXD1 PCINT26 INPUT	PCINT25 INPUT	RXD0 PCINT24 INPUT
AIO	–	–	–	–

Note:

1. When enabled, the two-wire Serial Interface enables Slew-Rate controls on the output pins PD0 and PD1. This is not shown in this table. In addition, spike filters are connected between the AIO outputs shown in the port figure and the digital logic of the TWI module.

15.4. Register Description

15.4.1. MCU Control 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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: MCUCR
Offset: 0x55
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x35

Bit	7	6	5	4	3	2	1	0
	JTD	BODS	BODSE	PUD			IVSEL	IVCE
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bit 7 – JTD
When this bit is zero, the JTAG interface is enabled if the JTAGEN Fuse is programmed. If this bit is one, the JTAG interface is disabled. In order to avoid unintentional disabling or enabling of the JTAG interface, a timed sequence must be followed when changing this bit: The application software must write this bit to the desired value twice within four cycles to change its value. Note that this bit must not be altered when using the On-chip Debug system.

Bit 6 – BODS: BOD Sleep
The BODS bit must be written to '1' in order to turn off BOD during sleep. Writing to the BODS bit is controlled by a timed sequence and the enable bit BODSE. To disable BOD in relevant sleep modes, both BODS and BODSE must first be written to '1'. Then, BODS must be written to '1' and BODSE must be written to zero within four clock cycles.

The BODS bit is active three clock cycles after it is set. A sleep instruction must be executed while BODS is active in order to turn off the BOD for the actual sleep mode. The BODS bit is automatically cleared after three clock cycles.

Bit 5 – BODSE: BOD Sleep Enable
BODSE enables setting of BODS control bit, as explained in BODS bit description. BOD disable is controlled by a timed sequence.

Bit 4 – PUD: Pull-up Disable
When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01).

Bit 1 – IVSEL: Interrupt Vector Select
When the IVSEL bit is cleared (zero), the Interrupt Vectors are placed at the start of the Flash memory. When this bit is set (one), the Interrupt Vectors are moved to the beginning of the Boot Loader section of the Flash. The actual address of the start of the Boot Flash Section is determined by the BOOTSZ Fuses. To avoid unintentional changes of Interrupt Vector tables, a special write procedure must be followed to change the IVSEL bit:

1. Write the Interrupt Vector Change Enable (IVCE) bit to one.
2. Within four cycles, write the desired value to IVSEL while writing a zero to IVCE.

Interrupts will automatically be disabled while this sequence is executed. Interrupts are disabled in the cycle IVCE is set, and they remain disabled until after the instruction following the write to IVSEL. If IVSEL is not written, interrupts remain disabled for four cycles. The I-bit in the Status Register is unaffected by the automatic disabling.

Note: If Interrupt Vectors are placed in the Boot Loader section and Boot Lock bit BLB02 is programmed, interrupts are disabled while executing from the Application section. If Interrupt Vectors are placed in the Application section and Boot Lock bit BLB12 is programed, interrupts are disabled while executing from the Boot Loader section.

Bit 0 – IVCE: Interrupt Vector Change Enable

The IVCE bit must be written to logic one to enable change of the IVSEL bit. IVCE is cleared by hardware four cycles after it is written or when IVSEL is written. Setting the IVCE bit will disable interrupts, as explained in the IVSEL description above. See Code Example below.

Assembly Code Example

```
Move_interrupts:
; Get MCUCR
in    r16, MCUCR
mov   r17, r16
; Enable change of Interrupt Vectors
ori   r16, (1<<IVCE)
out   MCUCR, r16
; Move interrupts to Boot Flash section
ori   r17, (1<<IVSEL)
out   MCUCR, r17
ret
```

C Code Example

```
void Move_interrupts(void)
{
uchar temp;
/* GET MCUCR*/
temp = MCUCR;
/* Enable change of Interrupt Vectors */
MCUCR = temp|(1<<IVCE);
/* Move interrupts to Boot Flash section */
MCUCR = temp|(1<<IVSEL);
}
```


15.4.2. Port A Data 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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: PORTA
Offset: 0x22
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x02

Bit	7	6	5	4	3	2	1	0
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – PORTAn: Port A Data [n = 0:7]

15.4.3. Port A Data Direction 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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: DDRA
Offset: 0x21
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x01

Bit	7	6	5	4	3	2	1	0
	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DDRAn: Port A Data Direction [n = 7:0]

15.4.4. Port A Input Pins Address

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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: PINA
Offset: 0x20
Reset: N/A
Property: When addressing as I/O Register: address offset is 0x00

Bit	7	6	5	4	3	2	1	0
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0
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 – PINAn: Port A Input Pins Address [n = 7:0]
Writing to the pin register provides toggle functionality for IO. Refer to Toggling the Pin.

15.4.5. Port B Data 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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: PORTB
Offset: 0x25
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x05

Bit	7	6	5	4	3	2	1	0
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – PORTBn: Port B Data [n = 0:7]

15.4.6. Port B Data Direction 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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: DDRB
Offset: 0x24
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x04

Bit	7	6	5	4	3	2	1	0
	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DDRBn: Port B Data Direction [n = 7:0]

15.4.7. Port B Input Pins Address

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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: PINB
Offset: 0x23
Reset: N/A
Property: When addressing as I/O Register: address offset is 0x03

Bit	7	6	5	4	3	2	1	0
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
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 – PINBn: Port B Input Pins Address [n = 7:0]
Writing to the pin register provides toggle functionality for IO. Refer to Toggling the Pin.

15.4.8. Port C Data 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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: PORTC
Offset: 0x28
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x08

Bit	7	6	5	4	3	2	1	0
	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – PORTCn: Port C Data [n = 7:0]

15.4.9. Port C Data Direction 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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: DDRC
Offset: 0x27
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x07

Bit	7	6	5	4	3	2	1	0
	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DDRCn: Port C Data Direction [n = 7:0]

15.4.10. Port C Input Pins Address

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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: PINC
Offset: 0x26
Reset: N/A
Property: When addressing as I/O Register: address offset is 0x06

Bit	7	6	5	4	3	2	1	0
	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	x	x	x	x	x	x	x

Bits 7:0 – PINCn: Port C Input Pins Address [n = 7:0]
Writing to the pin register provides toggle functionality for IO. Refer to Toggling the Pin.

15.4.11. Port D Data 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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: PORTD
Offset: 0x2B
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x0B

Bit	7	6	5	4	3	2	1	0
	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – PORTDn: Port D Data [n = 7:0]

15.4.12. Port D Data Direction 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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: DDRD
Offset: 0x2A
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x0A

Bit	7	6	5	4	3	2	1	0
	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – DDRDn: Port D Data Direction [n = 7:0]

15.4.13. Port D Input Pins Address

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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: PIND
Offset: 0x29
Reset: N/A
Property: When addressing as I/O Register: address offset is 0x09

Bit	7	6	5	4	3	2	1	0
	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
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 – PINDn: Port D Input Pins Address [n = 7:0]
Writing to the pin register provides toggle functionality for IO. Refer to Toggling the Pin.

16. TC0 - 8-bit Timer/Counter0 with PWM

Related Links
Timer/Counter0 and Timer/Counter1 Prescalers on page 185

16.1. Features

- Two independent Output Compare Units
- Double Buffered Output Compare Registers
- Clear Timer on Compare Match (Auto Reload)
- Glitch free, phase correct Pulse Width Modulator (PWM)
- Variable PWM period
- Frequency generator
- Three independent interrupt sources (TOV0, OCF0A, and OCF0B)

16.2. Overview

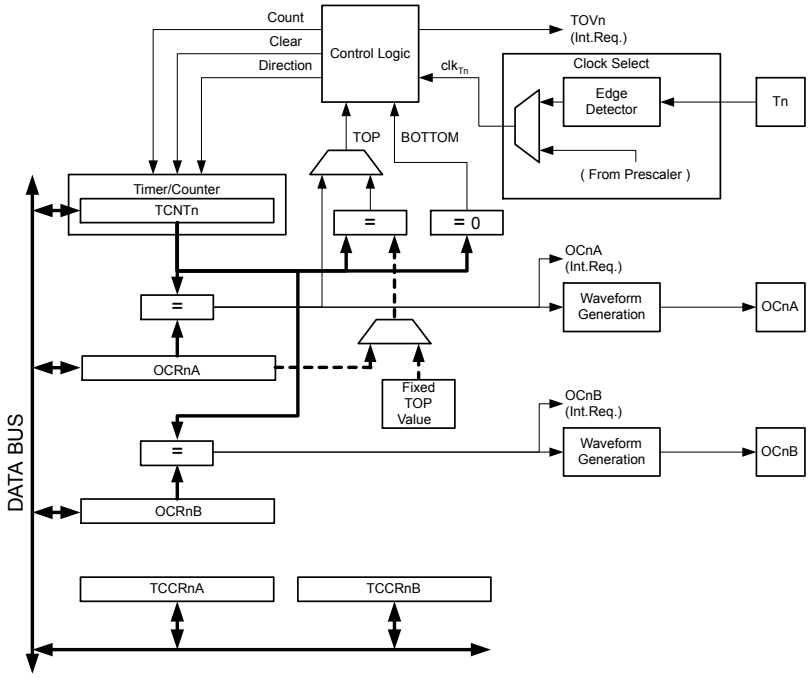
Timer/Counter0 (TC0) is a general purpose 8-bit Timer/Counter module, with two independent Output Compare Units, and PWM support. It allows accurate program execution timing (event management) and wave generation.

A simplified block diagram of the 8-bit Timer/Counter is shown below. CPU accessible I/O Registers, including I/O bits and I/O pins, are shown in bold. The device-specific I/O Register and bit locations are listed in the Register Description. For the actual placement of I/O pins, refer to the pinout diagram.

The TC0 is enabled by writing the PRTIM0 bit in ”Minimizing Power Consumption” to '0'.

The TC0 is enabled when the PRTIM0 bit in the Power Reduction Register (0.PRTIM0) is written to '1'.

Figure 16-1. 8-bit Timer/Counter Block Diagram



16.2.1. Definitions

Many register and bit references in this section are written in general form:

- n=0 represents the Timer/Counter number
- x=A,B represents the Output Compare Unit A or B

However, when using the register or bit definitions in a program, the precise form must be used, i.e., TCNT0 for accessing Timer/Counter0 counter value.

The following definitions are used throughout the section:

Table 16-1. Definitions

Constant	Description
BOTTOM	The counter reaches the BOTTOM when it becomes zero (0x00 for 8-bit counters, or 0x0000 for 16-bit counters).
MAX	The counter reaches its Maximum when it becomes 0xFF (decimal 255, for 8-bit counters) or 0xFFFF (decimal 65535, for 16-bit counters).
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 MAX or the value stored in the OCR0A Register. The assignment is dependent on the mode of operation.

16.2.2. Registers

The Timer/Counter 0 register (TCNT0) and Output Compare TC0x registers (OCR0x) are 8-bit registers. Interrupt request (abbreviated to Int.Req. in the block diagram) signals are all visible in the Timer Interrupt Flag Register 0 (TIFR0). All interrupts are individually masked with the Timer Interrupt Mask Register 0 (TIMSK0). TIFR0 and TIMSK0 are not shown in the figure.

The TC can be clocked internally, via the prescaler, or by an external clock source on the T0 pin. The Clock Select logic block controls which clock source and edge is used by the Timer/Counter to increment (or decrement) its value. The TC is inactive when no clock source is selected. The output from the Clock Select logic is referred to as the timer clock (clk_{T0}).

The double buffered Output Compare Registers (OCR0A and OCR0B) 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 (OC0A and OC0B). See Output Compare Unit for details. The compare match event will also set the Compare Flag (OCF0A or OCF0B) which can be used to generate an Output Compare interrupt request.

Related Links

Timer/Counter 0, 1 Prescalers on page 185

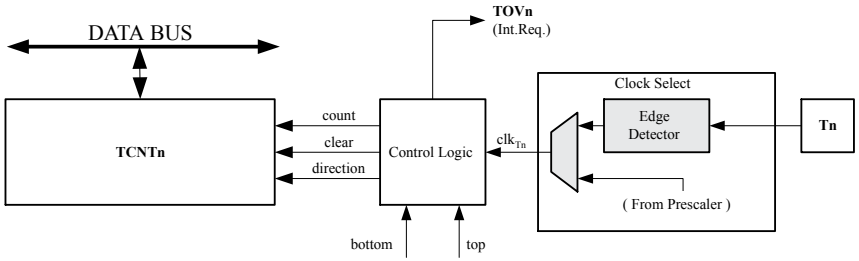
16.3. Timer/Counter Clock Sources

The TC can be clocked by an internal or an external clock source. The clock source is selected by writing to the Clock Select (CS0[2:0]) bits in the Timer/Counter Control Register (TCCR0B).

16.4. 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.

Figure 16-2. Counter Unit Block Diagram



Note: The “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).

Table 16-2. Signal description (internal signals)

Signal Name	Description
count	Increment or decrement TCNT0 by 1.
direction	Select between increment and decrement.
clear	Clear TCNT0 (set all bits to zero).
clk_{Tn}	Timer/Counter clock, referred to as clk_{T0} in the following.
top	Signalize that TCNT0 has reached maximum value.
bottom	Signalize that TCNT0 has reached minimum value (zero).

Depending of the mode of operation used, the counter is cleared, incremented, or decremented at each timer clock (clk_{T0}). clk_{T0} can be generated from an external or internal clock source, selected by the Clock Select bits (CS0[2:0]). When no clock source is selected (CS0=0x0) the timer is stopped. However, the TCNT0 value can be accessed by the CPU, regardless of whether clk_{T0} is present or not. A CPU write overrides (has priority over) all counter clear or count operations.

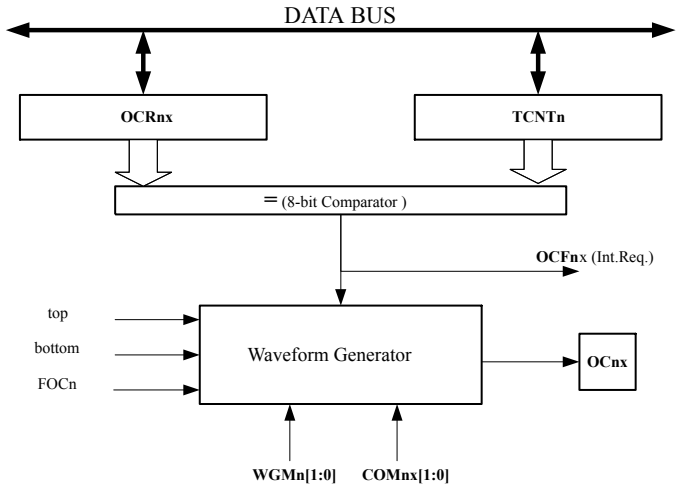
The counting sequence is determined by the setting of the WGM01 and WGM00 bits located in the Timer/Counter Control Register (TCCR0A) and the WGM02 bit located in the Timer/Counter Control Register B (TCCR0B). There are close connections between how the counter behaves (counts) and how waveforms are generated on the Output Compare outputs OC0A and OC0B. For more details about advanced counting sequences and waveform generation, see Modes of Operation.

The Timer/Counter Overflow Flag (TOV0) is set according to the mode of operation selected by the WGM0[2:0] bits. TOV0 can be used for generating a CPU interrupt.

16.5. Output Compare Unit

The 8-bit comparator continuously compares TCNT0 with the Output Compare Registers (OCR0A and OCR0B). Whenever TCNT0 equals OCR0A or OCR0B, the comparator signals a match. A match will set the Output Compare Flag (OCF0A or OCF0B) at the next timer clock cycle. If the corresponding interrupt is enabled, the Output Compare Flag generates an Output Compare interrupt. The Output Compare Flag is automatically cleared when the interrupt is executed. Alternatively, the flag can be cleared by software by writing a '1' to its I/O bit location. The Waveform Generator uses the match signal to generate an output according to operating mode set by the WGM02, WGM01, and WGM00 bits and Compare Output mode (COM0x[1:0]) bits. The max and bottom signals are used by the Waveform Generator for handling the special cases of the extreme values in some modes of operation.

Figure 16-3. Output Compare Unit, Block Diagram



Note: The “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).

The OCR0x Registers are double buffered when using any of the Pulse Width Modulation (PWM) modes. When double buffering is enabled, the CPU has access to the OCR0x Buffer Register. The double buffering synchronizes the update of the OCR0x Compare Registers to either top or bottom of the counting sequence. The synchronization prevents the occurrence of odd-length, non-symmetrical PWM pulses, thereby making the output glitch-free.

The double buffering is disabled for the normal and Clear Timer on Compare (CTC) modes of operation, and the CPU will access the OCR0x directly.

16.5.1. Force Output Compare

In non-PWM waveform generation modes, the match output of the comparator can be forced by writing a '1' to the Force Output Compare (TCCR0C.FOC0x) bit. Forcing compare match will not set the OCF0x Flag or reload/clear the timer, but the OC0x pin will be updated as if a real compare match had occurred (the TCCR0A.COM0x[1:0] bits define whether the OC0x pin is set, cleared or toggled).

16.5.2. Compare Match Blocking by TCNT0 Write

All CPU write operations to the TCNT0 Register will block any compare match that occur in the next timer clock cycle, even when the timer is stopped. This feature allows OCR0x to be initialized to the same value as TCNT0 without triggering an interrupt when the Timer/Counter clock is enabled.

16.5.3. Using the Output Compare Unit

Since writing TCNT0 in any mode of operation will block all compare matches for one timer clock cycle, there are risks involved when changing TCNT0 when using the Output Compare Unit, independently of whether the Timer/Counter is running or not. If the value written to TCNT0 equals the OCR0x value, the compare match will be missed, resulting in incorrect waveform generation. Similarly, do not write the TCNT0 value equal to BOTTOM when the counter is down counting.

The setup of the OC0x should be performed before setting the Data Direction Register for the port pin to output. The easiest way of setting the OC0x value is to use the Force Output Compare (FOC0x) strobe bits in Normal mode. The OC0x Registers keep their values even when changing between Waveform Generation modes.

Be aware that the TCCR0A.COM0x[1:0] bits are not double buffered together with the compare value. Changing the TCCR0A.COM0x[1:0] bits will take effect immediately.

16.6. Compare Match Output Unit

The Compare Output mode bits in the Timer/Counter Control Register A (TCCR0A.COM0x) have two functions:

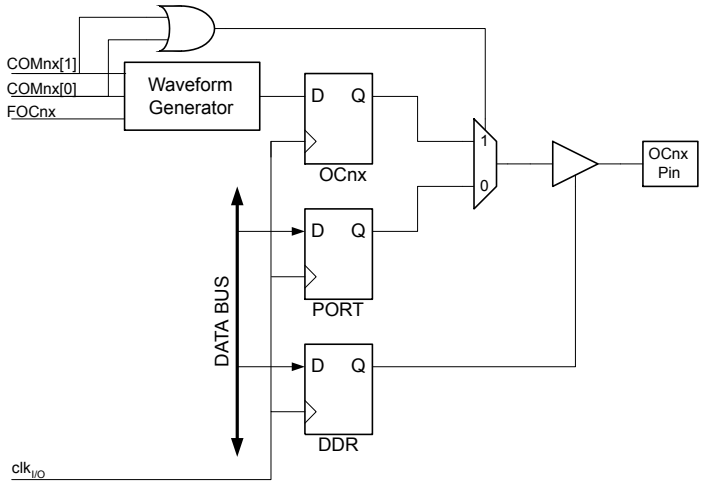
- The Waveform Generator uses the COM0x bits for defining the Output Compare (OC0x) register state at the next compare match.
- The COM0x bits control the OC0x pin output source

The figure below shows a simplified schematic of the logic affected by COM0x. The I/O Registers, I/O bits, and I/O pins in the figure are shown in bold. Only the parts of the general I/O port control registers that are affected by the COM0x bits are shown, namely PORT and DDR.

On system reset the OC0x Register is reset to 0x00.

Note: 'OC0x state' is always referring to internal OC0x *registers*, not the OC0x *pin*.

Figure 16-4. Compare Match Output Unit, Schematic



Note: The “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).

The general I/O port function is overridden by the Output Compare (OC0x) from the Waveform Generator if either of the COM0x[1:0] bits are set. However, the OC0x pin direction (input or output) is still controlled by the Data Direction Register (DDR) for the port pin. In the Data Direction Register, the bit for the OC1x pin (DDR.OC0x) must be set as output before the OC0x value is visible on the pin. The port override function is independent of the Waveform Generation mode.

The design of the Output Compare pin logic allows initialization of the OC0x register state before the output is enabled. Some TCCR0A.COM0x[1:0] bit settings are reserved for certain modes of operation.

The TCCR0A.COM0x[1:0] bits have no effect on the Input Capture unit.

Related Links

Register Description on page 139

16.6.1. Compare Output Mode and Waveform Generation

The Waveform Generator uses the TCCR0A.COM0x[1:0] bits differently in Normal, CTC, and PWM modes. For all modes, setting the TCCR0A.COM0x[1:0]=0x0 tells the Waveform Generator that no action on the OC0x Register is to be performed on the next compare match. Refer also to the descriptions of the output modes.

A change of the TCCR0A.COM0x[1:0] bits state will have effect at the first compare match after the bits are written. For non-PWM modes, the action can be forced to have immediate effect by using the TCCR0C.FOC0x strobe bits.

16.7. Modes of Operation

The mode of operation determines the behavior of the Timer/Counter and the Output Compare pins. It is defined by the combination of the Waveform Generation mode bits and Compare Output mode (TCCR0A.WGM0[2:0]) bits in the Timer/Counter control Registers A and B (TCCR0A.COM0x[1:0]). The Compare Output mode bits do not affect the counting sequence, while the Waveform Generation mode bits do. The COM0x[1:0] bits control whether the PWM output generated should be inverted or not (inverted or non-inverted PWM). For non-PWM modes the COM0x[1:0] bits control whether the output should be set, cleared, or toggled at a compare match (See previous section *Compare Match Output Unit*).

For detailed timing information refer to the following section *Timer/Counter Timing Diagrams*.

Related Links

Compare Match Output Unit on page 193
Timer/Counter Timing Diagrams on page 137

16.7.1. Normal Mode

The simplest mode of operation is the Normal mode (WGM0[2:0] = 0x0). In this mode the counting direction is always up (incrementing), and no counter clear is performed. The counter simply overruns when it passes its maximum 8-bit value (TOP=0xFF) and then restarts from the bottom (0x00). In Normal mode operation, the Timer/Counter Overflow Flag (TOV0) will be set in the same clock cycle in which the TCNT0 becomes zero. In this case, the TOV0 Flag behaves like a ninth bit, except that it is only set, not cleared. However, combined with the timer overflow interrupt that automatically clears the TOV0 Flag, the timer resolution can be increased by software. There are no special cases to consider in the Normal mode, a new counter value can be written anytime.

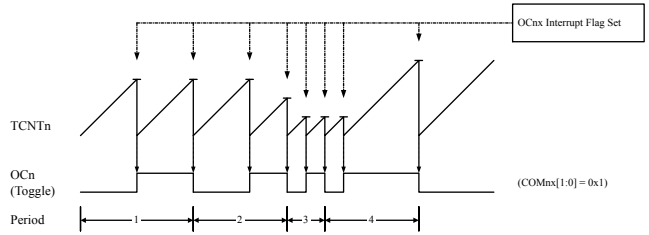
The Output Compare unit can be used to generate interrupts at some given time. Using the Output Compare to generate waveforms in Normal mode is not recommended, since this will occupy too much of the CPU time.

16.7.2. Clear Timer on Compare Match (CTC) Mode

In Clear Timer on Compare or CTC mode (WGM0[2:0]=0x2), the OCR0A Register is used to manipulate the counter resolution: the counter is cleared to ZERO when the counter value (TCNT0) matches the OCR0A. The OCR0A defines the top value for the counter, hence also its resolution. This mode allows greater control of the compare match output frequency. It also simplifies the counting of external events.

The timing diagram for the CTC mode is shown below. The counter value (TCNT0) increases until a compare match occurs between TCNT0 and OCR0A, and then counter (TCNT0) is cleared.

Figure 16-5. CTC Mode, Timing Diagram



An interrupt can be generated each time the counter value reaches the TOP value by setting the OCF0A Flag. If the interrupt is enabled, the interrupt handler routine can be used for updating the TOP value.

Note: Changing TOP to a value close to BOTTOM while the counter is running must be done with care, since the CTC mode does not provide double buffering. If the new value written to OCR0A is lower than the current value of TCNT0, the counter will miss the compare match. The counter will then count to its maximum value (0xFF for a 8-bit counter, 0xFFFF for a 16-bit counter) and wrap around starting at 0x00 before the compare match will occur.

For generating a waveform output in CTC mode, the OC0A output can be set to toggle its logical level on each compare match by writing the two least significant Compare Output mode bits in the Timer/Counter Control Register A Control to toggle mode (TCCR0A.COM0A[1:0]=0x1). The OC0A value will only be visible on the port pin unless the data direction for the pin is set to output. The waveform generated will have a maximum frequency of $f_{OC0} = f_{clk\_I/O}/2$ when OCR0A is written to 0x00. The waveform frequency is defined by the following equation:

$$f_{\mathrm{OCnx}} = \frac{f_{\mathrm{clk\_I/O}}}{2 \cdot N \cdot (1 + \mathrm{OCRnx})}$$

N represents the prescaler factor (1, 8, 64, 256, or 1024).

As for the Normal mode of operation, the Timer/Counter Overflow Flag TOV0 is set in the same clock cycle that the counter wraps from MAX to 0x00.

16.7.3. Fast PWM Mode

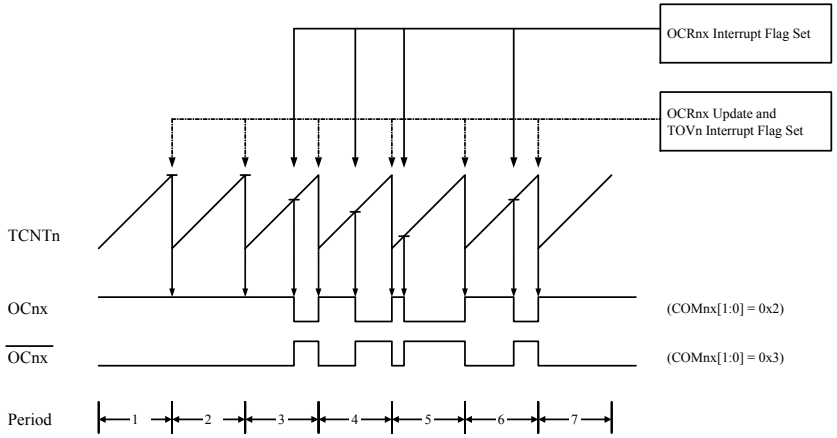
The Fast Pulse Width Modulation or Fast PWM modes (WGM0[2:0]=0x3 or WGM0[2:0]=0x7) provide a high frequency PWM waveform generation option. The Fast PWM modes differ from the other PWM options by their single-slope operation. The counter counts from BOTTOM to TOP, then restarts from BOTTOM. TOP is defined as 0xFF when WGM0[2:0]=0x3. TOP is defined as OCR0A when WGM0[2:0]=0x7.

In non-inverting Compare Output mode, the Output Compare register (OC0x) is cleared on the compare match between TCNT0 and OCR0x, and set at BOTTOM. In inverting Compare Output mode, the output is set on compare match and cleared at BOTTOM. Due to the single-slope operation, the operating

frequency of the Fast PWM mode can be twice as high as the phase correct PWM modes, which use dual-slope operation. This high frequency makes the Fast PWM mode well suited for power regulation, rectification, and DAC applications. High frequency allows physically small sized external components (coils, capacitors), and therefore reduces total system cost.

In Fast PWM mode, the counter is incremented until the counter value matches the TOP value. The counter is then cleared at the following timer clock cycle. The timing diagram for the Fast PWM mode is shown below. The TCNT0 value is in the timing diagram shown as a histogram for illustrating the single-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal lines on the TCNT0 slopes mark compare matches between OCR0x and TCNT0.

Figure 16-6. Fast PWM Mode, Timing Diagram



The Timer/Counter Overflow Flag (TOV0) is set each time the counter reaches TOP. If the interrupt is enabled, the interrupt handler routine can be used for updating the compare value.

In Fast PWM mode, the compare unit allows generation of PWM waveforms on the OC0x pins. Writing the TCCR0A.COM0x[1:0] bits to 0x2 will produce a non-inverted PWM; TCCR0A.COM0x[1:0]=0x3 will produce an inverted PWM output. Writing the TCCR0A.COM0A[1:0] bits to 0x1 allows the OC0A pin to toggle on Compare Matches if the TCCRnB.WGMn2 bit is set. This option is not available for the OC0B pin. The actual OC0x value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by setting (or clearing) the OC0x Register at the compare match between OCR0x and TCNT0, and clearing (or setting) the OC0x Register at the timer clock cycle the counter is cleared (changes from TOP to BOTTOM).

The PWM frequency for the output can be calculated by the following equation:

$$f_{\mathrm{OCnxPWM}} = \frac{f_{\mathrm{clk\_I/O}}}{N \cdot 256}$$

N represents the prescale divider (1, 8, 64, 256, or 1024).

The extreme values for the OCR0A register represents special cases for PWM waveform output in the Fast PWM mode: If OCR0A is written equal to BOTTOM, the output will be a narrow spike for each MAX +1 timer clock cycle. Writing OCR0A=MAX will result in a constantly high or low output (depending on the polarity of the output set by the COM0A[1:0] bits.)

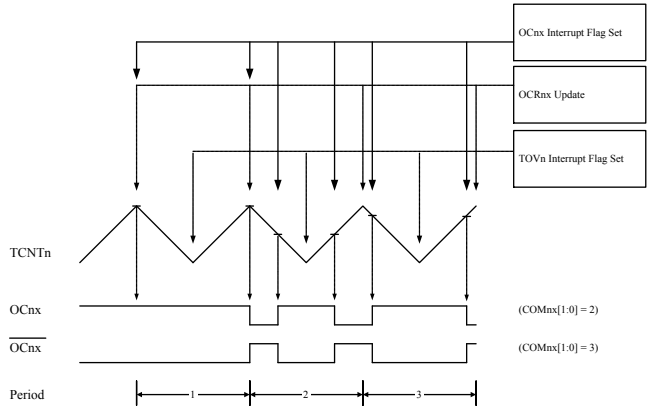
A frequency waveform output with 50% duty cycle can be achieved in Fast PWM mode by selecting OC0x to toggle its logical level on each compare match (COM0x[1:0]=0x1). The waveform generated will have a maximum frequency of $f_{OC0} = f_{clk\_I/O}/2$ when OCR0A=0x00. This feature is similar to the OC0A toggle in CTC mode, except double buffering of the Output Compare unit is enabled in the Fast PWM mode.

16.7.4. Phase Correct PWM Mode

The Phase Correct PWM mode (WGM0[2:0]=0x1 or WGM0[2:0]=0x5) provides a high resolution, phase correct PWM waveform generation. The Phase Correct PWM mode is based on dual-slope operation: The counter counts repeatedly from BOTTOM to TOP, and then from TOP to BOTTOM. When WGM0[2:0]=0x1 TOP is defined as 0xFF. When WGM0[2:0]=0x5, TOP is defined as OCR0A. In non-inverting Compare Output mode, the Output Compare (OC0x) bit is cleared on compare match between TCNT0 and OCR0x while up-counting, and OC0x is set on the compare match while down-counting. In inverting Output Compare mode, the operation is inverted. The dual-slope operation has a lower maximum operation frequency than single slope operation. Due to the symmetric feature of the dual-slope PWM modes, these modes are preferred for motor control applications.

In Phase Correct PWM mode the counter is incremented until the counter value matches TOP. When the counter reaches TOP, it changes the count direction. The TCNT0 value will be equal to TOP for one timer clock cycle. The timing diagram for the Phase Correct PWM mode is shown below. The TCNT0 value is shown as a histogram for illustrating the dual-slope operation. The diagram includes non-inverted and inverted PWM outputs. The small horizontal line marks on the TCNT0 slopes represent compare matches between OCR0x and TCNT0.

Figure 16-7. Phase Correct PWM Mode, Timing Diagram



Note: The “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).

The Timer/Counter Overflow Flag (TOV0) is set each time the counter reaches BOTTOM. The Interrupt Flag can be used to generate an interrupt each time the counter reaches the BOTTOM value.

In Phase Correct PWM mode, the compare unit allows generation of PWM waveforms on the OC0x pin. Writing the COM0x[1:0] bits to 0x2 will produce a non-inverted PWM. An inverted PWM output can be generated by writing COM0x[1:0]=0x3. Setting the Compare Match Output A Mode bit to '1' (TCCR0A.COM0A0) allows the OC0A pin to toggle on Compare Matches if the TCCR0B.WGM02 bit is set. This option is not available for the OC0B pin. The actual OC0x value will only be visible on the port pin if the data direction for the port pin is set as output. The PWM waveform is generated by clearing (or setting) the OC0x Register at the compare match between OCR0x and TCNT0 when the counter increments, and setting (or clearing) the OC0x Register at compare match between OCR0x and TCNT0 when the counter decrements. The PWM frequency for the output when using Phase Correct PWM can be calculated by:

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

N represents the prescaler factor (1, 8, 64, 256, or 1024).

The extreme values for the OCR0A Register represent special cases when generating a PWM waveform output in the Phase Correct PWM mode: If the OCR0A register is written equal to BOTTOM, the output will be continuously low. If OCR0A is written to MAX, the output will be continuously high for non-inverted PWM mode. For inverted PWM the output will have the opposite logic values.

At the very start of period 2 in the timing diagram above, OC0x has a transition from high to low even though there is no Compare Match. This transition serves to guarantee symmetry around BOTTOM. There are two cases that give a transition without Compare Match:

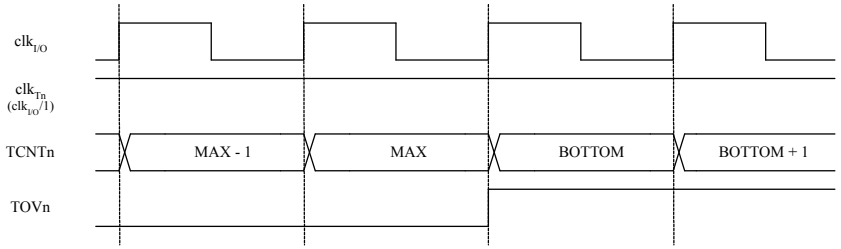
- OCR0x changes its value from MAX, as in the timing diagram. When the OCR0A value is MAX, the OC0 pin value is the same as the result of a down-counting Compare Match. To ensure symmetry around BOTTOM the OC0x value at MAX must correspond to the result of an up-counting Compare Match.
- The timer starts up-counting from a value higher than the one in OCR0x, and for that reason misses the Compare Match and consequently, the OC0x does not undergo the change that would have happened on the way up.

16.8. Timer/Counter Timing Diagrams

The Timer/Counter is a synchronous design and the timer clock (clk_{T0}) is therefore shown as a clock enable signal in the following figures. If the given instance of the TC0 supports an asynchronous mode, $clk_{I/O}$ should be replaced by the TC oscillator clock.

The figures include information on when interrupt flags are set. The first figure below illustrates timing data for basic Timer/Counter operation close to the MAX value in all modes other than Phase Correct PWM mode.

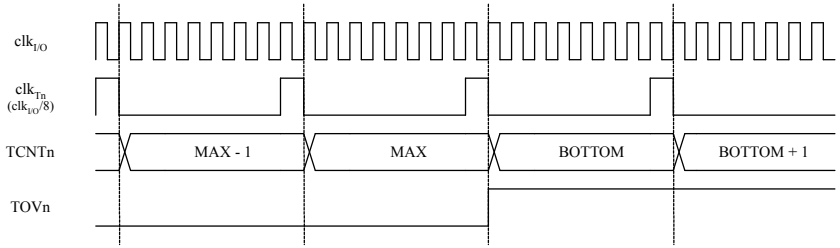
Figure 16-8. Timer/Counter Timing Diagram, no Prescaling



Note: The “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).

The next figure shows the same timing data, but with the prescaler enabled.

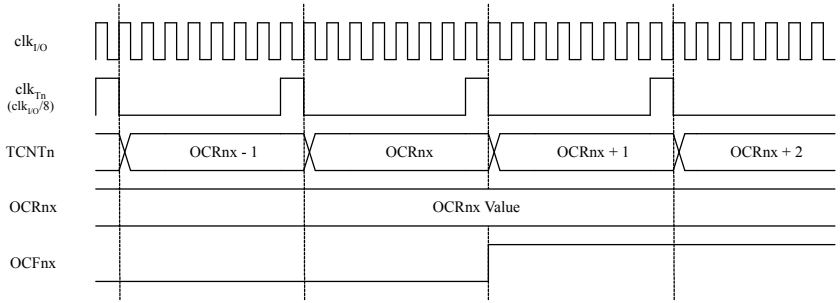
Figure 16-9. Timer/Counter Timing Diagram, with Prescaler ($f_{clk\_I/O}/8$)



Note: The “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).

The next figure shows the setting of OCF0B in all modes and OCF0A in all modes (except CTC mode and PWM mode where OCR0A is TOP).

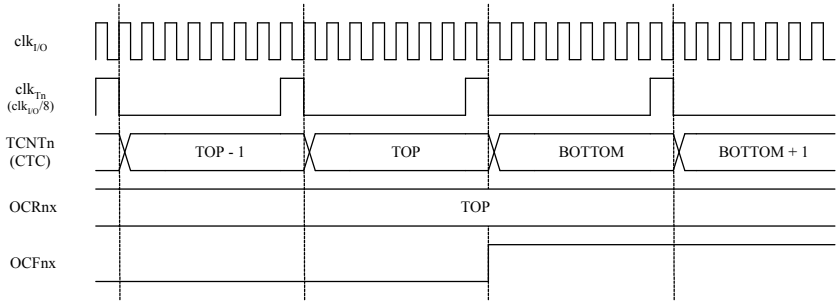
Figure 16-10. Timer/Counter Timing Diagram, Setting of OCF0x, with Prescaler ($f_{clk\_I/O}/8$)



Note: The “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).

The next figure shows the setting of OCF0A and the clearing of TCNT0 in CTC mode and fast PWM mode where OCR0A is TOP.

Figure 16-11. Timer/Counter Timing Diagram, Clear Timer on Compare Match mode, with Prescaler ($f_{clk\_I/O}/8$)



Note: The “n” in the register and bit names indicates the device number (n = 0 for Timer/Counter 0), and the “x” indicates Output Compare unit (A/B).

16.9. Register Description

16.9.1. TC0 Control Register A

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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: TCCR0A
Offset: 0x44
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x24

Bit	7	6	5	4	3	2	1	0
	COM0A1	COM0A0	COM0B1	COM0B0			WGM01	WGM00
Access	R/W	R/W	R/W	R/W			R/W	R/W
Reset	0	0	0	0			0	0

Bits 7:6 – COM0An: Compare Output Mode for Channel A [n = 1:0]
These bits control the Output Compare pin (OC0A) behavior. If one or both of the COM0A[1:0] bits are set, the OC0A output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC0A pin must be set in order to enable the output driver.

When OC0A is connected to the pin, the function of the COM0A[1:0] bits depends on the WGM0[2:0] bit setting. The table below shows the COM0A[1:0] bit functionality when the WGM0[2:0] bits are set to a normal or CTC mode (non- PWM).

Table 16-3. Compare Output Mode, non-PWM

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC0A on Compare Match.
1	0	Clear OC0A on Compare Match.
1	1	Set OC0A on Compare Match .

The table below shows the COM0A[1:0] bit functionality when the WGM0[1:0] bits are set to fast PWM mode.

Table 16-4. Compare Output Mode, Fast PWM(1)

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal Port Operation, OC0A Disconnected WGM02 = 1: Toggle OC0A on Compare Match
1	0	Clear OC0A on Compare Match, set OC0A at BOTTOM (non-inverting mode)
1	1	Set OC0A on Compare Match, clear OC0A at BOTTOM (inverting mode)

Note:
1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case the compare match is ignored, but the set or clear is done at BOTTOM. Refer to Fast PWM Mode for details.

The table below shows the COM0A[1:0] bit functionality when the WGM0[2:0] bits are set to phase correct PWM mode.

Table 16-5. Compare Output Mode, Phase Correct PWM Mode(1)

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	WGM02 = 0: Normal Port Operation, OC0A Disconnected. WGM02 = 1: Toggle OC0A on Compare Match.
1	0	Clear OC0A on Compare Match when up-counting. Set OC0A on Compare Match when down-counting.
1	1	Set OC0A on Compare Match when up-counting. Clear OC0A on Compare Match when down-counting.

Note:
1. A special case occurs when OCR0A equals TOP and COM0A1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. Refer to Phase Correct PWM Mode for details.

Bits 5:4 – COM0Bn: Compare Output Mode for Channel B [n = 1:0]
These bits control the Output Compare pin (OC0B) behavior. If one or both of the COM0B[1:0] bits are set, the OC0B output overrides the normal port functionality of the I/O pin it is connected to. However, note that the Data Direction Register (DDR) bit corresponding to the OC0B pin must be set in order to enable the output driver.

When OC0B is connected to the pin, the function of the COM0B[1:0] bits depends on the WGM0[2:0] bit setting. The table shows the COM0B[1:0] bit functionality when the WGM0[2:0] bits are set to a normal or CTC mode (non- PWM).

Table 16-6. Compare Output Mode, non-PWM

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Toggle OC0B on Compare Match.
1	0	Clear OC0B on Compare Match.
1	1	Set OC0B on Compare Match.

The table below shows the COM0B[1:0] bit functionality when the WGM0[2:0] bits are set to fast PWM mode.

Table 16-7. Compare Output Mode, Fast PWM(1)

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Reserved
1	0	Clear OC0B on Compare Match, set OC0B at BOTTOM, (non-inverting mode)
1	1	Set OC0B on Compare Match, clear OC0B at BOTTOM, (inverting mode)

Note:

1. A special case occurs when OCR0B equals TOP and COM0B1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. Refer to Fast PWM Mode for details.

The table below shows the COM0B[1:0] bit functionality when the WGM0[2:0] bits are set to phase correct PWM mode.

Table 16-8. Compare Output Mode, Phase Correct PWM Mode(1)

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Reserved
1	0	Clear OC0B on Compare Match when up-counting. Set OC0B on Compare Match when down-counting.
1	1	Set OC0B on Compare Match when up-counting. Clear OC0B on Compare Match when down-counting.

Note:

1. A special case occurs when OCR0B equals TOP and COM0B1 is set. In this case, the Compare Match is ignored, but the set or clear is done at TOP. Refer to Phase Correct PWM Mode for details.

Bits 1:0 – WGM0n: Waveform Generation Mode [n = 1:0]
Combined with the WGM02 bit found in the TCCR0B Register, these bits control the counting sequence of the counter, the source for maximum (TOP) counter value, and what type of waveform generation to be used. Modes of operation supported by the Timer/Counter unit are: Normal mode (counter), Clear Timer on Compare Match (CTC) mode, and two types of Pulse Width Modulation (PWM) modes (see Modes of Operation).

Table 16-9. Waveform Generation Mode Bit Description

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCR0x at	TOV Flag Set on(1)(2)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	-	-	-
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	-	-	-
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Note:

1. MAX = 0xFF
2. BOTTOM = 0x00

16.9.2. TC0 Control Register B

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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: TCCR0B
Offset: 0x45
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x25

Bit	7	6	5	4	3	2	1	0
	FOC0A	FOC0B			WGM02	CS0[2:0]		
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

Bit 7 – FOC0A: Force Output Compare A
The FOC0A 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 TCCR0B is written when operating in PWM mode. When writing a logical one to the FOC0A bit, an immediate Compare Match is forced on the Waveform Generation unit. The OC0A output is changed according to its COM0A[1:0] bits setting. The FOC0A bit is implemented as a strobe. Therefore it is the value present in the COM0A[1:0] bits that determines the effect of the forced compare.

A FOC0A strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR0A as TOP.

The FOC0A bit is always read as zero.

Bit 6 – FOC0B: Force Output Compare B
The FOC0B 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 TCCR0B is written when operating in PWM mode. When writing a logical one to the FOC0B bit, an immediate Compare Match is forced on the Waveform Generation unit. The OC0B output is changed according to its COM0B[1:0] bits setting. The FOC0B bit is implemented as a strobe. Therefore it is the value present in the COM0B[1:0] bits that determines the effect of the forced compare.

A FOC0B strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR0B as TOP.

The FOC0B bit is always read as zero.

Bit 3 – WGM02: Waveform Generation Mode
Refer to TCCR0A.

Bits 2:0 – CS0[2:0]: Clock Select 0 [n = 0..2]
The three Clock Select bits select the clock source to be used by the Timer/Counter.

Table 16-10. Clock Select Bit Description

CA02	CA01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	clkI/O/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

If external pin modes are used for the Timer/Counter0, transitions on the T0 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

16.9.3. TC0 Interrupt Mask Register

Name: TIMSK0
Offset: 0x6E
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
						OCIEB	OCIEA	TOIE
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 2 – OCIEB: Timer/Counter0, Output Compare B Match Interrupt Enable
When the OCIE0B bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter Compare Match B interrupt is enabled. The corresponding interrupt is executed if a Compare Match in Timer/Counter occurs, i.e., when the OCF0B bit is set in TIFR0.

Bit 1 – OCIEA: Timer/Counter0, Output Compare A Match Interrupt Enable
When the OCIE0A bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter0 Compare Match A interrupt is enabled. The corresponding interrupt is executed if a Compare Match in Timer/Counter0 occurs, i.e., when the OCF0A bit is set in TIFR0.

Bit 0 – TOIE: Timer/Counter0, Overflow Interrupt Enable
When the TOIE0 bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in TIFR0.

16.9.4. General Timer/Counter Control 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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: GTCCR
Offset: 0x43
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x23

Bit	7	6	5	4	3	2	1	0
	TSM						PSRASY	PSRSYNC
Access	R/W						R/W	R/W
Reset	0						0	0

Bit 7 – TSM: Timer/Counter Synchronization Mode
Writing the TSM bit to one activates the Timer/Counter Synchronization mode. In this mode, the value that is written to the PSRASY and PSRSYNC bits is kept, hence keeping the corresponding prescaler reset signals asserted. This ensures that the corresponding Timer/Counters are halted and can be configured to the same value without the risk of one of them advancing during configuration. When the TSM bit is written to zero, the PSRASY and PSRSYNC bits are cleared by hardware, and the Timer/Counters start counting simultaneously.

Bit 1 – PSRASY: Prescaler Reset Timer/Counter2
When this bit is one, the Timer/Counter2 prescaler will be reset. This bit is normally cleared immediately by hardware. If the bit is written when Timer/Counter2 is operating in asynchronous mode, the bit will remain one until the prescaler has been reset. The bit will not be cleared by hardware if the TSM bit is set.

Bit 0 – PSRSYNC: Prescaler Reset
When this bit is one, Timer/Counter1 and Timer/Counter0 prescaler will be Reset. This bit is normally cleared immediately by hardware, except if the TSM bit is set. Note that Timer/Counter1 and Timer/Counter0 share the same prescaler and a reset of this prescaler will affect both timers.

16.9.5. TC0 Counter Value 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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: TCNT0
Offset: 0x46
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x26

Bit	7	6	5	4	3	2	1	0
	TCNT0[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – TCNT0[7:0]: TC0 Counter Value
The Timer/Counter Register gives direct access, both for read and write operations, to the Timer/Counter unit 8-bit counter. Writing to the TCNT0 Register blocks (removes) the Compare Match on the following timer clock. Modifying the counter (TCNT0) while the counter is running, introduces a risk of missing a Compare Match between TCNT0 and the OCR0x Registers.

16.9.6. TC0 Output Compare Register A

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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: OCR0A
Offset: 0x47
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x27

Bit	7	6	5	4	3	2	1	0
	OCR0A[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – OCR0A[7:0]: Output Compare 0 A
The Output Compare Register A contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC0A pin.

16.9.7. TC0 Output Compare Register B

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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: OCR0B
Offset: 0x48
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x28

Bit	7	6	5	4	3	2	1	0
	OCR0B[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – OCR0B[7:0]: Output Compare 0 B
The Output Compare Register B contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC0B pin.

16.9.8. TC0 Interrupt Flag 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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: TIFR0
Offset: 0x35
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x15

Bit	7	6	5	4	3	2	1	0
						OCFB	OCFA	TOV
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 2 – OCFB: Timer/Counter0, Output Compare B Match Flag
The OCF0B bit is set when a Compare Match occurs between the Timer/Counter and the data in OCR0B – Output Compare Register0 B. OCF0B is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0B is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0B (Timer/Counter Compare B Match Interrupt Enable), and OCF0B are set, the Timer/Counter Compare Match Interrupt is executed.

Bit 1 – OCFA: Timer/Counter0, Output Compare A Match Flag
The OCF0A bit is set when a Compare Match occurs between the Timer/Counter0 and the data in OCR0A – Output Compare Register0. OCF0A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF0A is cleared by writing a logic one to the flag. When the I-bit in SREG, OCIE0A (Timer/Counter0 Compare Match Interrupt Enable), and OCF0A are set, the Timer/Counter0 Compare Match Interrupt is executed.

Bit 0 – TOV: Timer/Counter0, Overflow Flag
The bit TOV0 is set when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logic one to the flag. When the SREG I-bit, TOIE0 (Timer/Counter0 Overflow Interrupt Enable), and TOV0 are set, the Timer/Counter0 Overflow interrupt is executed.

The setting of this flag is dependent of the WGM02:0 bit setting. Refer to Table 16-9.

20. SPI – Serial Peripheral Interface

20.1. Features

- Full-duplex, Three-wire Synchronous Data Transfer
- Master or Slave Operation
- LSB First or MSB First Data Transfer
- Seven Programmable Bit Rates
- End of Transmission Interrupt Flag
- Write Collision Flag Protection
- Wake-up from Idle Mode
- Double Speed (CK/2) Master SPI Mode

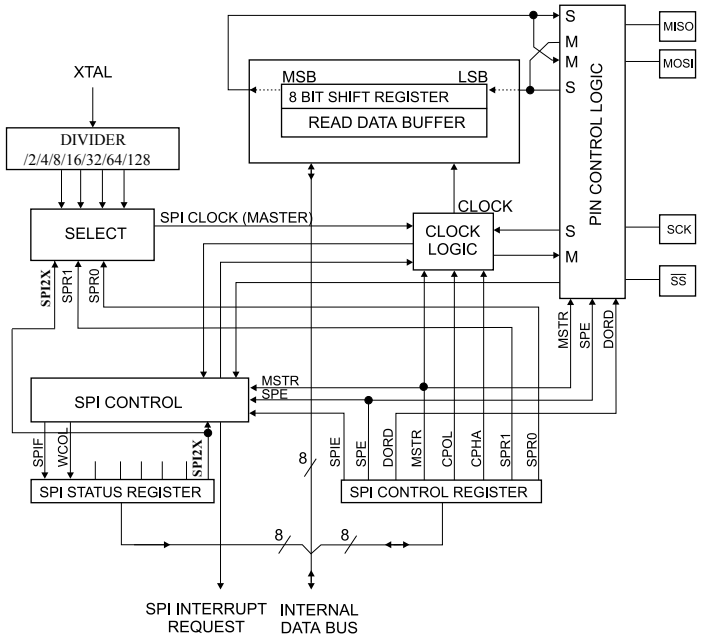
20.2. Overview

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the device and peripheral units, or between several AVR devices.

The USART can also be used in Master SPI mode, please refer to *USART in SPI Mode* chapter.

To enable the SPI module, Power Reduction Serial Peripheral Interface bit in the Power Reduction Register (0.PRSPI0) must be written to '0'.

Figure 20-1. SPI Block Diagram



Note: Refer to the pin-out description and the IO Port description for SPI pin placement.

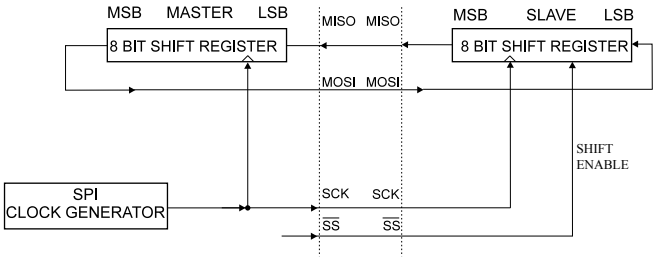
The interconnection between Master and Slave CPUs with SPI is shown in the figure below. The system consists of two shift registers, and a Master Clock generator. The SPI Master initiates the communication cycle when pulling low the Slave Select $\overline{SS}$ pin of the desired Slave. Master and Slave prepare the data to be sent in their respective shift Registers, and the Master generates the required clock pulses on the SCK line to interchange data. Data is always shifted from Master to Slave on the Master Out – Slave In, MOSI, line, and from Slave to Master on the Master In – Slave Out, MISO, line. After each data packet, the Master will synchronize the Slave by pulling high the Slave Select, $\overline{SS}$, line.

When configured as a Master, the SPI interface has no automatic control of the $\overline{SS}$ line. This must be handled by user software before communication can start. When this is done, writing a byte to the SPI Data Register starts the SPI clock generator, and the hardware shifts the eight bits into the Slave. After shifting one byte, the SPI clock generator stops, setting the end of Transmission Flag (SPIF). If the SPI Interrupt Enable bit (SPIE) in the SPCR Register is set, an interrupt is requested. The Master may continue to shift the next byte by writing it into SPDR, or signal the end of packet by pulling high the Slave Select, $\overline{SS}$ line. The last incoming byte will be kept in the Buffer Register for later use.

When configured as a Slave, the SPI interface will remain sleeping with MISO tri-stated as long as the $\overline{SS}$ pin is driven high. In this state, software may update the contents of the SPI Data Register, SPDR, but the data will not be shifted out by incoming clock pulses on the SCK pin until the $\overline{SS}$ pin is driven low. As one byte has been completely shifted, the end of Transmission Flag, SPIF is set. If the SPI Interrupt Enable bit, SPIE, in the SPCR Register is set, an interrupt is requested. The Slave may continue to place new

data to be sent into SPDR before reading the incoming data. The last incoming byte will be kept in the Buffer Register for later use.

Figure 20-2. SPI Master-slave Interconnection



The system is single buffered in the transmit direction and double buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SPI Data Register before the entire shift cycle is completed. When receiving data, however, a received character must be read from the SPI Data Register before the next character has been completely shifted in. Otherwise, the first byte is lost.

In SPI Slave mode, the control logic will sample the incoming signal of the SCK pin. To ensure correct sampling of the clock signal, the minimum low and high periods should be longer than two CPU clock cycles.

When the SPI is enabled, the data direction of the MOSI, MISO, SCK, and $\overline{SS}$ pins is overridden according to the table below. For more details on automatic port overrides, refer to the IO Port description.

Table 20-1. SPI Pin Overrides

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User Defined	Input
MISO	Input	User Defined
SCK	User Defined	Input
$\overline{SS}$	User Defined	Input

Note: 1. See the IO Port description for how to define the SPI pin directions.

The following code examples show how to initialize the SPI as a Master and how to perform a simple transmission. `DDR_SPI` in the examples must be replaced by the actual Data Direction Register controlling the SPI pins. `DD_MOSI`, `DD_MISO` and `DD_SCK` must be replaced by the actual data direction bits for these pins. E.g. if MOSI is placed on pin PB5, replace `DD_MOSI` with `DDB5` and `DDR_SPI` with `DDRB`.

Assembly Code Example

```
SPI_MasterInit:
   ; Set MOSI and SCK output, all others input
   ldi    r17,(1<<DD_MOSI)|(1<<DD_SCK)
   out    DDR_SPI,r17
   ; Enable SPI, Master, set clock rate fck/16
   ldi    r17,(1<<SPE)|(1<<MSTR)|(1<<SPR0)
   out    SPCR,r17
   ret
```

```
SPI_MasterTransmit:
   ; Start transmission of data (r16)
   out    SPDR,r16
Wait_Transmit:
   ; Wait for transmission complete
   in     r16, SPSR
   sbrs   r16, SPIF
   rjmp   Wait_Transmit
   ret
```

C Code Example

```
void SPI_MasterInit(void)
{
   /* Set MOSI and SCK output, all others input */
   DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK);
   /* Enable SPI, Master, set clock rate fck/16 */
   SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}

void SPI_MasterTransmit(char cData)
{
   /* Start transmission */
   SPDR = cData;
   /* Wait for transmission complete */
   while(!(SPSR & (1<<SPIF)))
      ;
}
```

The following code examples show how to initialize the SPI as a Slave and how to perform a simple reception.

Assembly Code Example

```
SPI_SlaveInit:
   ; Set MISO output, all others input
   ldi    r17,(1<<DD_MISO)
   out    DDR_SPI,r17
   ; Enable SPI
   ldi    r17,(1<<SPE)
   out    SPCR,r17
   ret
SPI_SlaveReceive:
   ; Wait for reception complete
   in     r16, SPSR
   sbrs   r16, SPIF
   rjmp SPI_SlaveReceive
   ; Read received data and return
   in     r16,SPDR
   ret
```

C Code Example

```
void SPI_SlaveInit(void)
{
   /* Set MISO output, all others input */
   DDR_SPI = (1<<DD_MISO);
   /* Enable SPI */
   SPCR = (1<<SPE);
}
char SPI_SlaveReceive(void)
{
   /* Wait for reception complete */
   while(!(SPSR & (1<<SPIF)))
      ;
   /* Return Data Register */
   return SPDR;
}
```

Related Links

Pin Descriptions on page 16
USARTSPI - USART in SPI Mode on page 252
PM - Power Management and Sleep Modes on page 56
I/O-Ports on page 95
About Code Examples on page 20

20.3. $\overline{SS}$ Pin Functionality

20.3.1. Slave Mode

When the SPI is configured as a Slave, the Slave Select ($\overline{SS}$) pin is always input. When $\overline{SS}$ is held low, the SPI is activated, and MISO becomes an output if configured so by the user. All other pins are inputs. When $\overline{SS}$ is driven high, all pins are inputs, and the SPI is passive, which means that it will not receive incoming data. The SPI logic will be reset once the SS pin is driven high.

The $\overline{SS}$ pin is useful for packet/byte synchronization to keep the slave bit counter synchronous with the master clock generator. When the $\overline{SS}$ pin is driven high, the SPI slave will immediately reset the send and receive logic, and drop any partially received data in the Shift Register.

20.3.2. Master Mode

When the SPI is configured as a Master (MSTR in SPCR is set), the user can determine the direction of the $\overline{SS}$ pin.

If $\overline{SS}$ is configured as an output, the pin is a general output pin which does not affect the SPI system. Typically, the pin will be driving the $\overline{SS}$ pin of the SPI Slave.

If $\overline{SS}$ is configured as an input, it must be held high to ensure Master SPI operation. If the $\overline{SS}$ pin is driven low by peripheral circuitry when the SPI is configured as a Master with the $\overline{SS}$ pin defined as an input, the SPI system interprets this as another master selecting the SPI as a slave and starting to send data to it. To avoid bus contention, the SPI system takes the following actions:

1. The MSTR bit in SPCR is cleared and the SPI system becomes a Slave. As a result of the SPI becoming a Slave, the MOSI and SCK pins become inputs.
2. The SPIF Flag in SPSR is set, and if the SPI interrupt is enabled, and the I-bit in SREG is set, the interrupt routine will be executed.

Thus, when interrupt-driven SPI transmission is used in Master mode, and there exists a possibility that $\overline{SS}$ is driven low, the interrupt should always check that the MSTR bit is still set. If the MSTR bit has been cleared by a slave select, it must be set by the user to re-enable SPI Master mode.

20.4. Data Modes

There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. Data bits are shifted out and latched in on opposite edges of the SCK signal, ensuring sufficient time for data signals to stabilize. The following table, summarizes SPCR.CPOL and SPCR.CPHA settings.

Table 20-2. SPI Modes

SPI Mode	Conditions	Leading Edge	Trailing Edge
0	CPOL=0, CPHA=0	Sample (Rising)	Setup (Falling)
1	CPOL=0, CPHA=1	Setup (Rising)	Sample (Falling)
2	CPOL=1, CPHA=0	Sample (Falling)	Setup (Rising)
3	CPOL=1, CPHA=1	Setup (Falling)	Sample (Rising)

The SPI data transfer formats are shown in the following figure.

Figure 20-3. SPI Transfer Format with CPHA = 0

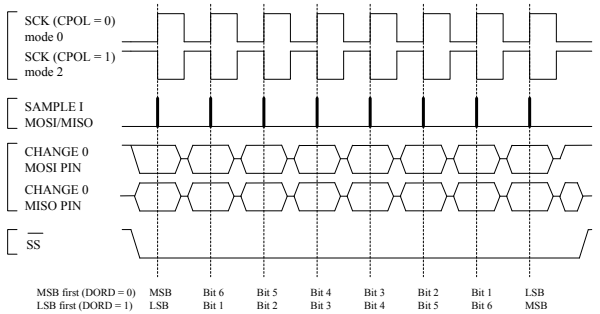
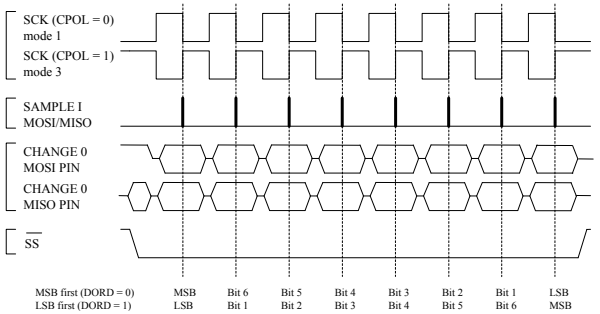


Figure 20-4. SPI Transfer Format with CPHA = 1



20.5. Register Description

20.5.1. SPI Control 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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: SPCR0
Offset: 0x4C
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x2C

Bit	7	6	5	4	3	2	1	0
	SPIE0	SPE0	DORD0	MSTR0	CPOL0	CPHA0	SPR01	SPR00
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – SPIE0: SPI0 Interrupt Enable
This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR Register is set and if the Global Interrupt Enable bit in SREG is set.

Bit 6 – SPE0: SPI0 Enable
When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any SPI operations.

Bit 5 – DORD0: Data0 Order
When the DORD bit is written to one, the LSB of the data word is transmitted first.

When the DORD bit is written to zero, the MSB of the data word is transmitted first.

Bit 4 – MSTR0: Master/Slave0 Select
This bit selects Master SPI mode when written to one, and Slave SPI mode when written logic zero. If SS is configured as an input and is driven low while MSTR is set, MSTR will be cleared, and SPIF in SPSR will become set. The user will then have to set MSTR to re-enable SPI Master mode.

Bit 3 – CPOL0: Clock0 Polarity
When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is low when idle. Refer to Figure 20-3 and Figure 20-4 for an example. The CPOL functionality is summarized below:

Table 20-3. CPOL0 Functionality

CPOL0	Leading Edge	Trailing Edge
0	Rising	Falling
1	Falling	Rising

Bit 2 – CPHA0: Clock0 Phase
The settings of the Clock Phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of SCK. Refer to Figure 20-3 and Figure 20-4 for an example. The CPHA functionality is summarized below:

Table 20-4. CPHA0 Functionality

CPHA0	Leading Edge	Trailing Edge
0	Sample	Setup
1	Setup	Sample

Bits 1:0 – SPR0n: SPI0 Clock Rate Select n [n = 1:0]
These two bits control the SCK rate of the device configured as a Master. SPR1 and SPR0 have no effect on the Slave. The relationship between SCK and the Oscillator Clock frequency f_{osc} is shown in the table below.

Table 20-5. Relationship between SCK and Oscillator Frequency

SPI2X	SPR01	SPR00	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

20.5.2. SPI Status 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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: SPSR0
Offset: 0x4D
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x2D

Bit	7	6	5	4	3	2	1	0
	SPIF0	WCOL0						SPI2X0
Access	R	R						R/W
Reset	0	0						0

Bit 7 – SPIF0: SPI Interrupt Flag
When a serial transfer is complete, the SPIF Flag is set. An interrupt is generated if SPIE in SPCR is set and global interrupts are enabled. If $\overline{SS}$ is an input and is driven low when the SPI is in Master mode, this will also set the SPIF Flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF set, then accessing the SPI Data Register (SPDR).

Bit 6 – WCOL0: Write Collision Flag
The WCOL bit is set if the SPI Data Register (SPDR) is written during a data transfer. The WCOL bit (and the SPIF bit) are cleared by first reading the SPI Status Register with WCOL set, and then accessing the SPI Data Register.

Bit 0 – SPI2X0: Double SPI Speed Bit
When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (refer to Table 20-5). This means that the minimum SCK period will be two CPU clock periods. When the SPI is configured as Slave, the SPI is only guaranteed to work at fosc/4 or lower.

The SPI interface is also used for program memory and EEPROM downloading or uploading. See *Serial Downloading* for serial programming and verification.

20.5.3. 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.

The device is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

Name: SPDR0
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.

21. USART - Universal Synchronous Asynchronous Receiver Transceiver

21.1. Features

- Two USART instances USART0, USART1
- Full Duplex Operation (Independent Serial Receive and Transmit Registers)
- Asynchronous or Synchronous Operation
- Master or Slave Clocked Synchronous Operation
- High Resolution Baud Rate Generator
- Supports Serial Frames with 5, 6, 7, 8, or 9 data bits and 1 or 2 stop bits
- Odd or Even Parity Generation and Parity Check Supported by Hardware
- Data OverRun Detection
- Framing Error Detection
- Noise Filtering Includes False Start Bit Detection and Digital Low Pass Filter
- Three Separate Interrupts on TX Complete, TX Data Register Empty and RX Complete
- Multi-processor Communication Mode
- Double Speed Asynchronous Communication Mode

21.2. Overview

The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a highly flexible serial communication device.

The USART can also be used in Master SPI mode. The Power Reduction USART bit in the Power Reduction Register (0.PRUSARTn) must be written to '0' in order to enable USARTn. USART 0 and 1 are in 0.

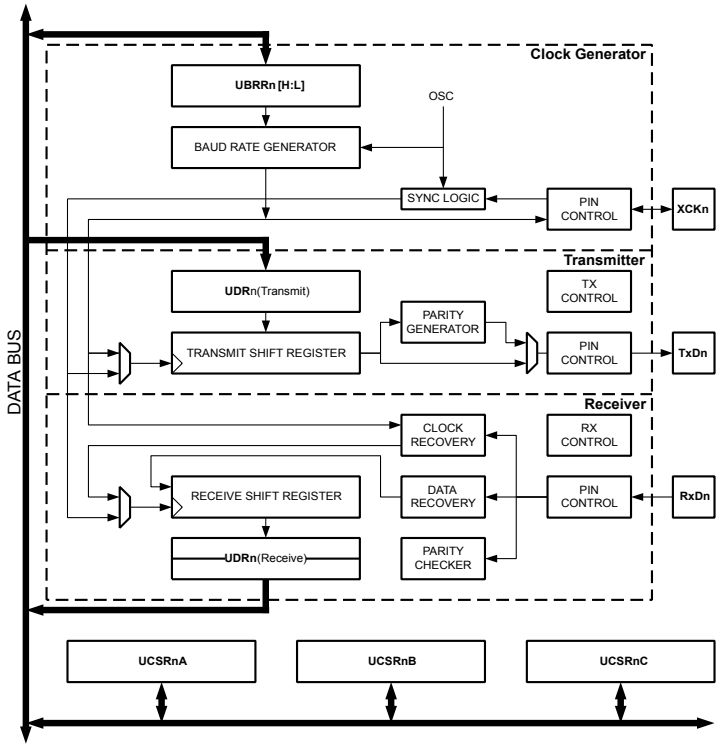
Related Links

USARTSPI - USART in SPI Mode on page 252
I/O-Ports on page 95
Pinout on page 15

21.3. Block Diagram

In the USART Block Diagram, the CPU accessible I/O Registers and I/O pins are shown in bold. The dashed boxes in the block diagram separate the three main parts of the USART (listed from the top): Clock Generator, Transmitter, and Receiver. Control Registers are shared by all units. The Clock Generation logic consists of synchronization logic for external clock input used by synchronous slave operation, and the baud rate generator. The XCKn (Transfer Clock) pin is only used by synchronous transfer mode. The Transmitter consists of a single write buffer, a serial Shift Register, Parity Generator, and Control logic for handling different serial frame formats. The write buffer allows a continuous transfer of data without any delay between frames. The Receiver is the most complex part of the USART module due to its clock and data recovery units. The recovery units are used for asynchronous data reception. In addition to the recovery units, the Receiver includes a Parity Checker, Control logic, a Shift Register, and a two level receive buffer (UDRn). The Receiver supports the same frame formats as the Transmitter, and can detect Frame Error, Data OverRun, and Parity Errors.

Figure 21-1. USART Block Diagram



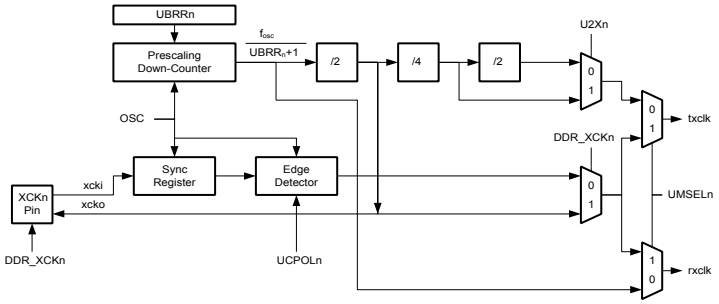
Note: Refer to the *Pin Configurations* and the *I/O-Ports* description for USART pin placement.

21.4. Clock Generation

The Clock Generation logic generates the base clock for the Transmitter and Receiver. The USART supports four modes of clock operation: Normal asynchronous, Double Speed asynchronous, Master synchronous and Slave synchronous mode. The USART Mode Select bit 0 in the USART Control and Status Register n C (UCSRnC.UMSELn0) selects between asynchronous and synchronous operation. Double Speed (asynchronous mode only) is controlled by the U2X found in the UCSRnA Register. When using synchronous mode (UMSELn0=1), the Data Direction Register for the XCKn pin (DDR_XCKn) controls whether the clock source is internal (Master mode) or external (Slave mode). The XCKn pin is only active when using synchronous mode.

Below is a block diagram of the clock generation logic.

Figure 21-2. Clock Generation Logic, Block Diagram



Signal description:

- txclk: Transmitter clock (internal signal).
- rxclk: Receiver base clock (internal signal).
- xcki: Input from XCKn pin (internal signal). Used for synchronous slave operation.
- xcko: Clock output to XCKn pin (internal signal). Used for synchronous master operation.
- f_{osc}: System clock frequency.

21.4.1. Internal Clock Generation – The Baud Rate Generator

Internal clock generation is used for the asynchronous and the synchronous master modes of operation. The description in this section refers to the Clock Generation Logic block diagram in the previous section..

The USART Baud Rate Register (UBRRn) and the down-counter connected to it function as a programmable prescaler or baud rate generator. The down-counter, running at system clock (f_{osc}), is loaded with the UBRRn value each time the counter has counted down to zero or when the UBRRnL Register is written. A clock is generated each time the counter reaches zero. This clock is the baud rate generator clock output (= f_{osc}/(UBRRn+1)). The Transmitter divides the baud rate generator clock output by 2, 8, or 16 depending on mode. The baud rate generator output is used directly by the Receiver's clock and data recovery units. However, the recovery units use a state machine that uses 2, 8, or 16 states depending on mode set by the state of the UMSEL, U2X and DDR_XCK bits.

The table below contains equations for calculating the baud rate (in bits per second) and for calculating the UBRRn value for each mode of operation using an internally generated clock source.

Table 21-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate(1)	Equation for Calculating UBRRn Value
Asynchronous Normal mode (U2X = 0)	$\mathrm{BAUD} = \frac{f_{\mathrm{OSC}}}{16(\mathbf{UBRR}n+1)}$	$\mathbf{UBRR}n = \frac{f_{\mathrm{OSC}}}{16\mathrm{BAUD}} - 1$
Asynchronous Double Speed mode (U2X = 1)	$\mathrm{BAUD} = \frac{f_{\mathrm{OSC}}}{8(\mathbf{UBRR}n+1)}$	$\mathbf{UBRR}n = \frac{f_{\mathrm{OSC}}}{8\mathrm{BAUD}} - 1$
Synchronous Master mode	$\mathrm{BAUD} = \frac{f_{\mathrm{OSC}}}{2(\mathbf{UBRR}n+1)}$	$\mathbf{UBRR}n = \frac{f_{\mathrm{OSC}}}{2\mathrm{BAUD}} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bits per second (bps)

BAUD Baud rate (in bits per second, bps)

f_{osc} System oscillator clock frequency

UBRRn Contents of the UBRRnH and UBRRnL Registers, (0-4095).
Some examples of UBRRn values for some system clock frequencies are found in Examples of Baud Rate Settings.

21.4.2. Double Speed Operation (U2X)

The transfer rate can be doubled by setting the U2X bit in UCSRnA. Setting this bit only has effect for the asynchronous operation. Set this bit to zero when using synchronous operation.

Setting this bit will reduce the divisor of the baud rate divider from 16 to 8, effectively doubling the transfer rate for asynchronous communication. However, in this case, the Receiver will only use half the number of samples (reduced from 16 to 8) for data sampling and clock recovery, and therefore a more accurate baud rate setting and system clock are required when this mode is used.

For the Transmitter, there are no downsides.

21.4.3. External Clock

External clocking is used by the synchronous slave modes of operation. The description in this section refers to the Clock Generation Logic block diagram in the previous section.

External clock input from the XCKn pin is sampled by a synchronization register to minimize the chance of meta-stability. The output from the synchronization register must then pass through an edge detector before it can be used by the Transmitter and Receiver. This process introduces a two CPU clock period delay and therefore the maximum external XCKn clock frequency is limited by the following equation:

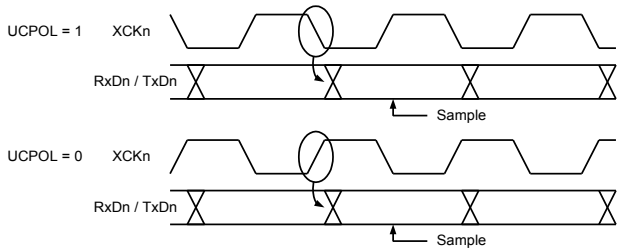
$$f_{\mathrm{XCKn}} < \frac{f_{\mathrm{OSC}}}{4}$$

The value of f_{osc} depends on the stability of the system clock source. It is therefore recommended to add some margin to avoid possible loss of data due to frequency variations.

21.4.4. Synchronous Clock Operation

When synchronous mode is used (UMSEL = 1), the XCKn pin will be used as either clock input (Slave) or clock output (Master). The dependency between the clock edges and data sampling or data change is the same. The basic principle is that data input (on RxDn) is sampled at the opposite XCKn clock edge of the edge the data output (TxDn) is changed.

Figure 21-3. Synchronous Mode XCKn Timing



The UCPOL bit UCRSC selects which XCKn clock edge is used for data sampling and which is used for data change. As the above timing diagram shows, when UCPOL is zero, the data will be changed at

rising XCKn edge and sampled at falling XCKn edge. If UCPOL is set, the data will be changed at falling XCKn edge and sampled at rising XCKn edge.

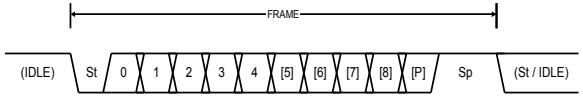
21.5. Frame Formats

A serial frame is defined to be one character of data bits with synchronization bits (start and stop bits), and optionally a parity bit for error checking. The USART accepts all 30 combinations of the following as valid frame formats:

- 1 start bit
- 5, 6, 7, 8, or 9 data bits
- no, even or odd parity bit
- 1 or 2 stop bits

A frame starts with the start bit, followed by the data bits (from five up to nine data bits in total): first the least significant data bit, then the next data bits ending with the most significant bit. If enabled, the parity bit is inserted after the data bits, before the one or two stop bits. When a complete frame is transmitted, it can be directly followed by a new frame, or the communication line can be set to an idle (high) state. the figure below illustrates the possible combinations of the frame formats. Bits inside brackets are optional.

Figure 21-4. Frame Formats



St	Start bit, always low.
(n)	Data bits (0 to 8).
P	Parity bit. Can be odd or even.
Sp	Stop bit, always high.
IDLE	No transfers on the communication line (RxDn or TxDn). An IDLE line must be high.

The frame format used by the USART is set by:

- Character Size bits (UCSRnC.UCSZn[2:0]) select the number of data bits in the frame.
- Parity Mode bits (UCSRnC.UPMn[1:0]) enable and set the type of parity bit.
- Stop Bit Select bit (UCSRnC.USBSn) select the number of stop bits. The Receiver ignores the second stop bit.

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. An FE (Frame Error) will only be detected in cases where the first stop bit is zero.

21.5.1. Parity Bit Calculation

The parity bit is calculated by doing an exclusive-or of all the data bits. If odd parity is used, the result of the exclusive or is inverted. The relation between the parity bit and data bits is as follows:

$$P_{\text{even}} = d_{n+-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0 \quad P_{\text{odd}} = d_{n+-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

P_{even} Parity bit using even parity

P_{odd} Parity bit using odd parity

d_n Data bit n of the character

If used, the parity bit is located between the last data bit and first stop bit of a serial frame.

21.6. USART Initialization

The USART has to be initialized before any communication can take place. The initialization process normally consists of setting the baud rate, setting frame format and enabling the Transmitter or the Receiver depending on the usage. For interrupt driven USART operation, the Global Interrupt Flag should be cleared (and interrupts globally disabled) when doing the initialization.

Before doing a re-initialization with changed baud rate or frame format, be sure that there are no ongoing transmissions during the period the registers are changed. The TXC Flag (UCSRnA.TXC) can be used to check that the Transmitter has completed all transfers, and the RXC Flag can be used to check that there are no unread data in the receive buffer. The UCSRnA.TXC 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 asynchronous operation using polling (no interrupts enabled) and a fixed frame format. 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

```
USART_Init:
   ; Set baud rate to UBRR0
   out    UBRR0H, r17
   out    UBRR0L, r16
   ; Enable receiver and transmitter
   ldi    r16, (1<<RXEN0)|(1<<TXEN0)
   out    UCSR0B,r16
   ; Set frame format: 8data, 2stop bit
   ldi    r16, (1<<USBS0)|(3<<UCSZ00)
   out    UCSR0C,r16
   ret
```

C Code Example

```
#define FOSC 1843200 // Clock Speed
#define BAUD 9600
#define MYUBRR FOSC/16/BAUD-1
void main( void )
{
   ...
   USART_Init(MYUBRR)
   ...
}
void USART_Init( unsigned int ubrr)
{
   /*Set baud rate */
   UBRR0H = (unsigned char)(ubrr>>8);
   UBRR0L = (unsigned char)ubrr;
   Enable receiver and transmitter */
   UCSR0B = (1<<RXEN0)|(1<<TXEN0);
   /* Set frame format: 8data, 2stop bit */
   UCSR0C = (1<<USBS0)|(3<<UCSZ00);
}
```

More advanced initialization routines can be written to include frame format as parameters, disable interrupts, and so on. However, many applications use a fixed setting

of the baud and control registers, and for these types of applications the initialization code can be placed directly in the main routine, or be combined with initialization code for other I/O modules.

Related Links

About Code Examples on page 20

21.7. Data Transmission – The USART Transmitter

The USART Transmitter is enabled by setting the Transmit Enable (TXEN) bit in the UCSRnB Register. When the Transmitter is enabled, the normal port operation of the TxDn pin is overridden by the USART and given the function as the Transmitter's serial output. The baud rate, mode of operation and frame format must be set up once before doing any transmissions. If synchronous operation is used, the clock on the XCKn pin will be overridden and used as transmission clock.

21.7.1. Sending Frames with 5 to 8 Data Bits

A data transmission is initiated by loading the transmit buffer with the data to be transmitted. The CPU can load the transmit buffer by writing to the UDRn I/O location. The buffered data in the transmit buffer will be moved to the Shift Register when the Shift Register is ready to send a new frame. The Shift Register is loaded with new data if it is in idle state (no ongoing transmission) or immediately after the last stop bit of the previous frame is transmitted. When the Shift Register is loaded with new data, it will transfer one complete frame at the rate given by the Baud Register, U2X bit or by XCKn depending on mode of operation.

The following code examples show a simple USART transmit function based on polling of the Data Register Empty (UDRE) Flag. When using frames with less than eight bits, the most significant bits written to the UDR0 are ignored. The USART 0 has to be initialized before the function can be used. For the assembly code, the data to be sent is assumed to be stored in Register R17.

Assembly Code Example

```
USART_Transmit:
   ; Wait for empty transmit buffer
   in      r17, UCSR0A
   sbrs    r17, UDRE
   rjmp    USART_Transmit
   ; Put data (r16) into buffer, sends the data
   out    UDR0,r16
   ret
```

C Code Example

```
void USART_Transmit( unsigned char data )
{
   /* Wait for empty transmit buffer */
   while ( !( UCSR0A & (1<<UDRE)) )
      ;
   /* Put data into buffer, sends the data */
   UDR0 = data;
}
```

The function simply waits for the transmit buffer to be empty by checking the UDRE Flag, before loading it with new data to be transmitted. If the Data Register Empty interrupt is utilized, the interrupt routine writes the data into the buffer.

Related Links

About Code Examples on page 20

21.7.2. Sending Frames with 9 Data Bit

If 9-bit characters are used (UCSZn = 7), the ninth bit must be written to the TXB8 bit in UCSRnB before the low byte of the character is written to UDRn.

The ninth bit can be used for indicating an address frame when using multi processor communication mode or for other protocol handling as for example synchronization.

The following code examples show a transmit function that handles 9-bit characters. For the assembly code, the data to be sent is assumed to be stored in registers R17:R16.

Assembly Code Example

```
USART_Transmit:
   ; Wait for empty transmit buffer
   in      r18, UCSR0A
   sbrs    r18, UDRE
   rjmp    USART_Transmit
   ; Copy 9th bit from r17 to TXB8
   cbi     UCSR0B,TXB8
   sbrc    r17,0
   sbi     UCSR0B,TXB8
   ; Put LSB data (r16) into buffer, sends the data
   out    UDR0,r16
   ret
```

C Code Example

```
void USART_Transmit( unsigned int data )
{
   /* Wait for empty transmit buffer */
   while ( !( UCSR0A & (1<<UDRE))) )
      ;
   /* Copy 9th bit to TXB8 */
   UCSR0B &= ~(1<<TXB8);
   if ( data & 0x0100 )
      UCSR0B |= (1<<TXB8);
   /* Put data into buffer, sends the data */
   UDR0 = data;
}
```

Note: These transmit functions are written to be general functions. They can be optimized if the contents of the UCSRnB is static. For example, only the TXB8 bit of the UCSRnB Register is used after initialization.

Related Links

About Code Examples on page 20

21.7.3. Transmitter Flags and Interrupts

The USART Transmitter has two flags that indicate its state: USART Data Register Empty (UDRE) and Transmit Complete (TXC). Both flags can be used for generating interrupts.

The Data Register Empty (UDRE) Flag indicates whether the transmit buffer is ready to receive new data. This bit is set when the transmit buffer is empty, and cleared when the transmit buffer contains data to be transmitted that has not yet been moved into the Shift Register. For compatibility with future devices, always write this bit to zero when writing the UCSRnA Register.

When the Data Register Empty Interrupt Enable (UDRIE) bit in UCSRnB is written to '1', the USART Data Register Empty Interrupt will be executed as long as UDRE is set (provided that global interrupts are enabled). UDRE is cleared by writing UDRn. When interrupt-driven data transmission is used, the Data Register Empty interrupt routine must either write new data to UDRn in order to clear UDRE or disable

the Data Register Empty interrupt - otherwise, a new interrupt will occur once the interrupt routine terminates.

The Transmit Complete (TXC) 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. The TXC Flag bit is either automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a '1' to its bit location. The TXC Flag is useful in half-duplex communication interfaces (like the RS-485 standard), where a transmitting application must enter receive mode and free the communication bus immediately after completing the transmission.

When the Transmit Compete Interrupt Enable (TXCIE) bit in UCSRnB is written to '1', the USART Transmit Complete Interrupt will be executed when the TXC Flag becomes set (provided that global interrupts are enabled). When the transmit complete interrupt is used, the interrupt handling routine does not have to clear the TXC Flag, this is done automatically when the interrupt is executed.

21.7.4. Parity Generator

The Parity Generator calculates the parity bit for the serial frame data. When parity bit is enabled (UCSRnC.UPM[1]=1), the transmitter control logic inserts the parity bit between the last data bit and the first stop bit of the frame that is sent.

21.7.5. Disabling the Transmitter

When writing the TX Enable bit in the USART Control and Status Register n B (UCSRnB.TXEN) to zero, the disabling of the Transmitter will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxDn pin.

21.8. Data Reception – The USART Receiver

The USART Receiver is enabled by writing the Receive Enable (RXEN) bit in the UCSRnB Register to '1'. When the Receiver is enabled, the normal pin operation of the RxDn pin is overridden by the USART and given the function as the Receiver's serial input. The baud rate, mode of operation and frame format must be set up once before any serial reception can be done. If synchronous operation is used, the clock on the XCKn pin will be used as transfer clock.

21.8.1. Receiving Frames with 5 to 8 Data Bits

The Receiver starts data reception when it detects a valid start bit. Each bit that follows the start bit will be sampled at the baud rate or XCKn clock, and shifted into the Receive Shift Register until the first stop bit of a frame is received. A second stop bit will be ignored by the Receiver. When the first stop bit is received, i.e., a complete serial frame is present in the Receive Shift Register, the contents of the Shift Register will be moved into the receive buffer. The receive buffer can then be read by reading the UDRn I/O location.

The following code example shows a simple USART receive function based on polling of the Receive Complete (RXC) Flag. When using frames with less than eight bits the most significant bits of the data read from the UDR0 will be masked to zero. The USART 0 has to be initialized before the function can be used. For the assembly code, the received data will be stored in R16 after the code completes.

Assembly Code Example

```
USART_Receive:
   ; Wait for data to be received
   in    r17, UCSR0A
   sbrs  r17, RXC
   rjmp  USART_Receive
   ; Get and return received data from buffer
   in    r16, UDR0
   ret
```

C Code Example

```
unsigned char USART_Receive( void )
{
   /* Wait for data to be received */
   while ( !(UCSR0A & (1<<RXC)) )
      ;
   /* Get and return received data from buffer */
   return UDR0;
}
```

For I/O Registers located in extended I/O map, "IN", "OUT", "SBIS", "SBIC", "CBI", and "SBI" instructions must be replaced with instructions that allow access to extended I/O. Typically "LDS" and "STS" combined with "SBRS", "SBRC", "SBR", and "CBR".

The function simply waits for data to be present in the receive buffer by checking the RXC Flag, before reading the buffer and returning the value.

Related Links

About Code Examples on page 20

21.8.2. Receiving Frames with 9 Data Bits

If 9-bit characters are used (UCSZn=7) the ninth bit must be read from the RXB8 bit in UCSRnB before reading the low bits from the UDRn. This rule applies to the FE, DOR and UPE Status Flags as well. Read status from UCSRnA, then data from UDRn. Reading the UDRn I/O location will change the state of the receive buffer FIFO and consequently the TXB8, FE, DOR and UPE bits, which all are stored in the FIFO, will change.

The following code example shows a simple receive function for USART0 that handles both nine bit characters and the status bits. For the assembly code, the received data will be stored in R17:R16 after the code completes.

Assembly Code Example

```
USART_Receive:
   ; Wait for data to be received
   in     r16, UCSR0A
   sbrs   r16, RXC
   rjmp   USART_Receive
   ; Get status and 9th bit, then data from buffer
   in     r18, UCSR0A
   in     r17, UCSR0B
   in     r16, UDR0
   ; If error, return -1
   andi   r18,(1<<FE)|(1<<DOR)|(1<<UPE)
   breq   USART_ReceiveNoError
   ldi    r17, HIGH(-1)
   ldi    r16, LOW(-1)
USART_ReceiveNoError:
   ; Filter the 9th bit, then return
   lsr    r17
   andi   r17, 0x01
   ret
```

C Code Example

```
unsigned int USART_Receive( void )
{
```

```
    unsigned char status, resh, resl;
    /* Wait for data to be received */
    while ( !(UCSR0A & (1<<RXC)) )
        ;
    /* Get status and 9th bit, then data */
    /* from buffer */
    status = UCSR0A;
    resh = UCSR0B;
    resl = UDR0;
    /* If error, return -1 */
    if ( status & (1<<FE)|(1<<DOR)|(1<<UPE) )
       return -1;
    /* Filter the 9th bit, then return */
    resh = (resh >> 1) & 0x01;
    return ((resh << 8) | resl);
}
```

The receive function example reads all the I/O Registers into the Register File before any computation is done. This gives an optimal receive buffer utilization since the buffer location read will be free to accept new data as early as possible.

Related Links

About Code Examples on page 20

21.8.3. Receive Compete Flag and Interrupt

The USART Receiver has one flag that indicates the Receiver state.

The Receive Complete (RXC) Flag indicates if there are unread data present in the receive buffer. This flag is one when unread data exist in the receive buffer, and zero when the receive buffer is empty (i.e., does not contain any unread data). If the Receiver is disabled (RXEN = 0), the receive buffer will be flushed and consequently the RXCn bit will become zero.

When the Receive Complete Interrupt Enable (RXCIE) in UCSRnB is set, the USART Receive Complete interrupt will be executed as long as the RXC Flag is set (provided that global interrupts are enabled). When interrupt-driven data reception is used, the receive complete routine must read the received data from UDR in order to clear the RXC Flag, otherwise a new interrupt will occur once the interrupt routine terminates.

21.8.4. Receiver Error Flags

The USART Receiver has three Error Flags: Frame Error (FE), Data OverRun (DOR) and Parity Error (UPE). All can be accessed by reading UCSRnA. Common for the Error Flags is that they are located in the receive buffer together with the frame for which they indicate the error status. Due to the buffering of the Error Flags, the UCSRnA must be read before the receive buffer (UDRn), since reading the UDRn I/O location changes the buffer read location. Another equality for the Error Flags is that they can not be altered by software doing a write to the flag location. However, all flags must be set to zero when the UCSRnA is written for upward compatibility of future USART implementations. None of the Error Flags can generate interrupts.

The Frame Error (FE) Flag indicates the state of the first stop bit of the next readable frame stored in the receive buffer. The FE Flag is zero when the stop bit was correctly read as '1', and the FE Flag will be one when the stop bit was incorrect (zero). This flag can be used for detecting out-of-sync conditions, detecting break conditions and protocol handling. The FE Flag is not affected by the setting of the USBS bit in UCSRnC since the Receiver ignores all, except for the first, stop bits. For compatibility with future devices, always set this bit to zero when writing to UCSRnA.

The Data OverRun (DOR) Flag indicates data loss due to a receiver buffer full condition. A Data OverRun occurs when the receive buffer is full (two characters), a new character is waiting in the Receive Shift Register, and a new start bit is detected. If the DOR Flag is set, one or more serial frames were lost between the last frame read from UDR, and the next frame read from UDR. For compatibility with future devices, always write this bit to zero when writing to UCSRnA. The DOR Flag is cleared when the frame received was successfully moved from the Shift Register to the receive buffer.

The Parity Error (UPE) Flag indicates that the next frame in the receive buffer had a Parity Error when received. If Parity Check is not enabled the UPE bit will always read '0'. For compatibility with future devices, always set this bit to zero when writing to UCSRnA. For more details see Parity Bit Calculation and 'Parity Checker' below.

21.8.5. Parity Checker

The Parity Checker is active when the high USART Parity Mode bit 1 in the USART Control and Status Register n C (UCSRnC.UPM[1]) is written to '1'. The type of Parity Check to be performed (odd or even) is selected by the UCSRnC.UPM[0] bit. When enabled, the Parity Checker calculates the parity of the data bits in incoming frames and compares the result with the parity bit from the serial frame. The result of the check is stored in the receive buffer together with the received data and stop bits. The USART Parity Error Flag in the USART Control and Status Register n A (UCSRnA.UPE) can then be read by software to check if the frame had a Parity Error.

The UPEn bit is set if the next character that can be read from the receive buffer had a Parity Error when received and the Parity Checking was enabled at that point (UPM[1] = 1). This bit is valid until the receive buffer (UDRn) is read.

21.8.6. Disabling the Receiver

In contrast to the Transmitter, disabling of the Receiver will be immediate. Data from ongoing receptions will therefore be lost. When disabled (i.e., UCSRnB.RXEN is written to zero) the Receiver will no longer override the normal function of the RxDn port pin. The Receiver buffer FIFO will be flushed when the Receiver is disabled. Remaining data in the buffer will be lost.

21.8.7. Flushing the Receive Buffer

The receiver buffer FIFO will be flushed when the Receiver is disabled, i.e., the buffer will be emptied of its contents. Unread data will be lost. If the buffer has to be flushed during normal operation, due to for instance an error condition, read the UDRn I/O location until the RXCn Flag is cleared.

The following code shows how to flush the receive buffer of USART0.

Assembly Code Example

```
USART_Flush:
   in      r16, UCSR0A
   sbrs    r16, RXC
   ret
   in      r16, UDR0
   rjmp    USART_Flush
```

C Code Example

```
void USART_Flush( void )
{
   unsigned char dummy;
   while ( UCSR0A & (1<<RXC) ) dummy = UDR0;
}
```

Related Links

About Code Examples on page 20

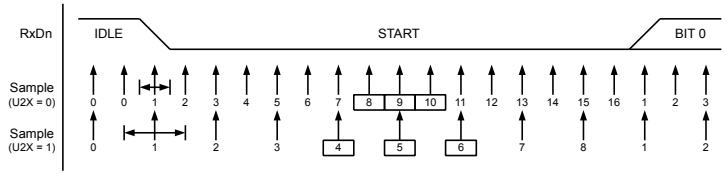
21.9. Asynchronous Data Reception

The USART includes a clock recovery and a data recovery unit for handling asynchronous data reception. The clock recovery logic is used for synchronizing the internally generated baud rate clock to the incoming asynchronous serial frames at the RxDn pin. The data recovery logic samples and low pass filters each incoming bit, thereby improving the noise immunity of the Receiver. The asynchronous reception operational range depends on the accuracy of the internal baud rate clock, the rate of the incoming frames, and the frame size in number of bits.

21.9.1. Asynchronous Clock Recovery

The clock recovery logic synchronizes internal clock to the incoming serial frames. The figure below illustrates the sampling process of the start bit of an incoming frame. The sample rate is 16-times the baud rate for Normal mode, and 8 times the baud rate for Double Speed mode. The horizontal arrows illustrate the synchronization variation due to the sampling process. Note the larger time variation when using the Double Speed mode (UCSRnA.U2X=1) of operation. Samples denoted '0' are samples taken while the RxDn line is idle (i.e., no communication activity).

Figure 21-5. Start Bit Sampling

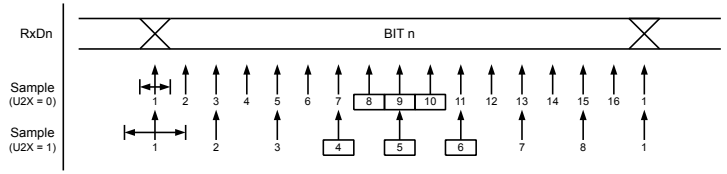


When the clock recovery logic detects a high (idle) to low (start) transition on the RxDn line, the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample as shown in the figure. The clock recovery logic then uses samples 8, 9, and 10 for Normal mode, and samples 4, 5, and 6 for Double Speed mode (indicated with sample numbers inside boxes on the figure), to decide if a valid start bit is received. If two or more of these three samples have logical high levels (the majority wins), the start bit is rejected as a noise spike and the Receiver starts looking for the next high to low-transition on RxDn. If however, a valid start bit is detected, the clock recovery logic is synchronized and the data recovery can begin. The synchronization process is repeated for each start bit.

21.9.2. Asynchronous Data Recovery

When the receiver clock is synchronized to the start bit, the data recovery can begin. The data recovery unit uses a state machine that has 16 states for each bit in Normal mode and eight states for each bit in Double Speed mode. The figure below shows the sampling of the data bits and the parity bit. Each of the samples is given a number that is equal to the state of the recovery unit.

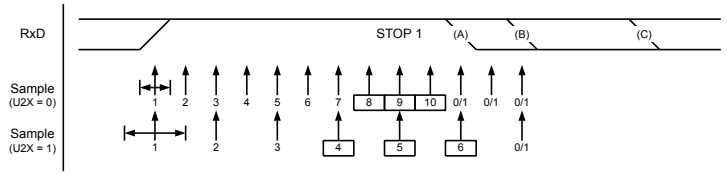
Figure 21-6. Sampling of Data and Parity Bit



The decision of the logic level of the received bit is taken by doing a majority voting of the logic value to the three samples in the center of the received bit: If two or all three center samples (those marked by their sample number inside boxes) have high levels, the received bit is registered to be a logic '1'. If two or all three samples have low levels, the received bit is registered to be a logic '0'. This majority voting process acts as a low pass filter for the incoming signal on the RxDn pin. The recovery process is then repeated until a complete frame is received, including the first stop bit. The Receiver only uses the first stop bit of a frame.

The following figure shows the sampling of the stop bit and the earliest possible beginning of the start bit of the next frame.

Figure 21-7. Stop Bit Sampling and Next Start Bit Sampling



The same majority voting is done to the stop bit as done for the other bits in the frame. If the stop bit is registered to have a logic '0' value, the Frame Error (UCSRnA.FE) Flag will be set.

A new high to low transition indicating the start bit of a new frame can come right after the last of the bits used for majority voting. For Normal Speed mode, the first low level sample can be taken at point marked (A) in the figure above. For Double Speed mode, the first low level must be delayed to (B). (C) marks a stop bit of full length. The early start bit detection influences the operational range of the Receiver.

21.9.3. Asynchronous Operational Range

The operational range of the Receiver is dependent on the mismatch between the received bit rate and the internally generated baud rate. If the Transmitter is sending frames at too fast or too slow bit rates, or the internally generated baud rate of the Receiver does not have a similar base frequency (see recommendations below), the Receiver will not be able to synchronize the frames to the start bit.

The following equations can be used to calculate the ratio of the incoming data rate and internal receiver baud rate.

$R_{\text{slow}} = \frac{(D+1)S}{S-1+D\cdot S+S_F}$	$R_{\text{fast}} = \frac{(D+2)S}{(D+1)S+S_M}$

- D: Sum of character size and parity size (D = 5 to 10 bit)
- S: Samples per bit. S = 16 for Normal Speed mode and S = 8 for Double Speed mode.
- S_F: First sample number used for majority voting. S_F = 8 for normal speed and S_F = 4 for Double Speed mode.
- S_M: Middle sample number used for majority voting. S_M = 9 for normal speed and S_M = 5 for Double Speed mode.
- R_{slow} : is the ratio of the slowest incoming data rate that can be accepted in relation to the receiver baud rate. R_{fast} is the ratio of the fastest incoming data rate that can be accepted in relation to the receiver baud rate.

The following tables list the maximum receiver baud rate error that can be tolerated. Note that Normal Speed mode has higher toleration of baud rate variations.

Table 21-2. Recommended Maximum Receiver Baud Rate Error for Normal Speed Mode (U2X = 0)

D # (Data+Parity Bit)	R_{slow} [%]	R_{fast} [%]	Max. Total Error [%]	Recommended Max. Receiver Error [%]
5	93.20	106.67	+6.67/-6.8	±3.0
6	94.12	105.79	+5.79/-5.88	±2.5
7	94.81	105.11	+5.11/-5.19	±2.0
8	95.36	104.58	+4.58/-4.54	±2.0
9	95.81	104.14	+4.14/-4.19	±1.5
10	96.17	103.78	+3.78/-3.83	±1.5

Table 21-3. Recommended Maximum Receiver Baud Rate Error for Double Speed Mode (U2X = 1)

D # (Data+Parity Bit)	R_{slow} [%]	R_{fast} [%]	Max Total Error [%]	Recommended Max Receiver Error [%]
5	94.12	105.66	+5.66/-5.88	±2.5
6	94.92	104.92	+4.92/-5.08	±2.0
7	95.52	104,35	+4.35/-4.48	±1.5
8	96.00	103.90	+3.90/-4.00	±1.5
9	96.39	103.53	+3.53/-3.61	±1.5
10	96.70	103.23	+3.23/-3.30	±1.0

The recommendations of the maximum receiver baud rate error was made under the assumption that the Receiver and Transmitter equally divides the maximum total error.

There are two possible sources for the receivers baud rate error. The Receiver's system clock (EXTCLK) will always have some minor instability over the supply voltage range and the temperature range. When using a crystal to generate the system clock, this is rarely a problem, but for a resonator, the system clock may differ more than 2% depending of the resonator's tolerance. The second source for the error is more controllable. The baud rate generator can not always do an exact division of the system frequency to get the baud rate wanted. In this case an UBRRn value that gives an acceptable low error can be used if possible.

21.10. Multi-Processor Communication Mode

Setting the Multi-Processor Communication mode (MPCMn) bit in UCSRnA enables a filtering function of incoming frames received by the USART Receiver. Frames that do not contain address information will be ignored and not put into the receive buffer. This effectively reduces the number of incoming frames that has to be handled by the CPU, in a system with multiple MCUs that communicate via the same serial bus. The Transmitter is unaffected by the MPCMn setting, but has to be used differently when it is a part of a system utilizing the Multi-processor Communication mode.

If the Receiver is set up to receive frames that contain 5 to 8 data bits, then the first stop bit indicates if the frame contains data or address information. If the Receiver is set up for frames with 9 data bits, then the ninth bit (RXB8) is used for identifying address and data frames. When the frame type bit (the first stop or the ninth bit) is '1', the frame contains an address. When the frame type bit is '0', the frame is a data frame.

The Multi-Processor Communication mode enables several slave MCUs to receive data from a master MCU. This is done by first decoding an address frame to find out which MCU has been addressed. If a particular slave MCU has been addressed, it will receive the following data frames as normal, while the other slave MCUs will ignore the received frames until another address frame is received.

21.10.1. Using MPCMn

For an MCU to act as a master MCU, it can use a 9-bit character frame format (UCSZ1=7). The ninth bit (TXB8) must be set when an address frame (TXB8=1) or cleared when a data frame (TXB=0) is being transmitted. The slave MCUs must in this case be set to use a 9-bit character frame format.

The following procedure should be used to exchange data in Multi-Processor Communication Mode:

1. All Slave MCUs are in Multi-Processor Communication mode (MPCM in UCSRnA is set).
2. The Master MCU sends an address frame, and all slaves receive and read this frame. In the Slave MCUs, the RXC Flag in UCSRnA will be set as normal.
3. Each Slave MCU reads the UDRn Register and determines if it has been selected. If so, it clears the MPCM bit in UCSRnA, otherwise it waits for the next address byte and keeps the MPCM setting.
4. The addressed MCU will receive all data frames until a new address frame is received. The other Slave MCUs, which still have the MPCM bit set, will ignore the data frames.
5. When the last data frame is received by the addressed MCU, the addressed MCU sets the MPCM bit and waits for a new address frame from master. The process then repeats from step 2.

Using any of the 5- to 8-bit character frame formats is possible, but impractical since the Receiver must change between using n and n+1 character frame formats. This makes full-duplex operation difficult since the Transmitter and Receiver uses the same character size setting. If 5- to 8-bit character frames are used, the Transmitter must be set to use two stop bit (USBS = 1) since the first stop bit is used for indicating the frame type.

Do not use Read-Modify-Write instructions (SBI and CBI) to set or clear the MPCM bit. The MPCM bit shares the same I/O location as the TXC Flag and this might accidentally be cleared when using SBI or CBI instructions.

21.11. Examples of Baud Rate Setting

For standard crystal and resonator frequencies, the most commonly used baud rates for asynchronous operation can be generated by using the UBRRn settings as listed in the table below.

UBRRn values which yield an actual baud rate differing less than 0.5% from the target baud rate, are bold in the table. Higher error ratings are acceptable, but the Receiver will have less noise resistance when the error ratings are high, especially for large serial frames (see also section *Asynchronous Operational Range*). The error values are calculated using the following equation:

$$Error\left[\%\right] = \left(\frac{\mathrm{BaudRate}_{\mathrm{Closest\ Match}}}{\mathrm{BaudRate}} - 1\right)^2 \ 100\,\%$$

Table 21-4. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies

Baud Rate [bps]	f_{osc} = 1.0000MHz				f_{osc} = 1.8432MHz				f_{osc} = 2.0000MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	25	0.2%	51	0.2%	47	0.0%	95	0.0%	51	0.2%	103	0.2%
4800	12	0.2%	25	0.2%	23	0.0%	47	0.0%	25	0.2%	51	0.2%
9600	6	-7.0%	12	0.2%	11	0.0%	23	0.0%	12	0.2%	25	0.2%
14.4k	3	8.5%	8	-3.5%	7	0.0%	15	0.0%	8	-3.5%	16	2.1%
19.2k	2	8.5%	6	-7.0%	5	0.0%	11	0.0%	6	-7.0%	12	0.2%
28.8k	1	8.5%	3	8.5%	3	0.0%	7	0.0%	3	8.5%	8	-3.5%
38.4k	1	-18.6%	2	8.5%	2	0.0%	5	0.0%	2	8.5%	6	-7.0%
57.6k	0	8.5%	1	8.5%	1	0.0%	3	0.0%	1	8.5%	3	8.5%
76.8k	–	–	1	-18.6%	1	-25.0%	2	0.0%	1	-18.6%	2	8.5%
115.2k	–	–	0	8.5%	0	0.0%	1	0.0%	0	8.5%	1	8.5%
230.4k	–	–	–	–	–	–	0	0.0%	–	–	–	–
250k	–	–	–	–	–	–	–	–	–	–	0	0.0%
Max.(1)	62.5kbps		125kbps		115.2kbps		230.4kbps		125kbps		250kbps	

Note: 1. UBRRn = 0, Error = 0.0%

Table 21-5. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies

Baud Rate [bps]	f_{osc} = 3.6864MHz				f_{osc} = 4.0000MHz				f_{osc} = 7.3728MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	95	0.0%	191	0.0%	103	0.2%	207	0.2%	191	0.0%	383	0.0%
4800	47	0.0%	95	0.0%	51	0.2%	103	0.2%	95	0.0%	191	0.0%
9600	23	0.0%	47	0.0%	25	0.2%	51	0.2%	47	0.0%	95	0.0%
14.4k	15	0.0%	31	0.0%	16	2.1%	34	-0.8%	31	0.0%	63	0.0%
19.2k	11	0.0%	23	0.0%	12	0.2%	25	0.2%	23	0.0%	47	0.0%
28.8k	7	0.0%	15	0.0%	8	-3.5%	16	2.1%	15	0.0%	31	0.0%
38.4k	5	0.0%	11	0.0%	6	-7.0%	12	0.2%	11	0.0%	23	0.0%
57.6k	3	0.0%	7	0.0%	3	8.5%	8	-3.5%	7	0.0%	15	0.0%
76.8k	2	0.0%	5	0.0%	2	8.5%	6	-7.0%	5	0.0%	11	0.0%
115.2k	1	0.0%	3	0.0%	1	8.5%	3	8.5%	3	0.0%	7	0.0%
230.4k	0	0.0%	1	0.0%	0	8.5%	1	8.5%	1	0.0%	3	0.0%
250k	0	-7.8%	1	-7.8%	0	0.0%	1	0.0%	1	-7.8%	3	-7.8%
0.5M	–	–	0	-7.8%	–	–	0	0.0%	0	-7.8%	1	-7.8%
1M	–	–	–	–	–	–	–	–	–	–	0	-7.8%
Max.(1)	230.4kbps		460.8kbps		250kbps		0.5Mbps		460.8kbps		921.6kbps	

(1) UBRRn = 0, Error = 0.0%

Table 21-6. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies

Baud Rate [bps]	f_{osc} = 8.0000MHz				f_{osc} = 11.0592MHz				f_{osc} = 14.7456MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	–	–	2	-7.8%	1	-7.8%	3	-7.8%
1M	–	–	0	0.0%	–	–	–	–	0	-7.8%	1	-7.8%
Max.(1)	0.5Mbps		1Mbps		691.2kbps		1.3824Mbps		921.6kbps		1.8432Mbps	

(1) UBRRn = 0, Error = 0.0%

Table 21-7. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies

Baud Rate [bps]	f_{osc} = 16.0000MHz				f_{osc} = 18.4320MHz				f_{osc} = 20.0000MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn	Error
2400	416	-0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.2%	416	-0.1%	239	0.0%	479	0.0%	259	0.2%	520	0.0%
9600	103	0.2%	207	0.2%	119	0.0%	239	0.0%	129	0.2%	259	0.2%
14.4k	68	0.6%	138	-0.1%	79	0.0%	159	0.0%	86	-0.2%	173	-0.2%
19.2k	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8k	34	-0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	-0.2%
38.4k	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	-1.4%	64	0.2%
57.6k	16	2.1%	34	-0.8%	19	0.0%	39	0.0%	21	-1.4%	42	0.9%
76.8k	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	-1.4%
115.2k	8	-3.5%	16	2.1%	9	0.0%	19	0.0%	10	-1.4%	21	-1.4%
230.4k	3	8.5%	8	-3.5%	4	0.0%	9	0.0%	4	8.5%	10	-1.4%
250k	3	0.0%	7	0.0%	4	-7.8%	8	2.4%	4	0.0%	9	0.0%
0.5M	1	0.0%	3	0.0%	–	–	4	-7.8%	–	–	4	0.0%
1M	0	0.0%	1	0.0%	–	–	–	–	–	–	–	–
Max.(1)	1Mbps		2Mbps		1.152Mbps		2.304Mbps		1.25Mbps		2.5Mbps	

(1) UBRRn = 0, Error = 0.0%

Related Links

Asynchronous Operational Range on page 237

21.12. Register Description

21.12.1. USART I/O Data Register n

The USART Transmit Data Buffer Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register or UDRn. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDR1 Register location. Reading the UDRn Register location will return the contents of the Receive Data Buffer Register (RXB).

For 5-, 6-, or 7-bit characters the upper unused bits will be ignored by the Transmitter and set to zero by the Receiver.

The transmit buffer can only be written when the UDRE Flag in the UCSRnA Register is set. Data written to UDRn when the UCSRnA.UDRE Flag is not set, will be ignored by the USART Transmitter n. When data is written to the transmit buffer, and the Transmitter is enabled, the Transmitter will load the data into the Transmit Shift Register when the Shift Register is empty. Then the data will be serially transmitted on the TxDn pin.

The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, do not use Read-Modify-Write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.

Name: UDRn
Offset: 0xC6 + n*0x08 [n=0..1]
Reset: 0x00
Property: -

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

Bits 7:0 – TXB / RXB[7:0]: USART Transmit / Receive Data Buffer

21.12.2. USART Control and Status Register n A

Name: UCSR0A, UCSR1A
Offset: 0xC0 + n*0x08 [n=0..1]
Reset: 0x20
Property: -

Bit	7	6	5	4	3	2	1	0
	RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM
Access	R	R/W	R	R	R	R	R/W	R/W
Reset	0	0	1	0	0	0	0	0

Bit 7 – RXC: 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 RXC bit will become zero. The RXC Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE bit).

Bit 6 – TXC: 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 (UDRn). The TXC 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 TXC Flag can generate a Transmit Complete interrupt (see description of the TXCIE bit).

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

Bit 4 – FE: 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 (UDRn) 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 UCSRnA.

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

Bit 3 – DOR: 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 (UDRn) is read. Always set this bit to zero when writing to UCSRnA.

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

Bit 2 – UPE: 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 (UCSRnC.UPM1 = 1). This bit is valid until the receive buffer (UDRn) is read. Always set this bit to zero when writing to UCSRnA.

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

Bit 1 – U2X: Double the USART Transmission Speed
This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation.

Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.

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

Bit 0 – MPCM: Multi-processor Communication Mode
This bit enables the Multi-processor Communication mode. When the MPCM bit is written to one, all the incoming frames received by the USART Receiver n that do not contain address information will be ignored. The Transmitter is unaffected by the MPCM setting. Refer to Multi-Processor Communication Mode for details.

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

21.12.3. USART Control and Status Register n B

Name: UCSR0B, UCSR1B
Offset: 0xC1 + n*0x08 [n=0..1]
Reset: 0x00
Property: -

Bit	7	6	5	4	3	2	1	0
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
Access	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – RXCIE: RX Complete Interrupt Enable
Writing this bit to one enables interrupt on the UCSRnA.RXC Flag. A USART Receive Complete interrupt will be generated only if the RXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXC bit in UCSRnA is set.

Bit 6 – TXCIE: TX Complete Interrupt Enable
Writing this bit to one enables interrupt on the TXC Flag. A USART Transmit Complete interrupt will be generated only if the TXCIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXC bit in UCSRnA is set.

Bit 5 – UDRIE: USART Data Register Empty Interrupt Enable
Writing this bit to one enables interrupt on the UDRE Flag. A Data Register Empty interrupt will be generated only if the UDRIE bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDRE bit in UCSRnA is set.

Bit 4 – RXEN: Receiver Enable
Writing this bit to one enables the USART Receiver. The Receiver will override normal port operation for the RxDn pin when enabled. Disabling the Receiver will flush the receive buffer invalidating the FE, DOR, and UPE Flags.

Bit 3 – TXEN: Transmitter Enable
Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxDn pin when enabled. The disabling of the Transmitter (writing TXEN to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the Transmit Shift Register and Transmit Buffer Register do not contain data to be transmitted. When disabled, the Transmitter will no longer override the TxDn port.

Bit 2 – UCSZ2: Character Size
The UCSZ2 bits combined with the UCSZ[1:0] bit in UCSRnC sets the number of data bits (Character Size) in a frame the Receiver and Transmitter use.

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

Bit 1 – RXB8: Receive Data Bit 8
RXB8 is the ninth data bit of the received character when operating with serial frames with nine data bits. Must be read before reading the low bits from UDRn.

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

Bit 0 – TXB8: Transmit Data Bit 8
TXB8 is the ninth data bit in the character to be transmitted when operating with serial frames with nine data bits. Must be written before writing the low bits to UDRn.

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

21.12.4. USART Control and Status Register n C

Name: UCSR0C, UCSR1C
Offset: 0xC2 + n*0x08 [n=0..1]
Reset: 0x06
Property: -

Bit	7	6	5	4	3	2	1	0
	UMSEL[1:0]		UPM[1:0]		USBS	UCSZ1 / UDORD	UCSZ0 / UCPHA	UCPOL
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	0

Bits 7:6 – UMSEL[1:0]: USART Mode Select
These bits select the mode of operation of the USARTn

Table 21-8. USART Mode Selection

UMSEL[1:0]	Mode
00	Asynchronous USART
01	Synchronous USART
10	Reserved
11	Master SPI (MSPIM)(1)

Note:
1. The UDORD, UCPHA, and UCPOL can be set in the same write operation where the MSPIM is enabled.

Bits 5:4 – UPM[1:0]: USART Parity Mode
These bits enable and set type of parity generation and check. If enabled, the Transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The Receiver will generate a parity value for the incoming data and compare it to the UPM setting. If a mismatch is detected, the UPE Flag in UCSRnA will be set.

Table 21-9. USART Mode Selection

UPM[1:0]	ParityMode
00	Disabled
01	Reserved
10	Enabled, Even Parity
11	Enabled, Odd Parity

These bits are reserved in Master SPI Mode (MSPIM).

Bit 3 – USBS: USART Stop Bit Select
This bit selects the number of stop bits to be inserted by the Transmitter n. The Receiver ignores this setting.

Table 21-10. Stop Bit Settings

USBS	Stop Bit(s)
0	1-bit
1	2-bit

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

Bit 2 – UCSZ1 / UDORD: USART Character Size / Data Order
UCSZ1[1:0]: USART Modes: The UCSZ1[1:0] bits combined with the UCSZ12 bit in UCSR1B sets the number of data bits (Character Size) in a frame the Receiver and Transmitter use.

Table 21-11. Character Size Settings

UCSZ1[2:0]	Character Size
000	5-bit
001	6-bit
010	7-bit
011	8-bit
100	Reserved
101	Reserved
110	Reserved
111	9-bit

UDORD0: Master SPI Mode: When set to one the LSB of the data word is transmitted first. When set to zero the MSB of the data word is transmitted first. Refer to the *USART in SPI Mode - Frame Formats* for details.

Bit 1 – UCSZ0 / UCPHA: USART Character Size / Clock Phase
UCSZ0: USART Modes: Refer to UCSZ1.

UCPHA: Master SPI Mode: The UCPHA bit setting determine if data is sampled on the leasing edge (first) or tailing (last) edge of XCK. Refer to the *SPI Data Modes and Timing* for details.

Bit 0 – UCPOL: Clock Polarity
USART n Modes: This bit is used for synchronous mode only. Write this bit to zero when asynchronous mode is used. The UCPOL bit sets the relationship between data output change and data input sample, and the synchronous clock (XCKn).

Table 21-12. USART Clock Polarity Settings

UCPOL	Transmitted Data Changed (Output of TxDn Pin)	Received Data Sampled (Input on RxDn Pin)
0	Rising XCKn Edge	Falling XCKn Edge
1	Falling XCKn Edge	Rising XCKn Edge

Master SPI Mode: The UCPOL bit sets the polarity of the XCKn clock. The combination of the UCPOL and UCPHA bit settings determine the timing of the data transfer. Refer to the *SPI Data Modes and Timing* for details.

21.12.5. USART Baud Rate n Register Low and High byte

The UBRRnL and UBRRnH register pair represents the 16-bit value, UBRRn (n=0,1).The low byte [7:0] (suffix L) is accessible at the original offset. The high byte [15:8] (suffix H) can be accessed at offset + 0x01. For more details on reading and writing 16-bit registers, refer to Accessing 16-bit Registers.

Name: UBRR0L and UBRR0H, UBRR1L and UBRR1H
Offset: 0xC4 + n*0x08 [n=0..1]
Reset: 0x00
Property: -

Bit	15	14	13	12	11	10	9	8
					UBRR[11:8]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

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

Bits 11:0 – UBRR[11:0]: USART Baud Rate
This is a 12-bit register which contains the USART baud rate. The UBRRnH contains the four most significant bits and the UBRRnL contains the eight least significant bits of the USART n baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing UBRRnL will trigger an immediate update of the baud rate prescaler.

31. Register Summary

Offset	Name	Bit Pos.								
0x20	PINA	7:0	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0
0x21	DDRA	7:0	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
0x22	PORTA	7:0	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0
0x23	PINB	7:0	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
0x24	DDRB	7:0	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
0x25	PORTB	7:0	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
0x26	PINC	7:0	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0
0x27	DDRC	7:0	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
0x28	PORTC	7:0	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0
0x29	PIND	7:0	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
0x2A	DDRD	7:0	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
0x2B	PORTD	7:0	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
0x2C ... 0x34	Reserved									
0x35	TIFR0	7:0						OCFB	OCFA	TOV
0x36	TIFR1	7:0			ICF			OCFB	OCFA	TOV
0x37	TIFR2	7:0						OCFB	OCFA	TOV
0x38 ... 0x3A	Reserved									
0x3B	PCIFR	7:0					PCIF3	PCIF2	PCIF1	PCIF0
0x3C	EIFR	7:0						INTF2	INTF1	INTF0
0x3D	EIMSK	7:0						INT2	INT1	INT0
0x3E	GPIOR0	7:0	GPIOR0[7:0]							
0x3F	EECR	7:0			EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE
0x40	EEDR	7:0	EEDR[7:0]							
0x41	EEARL and EEARH	7:0	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0
0x42		15:8							EEAR9	EEAR8
0x43	GTCCR	7:0	TSM						PSRASY	PSRSYNC
0x44	TCCR0A	7:0	COM0A1	COM0A0	COM0B1	COM0B0			WGM01	WGM00
0x45	TCCR0B	7:0	FOC0A	FOC0B			WGM02	CS0[2:0]		
0x46	TCNT0	7:0	TCNT0[7:0]							
0x47	OCR0A	7:0	OCR0A[7:0]							
0x48	OCR0B	7:0	OCR0B[7:0]							
0x49	Reserved									
0x4A	GPIOR1	7:0	GPIOR1[7:0]							
0x4B	GPIOR2	7:0	GPIOR2[7:0]							
0x4C	SPCR0	7:0	SPIE0	SPE0	DORD0	MSTR0	CPOL0	CPHA0	SPR01	SPR00
0x4D	SPSR0	7:0	SPIF0	WCOL0						SPI2X0
0x4E	SPDR0	7:0	SPID[7:0]							
0x4F	Reserved									
0x50	ACSR	7:0	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0
0x51	OCDR	7:0	IDRD/OCDR7	OCDR6	OCDR5	OCDR4	OCDR3	OCDR2	OCDR1	OCDR0

Offset	Name	Bit Pos.								
0x52	Reserved									
0x53	SMCR	7:0					SM2	SM1	SM0	SE
0x54	MCUSR	7:0				JTRF	WDRF	BORF	EXTRF	PORF
0x55	MCUCR	7:0	JTD	BODS	BODSE	PUD			IVSEL	IVCE
0x56	Reserved									
0x57	SPMCSR	7:0	SPMIE	RWWSB	SIGRD	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN
0x58 ... 0x5C	Reserved									
0x5D	SPL and SPH	7:0	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
0x5E		15:8					SP11	SP10	SP9	SP8
0x5F	SREG	7:0	I	T	H	S	V	N	Z	C
0x60	WDTCSR	7:0	WDIF	WDIE	WDP[3]	WDCE	WDE	WDP[2:0]		
0x61	CLKPR	7:0	CLKPCE				CLKPS3	CLKPS2	CLKPS1	CLKPS0
0x62 ... 0x63	Reserved									
0x64	PRR0	7:0	PRTWI	PRTIM2	PRTIM0	PRUSART1	PRTIM1	PRSPI0	PRUSART0	PRADC
0x65	Reserved									
0x66	OSCCAL	7:0	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0
0x67	Reserved									
0x68	PCICR	7:0					PCIE3	PCIE2	PCIE1	PCIE0
0x69	EICRA	7:0			ISC21	ISC20	ISC11	ISC10	ISC01	ISC00
0x6A	Reserved									
0x6B	PCMSK0	7:0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0
0x6C	PCMSK1	7:0	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8
0x6D	PCMSK2	7:0	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16
0x6E	TIMSK0	7:0						OCIEB	OCIEA	TOIE
0x6F	TIMSK1	7:0			ICIE			OCIEB	OCIEA	TOIE
0x70	TIMSK2	7:0						OCIEB	OCIEA	TOIE
0x71 ... 0x72	Reserved									
0x73	PCMSK3	7:0	PCINT31	PCINT30	PCINT29	PCINT28	PCINT27	PCINT26	PCINT25	PCINT24
0x74 ... 0x77	Reserved									
0x78	ADCL and ADCH	7:0	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
0x79		15:8							ADC9	ADC8
0x7A	ADCSRA	7:0	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
0x7B	ADCSRB	7:0		ACME				ADTS2	ADTS1	ADTS0
0x7C	ADMUX	7:0	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0
0x7D	Reserved									
0x7E	DIDR0	7:0	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D
0x7F	DIDR1	7:0	Reserved5	Reserved4	Reserved3	Reserved2	Reserved1	Reserved0	AIN1D	AIN0D
0x80	TCCR1A	7:0	COM1	COM1	COM1	COM1			WGM11	WGM10
0x81	TCCR1B	7:0	ICNC1	ICES1		WGM13	WGM12	CS12	CS11	CS10

Offset	Name	Bit Pos.								
0x82	TCCR1C	7:0	FOC1A	FOC1B						
0x83	Reserved									
0x84	TCNT1L and	7:0	TCNT1[7:0]							
0x85	TCNT1H	15:8	TCNT1[15:8]							
0x86	ICR1L and ICR1H	7:0	ICR1[7:0]							
0x87		15:8	ICR1[15:8]							
0x88	OCR1AL and	7:0	OCR1A[7:0]							
0x89	OCR1AH	15:8	OCR1A[15:8]							
0x8A	OCR1BL and	7:0	OCR1B[7:0]							
0x8B	OCR1BH	15:8	OCR1B[15:8]							
0x8C ... 0xAF	Reserved									
0xB0	TCCR2A	7:0	COM2A1	COM2A0	COM2B1	COM2B0			WGM21	WGM20
0xB1	TCCR2B	7:0	FOC2A	FOC2B			WGM22	CS2[2:0]		
0xB2	TCNT2	7:0	TCNT2[7:0]							
0xB3	OCR2A	7:0	OCR2A[7:0]							
0xB4	OCR2B	7:0	OCR2B[7:0]							
0xB5	Reserved									
0xB6	ASSR	7:0		EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB
0xB7	Reserved									
0xB8	TWBR	7:0	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0
0xB9	TWSR	7:0	TWS7	TWS6	TWS5	TWS4	TWS3		TWPS[1:0]	
0xBA	TWAR	7:0	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
0xBB	TWDR	7:0	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0
0xBC	TWCR	7:0	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN		TWIE
0xBD	TWAMR	7:0	TWAM6	TWAM5	TWAM4	TWAM3	TWAM2	TWAM1	TWAM0	
0xBE ... 0xBF	Reserved									
0xC0	UCSR0A	7:0	RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM
0xC1	UCSR0B	7:0	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
0xC2	UCSR0C	7:0	UMSEL[1:0]		UPM[1:0]		USBS	UCSZ1 / UDORD	UCSZ0 / UCPHA	UCPOL
0xC3	Reserved									
0xC4	UBRR0L and	7:0	UBRR[7:0]							
0xC5	UBRR0H	15:8					UBRR[11:8]			
0xC6	UDR0	7:0	TXB / RXB[7:0]							
0xC7	Reserved									
0xC8	UCSR1A	7:0	RXC	TXC	UDRE	FE	DOR	UPE	U2X	MPCM
0xC9	UCSR1B	7:0	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
0xCA	UCSR1C	7:0	UMSEL[1:0]		UPM[1:0]		USBS	UCSZ1 / UDORD	UCSZ0 / UCPHA	UCPOL
0xCB	Reserved									
0xCC	UBRR1L and	7:0	UBRR[7:0]							
0xCD	UBRR1H	15:8					UBRR[11:8]			
0xCE	UDR1	7:0	TXB / RXB[7:0]							

32. Instruction Set Summary

ARITHMETIC AND LOGIC INSTRUCTIONS					
Mnemonics	Operands	Description	Operation	Flags	#Clocks
ADD	Rd, Rr	Add two Registers without Carry	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add two Registers with Carry	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract two Registers with Carry	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract Constant from Reg with Carry.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd · Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd · K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← 0xFF - Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← 0x00 - Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd · (0xFF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd - 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd · Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← 0xFF	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2
MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd x Rr	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2

BRANCH INSTRUCTIONS					
Mnemonics	Operands	Description	Operation	Flags	#Clocks
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
JMP(1)	k	Direct Jump	PC ← k	None	3
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3

BRANCH INSTRUCTIONS					
Mnemonics	Operands	Description	Operation	Flags	#Clocks
ICALL		Indirect Call to (Z)	PC ← Z	None	3
CALL(1)	k	Direct Subroutine Call	PC ← k	None	4
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP	Rd,Rr	Compare	Rd - Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd - Rr - C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd - K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	A, b	Skip if Bit in I/O Register Cleared	if (I/O(A,b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIS	A, b	Skip if Bit in I/O Register is Set	if (I/O(A,b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1/2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1/2

BIT AND BIT-TEST INSTRUCTIONS					
Mnemonics	Operands	Description	Operation	Flags	#Clocks
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2

BIT AND BIT-TEST INSTRUCTIONS					
Mnemonics	Operands	Description	Operation	Flags	#Clocks
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C¬Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0...6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3...0)←Rd(7...4),Rd(7...4)¬Rd(3...0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Two's Complement Overflow.	V ← 1	V	1
CLV		Clear Two's Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1

DATA TRANSFER INSTRUCTIONS					
Mnemonics	Operands	Description	Operation	Flags	#Clocks
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Increment	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Decrement	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Increment	Rd ← (Y), Y ← Y + 1	None	2

DATA TRANSFER INSTRUCTIONS					
Mnemonics	**Operands**	**Description**	**Operation**	**Flags**	**#Clocks**
LD	Rd, - Y	Load Indirect and Pre-Decrement	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Increment	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Decrement	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Increment	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Decrement	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Increment	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Decrement	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Increment	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Decrement	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z+1	None	3
SPM		Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, A	In from I/O Location	Rd ← I/O (A)	None	1
OUT	A, Rr	Out to I/O Location	I/O (A) ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2

MCU CONTROL INSTRUCTIONS					
Mnemonics	**Operands**	**Description**	**Operation**	**Flags**	**#Clocks**
NOP		No Operation	No Operation	None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For On-chip Debug Only	None	N/A