Quelle: http://www.atmel.com

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 135 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-Chip 2-cycle Multiplier
- Non-volatile Program and Data Memories
  - 16/32K Bytes of In-System Self-Programmable Flash (ATmega16U4/ATmega32U4)
  - 1.25/2.5K Bytes Internal SRAM (ATmega16U4/ATmega32U4)
  - 512Bytes/1K Bytes Internal EEPROM (ATmega16U4/ATmega32U4)
  - 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
    - All supplied parts are preprogramed with a default USB bootloader
  - Programming Lock for Software Security
- 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
- USB 2.0 Full-speed/Low Speed Device Module with Interrupt on Transfer Completion
  - Complies fully with Universal Serial Bus Specification Rev 2.0
  - Supports data transfer rates up to 12 Mbit/s and 1.5 Mbit/s
  - Endpoint 0 for Control Transfers: up to 64-bytes
  - 6 Programmable Endpoints with IN or Out Directions and with Bulk, Interrupt or Isochronous Transfers
  - Configurable Endpoints size up to 256 bytes in double bank mode
  - Fully independent 832 bytes USB DPRAM for endpoint memory allocation
  - Suspend/Resume Interrupts
  - CPU Reset possible on USB Bus Reset detection
  - 48 MHz from PLL for Full-speed Bus Operation
  - USB Bus Connection/Disconnection on Microcontroller Request
  - Crystal-less operation for Low Speed mode
- Peripheral Features
  - On-chip PLL for USB and High Speed Timer: 32 up to 96 MHz operation
  - One 8-bit Timer/Counter with Separate Prescaler and Compare Mode
  - Two 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
  - One 10-bit High-Speed Timer/Counter with PLL (64 MHz) and Compare Mode
  - Four 8-bit PWM Channels
  - Four PWM Channels with Programmable Resolution from 2 to 16 Bits
  - Six PWM Channels for High Speed Operation, with Programmable Resolution from 2 to 11 Bits
  - Output Compare Modulator
  - 12-channels, 10-bit ADC (features Differential Channels with Programmable Gain)
  - Programmable Serial USART with Hardware Flow Control
  - Master/Slave SPI Serial Interface

8-bit AVR® Microcontroller with 16/32K Bytes of ISP Flash and USB Controller

ATmega16U4
ATmega32U4

Preliminary

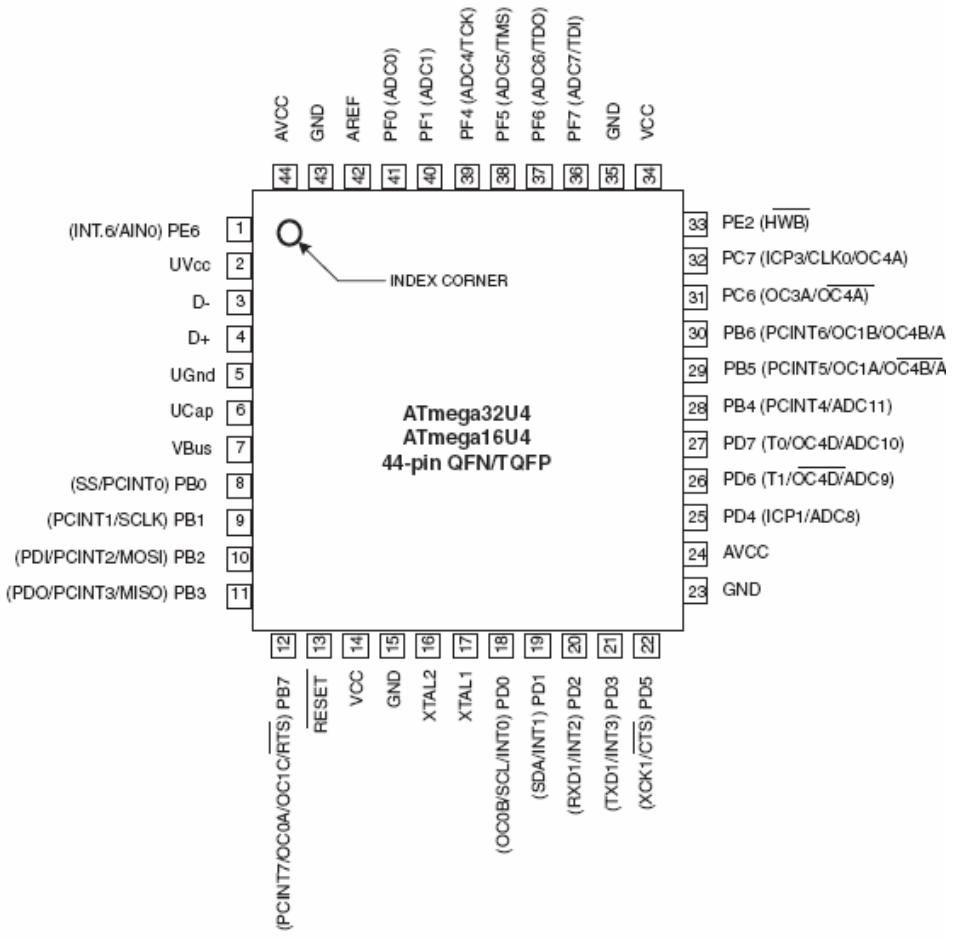
7766F–AVR–11/10

- Byte Oriented 2-wire Serial Interface
- Programmable Watchdog Timer with Separate On-chip Oscillator
- On-chip Analog Comparator
- Interrupt and Wake-up on Pin Change
- On-chip Temperature Sensor

- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal 8 MHz Calibrated Oscillator
  - Internal clock prescaler & On-the-fly Clock Switching (Int RC / Ext Osc)
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - All I/O combine CMOS outputs and LVTTL inputs
  - 26 Programmable I/O Lines
  - 44-lead TQFP Package, 10x10mm
  - 44-lead QFN Package, 7x7mm
- Operating Voltages
  - 2.7 - 5.5V
- Operating temperature
  - Industrial (-40°C to +85°C)
- Maximum Frequency
  - 8 MHz at 2.7V - Industrial range
  - 16 MHz at 4.5V - Industrial range

Note: 1. See "Data Retention" on page 8 for details.

Figure 1-1. Pinout ATmega16U4/ATmega32U4

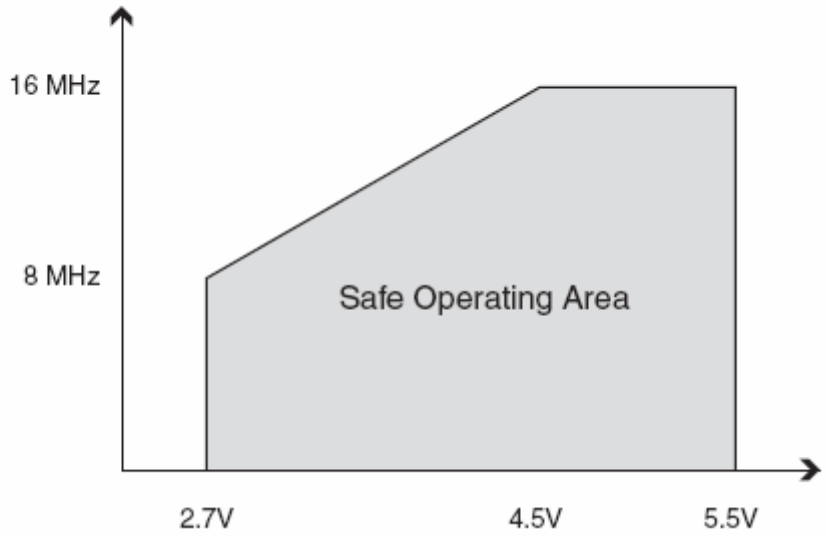


29.1 Absolute Maximum Ratings*

Operating Temperature	-40°C to +85°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{\text{RESET}}$ and VBUS with respect to Ground[8]	-0.5V to V_{CC}+0.5V
Voltage on $\overline{\text{RESET}}$ with respect to Ground	-0.5V to +13.0V
Voltage on VBUS with respect to Ground	-0.5V to +6.0V
Maximum Operating Voltage	6.0V
DC Current per I/O Pin	40.0 mA
DC Current V_{CC} and GND Pins	200.0 mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

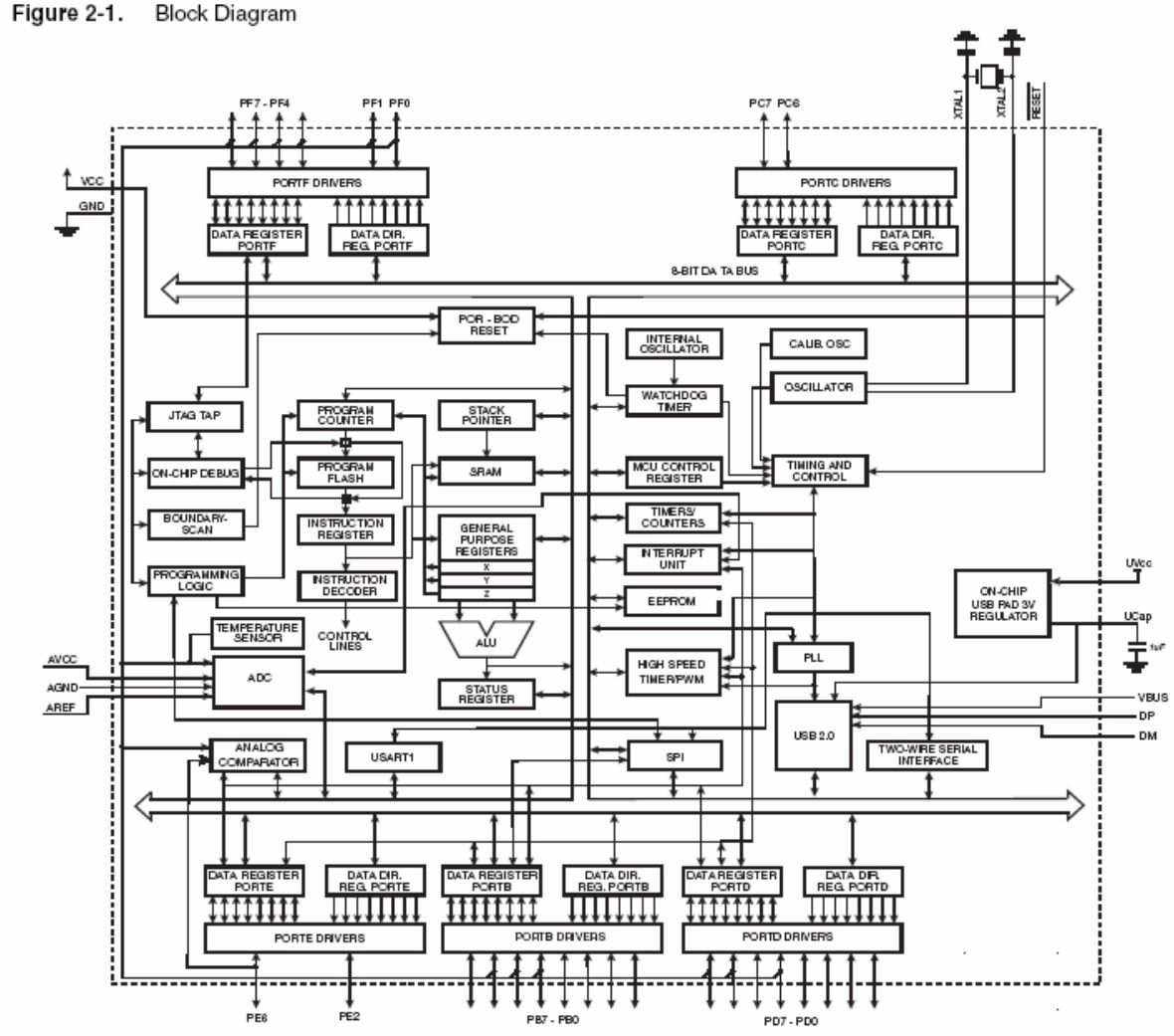
Figure 29-2. Maximum Frequency vs. V_{CC}, ATmega16U4/ATmega32U4



33.2 ATmega32U4

Speed (MHz)	Power Supply	Ordering Code	Default Oscillator	Package	Operation Range
16	2.7 - 5.5 V	ATmega32U4-AU	External XTAL	44ML	Industrial (-40° to +85°C)
		ATmega32U4RC-AU	Internal Calib. RC		
		ATmega32U4-MU	External XTAL	44PW	
		ATmega32U4RC-MU	Internal Calib. RC		

Figure 2-1. Block Diagram



31. Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xFF)	Reserved	-	-	-	-	-	-	-	-	
(0xFE)	Reserved	-	-	-	-	-	-	-	-	
(0xFD)	Reserved	-	-	-	-	-	-	-	-	
(0xFC)	Reserved	-	-	-	-	-	-	-	-	
(0xFB)	Reserved	-	-	-	-	-	-	-	-	
(0xFA)	Reserved	-	-	-	-	-	-	-	-	
(0xF9)	Reserved		-	-				-	-	
(0xF8)	Reserved	-	-	-	-	-	-	-	-	
(0xF7)	Reserved	-	-	-	-	-	-	-	-	
(0xF6)	Reserved	-	-	-	-	-	-	-	-	
(0xF5)	Reserved	-	-	-	-	-	-	-	-	
(0xF4)	UEINT	-	EPINT6:0							
(0xF3)	UEBCHX	-	-	-	-	-	BYCT10:8			
(0xF2)	UEBCLX	BYCT7:0								
(0xF1)	UEDATX	DAT7:0								
(0xF0)	UEIENX	FLERRE	NAKINE	-	NAKOUTE	RXSTPE	RXOUTE	STALLEDE	TXINE	
(0xEF)	UESTA1X	-	-	-	-	-	CTRLDIR	CURRBK1:0		
(0xEE)	UESTA0X	CFGOK	OVERFI	UNDERFI	-	DTSEQ1:0		NBUSYBK1:0		
(0xED)	UECFG1X		EPSIZE2:0			EPBK1:0		ALLOC	-	
(0xEC)	UECFG0X	EPTYPE1:0		-	-	-	-	-	EPDIR	
(0xEB)	UECONX	-	-	STALLRQ	STALLRQC	RSTDT	-	-	EPEN	
(0xEA)	UERST	-	EPRST6:0							
(0xE9)	UENUM	-	-	-	-	-	EPNUM2:0			
(0xE8)	UEINTX	FIFOCON	NAKINI	RWAL	NAKOUTI	RXSTPI	RXOUTI	STALLEDI	TXINI	
(0xE7)	Reserved			-	-	-	-			
(0xE6)	UDMFN	-	-	-	FNCERR	-	-	-	-	
(0xE5)	UDFNUMH	-	-	-	-	-	FNUM10:8			
(0xE4)	UDFNUML	FNUM7:0								
(0xE3)	UDADDR	ADDEN	UADD6:0							
(0xE2)	UDIEN	-	UPRSME	EORSME	WAKEUPE	EORSTE	SOFE	MSOFE	SUSPE	
(0xE1)	UDINT	-	UPRSMI	EORSMI	WAKEUPI	EORSTI	SOFI	MSOFI	SUSPI	
(0xE0)	UDCON	-	-	-	-	RSTCPU	LSM	RMWKUP	DETACH	
(0xDF)	Reserved									
(0xDE)	Reserved									
(0xDD)	Reserved									
(0xDC)	Reserved									
(0xDB)	Reserved									
(0xDA)	USBINT	-	-	-	-	-	-	-	VBUSTI	
(0xD9)	USBSTA	-	-	-	-	-	-	ID	VBUS	
(0xD8)	USBCON	USBE	-	FRZCLK	OTGPADE	-	-	-	VBUSTE	
(0xD7)	UHWCON	-	-	-	-	-	-	-	UVREGE	
(0xD6)	Reserved									
(0xD5)	Reserved									
(0xD4)	DT4	DT4H3	DT4H2	DT4H1	DT4H0	DT4L3	DT4L2	DT4L1	DT4L0	
(0xD3)	Reserved									
(0xD2)	OCR4D	Timer/Counter4 - Output Compare Register D								
(0xD1)	OCR4C	Timer/Counter4 - Output Compare Register C								
(0xD0)	OCR4B	Timer/Counter4 - Output Compare Register B								
(0xCF)	OCR4A	Timer/Counter4 - Output Compare Register A								
(0xCE)	UDR1	USART1 I/O Data Register								
(0xCD)	UBRR1H	-	-	-	-	USART1 Baud Rate Register High Byte				
(0xCC)	UBRR1L	USART1 Baud Rate Register Low Byte								
(0xCB)	Reserved	-	-	-	-	-	-	-	-	
(0xCA)	UCSR1C	UMSEL11	UMSEL10	UPM11	UPM10	USBS1	UCSZ11	UCSZ10	UCPOL1	
(0xC9)	UCSR1B	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81	
(0xC8)	UCSR1A	RXC1	TXC1	UDRE1	FE1	DOR1	PE1	U2X1	MPCM1	
(0xC7)	CLKSTA	-	-	-	-	-	-	RCON	EXTON	
(0xC6)	CLKSEL1	RCCKSEL3	RCCKSEL2	RCCKSEL1	RCCKSEL0	EXCKSEL3	EXCKSEL2	EXCKSEL1	EXCKSEL0	
(0xC5)	CLKSEL0	RCSUT1	RCSUT0	EXSUT1	EXSUT0	RCE	EXTE	-	CLKS	
(0xC4)	TCCR4E	TLOCK4	ENHC4	OC4OE5	OC4OE4	OC4OE3	OC4OE2	OC4OE1	OC4OE0	
(0xC3)	TCCR4D	FPIE4	FPEN4	FPNC4	FPES4	FPAC4	FPF4	WGM41	WGM40	
(0xC2)	TCCR4C	COM4A1S	COM4A0S	COM4B1S	COM4B0S	COM4D1S	COM4D0S	FOC4D	PWM4D	
(0xC1)	TCCR4B	PWM4X	PSR4	DTPS41	DTPS40	CS43	CS42	CS41	CS40	
(0xC0)	TCCR4A	COM4A1	COM4A0	COM4B1	COM4B0	FOC4A	FOC4B	PWM4A	PWM4B	
(0xBF)	TC4H	-	-	-	-	-	Timer/Counter4 High Byte			

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xBE)	TCNT4	Timer/Counter4 - Counter Register Low Byte								
(0xBD)	TWAMR	TWAM6	TWAM5	TWAM4	TWAM3	TWAM2	TWAM1	TWAM0	-	
(0xBC)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE	
(0xBB)	TWDR	2-wire Serial Interface Data Register								
(0xBA)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	
(0xB9)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	-	TWPS1	TWPS0	
(0xB8)	TWBR	2-wire Serial Interface Bit Rate Register								
(0xB7)	Reserved	-	-	-	-	-	-	-	-	
(0xB6)	Reserved	-								
(0xB5)	Reserved	-	-	-	-	-	-	-	-	
(0xB4)	Reserved	-	-	-	-	-	-	-	-	
(0xB3)	Reserved	-	-	-	-	-	-	-	-	
(0xB2)	Reserved	-	-	-	-	-	-	-	-	
(0xB1)	Reserved	-	-	-	-	-	-	-	-	
(0xB0)	Reserved	-	-	-	-	-	-	-	-	
(0xAF)	Reserved	-	-	-	-	-	-	-	-	
(0xAE)	Reserved	-	-	-	-	-	-	-	-	
(0xAD)	Reserved	-	-	-	-	-	-	-	-	
(0xAC)	Reserved	-	-	-	-	-	-	-	-	
(0xAB)	Reserved	-	-	-	-	-	-	-	-	
(0xAA)	Reserved	-	-	-	-	-	-	-	-	
(0xA9)	Reserved	-	-	-	-	-	-	-	-	
(0xA8)	Reserved	-	-	-	-	-	-	-	-	
(0xA7)	Reserved	-	-	-	-	-	-	-	-	
(0xA6)	Reserved	-	-	-	-	-	-	-	-	
(0xA5)	Reserved	-	-	-	-	-	-	-	-	
(0xA4)	Reserved	-	-	-	-	-	-	-	-	
(0xA3)	Reserved	-	-	-	-	-	-	-	-	
(0xA2)	Reserved	-	-	-	-	-	-	-	-	
(0xA1)	Reserved	-	-	-	-	-	-	-	-	
(0xA0)	Reserved	-	-	-	-	-	-	-	-	
(0x9F)	Reserved	-	-	-	-	-	-	-	-	
(0x9E)	Reserved	-	-	-	-	-	-	-	-	
(0x9D)	OCR3CH	Timer/Counter3 - Output Compare Register C High Byte								
(0x9C)	OCR3CL	Timer/Counter3 - Output Compare Register C Low Byte								
(0x9B)	OCR3BH	Timer/Counter3 - Output Compare Register B High Byte								
(0x9A)	OCR3BL	Timer/Counter3 - Output Compare Register B Low Byte								
(0x99)	OCR3AH	Timer/Counter3 - Output Compare Register A High Byte								
(0x98)	OCR3AL	Timer/Counter3 - Output Compare Register A Low Byte								
(0x97)	ICR3H	Timer/Counter3 - Input Capture Register High Byte								
(0x96)	ICR3L	Timer/Counter3 - Input Capture Register Low Byte								
(0x95)	TCNT3H	Timer/Counter3 - Counter Register High Byte								
(0x94)	TCNT3L	Timer/Counter3 - Counter Register Low Byte								
(0x93)	Reserved	-	-	-	-	-	-	-	-	
(0x92)	TCCR3C	FOC3A	-	-	-	-	-	-	-	
(0x91)	TCCR3B	ICNC3	ICES3	-	WGM33	WGM32	CS32	CS31	CS30	
(0x90)	TCCR3A	COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30	
(0x8F)	Reserved	-	-	-	-	-	-	-	-	
(0x8E)	Reserved	-	-	-	-	-	-	-	-	
(0x8D)	OCR1CH	Timer/Counter1 - Output Compare Register C High Byte								
(0x8C)	OCR1CL	Timer/Counter1 - Output Compare Register C Low Byte								
(0x8B)	OCR1BH	Timer/Counter1 - Output Compare Register B High Byte								
(0x8A)	OCR1BL	Timer/Counter1 - Output Compare Register B Low Byte								
(0x89)	OCR1AH	Timer/Counter1 - Output Compare Register A High Byte								
(0x88)	OCR1AL	Timer/Counter1 - Output Compare Register A Low Byte								
(0x87)	ICR1H	Timer/Counter1 - Input Capture Register High Byte								
(0x86)	ICR1L	Timer/Counter1 - Input Capture Register Low Byte								
(0x85)	TCNT1H	Timer/Counter1 - Counter Register High Byte								
(0x84)	TCNT1L	Timer/Counter1 - Counter Register Low Byte								
(0x83)	Reserved	-	-	-	-	-	-	-	-	
(0x82)	TCCR1C	FOC1A	FOC1B	FOC1C	-	-	-	-	-	
(0x81)	TCCR1B	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	
(0x80)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	
(0x7F)	DIDR1	-	-	-	-	-	-	-	AIN0D	
(0x7E)	DIDR0	ADC7D	ADC6D	ADC5D	ADC4D	-	-	ADC1D	ADC0D	
(0x7D)	DIDR2	-	-	ADC13D	ADC12D	ADC11D	ADC10D	ADC9D	ADC8D	

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x7C)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	
(0x7B)	ADCSRB	ADHSM	ACME	MUX5	-	ADTS3	ADTS2	ADTS1	ADTS0	
(0x7A)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	
(0x79)	ADCH	ADC Data Register High byte								
(0x78)	ADCL	ADC Data Register Low byte								
(0x77)	Reserved	-	-	-	-	-	-	-	-	
(0x76)	Reserved	-	-	-	-	-	-	-	-	
(0x75)	Reserved	-	-	-	-	-	-	-	-	
(0x74)	Reserved	-	-	-	-	-	-	-	-	
(0x73)	Reserved	-	-	-	-	-	-	-	-	
(0x72)	TIMSK4	OCIE4D	OCIE4A	OCIE4B	-	-	TOIE4	-	-	
(0x71)	TIMSK3	-	-	ICIE3	-	OCIE3C	OCIE3B	OCIE3A	TOIE3	
(0x70)	Reserved	-	-	-	-	-	-	-	-	
(0x6F)	TIMSK1	-	-	ICIE1	-	OCIE1C	OCIE1B	OCIE1A	TOIE1	
(0x6E)	TIMSK0	-	-	-	-	-	OCIE0B	OCIE0A	TOIE0	
(0x6D)	Reserved	-	-	-	-	-	-	-	-	
(0x6C)	Reserved	-	-	-	-	-	-	-	-	
(0x6B)	PCMSK0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	
(0x6A)	EICRB	-	-	ISC61	ISC60	-	-	-	-	
(0x69)	EICRA	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	
(0x68)	PCICR	-	-	-	-	-	-	-	PCIE0	
(0x67)	RCCTRL	-	-	-	-	-	-	-	RCFREQ	
(0x66)	OSCCAL	RC Oscillator Calibration Register								
(0x65)	PRR1	PRUSB	-	-	PRTIM4	PRTIM3	-	-	PRUSART1	
(0x64)	PRR0	PRTWI	-	PRTIM0	-	PRTIM1	PRSPI	-	PRADC	
(0x63)	Reserved	-	-	-	-	-	-	-	-	
(0x62)	Reserved	-	-	-	-	-	-	-	-	
(0x61)	CLKPR	CLKPCE	-	-	-	CLKPS3	CLKPS2	CLKPS1	CLKPS0	
(0x60)	WDTCSR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	
0x3E (0x5E)	SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	
0x3C (0x5C)	Reserved	-	-	-	-	-	-	-	-	
0x3B (0x5B)	RAMPZ	-	-	-	-	-	-	RAMPZ1	RAMPZ0	
0x3A (0x5A)	Reserved	-	-	-	-	-	-	-	-	
0x39 (0x59)	Reserved	-	-	-	-	-	-	-	-	
0x38 (0x58)	Reserved	-	-	-	-	-	-	-	-	
0x37 (0x57)	SPMCSR	SPMIE	RWWSB	SIGRD	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	
0x36 (0x56)	Reserved	-	-	-	-	-	-	-	-	
0x35 (0x55)	MCUCR	JTD	-	-	PUD	-	-	IVSEL	IVCE	
0x34 (0x54)	MCUSR	-	-	USBRF	JTRF	WDRF	BORF	EXTRF	PORF	
0x33 (0x53)	SMCR	-	-	-	-	SM2	SM1	SM0	SE	
0x32 (0x52)	PLLFRQ	PINMUX	PLLUSB	PLLTM1	PLLTM0	PDIV3	PDIV2	PDIV1	PDIV0	
0x31 (0x51)	OCDR/ MONDR	OCDR7	OCDR6	OCDR5	OCDR4	OCDR3	OCDR2	OCDR1	OCDR0	
		Monitor Data Register								
0x30 (0x50)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	
0x2F (0x4F)	Reserved	-	-	-	-	-	-	-	-	
0x2E (0x4E)	SPDR	SPI Data Register								
0x2D (0x4D)	SPSR	SPIF	WCOL	-	-	-	-	-	SPI2X	
0x2C (0x4C)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	
0x2B (0x4B)	GPIOR2	General Purpose I/O Register 2								
0x2A (0x4A)	GPIOR1	General Purpose I/O Register 1								
0x29 (0x49)	PLLCSR	-	-	-	PINDIV	-	-	PLLE	PLOCK	
0x28 (0x48)	OCR0B	Timer/Counter0 Output Compare Register B								
0x27 (0x47)	OCR0A	Timer/Counter0 Output Compare Register A								
0x26 (0x46)	TCNT0	Timer/Counter0 (8 Bit)								
0x25 (0x45)	TCCR0B	FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00	
0x24 (0x44)	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00	
0x23 (0x43)	GTCCR	TSM	-	-	-	-	-	PSRASY	PSRSYNC	
0x22 (0x42)	EEARH	-	-	-	-	EEPROM Address Register High Byte				
0x21 (0x41)	EEARL	EEPROM Address Register Low Byte								
0x20 (0x40)	EEDR	EEPROM Data Register								
0x1F (0x3F)	EECR	-	-	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	
0x1E (0x3E)	GPIOR0	General Purpose I/O Register 0								
0x1D (0x3D)	EIMSK	-	INT6	-	-	INT3	INT2	INT1	INT0	
0x1C (0x3C)	EIFR	-	INTF6	-	-	INTF3	INTF2	INTF1	INTF0	

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x1B (0x3B)	PCIFR	-	-	-	-	-	-	-	PCIF0	
0x1A (0x3A)	Reserved	-	-	-	-	-	-	-	-	
0x19 (0x39)	TIFR4	OCF4D	OCF4A	OCF4B	-	-	TOV4	-	-	
0x18 (0x38)	TIFR3	-	-	ICF3	-	OCF3C	OCF3B	OCF3A	TOV3	
0x17 (0x37)	Reserved	-	-	-	-	-	-	-	-	
0x16 (0x36)	TIFR1	-	-	ICF1	-	OCF1C	OCF1B	OCF1A	TOV1	
0x15 (0x35)	TIFR0	-	-	-	-	-	OCF0B	OCF0A	TOV0	
0x14 (0x34)	Reserved	-	-	-	-	-	-	-	-	
0x13 (0x33)	Reserved	-	-	-	-	-	-	-	-	
0x12 (0x32)	Reserved	-	-	-	-	-	-	-	-	
0x11 (0x31)	PORTF	PORTF7	PORTF6	PORTF5	PORTF4	-	-	PORTF1	PORTF0	
0x10 (0x30)	DDRF	DDF7	DDF6	DDF5	DDF4	-	-	DDF1	DDF0	
0x0F (0x2F)	PINF	PINF7	PINF6	PINF5	PINF4	-	-	PINF1	PINF0	
0x0E (0x2E)	PORTE	-	PORTE6	-	-	-	PORTE2	-	-	
0x0D (0x2D)	DDRE	-	DDE6	-	-	-	DDE2	-	-	
0x0C (0x2C)	PINE	-	PINE6	-	-	-	PINE2	-	-	
0x0B (0x2B)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	
0x0A (0x2A)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	
0x09 (0x29)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	
0x08 (0x28)	PORTC	PORTC7	PORTC6	-	-	-	-	-	-	
0x07 (0x27)	DDRC	DDC7	DDC6	-	-	-	-	-	-	
0x06 (0x26)	PINC	PINC7	PINC6	-	-	-	-	-	-	
0x05 (0x25)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	
0x04 (0x24)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	
0x03 (0x23)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	
0x02 (0x22)	Reserved	-	-	-	-	-	-	-	-	
0x01 (0x21)	Reserved	-	-	-	-	-	-	-	-	
0x00 (0x20)	Reserved	-	-	-	-	-	-	-	-	

Note:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O registers within the address range $00 - $1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
3. Some of the status flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers 0x00 to 0x1F only.
4. When using the I/O specific commands IN and OUT, the I/O addresses $00 - $3F must be used. When addressing I/O registers as data space using LD and ST instructions, $20 must be added to these addresses. The ATmega16U4/ATmega32U4 is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from $60 - $1FF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

PORTS

Register: MCUCR, PORTx, DDRx, PINx

3. Although each I/O port can sink more than the test conditions (20mA at VCC = 5V, 10mA at VCC = 3V) under steady state conditions (non-transient), the following must be observed:
   ATmega16U4/ATmega32U4:
   1.)The sum of all IOL, for ports A0-A7, G2, C4-C7 should not exceed 100 mA.
   2.)The sum of all IOL, for ports C0-C3, G0-G1, D0-D7 should not exceed 100 mA.
   3.)The sum of all IOL, for ports G3-G5, B0-B7, E0-E7 should not exceed 100 mA.
   4.)The sum of all IOL, for ports F0-F7 should not exceed 100 mA.
   If IOL exceeds the test condition, VOL may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test condition.
4. Although each I/O port can source more than the test conditions (20mA at VCC = 5V, 10mA at VCC = 3V) under steady state conditions (non-transient), the following must be observed:
   ATmega16U4/ATmega32U4:
   1)The sum of all IOH, for ports A0-A7, G2, C4-C7 should not exceed 100 mA.
   2)The sum of all IOH, for ports C0-C3, G0-G1, D0-D7 should not exceed 100 mA.
   3)The sum of all IOH, for ports G3-G5, B0-B7, E0-E7 should not exceed 100 mA.
   4)The sum of all IOH, for ports F0-F7 should not exceed 100 mA.

Figure 30-18. I/O Pin Output Voltage vs. Sink Current (V_{CC} = 5 V)

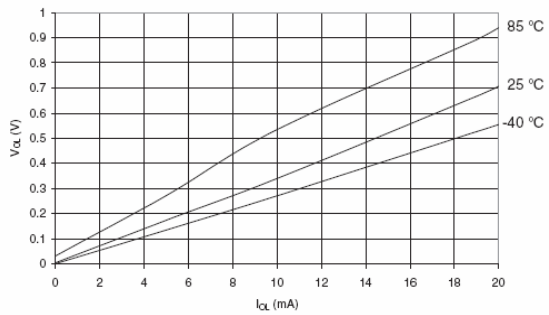


Figure 30-20. I/O Pin Output Voltage vs. Source Current (V_{CC} = 5 V)

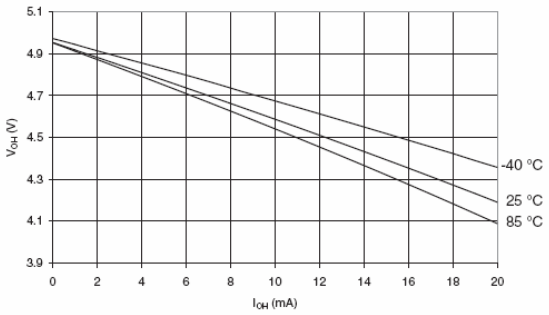
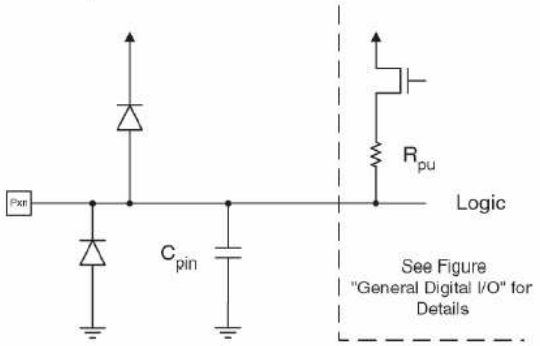


Figure 10-1. I/O Pin Equivalent Schematic



R_{RST}	Reset Pull-up Resistor		30		60	kΩ
R_{PU}	I/O Pin Pull-up Resistor		20		50	kΩ

MCU Control Register – MCUCR

Bit	7	6	5	4	3	2	1	0	
	JTD	–	–	PUD	–	–	IVSEL	IVCE	MCUCR
Read/Write	R/W	R	R	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **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). See "Configuring the Pin" on page 66 for more details about this feature.

Port B Data Register – PORTB

Bit	7	6	5	4	3	2	1	0	
	PORTB 7	PORTB 6	PORTB 5	PORTB 4	PORTB 3	PORTB 2	PORTB 1	PORTB 0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port B Data Direction Register – DDRB

Bit	7	6	5	4	3	2	1	0	
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port B Input Pins Address – PINB

Bit	7	6	5	4	3	2	1	0	
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Port C Data Register – PORTC

Bit	7	6	5	4	3	2	1	0	
	PORTC 7	PORTC 6	-	-	-	-	-	-	PORTC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port C Data Direction Register – DDRC

Bit	7	6	5	4	3	2	1	0	
	DDC7	DDC6	-	-	-	-	-	-	DDRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port C Input Pins Address – PINC

Bit	7	6	5	4	3	2	1	0	
	PINC7	PINC6	-	-	-	-	-	-	PINC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Port D Data Register – PORTD

Bit	7	6	5	4	3	2	1	0	
	PORTD 7	PORTD 6	PORTD 5	PORTD 4	PORTD 3	PORTD 2	PORTD 1	PORTD 0	PORTD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port D Data Direction Register – DDRD

Bit	7	6	5	4	3	2	1	0	
	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port D Input Pins Address – PIND

Bit	7	6	5	4	3	2	1	0	
	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Port E Data Register – PORTE

Bit	7	6	5	4	3	2	1	0	
	-	PORTE6	-	-	-	PORTE2	-	-	PORTE
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port E Data Direction Register – DDRE

Bit	7	6	5	4	3	2	1	0	
	-	DDE6	-	-	-	DDE2	-	-	DDRE
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port E Input Pins Address – PINE

Bit	7	6	5	4	3	2	1	0	
	-	PINE6	-	-	-	PINE2	-	-	PINE
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Port F Data Register – PORTF

Bit	7	6	5	4	3	2	1	0	
	PORTF7	PORTF6	PORTF5	PORTF4	-	-	PORTF1	PORTF0	PORTF
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port F Data Direction Register – DDRF

Bit	7	6	5	4	3	2	1	0	
	DDF7	DDF6	DDF5	DDF4	-	-	DDF1	DDF0	DDRF
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port F Input Pins Address – PINF

Bit	7	6	5	4	3	2	1	0	
	PINF7	PINF6	PINF5	PINF4	-	-	PINF1	PINF0	PINF
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Table 10-1. 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)

```
////////////////////////////////////////////////////////////////////////////
//
// Bsp. Konfiguration von I/O Port B:
//
// PB0  Output  HIGH
// PB1  Output  HIGH
// PB2  Output  LOW
// PB3  Output  LOW
// PB4  Input   kein Pull-Up
// PB5  Input   kein Pull-Up
// PB6  Input   mit Pull-Up
// PB7  Input   mit Pull-Up
//
//
////////////////////////////////////////////////////////////////////////////

#include <avr/io.h>           //SFR Deklerationen

int main(void)
{
        unsigned char i;      //zum Einlesen des Ports

        DDRB  = 0x0F;         //PB7...PB4 Input, PB3...PB0 Output
        PORTB = 0xC3;         //Ausgabe 1100 0011

        _NOP();               //zur Synchronisation der Anschlüsse vor dem Rücklesen warten
        i = PINB;             //Lesen der Portpins

        While(1);             //Endlosschleife
        return 0;             //das Hauptprogramm main gibt einen Wert zurück

}
```

oder länger aber übersichtlicher

```
int main(void)
{
        unsigned char i;

        DDRB  = DDRB|(1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
        PORTB = PORTB|(1<<PORTB7)|(1<<PORTB6)|(1<<PORTB1)|(1<<PORTB0);

        _NOP();
        i = PINB;

        While(1);
        return 0;
}
```

```
////////////////////////////////////////////////////////////////////////////
//
// Pins PB0 und PB1 blinken abwechselnd mit 500ms
//
////////////////////////////////////////////////////////////////////////////

#define F_CPU 12000000UL

#include <avr/io.h>
#include <util/delay.h>              // _delay_ms() geht nur bis max. 262.14 ms / F_CPU !!!


void delay_ms (unsigned int ms)
{
        for (unsigned int i=0; i<ms; i++)
        {
                _delay_ms(1);
        }

}


int main(void)
{
        DDRB = 0xFF;                            //PORTB als Ausgang
        while(1)
        {
                PORTB = PORTB |(1<<PB0);        //Maske 0b00000001    Pin PB0 = 1
                PORTB = PORTB & ~(1<<PB1);      //Maske 0b11111101    Pin PB1 = 0
                delay_ms(500);
                PORTB = PORTB |(1<<PB1);        //Maske 0b00000010    Pin PB1 = 1
                PORTB = PORTB & ~(1<<PB0);      //Maske 0b11111110    Pin PB0 = 0
                delay_ms(500);
        }
        return 0;
}
```

besser:

```
int main(void)
{
        DDRB = 0xFF;

        PORTB = PORTB |(1<<PB0);        //Pin PB0 = 1
        PORTB = PORTB & ~(1<<PB1);      //Pin PB1 = 0

        while(1)
        {
                PORTB = PORTB ^0b00000011; //toggelt PB0 und PB1
                delay_ms(500);
        }
        return 0;
}
```

Alternate Functions:

Table 10-3. Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB7	OC0A/OC1C/PCINT7/$\overline{\text{RTS}}$ (Output Compare and PWM Output A for Timer/Counter0, Output Compare and PWM Output C for Timer/Counter1 or Pin Change Interrupt 7 or UART flow control $\overline{\text{RTS}}$ signal)
PB6	OC1B/PCINT6/OC.4B/ADC13 (Output Compare and PWM Output B for Timer/Counter1 or Pin Change Interrupt 6 or Timer 4 Output Compare B / PWM output or Analog to Digital Converter channel 13)
PB5	OC1A/PCINT5/$\overline{\text{OC.4B}}$/ADC12 (Output Compare and PWM Output A for Timer/Counter1 or Pin Change Interrupt 5 or Timer 4 Complementary Output Compare B / PWM output or Analog to Digital Converter channel 12)
PB4	PCINT4/ADC11 (Pin Change Interrupt 4 or Analog to Digital Converter channel 11)
PB3	PDO/MISO/PCINT3 (Programming Data Output or SPI Bus Master Input/Slave Output or Pin Change Interrupt 3)
PB2	PDI/MOSI/PCINT2 (Programming Data Input or SPI Bus Master Output/Slave Input or Pin Change Interrupt 2)
PB1	SCK/PCINT1 (SPI Bus Serial Clock or Pin Change Interrupt 1)
PB0	$\overline{\text{SS}}$/PCINT0 (SPI Slave Select input or Pin Change Interrupt 0)

Table 10-6. Port C Pins Alternate Functions

Port Pin	Alternate Function
PC7	ICP3/CLKO/OC4A(Input Capture Timer 3 or CLK0 (Divided System Clock) or Output Compare and direct PWM output A for Timer 4)
PC6	OC.3A/$\overline{\text{OC4A}}$ (Output Compare and PWM output A for Timer/Counter3 or Output Compare and complementary PWM output $\overline{\text{A}}$ for Timer 4)
PC5	Not present on pin-out.
PC4	
PC3	
PC2	
PC1	
PC0	

Table 10-8. Port D Pins Alternate Functions

Port Pin	Alternate Function
PD7	T0/OC.4D/ADC10 (Timer/Counter0 Clock Input or Timer 4 Output Compare D / PWM output or Analog to Digital Converter channel 10)
PD6	T1/$\overline{\text{OC.4D}}$/ADC9 (Timer/Counter1 Clock Input or Timer 4 Output Complementary Compare D / PWM output or Analog to Digital Converter channel 9)
PD5	XCK1/$\overline{\text{CTS}}$ (USART1 External Clock Input/Output or UART flow control $\overline{\text{CTS}}$ signal)
PD4	ICP1/ADC8 (Timer/Counter1 Input Capture Trigger or Analog to Digital Converter channel 8)
PD3	$\overline{\text{INT3}}$/TXD1 (External Interrupt3 Input or USART1 Transmit Pin)
PD2	$\overline{\text{INT2}}$/RXD1 (External Interrupt2 Input or USART1 Receive Pin)
PD1	$\overline{\text{INT1}}$/SDA (External Interrupt1 Input or TWI Serial DAta)
PD0	$\overline{\text{INT0}}$/SCL/OC0B (External Interrupt0 Input or TWI Serial CLock or Output Compare for Timer/Counter0)

Table 10-11. Port E Pins Alternate Functions

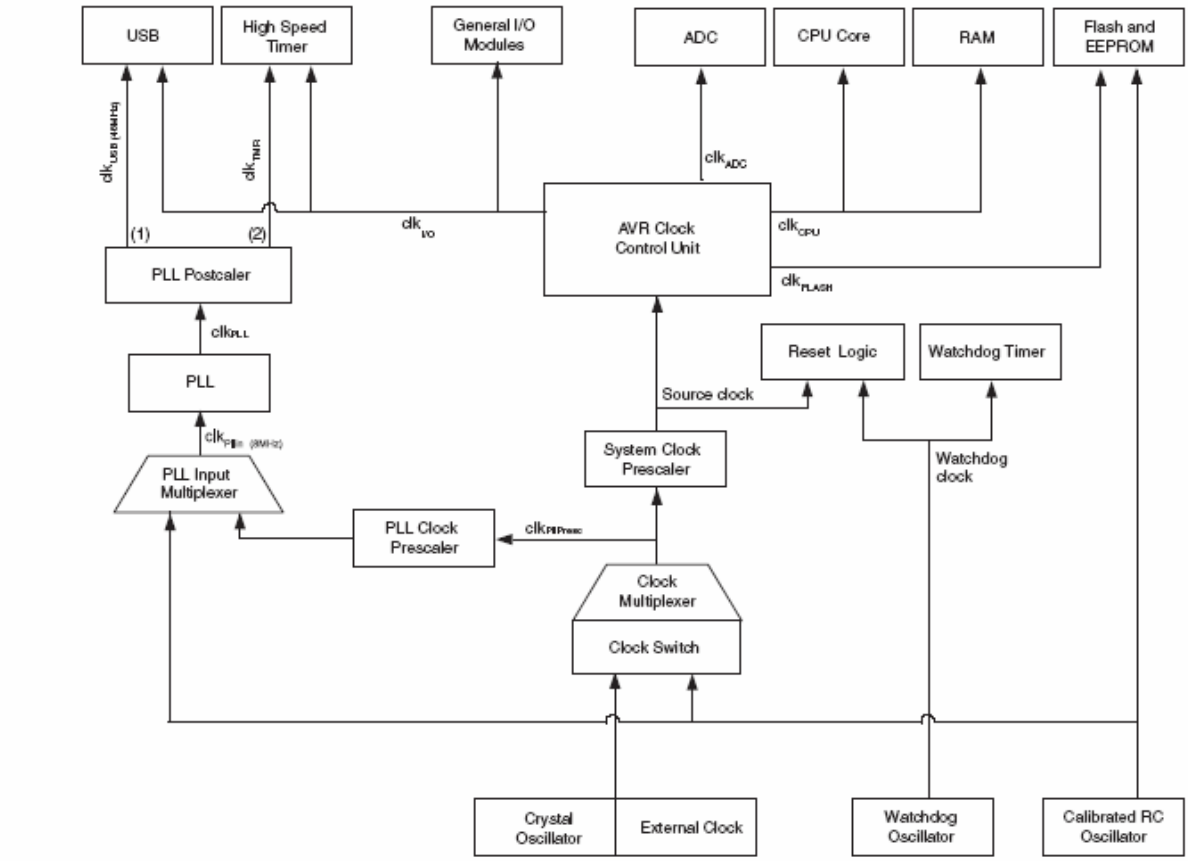
Port Pin	Alternate Function
PE7	Not present on pin-out.
PE6	INT6/AIN0 (External Interrupt 6 Input or Analog Comparator Positive Input)
PE5	Not present on pin-out.
PE4	
PE3	
PE2	$\overline{\text{HWB}}$ (Hardware bootloader activation)
PE1	Not present on pin-out.
PE0	

Table 10-13. Port F Pins Alternate Functions

Port Pin	Alternate Function
PF7	ADC7/TDI (ADC input channel 7 or JTAG Test Data Input)
PF6	ADC6/TDO (ADC input channel 6 or JTAG Test Data Output)
PF5	ADC5/TMS (ADC input channel 5 or JTAG Test Mode Select)
PF4	ADC4/TCK (ADC input channel 4 or JTAG Test ClocK)
PF3	Not present on pin-out.
PF2	
PF1	ADC1 (ADC input channel 1)
PF0	ADC0 (ADC input channel 0)

CLOCK

Figure 6-1. Clock Distribution



Fuse Bits:

Table 6-1. Device Clocking Options Select(1)

Device Clocking Option	CKSEL[3:0] (or EXCKSEL[3:0])
Low Power Crystal Oscillator	1111 - 1000
Reserved	0111 - 0110
Low Frequency Crystal Oscillator	0101 - 0100
Reserved	0011
Calibrated Internal RC Oscillator	0010
External Clock	0000
Reserved	0001

Note: 1. For all fuses "1" means unprogrammed while "0" means programmed.

Default Clock Source ATmega16U4 and ATmega32U4

The device is shipped with Low Power Crystal Oscillator (8.0MHz-16MHz) enabled and with the fuse CKDIV8 programmed, resulting in 1.0MHz system clock with an 8 MHz crystal. See Table 28-5 on page 348 for an overview of the default Clock Selection Fuse setting.

Figure 6-2. Crystal Oscillator Connections

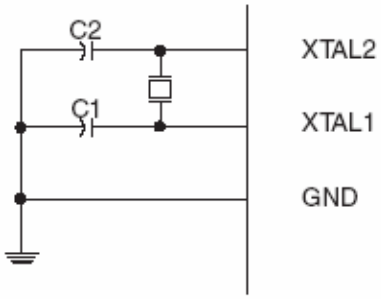


Table 6-3. Low Power Crystal Oscillator Operating Modes[3]

Frequency Range[1] (MHz)	CKSEL3..1	Recommended Range for Capacitors C1 and C2 (pF)
0.4 - 0.9	100[2]	–
0.9 - 3.0	101	12 - 22
3.0 - 8.0	110	12 - 22
8.0 - 16.0	111	12 - 22

The CKSEL0 Fuse together with the SUT1..0 Fuses select the start-up times as shown in Table 6-4.

Table 6-4. Start-up Times for the Low Power Crystal Oscillator Clock Selection

Oscillator Source / Power Conditions	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	CKSEL0	SUT1..0
Ceramic resonator, fast rising power	258 CK	14CK + 4.1 ms[1]	0	00
Ceramic resonator, slowly rising power	258 CK	14CK + 65 ms[1]	0	01
Ceramic resonator, BOD enabled	1K CK	14CK[2]	0	10
Ceramic resonator, fast rising power	1K CK	14CK + 4.1 ms[2]	0	11

Table 6-4. Start-up Times for the Low Power Crystal Oscillator Clock Selection (Continued)

Oscillator Source / Power Conditions	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	CKSEL0	SUT1..0
Ceramic resonator, slowly rising power	1K CK	14CK + 65 ms[2]	1	00
Crystal Oscillator, BOD enabled	16K CK	14CK	1	01
Crystal Oscillator, fast rising power	16K CK	14CK + 4.1 ms	1	10
Crystal Oscillator, slowly rising power	16K CK	14CK + 65 ms	1	11

Interner 8 MHz Oszillator:

Table 6-7. Internal Calibrated RC Oscillator Operating Modes[1][3]

Frequency Range[2] (MHz)	CKSEL[3:0]
7.3 - 8.1	0010

Register: OSCCAL zur Kalibrierung
RCCTRL zur Auswahl ob 8 MHz (= default) oder 1 MHz

Ändern des System Clock Prescalers

Gilt für int. RC-Oszillator und Beschaltung mit ext. Quarz

Register: CLKPR

CLKPR – Clock Prescaler Register

Bit	7	6	5	4	3	2	1	0	
	CLKPCE	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	CLKPR
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	See Bit Description				

- **Bit 7 – CLKPCE: Clock Prescaler Change Enable**

The CLKPCE bit must be written to logic one to enable change of the CLKPS bits. The CLKPCE bit is only updated when the other bits in CLKPR are simultaneously written to zero. CLKPCE is cleared by hardware four cycles after it is written or when CLKPS bits are written. Rewriting the CLKPCE bit within this time-out period does neither extend the time-out period, nor clear the CLKPCE bit.

- **Bits 3..0 – CLKPS[3..0]: Clock Prescaler Select Bits 3 - 0**

These bits define the division factor between the selected clock source and the internal system clock. These bits can be written run-time to vary the clock frequency to suit the application requirements. As the divider divides the master clock input to the MCU, the speed of all synchronous peripherals is reduced when a division factor is used. The division factors are given in Table 6-10.

The CKDIV8 Fuse determines the initial value of the CLKPS bits. If CKDIV8 is unprogrammed, the CLKPS bits will be reset to "0000". If CKDIV8 is programmed, CLKPS bits are reset to "0011", giving a division factor of 8 at start up. This feature should be used if the selected clock source has a higher frequency than the maximum frequency of the device at the present operating conditions. Note that any value can be written to the CLKPS bits regardless of the CKDIV8 Fuse setting. The Application software must ensure that a sufficient division factor is chosen if the selected clock source has a higher frequency than the maximum frequency of the device at the present operating conditions. The device is shipped with the CKDIV8 Fuse programmed.

Table 6-10. Clock Prescaler Select

CLKPS3	CLKPS2	CLKPS1	CLKPS0	Clock Division Factor
0	0	0	0	1
0	0	0	1	2
0	0	1	0	4
0	0	1	1	8
0	1	0	0	16
0	1	0	1	32
0	1	1	0	64
0	1	1	1	128
1	0	0	0	256
1	0	0	1	Reserved
1	0	1	0	Reserved
1	0	1	1	Reserved
1	1	0	0	Reserved
1	1	0	1	Reserved
1	1	1	0	Reserved
1	1	1	1	Reserved

To avoid unintentional changes of clock frequency, a special write procedure must be followed to change the CLKPS bits:

1. Write the Clock Prescaler Change Enable (CLKPCE) bit to one and all other bits in CLKPR to zero.
2. Within four cycles, write the desired value to CLKPS while writing a zero to CLKPCE.

Interrupts must be disabled when changing prescaler setting to make sure the write procedure is not interrupted.

d.h.: z.B.:

```
CLKPR = 0x80;        //CLKPCE-Bit 1 und alle andern Bits 0
CLKPR = 0x02;        //CLKPCE-Bit 0 und sonst Teilerfaktor (hier 4) einstellen
```

Umschalten zwischen internem RC-Oszillator und externen Quarz

- durch die Fuse Bits oder per SW im laufenden Betrieb mit CLKSEL0

Register: CLKSEL0

CLKSEL0 – Clock Selection Register 0

Bit	7	6	5	4	3	2	1	0	
	RCSUT1	RCSUT0	EXSUT1	EXSUT0	RCE	EXTE	-	CLKS	CLKSEL0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	See Bit Description				

- **Bit 7-6 – RCSUT[1:0]: SUT for RC oscillator**

These 2 bits are the SUT value for the RC Oscillator. If the RC oscillator is selected by fuse bits, the SUT fuse are copied into these bits. A firmware change will not have any effect because this additional start-up time is only used after a reset and not after a clock switch.

- **Bit 5-4 – EXSUT[1:0]: SUT for External Clock/ Low Power Crystal Oscillator**

These 2 bits are the SUT value for the External Clock / Low Power Crystal Oscillator. If the External Clock / Low Power Crystal Oscillator is selected by fuse bits, the SUT fuses are copied into these bits. The firmware can modify these bits by writing a new value. This value will be used at the next start of the External Clock / Low Power Crystal Oscillator.

- **Bit 3 – RCE: Enable RC Oscillator**

The RCE bit must be written to logic one to enable the RC Oscillator. The RCE bit must be written to logic zero to disable the RC Oscillator.

- **Bit 2 – EXTE: Enable External Clock / Low Power Crystal Oscillator**

The OSCE bit must be written to logic one to enable External Clock / Low Power Crystal Oscillator. The OSCE bit must be written to logic zero to disable the External Clock / Low Power Crystal Oscillator.

- **Bit 0 – CLKS: Clock Selector**

The CLKS bit must be written to logic one to select the External Clock / Low Power Crystal Oscillator as CPU clock. The CLKS bit must be written to logic zero to select the RC Oscillator as CPU clock. After a reset, the CLKS bit is set by hardware if the External Clock / Low Power Crystal Oscillator is selected by the fuse bits configuration.
The firmware has to check if the clock is correctly started before selected it.

z.B. umschalten von ext. Quarz auf int. RC-Oszillator:

```
CLKSEL0 = CLKSEL0 | (1<<RCE);        //enable internal RC-Oszillator
CLKSEL0 = CLKSEL0 &~(1<<CLKS);       //select internal RC-Oszillator
CLKSEL0 = CLKSEL0 &~(1<<EXTE);       //disable external Crystal
```

RESET

Figure 8-1. Reset Logic

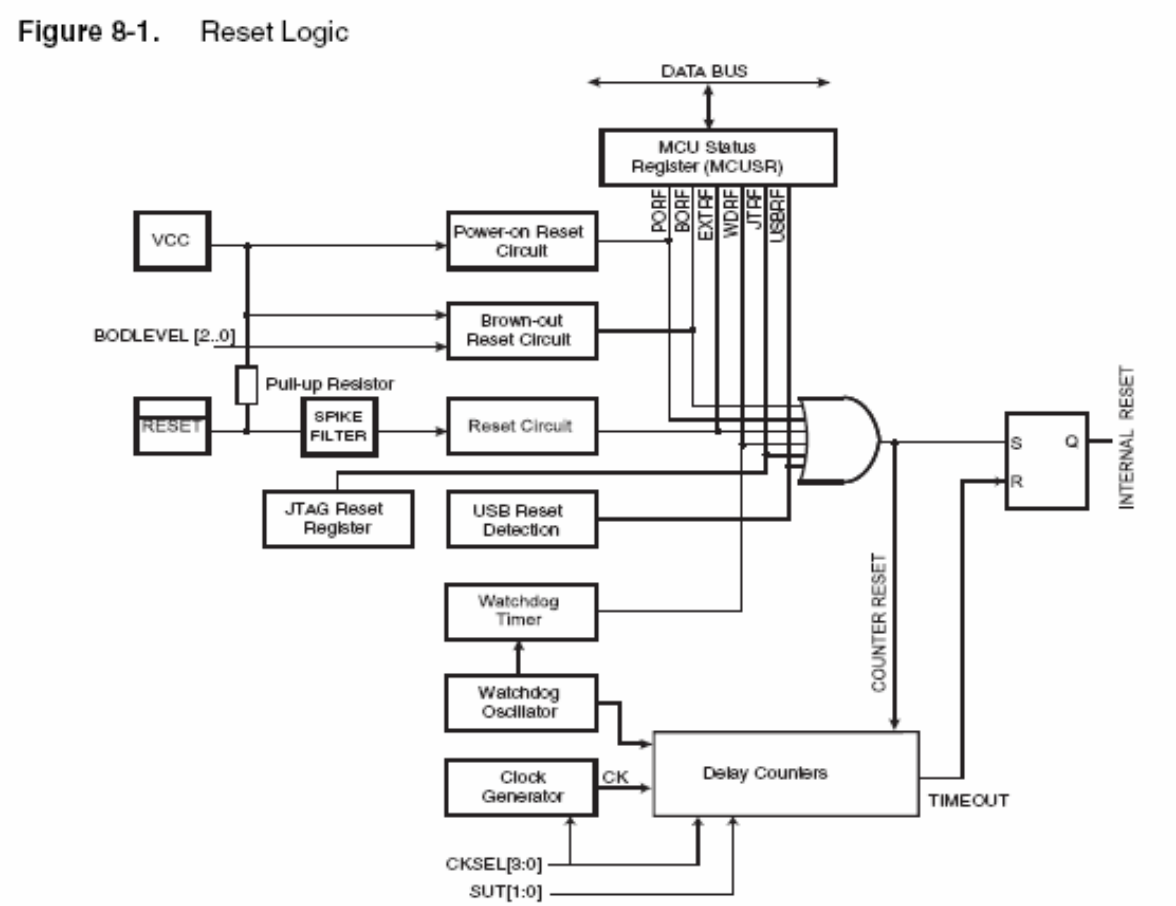
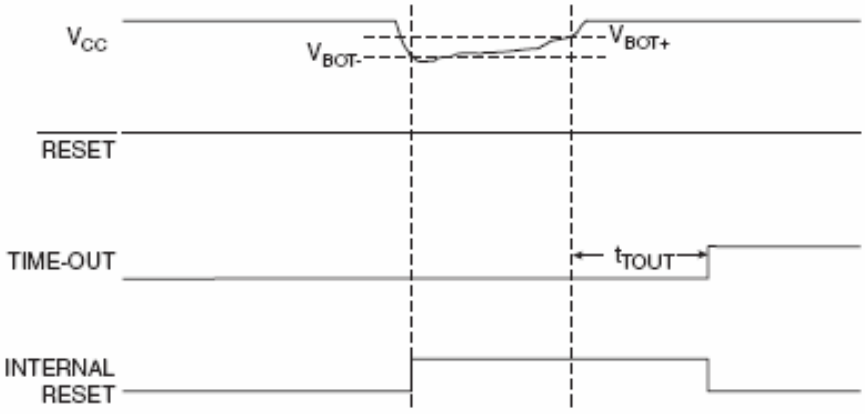


Figure 8-5. Brown-out Reset During Operation



INTERRUPTS

Table 9-1. Reset and Interrupt Vectors

Vector No.	Program Address[2]	Source	Interrupt Definition
1	$0000[1]	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$0002	INT0	External Interrupt Request 0
3	$0004	INT1	External Interrupt Request 1
4	$0006	INT2	External Interrupt Request 2
5	$0008	INT3	External Interrupt Request 3
6	$000A	Reserved	Reserved
7	$000C	Reserved	Reserved
8	$000E	INT6	External Interrupt Request 6
9	$0010	Reserved	Reserved
10	$0012	PCINT0	Pin Change Interrupt Request 0
11	$0014	USB General	USB General Interrupt request
12	$0016	USB Endpoint	USB Endpoint Interrupt request
13	$0018	WDT	Watchdog Time-out Interrupt
14	$001A	Reserved	Reserved
15	$001C	Reserved	Reserved
16	$001E	Reserved	Reserved
17	$0020	TIMER1 CAPT	Timer/Counter1 Capture Event
18	$0022	TIMER1 COMPA	Timer/Counter1 Compare Match A
19	$0024	TIMER1 COMPB	Timer/Counter1 Compare Match B
20	$0026	TIMER1 COMPC	Timer/Counter1 Compare Match C
21	$0028	TIMER1 OVF	Timer/Counter1 Overflow
22	$002A	TIMER0 COMPA	Timer/Counter0 Compare Match A
23	$002C	TIMER0 COMPB	Timer/Counter0 Compare match B
24	$002E	TIMER0 OVF	Timer/Counter0 Overflow
25	$0030	SPI (STC)	SPI Serial Transfer Complete
26	$0032	USART1 RX	USART1 Rx Complete
27	$0034	USART1 UDRE	USART1 Data Register Empty
28	$0036	USART1TX	USART1 Tx Complete
29	$0038	ANALOG COMP	Analog Comparator

Ergebnis	Funktion	Bemerkung
void	**cli(void)**	I=0 Interrupts global sperren
void	**sei(void)**	I=1 Interrupts global freigeben
	ISR(Name)	Interruptroutine kann nicht unterbrochen werden
	INTERRUPT (Name)	Interruptroutine kann unterbrochen werden

<avr/iom32u4.h>

```
INT0_vect           /* External Interrupt Request 0 */
INT1_vect           /* External Interrupt Request 1 */
INT2_vect           /* External Interrupt Request 2 */
INT3_vect           /* External Interrupt Request 3 */
INT6_vect           /* External Interrupt Request 6 */
PCINT0_vect         /* Pin Change Interrupt Request 0 */
USB_GEN_vect        /* USB General Interrupt Request */
USB_COM_vect        /* USB Endpoint/Pipe Interrupt Communication Request */
WDT_vect            /* Watchdog Time-out Interrupt */
TIMER1_CAPT_vect    /* Timer/Counter1 Capture Event */
TIMER1_COMPA_vect   /* Timer/Counter1 Compare Match A */
TIMER1_COMPB_vect   /* Timer/Counter1 Compare Match B */
TIMER1_COMPC_vect   /* Timer/Counter1 Compare Match C */
TIMER1_OVF_vect     /* Timer/Counter1 Overflow */
TIMER0_COMPA_vect   /* Timer/Counter0 Compare Match A */
TIMER0_COMPB_vect   /* Timer/Counter0 Compare Match B */
TIMER0_OVF_vect     /* Timer/Counter0 Overflow */
SPI_STC_vect        /* SPI Serial Transfer Complete */
USART1_RX_vect      /* USART1, Rx Complete */
USART1_UDRE_vect    /* USART1 Data register Empty */
USART1_TX_vect      /* USART1, Tx Complete */
ANALOG_COMP_vect    /* Analog Comparator */
ADC_vect            /* ADC Conversion Complete */
EE_READY_vect       /* EEPROM Ready */
TIMER3_CAPT_vect    /* Timer/Counter3 Capture Event */
TIMER3_COMPA_vect   /* Timer/Counter3 Compare Match A */
TIMER3_COMPB_vect   /* Timer/Counter3 Compare Match B */
TIMER3_COMPC_vect   /* Timer/Counter3 Compare Match C */
TIMER3_OVF_vect     /* Timer/Counter3 Overflow */
TWI_vect            /* 2-wire Serial Interface         */
SPM_READY_vect      /* Store Program Memory Read */
TIMER4_COMPA_vect   /* Timer/Counter4 Compare Match A */
TIMER4_COMPB_vect   /* Timer/Counter4 Compare Match B */
TIMER4_COMPD_vect   /* Timer/Counter4 Compare Match D */
TIMER4_OVF_vect     /* Timer/Counter4 Overflow */
TIMER4_FPF_vect     /* Timer/Counter4 Fault Protection Interrupt */
```

EXTERNE INTERRUPTS

Register: SREG, EICRA, EICRB, EIMSK, EIFR, PCICR, PCIFR, PCMSK

Pins: INT3:0, INT6 flanken- / pegel- gesteuerte Interrupts
PCINT7:0 Pin Change Interrupt

The AVR Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – I: Global Interrupt Enable**

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

ext. Interrupts INT3:0, INT6

External Interrupt Control Register A – EICRA

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

Bit	7	6	5	4	3	2	1	0	
	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..0 – ISC31, ISC30 – ISC00, ISC00: External Interrupt 3 - 0 Sense Control Bits**

The External Interrupts 3 - 0 are activated by the external pins INT3:0 if the SREG I-flag and the corresponding interrupt mask in the EIMSK is set. The level and edges on the external pins that activate the interrupts are defined in Table 11-1. Edges on INT3..INT0 are registered asynchronously. Pulses on INT3:0 pins wider than the minimum pulse width given in Table 11-2 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. If enabled, a level triggered interrupt will generate an interrupt request as long as the pin is held low. When changing the ISCn bit, an interrupt can occur. Therefore, it is recommended to first disable INTn by clearing its Interrupt Enable bit in the EIMSK Register. Then, the ISCn bit can be changed. Finally, the INTn interrupt flag should be cleared by writing a logical one to its Interrupt Flag bit (INTFn) in the EIFR Register before the interrupt is re-enabled.

Table 11-1. Interrupt Sense Control[1]

ISCn1	ISCn0	Description
0	0	The low level of INTn generates an interrupt request.
0	1	Any edge of INTn generates asynchronously an interrupt request.
1	0	The falling edge of INTn generates asynchronously an interrupt request.
1	1	The rising edge of INTn generates asynchronously an interrupt request.

Note: 1. n = 3, 2, 1or 0.
When changing the ISCn1/ISCn0 bits, the interrupt must be disabled by clearing its Interrupt Enable bit in the EIMSK Register. Otherwise an interrupt can occur when the bits are changed.

External Interrupt Control Register B – EICRB

Bit	7	6	5	4	3	2	1	0	
	-	-	ISC61	ISC60	-	-	-	-	EICRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7..6 – Res: Reserved Bits**

These bits are reserved bits in the ATmega16U4/ATmega32U4 and always read as zero.

- **Bits 5, 4 – ISC61, ISC60: External Interrupt 6 Sense Control Bits**

The External Interrupt 6 is activated by the external pin INT6 if the SREG I-flag and the corresponding interrupt mask in the EIMSK is set. The level and edges on the external pin that activate the interrupt are defined in Table 11-3. The value on the INT6 pin are 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. Observe that CPU clock frequency can be lower than the XTAL frequency if the XTAL divider is enabled. If low level interrupt is selected, the low level must be held until the completion of the currently executing instruction to generate an interrupt. If enabled, a level triggered interrupt will generate an interrupt request as long as the pin is held low.

Table 11-3. Interrupt Sense Control(1)

ISC61	ISC60	Description
0	0	The low level of INT6 generates an interrupt request.
0	1	Any logical change on INT6 generates an interrupt request
1	0	The falling edge between two samples of INT6 generates an interrupt request.
1	1	The rising edge between two samples of INT6 generates an interrupt request.

Note: 1. When changing the ISC61/ISC60 bits, the interrupt must be disabled by clearing its Interrupt Enable bit in the EIMSK Register. Otherwise an interrupt can occur when the bits are changed.

External Interrupt Mask Register – EIMSK

Bit	7	6	5	4	3	2	1	0	
	-	INT6	-	-	INT3	INT2	INT1	IINT0	EIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..0 – INT6, INT3 – INT0: External Interrupt Request 6, 3 - 0 Enable**

When an INT[6;3:0] bit is written to one and the I-bit in the Status Register (SREG) is set (one), the corresponding external pin interrupt is enabled. The Interrupt Sense Control bits in the External Interrupt Control Registers – EICRA and EICRB – defines whether the external interrupt is activated on rising or falling edge or level sensed. Activity on any of these pins will trigger an interrupt request even if the pin is enabled as an output. This provides a way of generating a software interrupt.

External Interrupt Flag Register – EIFR

Bit	7	6	5	4	3	2	1	0	
	-	INTF6	-	-	INTF3	INTF2	INTF1	IINTF0	EIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..0 – INTF6, INTF3 - INTF0: External Interrupt Flags 6, 3 - 0**

When an edge or logic change on the INT[6;3:0] pin triggers an interrupt request, INTF7:0 becomes set (one). If the I-bit in SREG and the corresponding interrupt enable bit, INT[6;3:0] in EIMSK, are set (one), the MCU will jump to the interrupt vector. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it. These flags are always cleared when INT[6;3:0] are configured as level interrupt. Note that when entering sleep mode with the INT3:0 interrupts disabled, the input buffers on these pins will be disabled. This may cause a logic change in internal signals which will set the INTF3:0 flags. See "Digital Input Enable and Sleep Modes" on page 69 for more information.

Pin Change Interrupts PCINT7:0

Pin Change Interrupt Control Register - PCICR

Bit	7	6	5	4	3	2	1	0	
			–	–	–	–	–	PCIE0	PCICR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 0 – PCIE0: Pin Change Interrupt Enable 0**

When the PCIE0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), pin change interrupt 0 is enabled. Any change on any enabled PCINT7..0 pin will cause an interrupt. The corresponding interrupt of Pin Change Interrupt Request is executed from the PCI0 Interrupt Vector. PCINT7..0 pins are enabled individually by the PCMSK0 Register.

Pin Change Interrupt Flag Register – PCIFR

Bit	7	6	5	4	3	2	1	0	
			–	–	–	–	–	PCIF0	PCIFR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 0 – PCIF0: Pin Change Interrupt Flag 0**

When a logic change on any PCINT7..0 pin triggers an interrupt request, PCIF0 becomes set (one). If the I-bit in SREG and the PCIE0 bit in EIMSK are set (one), 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 a logical one to it.

Pin Change Mask Register 0 – PCMSK0

Bit	7	6	5	4	3	2	1	0	
	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7..0 – PCINT7..0: Pin Change Enable Mask 7..0**

Each PCINT7..0 bit selects whether pin change interrupt is enabled on the corresponding I/O pin. If PCINT7..0 is set and the PCIE0 bit in PCICR is set, pin change interrupt is enabled on the corresponding I/O pin. If PCINT7..0 is cleared, pin change interrupt on the corresponding I/O pin is disabled.

```
////////////////////////////////////////////////////////////////
//
//Atmega32u4
//GNU-C WinAVR Compiler
//
//ext. INT0 (PD0) erhöht Port B um 1 mit neg. Flanke und
//PD6 löscht mit fallender Flanke Zählerstand auf Port B
//
////////////////////////////////////////////////////////////////

#include <avr/io.h>           //Registerdeklerationen
#include <avr/interrupt.h>    //Makros sei(),cli()


ISR(INT0_vect)                //Interruptroutine INT0
{
      PORTB++;
}

int main(void)
{
      DDRB  = 0xFF;                         //PORTB ist Ausgang

      DDRD  = DDRD&~(1<<DDD6)&~(1<<DDD0);//DDRD6,0 = 0 (Input)
      PORTD = PORTD|(1<<PD6)|(1<<PD0);    //PORTD6,0= 1 (Pull Up on)

      EICRA = EICRA | (1<<ISC01);         //INT0 fallende Flanke
      EIMSK = EIMSK | (1<<INT0);          //INT0 freigeben
      sei();                              //Interrupts generell frei

      while(1)
      {
            while (PIND & (1<<PD6));      //warten auf fallende Flanke
            PORTB = 0x00;                 //Port B löschen
            while (!(PIND & (1<<PD6)));   //warten auf steigende Flanke
      }
      return 0;                           //main gibt Wert zurück

}
```

8-Bit TIMER / COUNTER 0

- 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)

Register: SREG, GTCCR, TCNT0, TIFR0, TIMSK0, OCR0A, OCR0B, TCCR0A, TCCR0B

Pins: T0 (PD7)……Counter Input
OC0A (PB7), OC0B (PD0)…….PWM Output

Timer/Counter Register – TCNT0

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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.

Output Compare Register A – OCR0A

Bit	7	6	5	4	3	2	1	0	
	OCR0A[7:0]								OCR0A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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.

Output Compare Register B – OCR0B

Bit	7	6	5	4	3	2	1	0	
	OCR0B[7:0]								OCR0B
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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.

General Timer/Counter Control Register – GTCCR

Bit	7	6	5	4	3	2	1	0	
	TSM	–	–	–	–	–	PSRASY	PSRSYNC	GTCCR
Read/Write	R/W	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	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 0 – PSRSYNC: Prescaler Reset for Synchronous Timer/Counters**

When this bit is one, Timer/Counter0 and Timer/Counter1 and Timer/Counter3 prescaler will be Reset. This bit is normally cleared immediately by hardware, except if the TSM bit is set. Note that Timer/Counter0, Timer/Counter1 and Timer/Counter3 share the same prescaler and a reset of this prescaler will affect all timers.

Timer/Counter Interrupt Mask Register – TIMSK0

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..3, 0 – Res: Reserved Bits**

These bits are reserved bits and will always read as zero.

- **Bit 2 – OCIE0B: Timer/Counter Output Compare Match B 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 the Timer/Counter Interrupt Flag Register – TIFR0.

- **Bit 1 – OCIE0A: Timer/Counter0 Output Compare Match A 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 the Timer/Counter 0 Interrupt Flag Register – TIFR0.

- **Bit 0 – TOIE0: 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 the Timer/Counter 0 Interrupt Flag Register – TIFR0.

Timer/Counter 0 Interrupt Flag Register – TIFR0

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	OCF0B	OCF0A	TOV0	TIFR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..3, 0 – Res: Reserved Bits**

These bits are reserved bits in the ATmega16U4/ATmega32U4 and will always read as zero.

- **Bit 2 – OCF0B: Timer/Counter 0 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 – OCF0A: Timer/Counter 0 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 – TOV0: 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 13-8, "Waveform Generation Mode Bit Description" on page 104.

Timer/Counter Control Register A – TCCR0A

Bit	7	6	5	4	3	2	1	0	
	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:6 – COM01A:0: Compare Match Output A Mode**

These bits control the Output Compare pin (OC0A) behavior. If one or both of the COM0A1:0 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 COM0A1:0 bits depends on the WGM02:0 bit setting. Table 13-2 shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 13-2. Compare Output Mode, non-PWM Mode

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

Table 13-3 shows the COM0A1:0 bit functionality when the WGM01:0 bits are set to fast PWM mode.

Table 13-3. Compare Output Mode, Fast 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, set OC0A at TOP
1	1	Set OC0A on Compare Match, clear OC0A at TOP

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. See "Fast PWM Mode" on page 97 for more details.

Table 13-4 shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to phase correct PWM mode.

Table 13-4. 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. See "Phase Correct PWM Mode" on page 99 for more details.

- **Bits 5:4 – COM0B1:0: Compare Match Output B Mode**

These bits control the Output Compare pin (OC0B) behavior. If one or both of the COM0B1:0 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 COM0B1:0 bits depends on the WGM02:0 bit setting. Table 13-2 shows the COM0A1:0 bit functionality when the WGM02:0 bits are set to a normal or CTC mode (non-PWM).

Table 13-5. Compare Output Mode, non-PWM Mode

COM01	COM00	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

Table 13-3 shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to fast PWM mode.

Table 13-6. Compare Output Mode, Fast PWM Mode[1]

COM01	COM00	Description
0	0	Normal port operation, OC0B disconnected.
0	1	Reserved
1	0	Clear OC0B on Compare Match, set OC0B at TOP
1	1	Set OC0B on Compare Match, clear OC0B at TOP

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. See "Fast PWM Mode" on page 97 for more details.

Table 13-4 shows the COM0B1:0 bit functionality when the WGM02:0 bits are set to phase correct PWM mode.

Table 13-7. Compare Output Mode, Phase Correct PWM Mode[1]

COM0A1	COM0A0	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. See "Phase Correct PWM Mode" on page 99 for more details.

- **Bits 3, 2 – Res: Reserved Bits**

These bits are reserved bits in the ATmega16U4/ATmega32U4 and will always read as zero.

- **Bits 1:0 – WGM01:0: Waveform Generation Mode**

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, see Table 13-8. 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" on page 96).

Table 13-8. Waveform Generation Mode Bit Description

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRx 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	TOP	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	TOP	TOP

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

Timer/Counter Control Register B – TCCR0B

Bit	7	6	5	4	3	2	1	0	
	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	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.

However, for ensuring 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 COM0A1:0 bits setting. Note that the FOC0A bit is implemented as a strobe. Therefore it is the value present in the COM0A1:0 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.

However, for ensuring 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 COM0B1:0 bits setting. Note that the FOC0B bit is implemented as a strobe. Therefore it is the value present in the COM0B1:0 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.

- **Bits 5:4 – Res: Reserved Bits**

These bits are reserved bits and will always read as zero.

- **Bit 3 – WGM02: Waveform Generation Mode**

See the description in the "Timer/Counter Control Register A – TCCR0A" on page 102.

- **Bits 2:0 – CS02:0: Clock Select**

The three Clock Select bits select the clock source to be used by the Timer/Counter.

Table 13-9. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)

Table 13-9. Clock Select Bit Description (Continued)

CS02	CS01	CS00	Description
1	0	0	$clk_{I/O}$/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.

```
////////////////////////////////////////////////////////////////////////
//
// ATmega32u4
//
// Rechteck am PD7 mit 8-Bit Timer_0 OVERFLOW
// tein = taus = 500us
//
// f_CLK_IO = 8MHz
//
////////////////////////////////////////////////////////////////////////


#include <avr/io.h>                 //Registerdeklerationen
#include <avr/interrupt.h>          //Makros sei(), cli() und Interruptnamen


ISR (TIMER0_OVF_vect)
{
      PORTD = PORTD^(1<<PORTD7);    //toggle PD7
      TCNT0 = 255+1-62;             //Reinitialisierung
}



int main (void)
{
      DDRD = DDRD | (1<<DDD7);      //PD7 als Ausgang

      TIMSK0 = TIMSK0 | (1<<TOIE0); //Timer_0 Overflow Interrupt freigeben
      TCNT0 = 255+1-62;             //Startwert für 500us (62*8us=500us)
      TCCR0B = TCCR0B | (1<<CS01)|(1<<CS00);//:64 Teiler => dt=64/8MHz=8us
      sei();

      while(1);
}
```

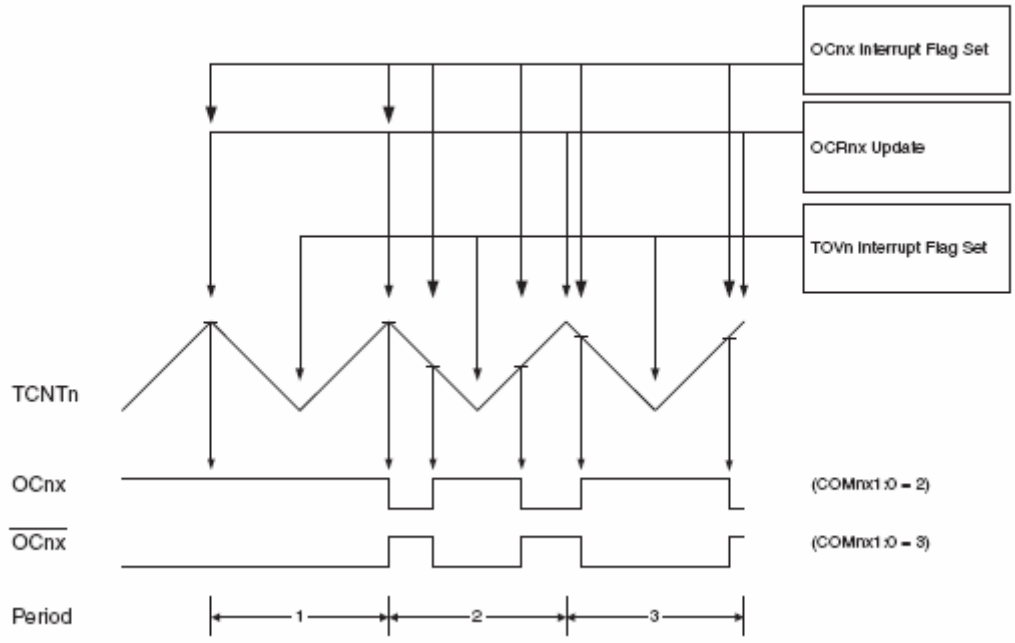
```
////////////////////////////////////////////////////////////////////
//
// ATmega32u4
//
// Rechteck am PD7 mit 8-Bit Timer_0 COMPARE_A
// tein = taus = 400us
//
// f_CLK_IO = 8MHz
//
////////////////////////////////////////////////////////////////////


#include <avr/io.h>                 //Registerdeklerationen
#include <avr/interrupt.h>          //Makros sei(), cli() und Interruptnamen


ISR (TIMER0_COMPA_vect)
{
      PORTD = PORTD^(1<<PORTD7);    //toggle PD7
      TCNT0 = 0x00;                 //Zählerstand rücksetzen
}



int main (void)
{
      DDRD = DDRD | (1<<DDD7);      //PD7 als Ausgang

      TIMSK0 = TIMSK0|(1<<OCIE0A);  //Timer_0 Compare_A Interrupt freigeben
      OCR0A = 50-1;                 //Endwert für 400us (50*8us=400us)
      TCCR0B = TCCR0B|(1<<CS01)|(1<<CS00);//:64 Teiler => dt=64/8MHz=8us
      sei();

      while(1);
}
```

Table 13-8. Waveform Generation Mode Bit Description

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRx 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	TOP	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	TOP	TOP

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

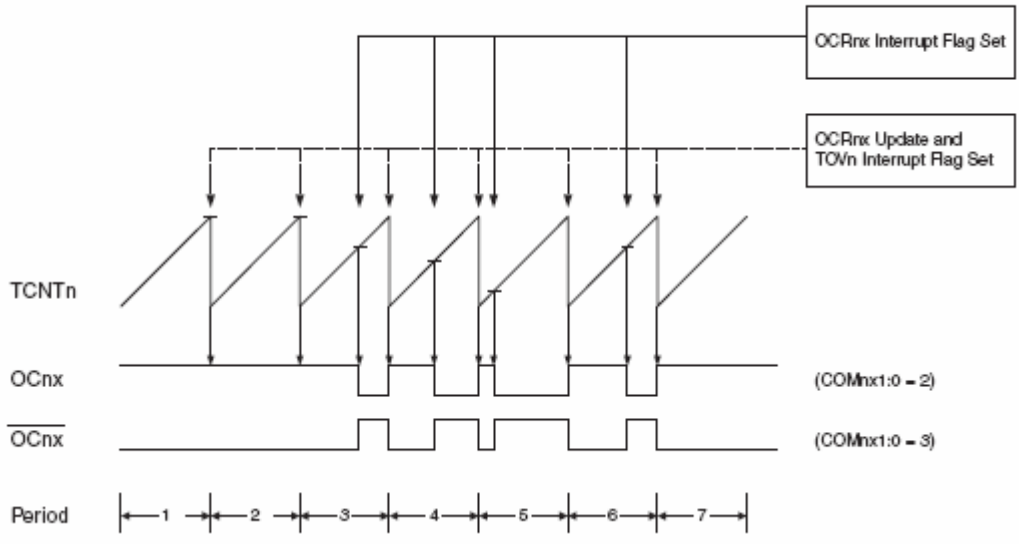
Figure 13-7. Phase Correct PWM Mode, Timing Diagram



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

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

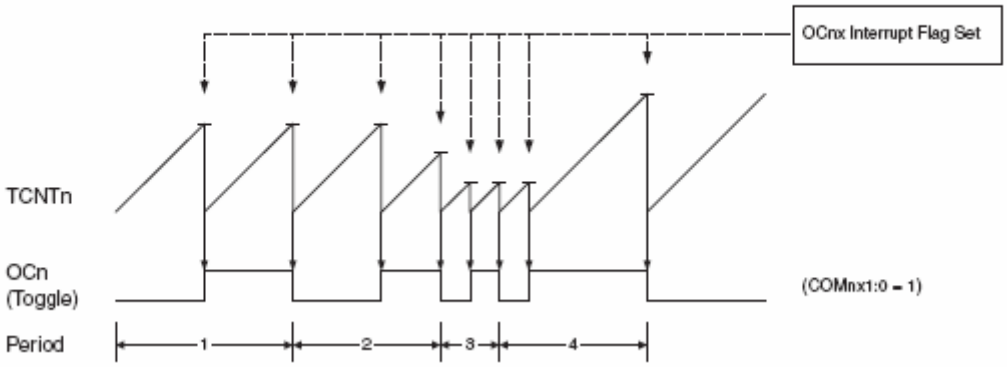
Figure 13-6. Fast PWM Mode, Timing Diagram



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

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

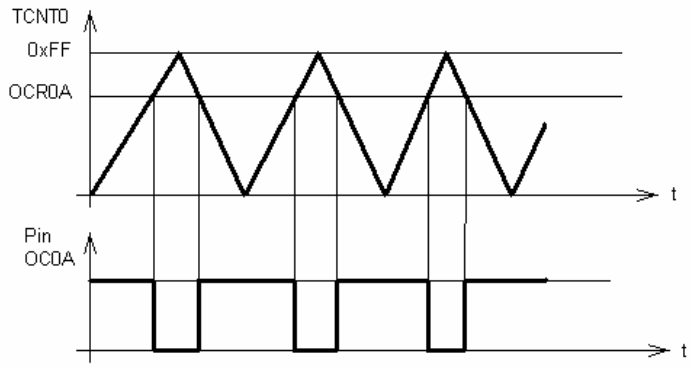
Figure 13-5. CTC Mode, Timing Diagram



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

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

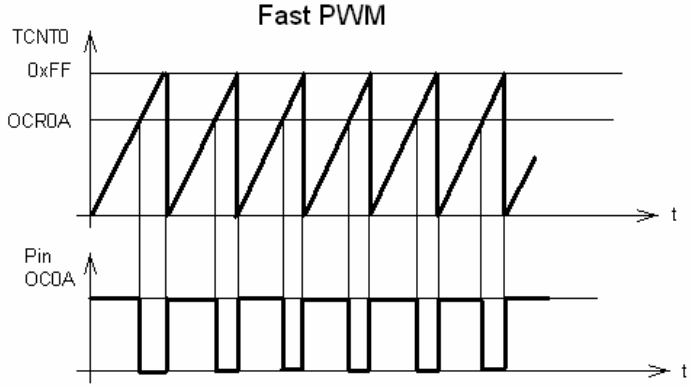
Phase Correct PWM



WGM2:0 = 1 Phase Correct PWM

COM1:0 =2 Non Inverting Pin OC0A

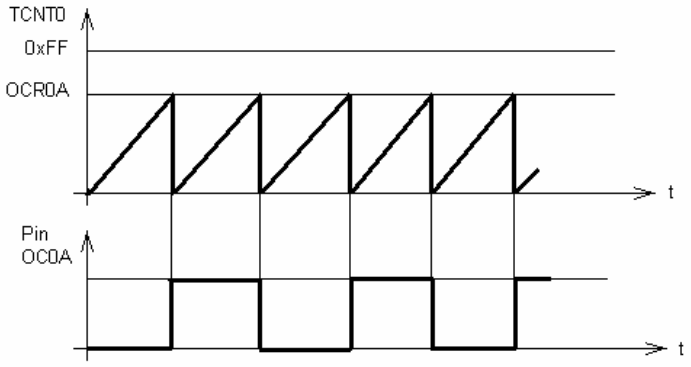
Fast PWM



WGM2:0 = 3 Fast PWM

COM1:0 =2 Non Inverting Pin OC0A

CTC (Claer Timer on Compare Match)



WGM2:0 = 2 CTC

COM1:0=1

```
////////////////////////////////////////////////////////////////////////////
//
// ATmega32u4
//
// PWM "PWM Phase Correct" am OC0A (PB7)
//
// Das Tastverhältnis kann durch drücken
// der Taste am PB0 kontinuierlich verändert werden
//
// WGM2:0 = 1 PWM Phase Correct Mode
// COM1:0 = 2 Non Inverting (Clear OC0A on Compare Match when Up Counting
//                          Set   OC0A on Compare Match when Down Counting)
// OCF0A-Flag => ISR(TIMER0_COMPA_vect)
// TOV0-Flag  => ISR(TIMER0_OVF_vect)
//
// fPWM = f_CLK_IO/(Teiler*512) = 8MHz/(64*512) = 244 Hz
//
// f_CLK_IO = 8MHz
//
////////////////////////////////////////////////////////////////////////////

#include <avr/io.h>                          //Registerdeklerationen

int main (void)
{
        DDRB = DDRB | (1<<DDB7);             //PB7 als Ausgang
        DDRB = DDRB &~(1<<DDB0);             //PB0 als Input
        PORTB = PORTB | (1<<PORTB0);         //PB0 int. Pull Up ein

        TCCR0A = TCCR0A | (1<<WGM00);
        TCCR0A = TCCR0A &~(1<<WGM01);
        TCCR0B = TCCR0B &~(1<<WGM02);        //WGM2:0=1     Phase Correct PWM


        TCCR0A = TCCR0A &~(1<<COM0A0);
        TCCR0A = TCCR0A | (1<<COM0A1);       //COM1:0=2      Non Inverting

        TCCR0B = TCCR0B|(1<<CS01)|(1<<CS00); //:64 Teiler, startet die PWM


        while(1)
                {
                        while(PINB&(1<<PINB0));//warten auf neg. Flanke von Taste PB0
                        OCR0A = OCR0A + 10;
                        while(!(PINB&(1<<PINB0)));//warten auf pos. Flanke von Taste PB0
                }
        return 0;
}
```

16-Bit TIMER / COUNTER 1 und TIMER / COUNTER 3

- True 16-bit Design (i.e., Allows 16-bit PWM)
- Three independent Output Compare Units
- Double Buffered Output Compare Registers
- One Input Capture Unit
- Input Capture Noise Canceler
- Clear Timer on Compare Match (Auto Reload)
- Glitch-free, Phase Correct Pulse Width Modulator (PWM)
- Variable PWM Period
- Frequency Generator
- External Event Counter
- Ten independent interrupt sources (TOV1, OCF1A, OCF1B, OCF1C, ICF1, TOV3, OCF3A, OCF3B, OCF3C and ICF3)

n=1 für Timer/Counter1 n=3 für Timer/Counter3

Register: *16-Bit Register*

TCNTn	Timer/Counter Zählerstand
OCRnA/B/C	Output Compare
ICRn	Input Capture

8-Bit Register

SREG	Statusregister
TCCRnA/B/C	Control Register
TIFRn	Timer Interrupt Flag Register
TIMSKn	Timer Interrupt Mask Register

Pins:

T1 (PD6)	Input Counter1
T3 existiert nicht	Input Counter3
ICP1 (PD4)	Input Capture1
ICP3 (PC7)	Input Capture3
OC1A (PB5)	Output Compare1A
OC1B (PB6)	Output Compare1B
OC1C (PB7)	Output Compare1C
OC3A (PC6)	Outout Compare 3A
OC3B existert nicht	Output Compare 3B
OC3C existiert nicht	Output Compare 3C

Timer/Counter1 – TCNT1H and TCNT1L

Bit	7	6	5	4	3	2	1	0	
	TCNT1[15:8]								TCNT1H
	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter3 – TCNT3H and TCNT3L

Bit	7	6	5	4	3	2	1	0	
	TCNT3[15:8]								TCNT3H
	TCNT3[7:0]								TCNT3L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Output Compare Register 1 A – OCR1AH and OCR1AL

Bit	7	6	5	4	3	2	1	0	
	OCR1A[15:8]								OCR1AH
	OCR1A[7:0]								OCR1AL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Output Compare Register 1 B – OCR1BH and OCR1BL

Bit	7	6	5	4	3	2	1	0	
	OCR1B[15:8]								OCR1BH
	OCR1B[7:0]								OCR1BL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Output Compare Register 1 C – OCR1CH and OCR1CL

Bit	7	6	5	4	3	2	1	0	
	OCR1C[15:8]								OCR1CH
	OCR1C[7:0]								OCR1CL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Output Compare Register 3 A – OCR3AH and OCR3AL

Bit	7	6	5	4	3	2	1	0	
	OCR3A[15:8]								OCR3AH
	OCR3A[7:0]								OCR3AL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Output Compare Register 3 B – OCR3BH and OCR3BL

Bit	7	6	5	4	3	2	1	0	
	OCR3B[15:8]								OCR3BH
	OCR3B[7:0]								OCR3BL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Output Compare Register 3 C – OCR3CH and OCR3CL

Bit	7	6	5	4	3	2	1	0	
	OCR3C[15:8]								OCR3CH
	OCR3C[7:0]								OCR3CL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Input Capture Register 1 – ICR1H and ICR1L

Bit	7	6	5	4	3	2	1	0	
	ICR1[15:8]								ICR1H
	ICR1[7:0]								ICR1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Input Capture Register 3 – ICR3H and ICR3L

Bit	7	6	5	4	3	2	1	0	
	ICR3[15:8]								ICR3H
	ICR3[7:0]								ICR3L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter1 Interrupt Mask Register – TIMSK1

Bit	7	6	5	4	3	2	1	0	
	–	–	ICIE1	–	OCIE1C	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter3 Interrupt Mask Register – TIMSK3

Bit	7	6	5	4	3	2	1	0	
	–	–	ICIE3	–	OCIE3C	OCIE3B	OCIE3A	TOIE3	TIMSK3
Read/Write	R	R	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 5 – ICIEn: Timer/Countern, Input Capture Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Countern Input Capture interrupt is enabled. The corresponding Interrupt Vector (See “Interrupts” on page 61.) is executed when the ICFn Flag, located in TIFRn, is set.

- **Bit 3 – OCIEnC: Timer/Countern, Output Compare C Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Countern Output Compare C Match interrupt is enabled. The corresponding Interrupt Vector (See “Interrupts” on page 61.) is executed when the OCFnC Flag, located in TIFRn, is set.

- **Bit 2 – OCIEnB: Timer/Countern, Output Compare B Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Countern Output Compare B Match interrupt is enabled. The corresponding Interrupt Vector (See “Interrupts” on page 61.) is executed when the OCFnB Flag, located in TIFRn, is set.

- **Bit 1 – OCIEnA: Timer/Countern, Output Compare A Match Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Countern Output Compare A Match interrupt is enabled. The corresponding Interrupt Vector (See “Interrupts” on page 61.) is executed when the OCFnA Flag, located in TIFRn, is set.

- **Bit 0 – TOIEn: Timer/Countern, Overflow Interrupt Enable**

When this bit is written to one, and the I-flag in the Status Register is set (interrupts globally enabled), the Timer/Countern Overflow interrupt is enabled. The corresponding Interrupt Vector (See “Interrupts” on page 61.) is executed when the TOVn Flag, located in TIFRn, is set.

Timer/Counter1 Interrupt Flag Register – TIFR1

Bit	7	6	5	4	3	2	1	0	
	–	–	ICF1	–	OCF1C	OCF1B	OCF1A	TOV1	TIFR1
Read/Write	R	R	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter3 Interrupt Flag Register – TIFR3

Bit	7	6	5	4	3	2	1	0	
	–	–	ICF3	–	OCF3C	OCF3B	OCF3A	TOV3	TIFR3
Read/Write	R	R	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter1 Control Register A – TCCR1A

Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter3 Control Register A – TCCR3A

Bit	7	6	5	4	3	2	1	0	
	COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30	TCCR3A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – COMnA1:0: Compare Output Mode for Channel A**
- **Bit 5:4 – COMnB1:0: Compare Output Mode for Channel B**
- **Bit 3:2 – COMnC1:0: Compare Output Mode for Channel C**

Timer/Counter1 Control Register B – TCCR1B

Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter3 Control Register B – TCCR3B

Bit	7	6	5	4	3	2	1	0	
	ICNC3	ICES3	–	WGM33	WGM32	CS32	CS31	CS30	TCCR3B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 14-6. Clock Select Bit Description

CSn2	CSn1	CSn0	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	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on Tn pin. Clock on falling edge
1	1	1	External clock source on Tn pin. Clock on rising edge

Timer/Counter1 Control Register C – TCCR1C

Bit	7	6	5	4	3	2	1	0	
	FOC1A	FOC1B	FOC1C	–	–	–	–	–	TCCR1C
Read/Write	W	W	W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Timer/Counter3 Control Register C – TCCR3C

Bit	7	6	5	4	3	2	1	0	
	FOC3A	–	–	–	–	–	–	–	TCCR3C
Read/Write	W	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

PWM

Table 14-5. Waveform Generation Mode Bit Description [1]

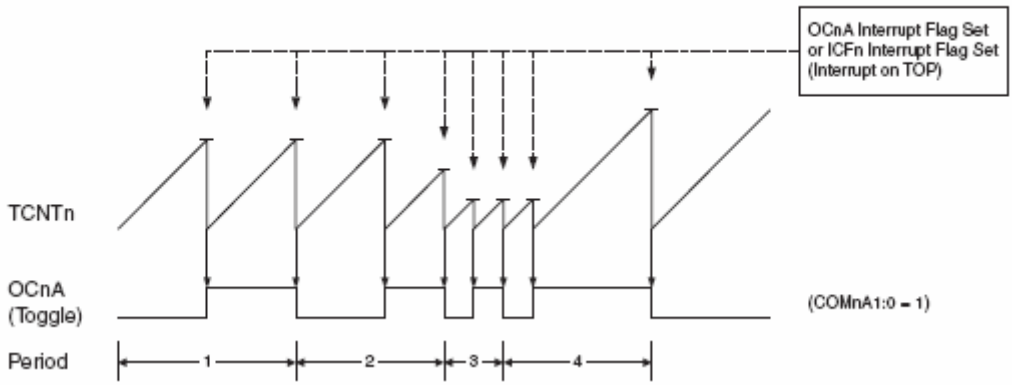
Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter Mode of Operation	TOP	Update of OCRnX at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX

Table 14-5. Waveform Generation Mode Bit Description (Continued)[1]

Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter Mode of Operation	TOP	Update of OCRnX at	TOVn Flag Set on
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICRn	TOP	TOP
15	1	1	1	1	Fast PWM	OCRnA	TOP	TOP

Note: 1. The CTCn and PWMn1:0 bit definition names are obsolete. Use the WGMn2:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

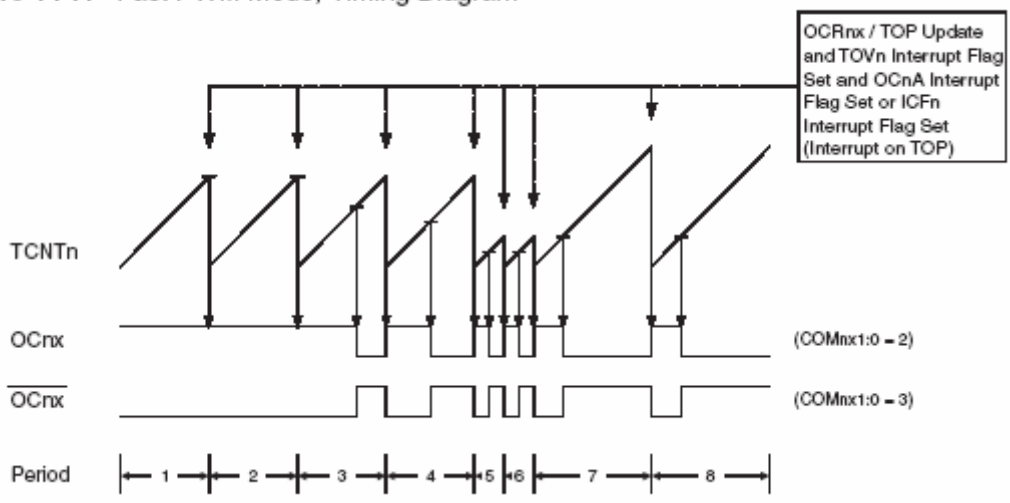
Figure 14-6. CTC Mode, Timing Diagram



$$f_{OCnA} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRnA)}$$

The *N* variable represents the prescaler factor (1, 8, 64, 256, or 1024)

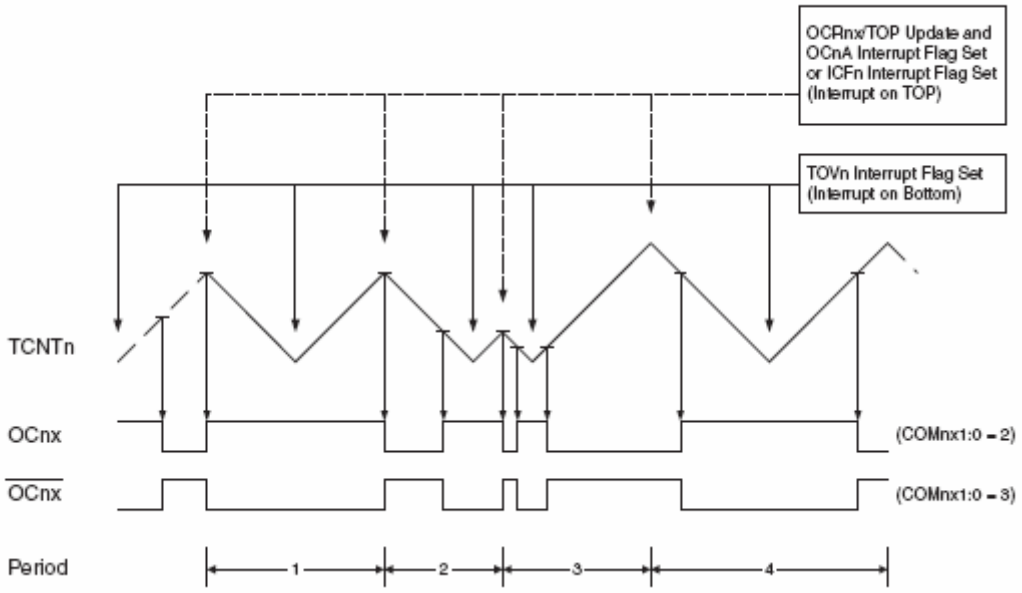
Figure 14-7. Fast PWM Mode, Timing Diagram



$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot (1 + TOP)}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024)

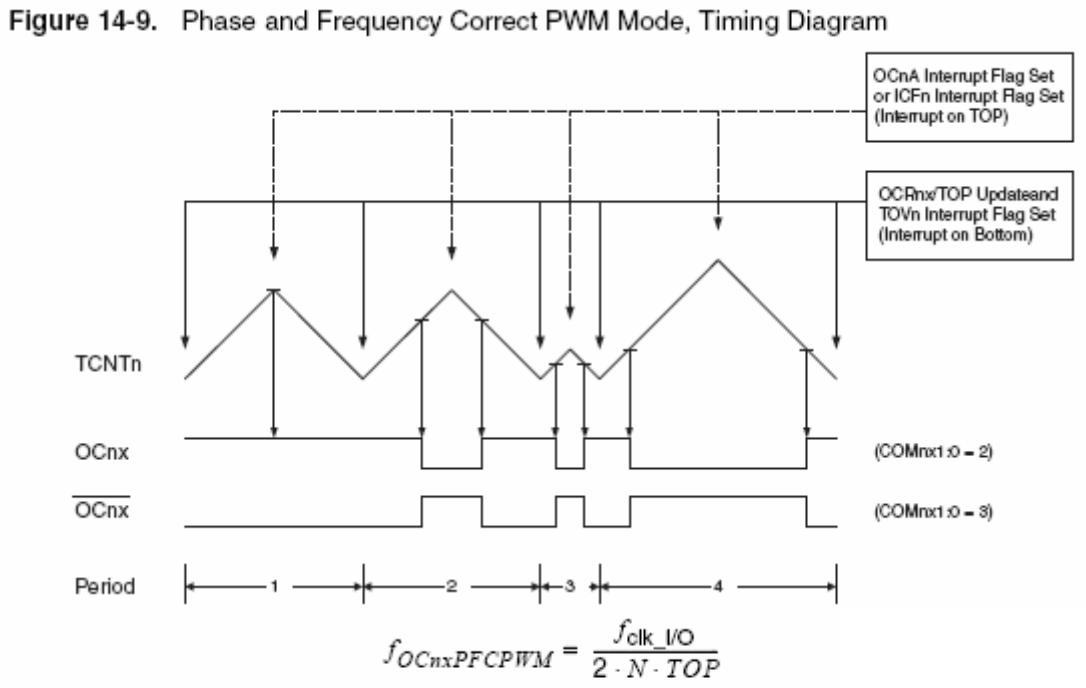
Figure 14-8. Phase Correct PWM Mode, Timing Diagram



$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot TOP}$$

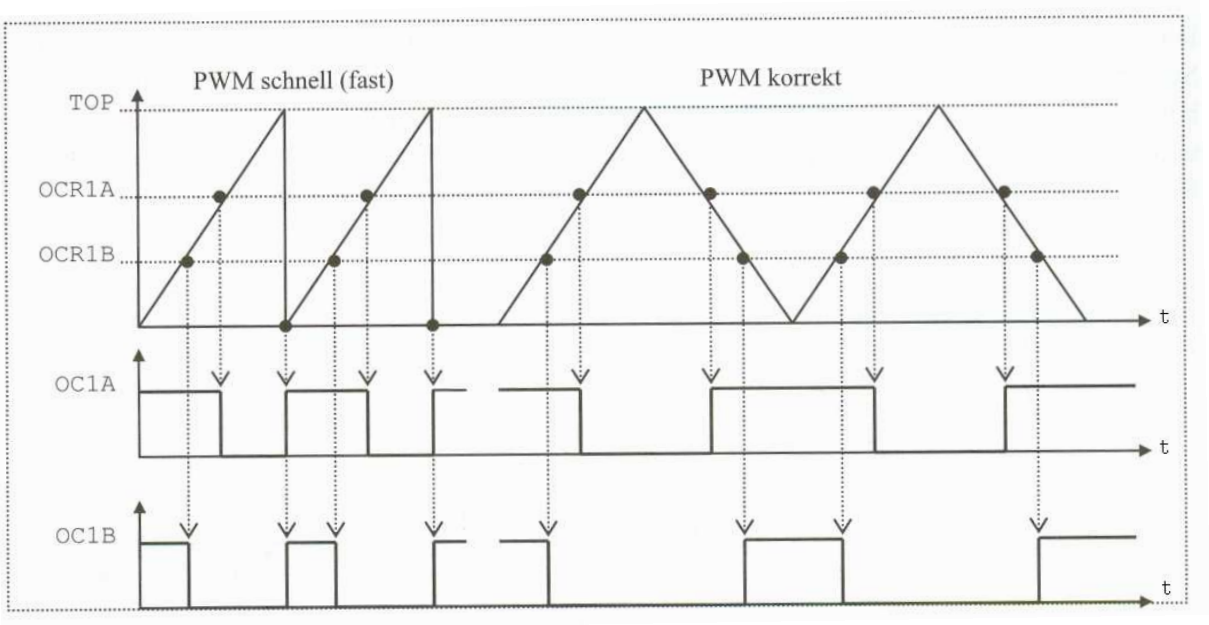
The N variable represents the prescaler divider (1, 8, 64, 256, or 1024)

Figure 14-9. Phase and Frequency Correct PWM Mode, Timing Diagram



$$f_{OCnxPFCPWM} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot TOP}$$

The N variable represents the prescaler divider (1, 8, 64, 256, or 1024)



```
////////////////////////////////////////////////////////////////////////////
//
// ATmega32u4
//
// Rechteck am PD7 mit 16-Bit Timer_1 OVERFLOW
// tein = taus = 2 ms
//
// f_CLK_IO = 8MHz
//
////////////////////////////////////////////////////////////////////////////


#include <avr/io.h>                    //Registerdeklerationen
#include <avr/interrupt.h>             //Makros sei(), cli() und Interruptnamen


ISR (TIMER1_OVF_vect)
{
      PORTD = PORTD^(1<<PORTD7);       //toggle PD7
      TCNT1 = 0xFFFF+1-250;            //Reinitialisierung
}



int main (void)
{
      DDRD = DDRD | (1<<DDD7);         //PD7 als Ausgang

      TIMSK1 = TIMSK1 | (1<<TOIE1);    //Timer_0 Overflow Interrupt freigeben
      TCNT1 = 0xFFFF+1-250;            //Startwert für 2ms (250*8us=2ms)
      TCCR1B = TCCR1B | (1<<CS11)|(1<<CS10);//:64 Teiler => dt=64/8MHz=8us
      sei();

      while(1);
}
```

```
//////////////////////////////////////////////////////////////////////
//
// ATmega32u4
//
// Timer 1 PWM "10-Bit Fast PWM" am OC1A (PB5)
// PD7 ändert 1:1 durch PWM-Interrupts sein Tastverhältnis (0-100)%
//
// Das Tastverhältnis kann durch drücken
// der Taste am PB0 kontinuierlich verändert werden
//
// WGM3:0 = 7 10-Bit Fast PWM mit TOP=0x03FF
// COM1:0 = 2 (Clear OC1A on Compare Match when Up Counting
//             Set   OC1A at TOP)
// OCF1A-Flag => ISR(TIMER1_COMPA_vect)
// TOV1-Flag  => ISR(TIMER1_OVF_vect)
//
// fPWM = f_CLK_IO/(Teiler*(1+TOP)) = 8MHz/(64*(1+0x03FF)) = 122Hz
//
// f_CLK_IO = 8MHz
//
//////////////////////////////////////////////////////////////////////

#include <avr/io.h>            //Registerdeklerationen
#include <avr/interrupt.h>     //Makros sei(), cli() und Interruptnamen

ISR(TIMER1_COMPA_vect)
{
      PORTD = PORTD &~ (1<<PORTD7);
}

ISR(TIMER1_OVF_vect)
{
      PORTD = PORTD | (1<<PORTD7);
}

int main (void)
{
      DDRB = DDRB | (1<<DDB5);              //PB7 als Ausgang
      DDRB = DDRB &~(1<<DDB0);              //PB0 als Input
      PORTB = PORTB | (1<<PORTB0);          //PB0 int. Pull Up ein
      DDRD = DDRD|(1<<DDD7);

      TCCR1A = TCCR1A | (1<<WGM10);
      TCCR1A = TCCR1A | (1<<WGM11);
      TCCR1B = TCCR1B | (1<<WGM12);
      TCCR1B = TCCR1B &~(1<<WGM13);         //WGM3:0=7      10-Bit Fast PWM

      TCCR1A = TCCR1A &~(1<<COM1A0);
      TCCR1A = TCCR1A | (1<<COM1A1);        //COM1:0=2      Non Inverting

      TCCR1B = TCCR1B|(1<<CS11)|(1<<CS10); //:64 Teiler, startet PWM

      TIMSK1 = TIMSK1|(1<<OCIE1A)|(1<<TOIE1);
      sei();                                //Interrupts freigeben

      while(1)
            {
                  while(PINB & (1<<PINB0));   //warten auf neg. Flanke von PB0
                  OCR1A = OCR1A + 10;
                  while(!(PINB & (1<<PINB0)));//warten auf pos. Flanke von PB0
            }
}
```

10-Bit HIGH SPEED TIMER / COUNTER 4

- **Up to 10-Bit Accuracy**
- **Three Independent Output Compare Units**
- **Clear Timer on Compare Match (Auto Reload)**
- **Glitch Free, Phase and Frequency Correct Pulse Width Modulator (PWM)**
- **Enhanced PWM mode: one optional additional accuracy bit without effect on output frequency**
- **Variable PWM Period**
- **Independent Dead Time Generators for each PWM channels**
- **Synchronous update of PWM registers**
- **Five Independent Interrupt Sources (TOV4, OCF4A, OCF4B, OCF4D, FPF4)**
- **High Speed Asynchronous and Synchronous Clocking Modes**
- **Separate Prescaler Unit**

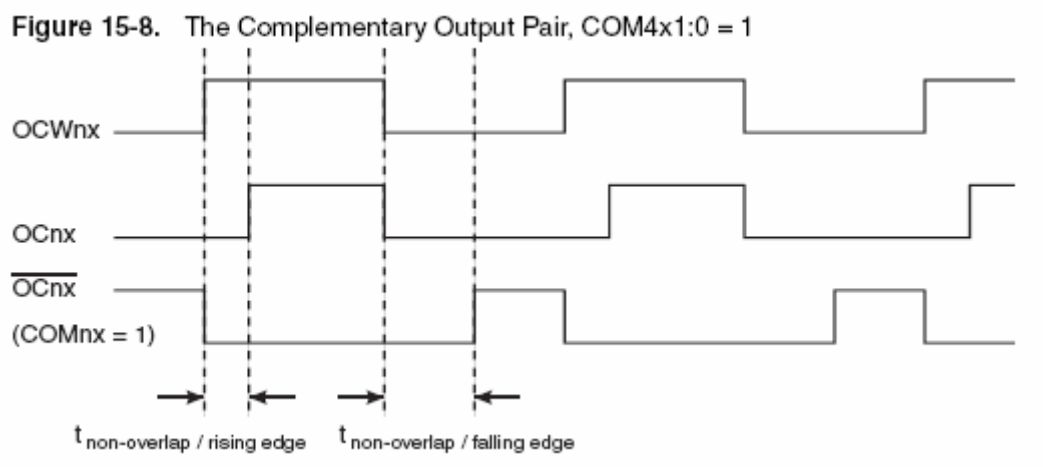
Register: TCNT4, TC4H, OCR4A/B/C/D, TIFR4, TIMSK4, TCCR4A/B/C/D/E, DT4

Pins: OC4A (PC7) und #OC4A (PC6)
OC4B (PB6) und #OC4B (PB5)
OC4C (PD7) und #OC4C (PD6)

Interne Clock Speed bis zu max. 64 MHz

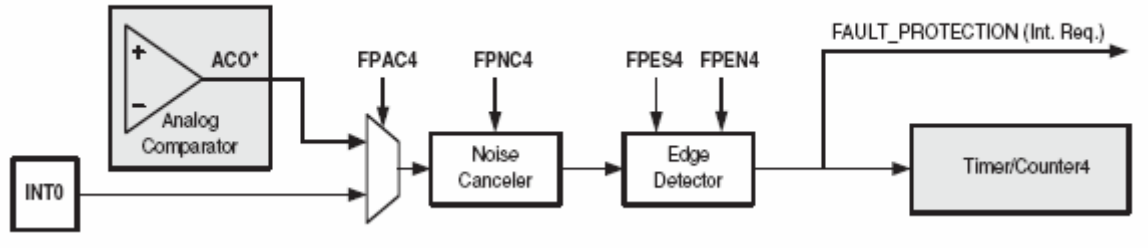
Dead Time Generator fügt Verzögerungszeiten zwischen den Schaltflanken der komplementären Ausgangspins (OC4nx) ein (z.B. zur Ansteuerung von Gegentaktstufen, Schaltnetzteilen usw. nötig)

Figure 15-8. The Complementary Output Pair, COM4x1:0 = 1



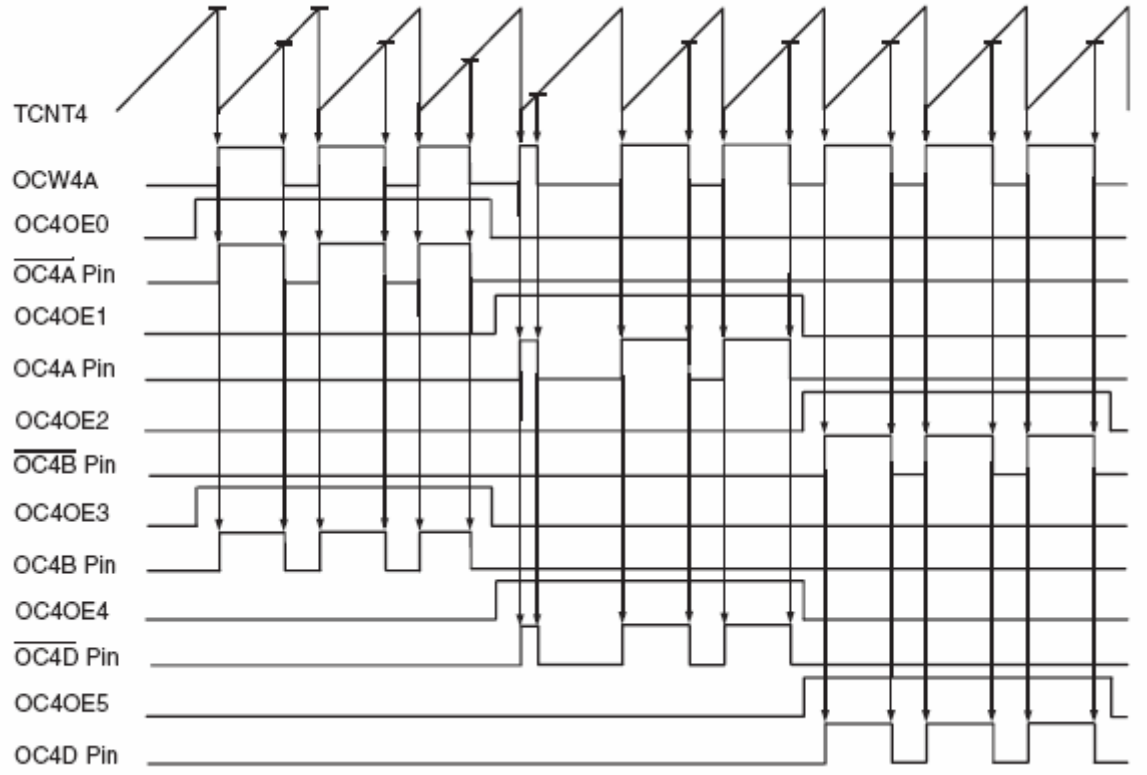
Fault Protection Unit deaktiviert die PWM Output Pins wenn am INT0 Pin oder am analogen Komparator ein Ereignis auftritt. Der **Noise Canceler** ist ein digitales Filter an dem min. 4 Samples gleich sein müssen um dann den Edge Detector anzusteuern.

Figure 15-20. Fault Protection Unit Block Diagram



PWM6 Mode zur Ansteuerung von Brushless DC Motoren (BLDC-Motoren)

Figure 15-15. PWM6 Mode, Single-slope Operation, Timing Diagram

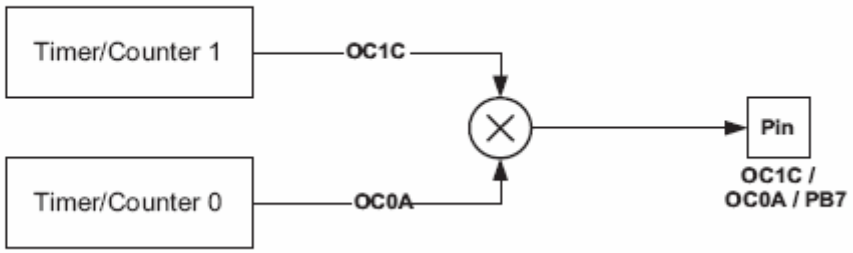


Output Compare Modulator (OCM1C0A)

Register: von Timer0 und Timer1

Pin: OC1C, OC0A, PB7 (alles gleicher Pin)

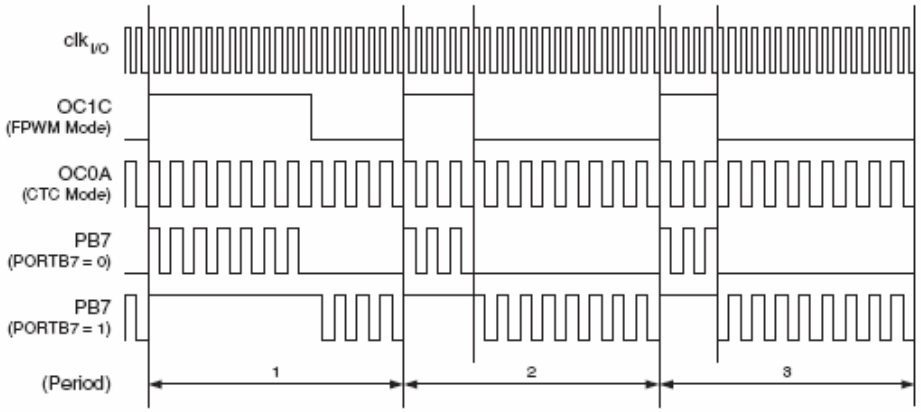
Figure 16-1. Output Compare Modulator, Block Diagram



Timing Example

Figure 16-3 illustrates the modulator in action. In this example the Timer/Counter1 is set to operate in fast PWM mode (non-inverted) and Timer/Counter0 uses CTC waveform mode with toggle Compare Output mode (COMnx1:0 = 1).

Figure 16-3. Output Compare Modulator, Timing Diagram



In this example, Timer/Counter0 provides the carrier, while the modulating signal is generated by the Output Compare unit C of the Timer/Counter1.

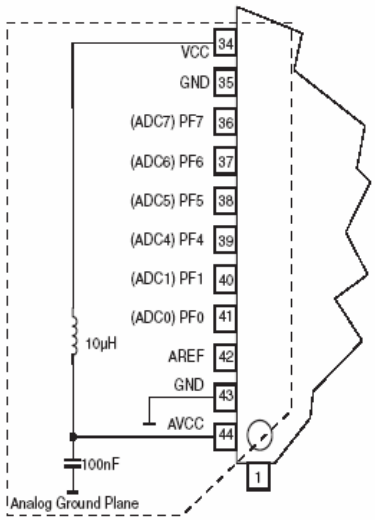
ADC

Features

- 10/8-bit Resolution
- 0.5 LSB Integral Non-linearity
- ± 2 LSB Absolute Accuracy
- 65 - 260 µs Conversion Time
- Up to 15 kSPS at Maximum Resolution
- Twelve Multiplexed Single-Ended Input Channels
- One Differential amplifier providing gain of 1x - 10x - 40x - 200x
- Temperature sensor
- Optional Left Adjustment for ADC Result Readout
- 0 - V_{CC} ADC Input Voltage Range
- Selectable 2.56 V ADC Reference Voltage
- Free Running or Single Conversion Mode
- ADC Start Conversion by Auto Triggering on Interrupt Sources
- Interrupt on ADC Conversion Complete
- Sleep Mode Noise Canceler

Register: ADMUX, ADCSRA, ADCx, ADCSRB, DIDR0, DIDR2

Figure 24-10. ADC Power Connections



Note: The same circuitry should be used for AVCC filtering on the ADC8-ADC13 side

50 kHz ≤ f_{ADC} ≤ 200 kHz für max. Auflösung ideal (Bits ADPS[2..0] im ADCSRA)
(f_{ADC} > 200kHz möglich, aber Auflösung dann geringer)

f_{in_max} ≤ f_{ADC}/2 (Nyquist)
max. Innenwiderstand der Signalquelle: ≤ 10 kΩ bei single ended mode
≤ einige 100 kΩ bei differential mode

ADC Multiplexer Selection Register – ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – REFS1:0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 24-3. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 24-3. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AV_{CC} with external capacitor on AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor on AREF pin

- **Bit 5 – ADLAR: ADC Left Adjust Result**

The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conversions. For a complete description of this bit, see "The ADC Data Register – ADCL and ADCH" on page 311.

- **Bits 4:0 – MUX4:0: Analog Channel Selection Bits**

The value of these bits selects which combination of analog inputs are connected to the ADC. These bits also select the gain for the differential channels. See Table 24-4 for details. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set).

Table 24-4. Input Channel and Gain Selections

MUX5..0[1]	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
000000	ADC0	N/A		
000001	ADC1			
000010	N/A			
000011				
000100	ADC4			
000101	ADC5			
000110	ADC6			
000111	ADC7			
001000	N/A	N/A	N/A	N/A
001001		ADC1	ADC0	10x
001010		N/A	N/A	N/A
001011		ADC1	ADC0	200x
001100		N/A		
001101				
001110				
001111				
010000		ADC0	ADC1	1x

Table 24-4. Input Channel and Gain Selections (Continued)

MUX5..0[1]	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
010001	N/A	N/A		
010010				
010011				
010100		ADC4	ADC1	1x
010101		ADC5	ADC1	1x
010110		ADC6	ADC1	1x
010111		ADC7	ADC1	1x
011000		N/A		
011001				
011010				
011011				
011100				
011101				
011110	1.1V ($V_{Band\ Gap}$)			
011111	0V (GND)			
100000	ADC8			
100001	ADC9			
100010	ADC10			
100011	ADC11			
100100	ADC12			
100101	ADC13			
100110	N/A	ADC1	ADC0	40x
100111	Temperature Sensor			
101000	N/A	ADC4	ADC0	10x
101001		ADC5	ADC0	10x
101010		ADC6	ADC0	10x
101011		ADC7	ADC0	10x
101100		ADC4	ADC1	10x
101101		ADC5	ADC1	10x
101110		ADC6	ADC1	10x
101111		ADC7	ADC1	10x
110000		ADC4	ADC0	40x
110001		ADC5	ADC0	40x
110010		ADC6	ADC0	40x
110011		ADC7	ADC0	40x

Table 24-4. Input Channel and Gain Selections (Continued)

MUX5..0[1]	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
110100	N/A	ADC4	ADC1	40x
110101		ADC5	ADC1	40x
110110		ADC6	ADC1	40x
110111		ADC7	ADC1	40x
111000		ADC4	ADC0	200x
111001		ADC5	ADC0	200x
111010		ADC6	ADC0	200x
111011		ADC7	ADC0	200x
111100		ADC4	ADC1	200x
111101		ADC5	ADC1	200x
111110		ADC6	ADC1	200x
111111		ADC7	ADC1	200x

Note: 1. MUX5 bit make part of ADCSRB register

ADC Control and Status Register A – ADCSRA

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADEN: ADC Enable**

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

In Single Conversion mode, write this bit to one to start each conversion. In Free Running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

- **Bit 5 – ADATE: ADC Auto Trigger Enable**

When this bit is written to one, Auto Triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in ADCSRB.

- **Bit 4 – ADIF: ADC Interrupt Flag**

This bit is set when an ADC conversion completes and the Data Registers are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

- **Bit 3 – ADIE: ADC Interrupt Enable**

When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

- **Bits 2:0 – ADPS2:0: ADC Prescaler Select Bits**

These bits determine the division factor between the XTAL frequency and the input clock to the ADC.

Table 24-5. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

The ADC Data Register – ADCL and ADCH

ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
Bit	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	–	–	–	–	–	–	ADCL
Bit	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

When an ADC conversion is complete, the result is found in these two registers. If differential channels are used, the result is presented in two's complement form.

When ADCL is read, the ADC Data Register is not updated until ADCH is read. Consequently, if the result is left adjusted and no more than 8-bit precision (7 bit + sign bit for differential input channels) is required, it is sufficient to read ADCH. Otherwise, ADCL must be read first, then ADCH.

The ADLAR bit in ADMUX, and the MUXn bits in ADMUX affect the way the result is read from the registers. If ADLAR is set, the result is left adjusted. If ADLAR is cleared (default), the result is right adjusted.

- **ADC9:0: ADC Conversion Result**

These bits represent the result from the conversion, as detailed in "ADC Conversion Result" on page 305.

ADC Control and Status Register B – ADCSRB

Bit	7	6	5	4	3	2	1	0	
	ADHSM	ACME	MUX5	–	ADTS3	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R/W	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADHSM: ADC High Speed Mode**

Writing this bit to one enables the ADC High Speed mode. This mode enables higher conversion rate at the expense of higher power consumption.

- **Bit 5 – MUX5: Analog Channel Additional Selection Bits**

This bit make part of MUX5:0 bits of ADRCSRB and ADMUX register, that select the combination of analog inputs connected to the ADC (including differential amplifier configuration).

- **Bit 3:0 – ADTS3:0: ADC Auto Trigger Source**

If ADATE in ADCSRA is written to one, the value of these bits selects which source will trigger an ADC conversion. If ADATE is cleared, the ADTS3:0 settings will have no effect. A conversion will be triggered by the rising edge of the selected interrupt flag. Note that switching from a trigger source that is cleared to a trigger source that is set, will generate a positive edge on the trigger signal. If ADEN in ADCSRA is set, this will start a conversion. Switching to Free Running mode (ADTS[3:0]=0) will not cause a trigger event, even if the ADC Interrupt Flag is set.

Table 24-6. ADC Auto Trigger Source Selections

ADTS3	ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	0	Free Running mode
0	0	0	1	Analog Comparator
0	0	1	0	External Interrupt Request 0
0	0	1	1	Timer/Counter0 Compare Match
0	1	0	0	Timer/Counter0 Overflow
0	1	0	1	Timer/Counter1 Compare Match B
0	1	1	0	Timer/Counter1 Overflow
0	1	1	1	Timer/Counter1 Capture Event
1	0	0	0	Timer/Counter4 Overflow

Table 24-6. ADC Auto Trigger Source Selections (Continued)

ADTS3	ADTS2	ADTS1	ADTS0	Trigger Source
1	0	0	1	Timer/Counter4 Compare Match A
1	0	1	0	Timer/Counter4 Compare Match B
1	0	1	1	Timer/Counter4 Compare Match D

Digital Input Disable Register 0 – DIDR0

Bit	7	6	5	4	3	2	1	0	
	ADC7D	ADC6D	ADC5D	ADC4D	-	-	ADC1D	ADC0D	DIDR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:4, 1:0 – ADC7D..4D - ADC1D..0D : ADC7:4 - ADC1:0 Digital Input Disable**

When this bit is written logic one, the digital input buffer on the corresponding ADC pin is disabled. The corresponding PIN Register bit will always read as zero when this bit is set. When an analog signal is applied to the ADC7..4 / ADC1..0 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

Digital Input Disable Register 2 – DIDR2

Bit	7	6	5	4	3	2	1	0	
	-	-	ADC13D	ADC12D	ADC11D	ADC10D	ADC9D	ADC8D	DIDR2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 5:0 – ADC13D..ADC8D: ADC13:8 Digital Input Disable**

When this bit is written logic one, the digital input buffer on the corresponding ADC pin is disabled. The corresponding PIN Register bit will always read as zero when this bit is set. When an analog signal is applied to the ADC13..8 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

```
////////////////////////////////////////////////////////////////////////////
//
//ATmega32u4
//8 MHz = f_ADC, :64 Teiler => 125 kHz Wandlertakt
//
//ADC Single Ended Messung am Pin ADC0 (PF0)
//Vref = AVCC
//
//linksbündige Ausgabe des gewandelten Werts
//Port B7:0 => Bit9 – Bit2 des 10-Bit Digitalwertes
//Port D7:6 => Bit1 – Bit0 des 10-Bit Digitalwertes
//
////////////////////////////////////////////////////////////////////////////


#include <avr/io.h>

#define F_CPU 8000000UL

int main(void)
{
unsigned char adc_low,adc_high;             //Hilfsvariablen

      DDRF = DDRF &~(1<<DDF7);              //PF7 INPUT
      DDRB = 0xFF;                          //ganzer PB OUTPUT
      DDRD = DDRD | (1<<DDD7)|(1<<DDD6);    //PD7:6 OUTPUT

      ADMUX  = ADMUX &~(1<<REFS1);
      ADMUX  = ADMUX | (1<<REFS0)|(1<<ADLAR);     //Vref=AVCC
      ADMUX  = ADMUX | (1<<ADLAR);                //linksbündig
      ADMUX  = ADMUX & 0b11100000;                //MUX4:0 = 0
      ADCSRB = ADCSRB &~ (1<<MUX5);               //MUX5   = 0  => MUX5:0=0 =>
                                                  //ADC0 (PF0) Single Ended
      ADCSRA = ADCSRA|(1<<ADEN)|(1<<ADSC)|(1<<ADPS2)|(1<<ADPS1); //Wandler ein,
                                                  //Start Wandler,:64 Teiler

      while(1)
      {
            while(ADCSRA & (1<<ADSC));      //warten auf Wandlungsende
            adc_low = ADCL;                 //zuerst immer Low Bits holen
            adc_high = ADCH;                //dann High Bits holen
            PORTB = adc_high;               //Ausgabe Bit9-Bit2 vom 10Bit Wert
            PORTD = adc_low;                //Ausgabe Bit1-Bit0 vom 10Bit Wert
            ADCSRA = ADCSRA|(1<<ADSC);      //Wandler neu starten
      }
      return(0);
}
```

Interpretation des Messergebnisses aus den ADCx Registern:

For single ended conversion, the result is:

$$ADC = \frac{V_{IN} \cdot 1023}{V_{REF}}$$

where V_{IN} is the voltage on the selected input pin and V_{REF} the selected voltage reference (see Table 24-3 on page 307 and Table 24-4 on page 308). 0x000 represents analog ground, and 0x3FF represents the selected reference voltage minus one LSB.

Im differential Mode wird das Ergebnis als 2er Komplement dargestellt:

If differential channels are used, the result is:

$$ADC = \frac{(V_{POS} - V_{NEG}) \cdot GAIN \cdot 512}{V_{REF}}$$

where V_{POS} is the voltage on the positive input pin, V_{NEG} the voltage on the negative input pin, GAIN the selected gain factor and V_{REF} the selected voltage reference. The result is presented in two's complement form, from 0x200 (-512d) through 0x1FF (+511d). Note that if the user wants to perform a quick polarity check of the result, it is sufficient to read the MSB of the result (ADC9 in ADCH). If the bit is one, the result is negative, and if this bit is zero, the result is positive. Figure 24-15 shows the decoding of the differential input range.

Figure 24-15. Differential Measurement Range

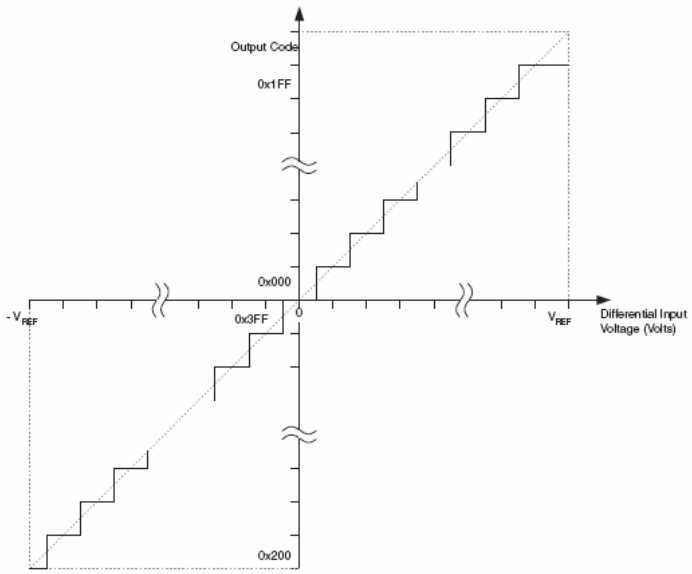


Table 24-2. Correlation Between Input Voltage and Output Codes

V_{ADCn}	Read code	Corresponding decimal value
$V_{ADCm} + V_{REF}/GAIN$	0x1FF	511
$V_{ADCm} + 0.999\ V_{REF}/GAIN$	0x1FF	511
$V_{ADCm} + 0.998\ V_{REF}/GAIN$	0x1FE	510
...	...	...
$V_{ADCm} + 0.001\ V_{REF}/GAIN$	0x001	1
V_{ADCm}	0x000	0
$V_{ADCm} - 0.001\ V_{REF}/GAIN$	0x3FF	-1
...	...	...
$V_{ADCm} - 0.999\ V_{REF}/GAIN$	0x201	-511
$V_{ADCm} - V_{REF}/GAIN$	0x200	-512

Example 1:

- ADMUX = 0xE9, MUX5 = 0 (ADC1 - ADC0, 10x gain, 2.56V reference, left adjusted result)
- Voltage on ADC1 is 300 mV, voltage on ADC0 is 500 mV.
- ADCR = 512 * 10 * (300 - 500) / 2560 = -400 = 0x270
- ADCL will thus read 0x00, and ADCH will read 0x9C. Writing zero to ADLAR right adjusts the result: ADCL = 0x70, ADCH = 0x02.

Hinweis: Umrechnung von **-** 400 auf einen Hexwert

$+400_{dez} = 190_{hex} = 01\ 1001\ 0000_{bin}$

10 0110 1111	1er Komplement
+1	
10 0111 0000	2er Komplement

$10\ 0111\ 0000_{bin} = 270_{hex} = -\ 400_{dez}$

Bsp.: Wie groß ist die Differenzspannung wenn im ADC Register bei obiger Konfiguration als Ergebnis 0x270 steht?

0x270 $= 10\ 0111\ 0000_{bin}$ 10-Bit Darstellung MSB=1 => neg. Zahl !!!

$01\ 1000\ 1111_{bin}$	1er Komplement
+1	
$01\ 1001\ 0000_{bin}$	2er Komplement $= 0x190 = 400_{dez} \Rightarrow -\ 400_{dez}$

$$ADC = \frac{(Vpos - Vneg) * GAIN * 512}{Vref}$$

$$-400 = \frac{(Vpos - Vneg) * 10 * 512}{2.56V}$$

$$(Vpos - Vneg) = \frac{-400 * 2.56V}{10 * 512} = -200mV$$

Temparatursensor

Sensortoleranz ±10℃ aber linear => Kalibrierung für absolute Messungen nötig

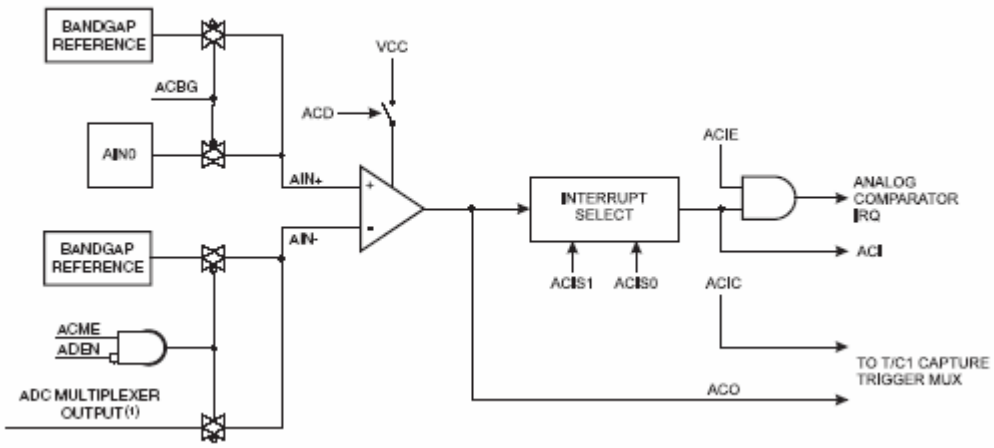
- Temparatursensor mit MUX[5..0] an single ended ADC einstellen
- Int. Vref muss verwendet werden
- Wegen der sukzessiven Approximation 2-3x Wert auslesen bis der richtige geliefert wird

Analog Comparator

Register: ADMUX, ADCSRA, ADCSRB, ACSR, DIDR1

Pins: für AIN+ => AIN0 (PE6) oder
int. Bandgap Reference
für AIN- => wahlweise einer der Pins ADC0:1 (PF0:1), ADC4:7 (PF4:7) oder
int. Bandgap Reference

Figure 23-1. Analog Comparator Block Diagram[2]



ADC Control and Status Register B – ADCSRB

Bit	7	6	5	4	3	2	1	0	
	ADHSM	ACME	MUX5	–	ADTS3	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 6 – ACME: Analog Comparator Multiplexer Enable**

When this bit is written logic one and the ADC is switched off (ADEN in ADCSRA is zero), the ADC multiplexer is connected to the negative input to the Analog Comparator. When this bit is written logic zero, the Bandgap reference is connected to the negative input of the Analog Comparator (See “Internal Voltage Reference” on page 54.). For a detailed description of this bit, see “Analog Comparator Multiplexed Input” on page 291.

Analog Comparator Control and Status Register – ACSR

Bit	7	6	5	4	3	2	1	0	
	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	N/A	0	0	0	0	0	

- **Bit 7 – ACD: Analog Comparator Disable**

When this bit is written logic one, the power to the Analog Comparator is switched off. This bit can be set at any time to turn off the Analog Comparator. This will reduce power consumption in Active and Idle mode. When changing the ACD bit, the Analog Comparator Interrupt must be disabled by clearing the ACIE bit in ACSR. Otherwise an interrupt can occur when the bit is changed.

- **Bit 6 – ACBG: Analog Comparator Bandgap Select**

When this bit is set, a fixed bandgap reference voltage replaces the positive input to the Analog Comparator. When this bit is cleared, AIN0 is applied to the positive input of the Analog Comparator. See "Internal Voltage Reference" on page 54.

- **Bit 5 – ACO: Analog Comparator Output**

The output of the Analog Comparator is synchronized and then directly connected to ACO. The synchronization introduces a delay of 1 - 2 clock cycles.

- **Bit 4 – ACI: Analog Comparator Interrupt Flag**

This bit is set by hardware when a comparator output event triggers the interrupt mode defined by ACIS1 and ACIS0. The Analog Comparator interrupt routine is executed if the ACIE bit is set and the I-bit in SREG is set. ACI is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ACI is cleared by writing a logic one to the flag.

- **Bit 3 – ACIE: Analog Comparator Interrupt Enable**

When the ACIE bit is written logic one and the I-bit in the Status Register is set, the Analog Comparator interrupt is activated. When written logic zero, the interrupt is disabled.

- **Bit 2 – ACIC: Analog Comparator Input Capture Enable**

When written logic one, this bit enables the input capture function in Timer/Counter1 to be triggered by the Analog Comparator. The comparator output is in this case directly connected to the input capture front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/Counter1 Input Capture interrupt. When written logic zero, no connection between the Analog Comparator and the input capture function exists. To make the comparator trigger the Timer/Counter1 Input Capture interrupt, the ICIE1 bit in the Timer Interrupt Mask Register (TIMSK1) must be set.

- **Bits 1, 0 – ACIS1, ACIS0: Analog Comparator Interrupt Mode Select**

These bits determine which comparator events that trigger the Analog Comparator interrupt. The different settings are shown in Table 23-1.

Table 23-1. ACIS1/ACIS0 Settings

ACIS1	ACIS0	Interrupt Mode
0	0	Comparator Interrupt on Output Toggle.
0	1	Reserved
1	0	Comparator Interrupt on Falling Output Edge.
1	1	Comparator Interrupt on Rising Output Edge.

When changing the ACIS1/ACIS0 bits, the Analog Comparator Interrupt must be disabled by clearing its Interrupt Enable bit in the ACSR Register. Otherwise an interrupt can occur when the bits are changed.

Analog Comparator Multiplexed Input

It is possible to select any of the ADC13..0 pins to replace the negative input to the Analog Comparator. The ADC multiplexer is used to select this input, and consequently, the ADC must be switched off to utilize this feature. If the Analog Comparator Multiplexer Enable bit (ACME in ADCSRB) is set and the ADC is switched off (ADEN in ADCSRA is zero), and MUX2..0 in ADMUX select the input pin to replace the negative input to the Analog Comparator, as shown in Table 23-2. If ACME is cleared or ADEN is set, the Bandgap reference is applied to the negative input to the Analog Comparator.

Table 23-2. Analog Comparator Multiplexed Input

ACME	ADEN	MUX2..0	Analog Comparator Negative Input
0	x	xxx	Bandgap Ref.
1	1	xxx	Bandgap Ref.
1	0	000	ADC0
1	0	001	ADC1
1	0	010	N/A
1	0	011	
1	0	100	ADC4
1	0	101	ADC5
1	0	110	ADC6
1	0	111	ADC7

Digital Input Disable Register 1 – DIDR1

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	–	–	AIN0D	DIDR1
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 0 – AIN0D: AIN0 Digital Input Disable**

When this bit is written logic one, the digital input buffer on the AIN0 pin is disabled. The corresponding PIN Register bit will always read as zero when this bit is set. When an analog signal is applied to the AIN0 pin and the digital input from this pin is not needed, this bit should be written logic one to reduce power consumption in the digital input buffer.

```
////////////////////////////////////////////////////////////////////////////
//
// Atmega32u4      Analog Comparator
//
// AIN+ (AIN0, PE6)
// AIN- (ADC4, PF4)
//
// AIN+ > AIN-     LED gegen Vcc am PB0 ein
// AIN+ < AIN-     LED gegen Vcc am PB0 aus
// d.h. Comparator toggle Interrupt => ACIS1=0 und ACIS0=0 im ACSR Register
//                                                        (default Werte)
//
////////////////////////////////////////////////////////////////////////////

#include <avr/io.h>
#include <avr/interrupt.h>

int main(void)
{
    DDRB = DDRB | (1<<DDB0);          //PB0 OUTPUT (LED)
      DDRE = DDRE &~(1<<DDE6);        //PE6 INPUT        (AIN+)
      DDRF = DDRF &~(1<<DDF4);        //PF4 INPUT        (AIN-)

      ADCSRA = ADCSRA &~(1<<ADEN);    //ADC aus
      ADCSRB = ADCSRB | (1<<ACME);    //AIN- hängt am ADC Multiplexer
      ADMUX  = ADMUX  | (1<<MUX2);    //ADC4 als AIN- Input gewählt

      DIDR1 = DIDR1 | (1<<AIN0D);     //dig. Input Buffer am AIN+ (PE6) disable


      ACSR = ACSR | (1<<ACIE);        //Analog Comparator Interrupt enable
      sei();                          //Interrupts global freigeben

      while(1);
}

ISR (ANALOG_COMP_vect)
{
      if (ACSR & (1<<ACO))            //Comparator Ausgang AC0=1 (AIN+ > AIN-)
      {
            PORTB = PORTB &~(1<<PORTB0);   //LED ein    (AIN+ > AIN-)
      }
      else
      {
            PORTB = PORTB | (1<<PORTB0);   //LED aus    (AIN+ < AIN-)
      }
}
```

SLEEPING MODES UND POWER MANAGEMENT

Register: SMCR, PRR0, PRR1

Table 7-2. Active Clock Domains and Wake-up Sources in the Different Sleep Modes.

	Active Clock Domains				Oscillators	Wake-up Sources							
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	Main Clock Source Enabled	INT6, INT3:0 and Pin Change	TWI Address Match	SPM/ EEPROM Ready	ADC	WDT Interrupt	Other I/O	USB Synchronous Interrupts	USB Asynchronous Interrupts(3)
Idle			X	X	X	X	X	X	X	X	X	X	X
ADCNRM				X	X	X(2)	X	X	X	X		X	X
Power-down						X(2)	X			X			X
Power-save						X(2)	X			X			X
Standby(1)					X	X(2)	X			X			X
Extended Standby					X	X(2)	X			X			X

Notes: 1. Only recommended with external crystal or resonator selected as clock source.
2. For INT6, only level interrupt.
3. Asynchronous USB interrupts are VBUSTI and WAKEUPI.

Figure 30-4. Active Supply Current vs. Frequency (1-16 MHz) and T = 25°C

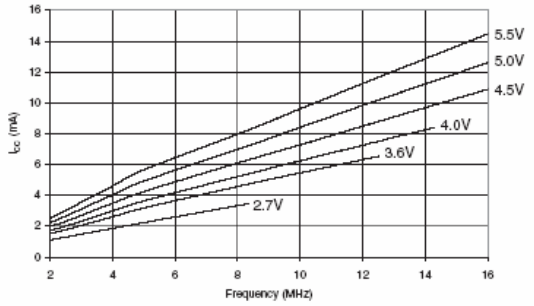


Figure 30-8. Idle Supply Current vs. Frequency (1-16 MHz) T = 25°C

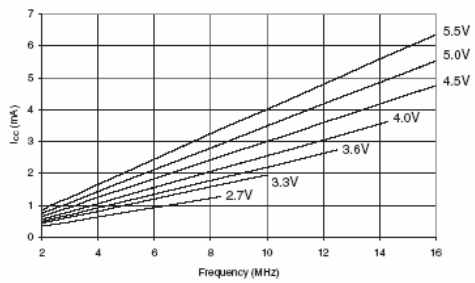


Figure 30-10. Power-Down Supply Current vs. V_{CC} (WDT Disabled)

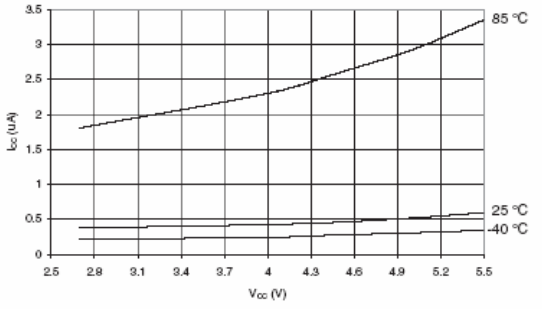
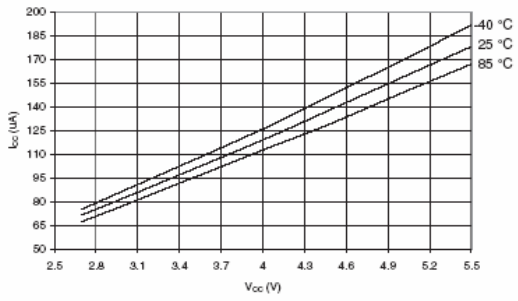


Figure 30-13. Power-Save Supply Current vs. V_{CC} (WDT Disabled)



Sleep Mode Control Register – SMCR

The Sleep Mode Control Register contains control bits for power management.

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	SM2	SM1	SM0	SE	SMCR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 3, 2, 1 – SM2..0: Sleep Mode Select Bits 2, 1, and 0**

These bits select between the six available sleep modes as shown in Table 7-1.

Table 7-1. Sleep Mode Select

SM2	SM1	SM0	Sleep Mode
0	0	0	Idle
0	0	1	ADC Noise Reduction
0	1	0	Power-down
0	1	1	Power-save
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Standby(1)
1	1	1	Extended Standby(1)

Note: 1. Standby modes are only recommended for use with external crystals or resonators.

- **Bit 1 – SE: Sleep Enable**

The SE bit must be written to logic one to make the MCU enter the sleep mode when the SLEEP instruction is executed. To avoid the MCU entering the sleep mode unless it is the programmer's purpose, it is recommended to write the Sleep Enable (SE) bit to one just before the execution of the SLEEP instruction and to clear it immediately after waking up.

SLEEP ist ein Assemblerbefehl der in C durch Makros aus sleep.h abgedeckt wird.

```
#include <avr/sleep.h>
```

Ergebnis	Funktion	Anwendung
void	`set_sleep_mode(mode)`	bereitet Sleep Betrieb im SMCR durch SM2 SM1 SM0 vor mode 0 0 0: SLEEP_MODE_IDLE mode 0 0 1: SLEEP_MODE_ADC mode 0 1 0: SLEEP_MODE_PWR_DOWN mode 0 1 1: SLEEP_MODE_PWR_SAVE mode 1 1 0: SLEEP_MODE_STANDBY mode 1 1 1: SLEEP_MODE_EXT_STANDBY
void	`sleep_mode(void)`	versetzte den μC in einen der gewählten Sleep Modes
void	`sleep_enable(void)`	setzt SE-Bit im SMCR
void	`sleep_disable(void)`	löscht SE-Bit im SMCR
void	`sleep_cpu(void)`	führt SLEEP Assemblerbefehl aus
void	`speep_bod_disable(void)`	schaltet die Braun Out Detection aus

Bem.: `sleep_mode();`
fasst folgende Befehlsabfolge zusammen:

```
sleep_enable();
sleep_cpu();
sleep_disable();
```

```
////////////////////////////////////////////////////////////////////////
//
//ATmega32u4 im Idle Sleep Mode
//
//Unterbrechung: ext. INT1 (PD1) erhöht Port B um 1
//
////////////////////////////////////////////////////////////////////////

#include <avr/io.h>           //SFR's, Deklerationen
#include <avr/sleep.h>        //Sleep Funktionen
#include <avr/interrupt.h>    //Interruptroutinen

ISR(INT1_vect)
{
      PORTB++;
}

void main(void)
{
      DDRD  = DDRD &~(1<<DDD1)            //INT1 (PD1) INPUT
      DDRB  = 0xFF;                       //Port B OUTPUT
      PORTB = 0x00;                       //Startwert 0

      EICRA |= (1<<ISC11);                //INT1 fallende Flanke
      EIMSK |= (1<<INT1);                 //INT1 freigeben
      set_sleep_mode(SLEEP_MODE_IDLE);    //Idle Mode
      sei();                              //global Interrupt enable

      while(1)
      {
            sleep_mode();                 //µC schläft
      }
}
```

Um Strom zu sparen, können im normalen Betrieb (Active Mode) und im Idle Mode die Clocks für einzelne Peripherieeinheiten mit Hilfe der Power Reduction Register (PRR0, PRR1) angehalten werden. In allen anderen Speep Modes passiert das automatisch.

Power Reduction Register 0 - PRR0

Bit	7	6	5	4	3	2	1	0	
	PRTWI	PRTIM2	PRTIM0	–	PRTIM1	PRSPI	-	PRADC	PRR0
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 - PRTWI: Power Reduction TWI**

Writing a logic one to this bit shuts down the TWI by stopping the clock to the module. When waking up the TWI again, the TWI should be re initialized to ensure proper operation.

- **Bit 6 - Res: Reserved bit**

This bits is reserved and will always read as zero.

- **Bit 5 - PRTIM0: Power Reduction Timer/Counter0**

Writing a logic one to this bit shuts down the Timer/Counter0 module. When the Timer/Counter0 is enabled, operation will continue like before the shutdown.

- **Bit 4 - Res: Reserved bit**

This bit is reserved and will always read as zero.

- **Bit 3 - PRTIM1: Power Reduction Timer/Counter1**

Writing a logic one to this bit shuts down the Timer/Counter1 module. When the Timer/Counter1 is enabled, operation will continue like before the shutdown.

- **Bit 2 - PRSPI: Power Reduction Serial Peripheral Interface**

Writing a logic one to this bit shuts down the Serial Peripheral Interface by stopping the clock to the module. When waking up the SPI again, the SPI should be re initialized to ensure proper operation.

- **Bit 1 - Res: Reserved bit**

These bits are reserved and will always read as zero.

- **Bit 0 - PRADC: Power Reduction ADC**

Writing a logic one to this bit shuts down the ADC. The ADC must be disabled before shut down. The analog comparator cannot use the ADC input MUX when the ADC is shut down.

Power Reduction Register 1 - PRR1

Bit	7	6	5	4	3	2	1	0	
	PRUSB	–	–	PRTIM4	PRTIM3	–	–	PRUSART1	PRR1
Read/Write	R/W	R	R	R	R/W	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 - PRUSB: Power Reduction USB**

Writing a logic one to this bit shuts down the USB by stopping the clock to the module. When waking up the USB again, the USB should be re initialized to ensure proper operation.

- **Bit 6..5 - Res: Reserved bits**

These bits are reserved and will always read as zero.

- **Bit 4- PRTIM4: Power Reduction Timer/Counter4**

Writing a logic one to this bit shuts down the Timer/Counter4 module. When the Timer/Counter4 is enabled, operation will continue like before the shutdown.

- **Bit 3 - PRTIM3: Power Reduction Timer/Counter3**

Writing a logic one to this bit shuts down the Timer/Counter3 module. When the Timer/Counter3 is enabled, operation will continue like before the shutdown.

- **Bit 2..1 - Res: Reserved bits**

These bits are reserved and will always read as zero.

- **Bit 0 - PRUSART1: Power Reduction USART1**

Writing a logic one to this bit shuts down the USART1 by stopping the clock to the module. When waking up the USART1 again, the USART1 should be re initialized to ensure proper operation.

UART / USART

- **Full Duplex Operation (Independent Serial Receive and Transmit Registers)**
- **Asynchronous or Synchronous Operation**
- **Flow control CTS/RTS signals hardware management**
- **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**

Register: UDR1, UBRR1 (UBRR1L, UBRR1H), UCSR1A, UCSR1B, UCSR1C

Pins: RxD1 (PD2) $\overline{CTS}$ (PD5)
TxD1 (PD3) $\overline{RTS}$ (PB7)
XCK1 (PD5)

The hardware flow control can be enabled by software.
$\overline{CTS}$: (Clear to Send)
$\overline{RTS}$: (Request to Send)

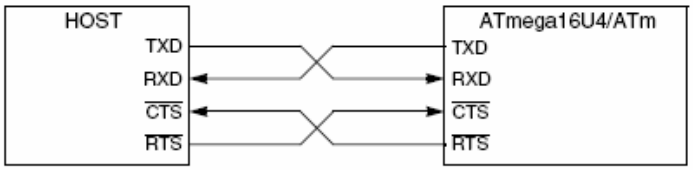
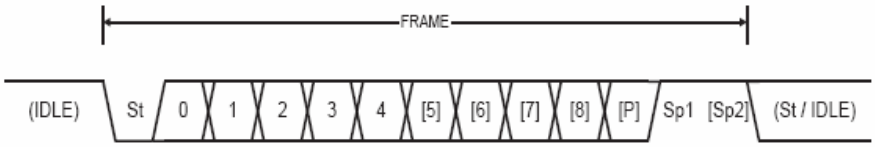


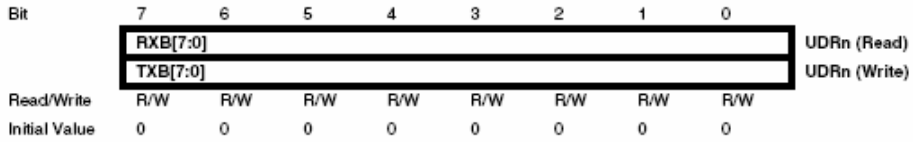
Figure 18-4. Frame Formats



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.

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

USART I/O Data Register n– UDRn



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 UDRn 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 UDREn Flag in the UCSRnA Register is set. Data written to UDRn when the UDREn Flag is not set, will be ignored by the USART Transmitter. 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.

USART Control and Status Register A – UCSRnA

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

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

- **Bit 6 – TXCn: 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 TXCn 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 TXCn Flag can generate a Transmit Complete interrupt (see description of the TXCIEn bit).

- **Bit 5 – UDREn: USART Data Register Empty**

The UDREn Flag indicates if the transmit buffer (UDRn) is ready to receive new data. If UDREn is one, the buffer is empty, and therefore ready to be written. The UDREn Flag can generate a Data Register Empty interrupt (see description of the UDRIEn bit).

UDREn is set after a reset to indicate that the Transmitter is ready.

- **Bit 4 – FEn: 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.

- **Bit 3 – DORn: 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.

- **Bit 2 – UPEn: 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 (UPMn1 = 1). This bit is valid until the receive buffer (UDRn) is read. Always set this bit to zero when writing to UCSRnA.

- **Bit 1 – U2Xn: 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.

- **Bit 0 – MPCMn: Multi-processor Communication Mode**

This bit enables the Multi-processor Communication mode. When the MPCMn bit is written to one, all the incoming frames received by the USART Receiver that do not contain address information will be ignored. The Transmitter is unaffected by the MPCMn setting. For more detailed information see "Multi-processor Communication Mode" on page 202.

USART Control and Status Register n B – UCSRnB

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – RXCIEn: RX Complete Interrupt Enable n**

Writing this bit to one enables interrupt on the RXCn Flag. A USART Receive Complete interrupt will be generated only if the RXCIEn bit is written to one, the Global Interrupt Flag in SREG is written to one and the RXCn bit in UCSRnA is set.

- **Bit 6 – TXCIEn: TX Complete Interrupt Enable n**

Writing this bit to one enables interrupt on the TXCn Flag. A USART Transmit Complete interrupt will be generated only if the TXCIEn bit is written to one, the Global Interrupt Flag in SREG is written to one and the TXCn bit in UCSRnA is set.

- **Bit 5 – UDRIEn: USART Data Register Empty Interrupt Enable n**

Writing this bit to one enables interrupt on the UDREn Flag. A Data Register Empty interrupt will be generated only if the UDRIEn bit is written to one, the Global Interrupt Flag in SREG is written to one and the UDREn bit in UCSRnA is set.

- **Bit 4 – RXENn: Receiver Enable n**

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 FEn, DORn, and UPEn Flags.

- **Bit 3 – TXENn: Transmitter Enable n**

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 TXENn 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 – UCSZn2: Character Size n**

The UCSZn2 bits combined with the UCSZn1:0 bit in UCSRnC sets the number of data bits (Character SiZe) in a frame the Receiver and Transmitter use.

- **Bit 1 – RXB8n: Receive Data Bit 8 n**

RXB8n 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.

- **Bit 0 – TXB8n: Transmit Data Bit 8 n**

TXB8n 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.

USART Control and Status Register n C – UCSRnC

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

- **Bits 7:6 – UMSELn1:0 USART Mode Select**

These bits select the mode of operation of the USARTn as shown in Table 18-4.

Table 18-4. UMSELn Bit Settings

UMSELn1	UMSELn0	Mode
0	0	Asynchronous USART
0	1	Synchronous USART
1	0	(Reserved)
1	1	Master SPI (MSPIM)(1)

Note: 1. See "USART in SPI Mode" on page 214 for full description of the Master SPI Mode (MSPIM) operation

- **Bits 5:4 – UPMn1:0: 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 UPMn setting. If a mismatch is detected, the UPEn Flag in UCSRnA will be set.

Table 18-5. UPMn Bit Settings

UPMn1	UPMn0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

- **Bit 3 – USBSn: Stop Bit Select**

This bit selects the number of stop bits to be inserted by the Transmitter. The Receiver ignores this setting.

Table 18-6. USBS Bit Settings

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

- **Bit 2:1 – UCSZn1:0: Character Size**

The UCSZn1:0 bits combined with the UCSZn2 bit in UCSRnB sets the number of data bits (Character SiZe) in a frame the Receiver and Transmitter use.

Table 18-7. UCSZn Bit Settings

UCSZn2	UCSZn1	UCSZn0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

- **Bit 0 – UCPOLn: Clock Polarity**

This bit is used for synchronous mode only. Write this bit to zero when asynchronous mode is used. The UCPOLn bit sets the relationship between data output change and data input sample, and the synchronous clock (XCKn).

Table 18-8. UCPOLn Bit Settings

UCPOLn	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

USART Control and Status Register n D– UCSRnD

Bit	7	6	5	4	3	2	1	0	
	–	–	–	–	–	–	CTSEN	RTSEN	UCSRnD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7:2 – Reserved bits**

These bits are reserved and will be read as '0'. Do not set these bits.

- **Bits 1 – CTSEN: UART $\overline{\text{CTS}}$ Signal Enable**

Set this bit by firmware to enable the transmission flow control signal ($\overline{\text{CTS}}$). Transmission will be enabled only if $\overline{\text{CTS}}$ input = 0. Clear this bit to disable the transmission flow control signal. Transmission will occur without hardware condition. Data Direction Register bit must be correctly clear to enable the pin as an input.

- **Bits 0 – RTSEN: UART $\overline{\text{RTS}}$ Signal Enable**

Set this bit by firmware to enable the reception flow control signal (RTS). In this case the $\overline{\text{RTS}}$ line will automatically rise when the FIFO is full. Clear this bit to disable the reception flow control signal. Data Direction Register bit must be correctly set to enable the pin as an output.

USART Baud Rate Registers – UBRRLn and UBRRHn

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRR[11:8]				UBRRHn
	UBRR[7:0]								UBRRLn
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

- **Bit 15:12 – Reserved Bits**

These bits are reserved for future use. For compatibility with future devices, these bit must be written to zero when UBRRH is written.

- **Bit 11:0 – UBRR11:0: USART Baud Rate Register**

This is a 12-bit register which contains the USART baud rate. The UBRRH contains the four most significant bits, and the UBRRL contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the Transmitter and Receiver will be corrupted if the baud rate is changed. Writing UBRRL will trigger an immediate update of the baud rate prescaler.

Table 18-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal mode (U2Xn = 0)	$BAUD = \frac{f_{OSC}}{16(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed mode (U2Xn = 1)	$BAUD = \frac{f_{OSC}}{8(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master mode	$BAUD = \frac{f_{OSC}}{2(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{2BAUD} - 1$

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

BAUD	Baud rate (in bits per second, bps)
f_{osc}	System Oscillator clock frequency
UBRRn	Contents of the UBRRHn and UBRRLn Registers, (0-4095)

C Code Example[1]

```
void USART_Init( unsigned int baud )
{
  /* Set baud rate */
  UBRRHn = (unsigned char)(baud>>8);
  UBRRLn = (unsigned char)baud;
  /* Enable receiver and transmitter */
  UCSRnB = (1<<RXENn)|(1<<TXENn);
  /* Set frame format: 8data, 2stop bit */
  UCSRnC = (1<<USBSn)|(3<<UCSZn0);
}
```

```
////////////////////////////////////////////////////////////////////////////
//
// Atmega 32u4        UART einzelnes Zeichen schaltet LED ein/aus
//
// Taste 'e' schaltet LED gegen Vcc an PB0 ein
// Taste 'a' schaltet LED gegen Vcc an PB0 aus
//
// f_CPU = 16 MHz
// 9600 Baud, 8 Datenbits, kein Parity, 1 Stopbit  (9600 8-N-1)
// Asynchronous Normal Mode
//
////////////////////////////////////////////////////////////////////////////

#define F_CPU         16000000UL                    //Takt
#define BAUD          9600UL                        //gewünschte Baudrate
#define UBRR_CALC     (F_CPU/16UL/BAUD-1)           //Baudrtate aus Takt berechnen

#include <avr/io.h>
#include <avr/interrupt.h>

char out_text[]="\n\rTaste 'e' ==> LED ein\n\rTaste 'a' ==> LED aus\n\rBitte um Eingabe: \0";
                                              //\n New Line, \r Carrige Return, \0 String Ende
char i=0;                                     //Laufvariable für out_text[i]
char in;                                      //Variable für empf. Zeichen

void init_usart (void)
{
       UBRR1H = (unsigned char)(UBRR_CALC>>8);      //Baudrate einstellen
       UBRR1L = (unsigned char)(UBRR_CALC);

       UCSR1B |= (1<<RXEN1)|(1<<RXCIE1)|(1<<TXEN1);//Empf. ein, Empf.-Interr. ein, Sender ein
       UCSR1C |= (1<<UCSZ11)|(1<<UCSZ10);          //asynchr., 8 Daten-, kein Parity-, 1 Stop
}

ISR(USART1_RX_vect)                           //Interrupt für Empfang
{
       while(!(UCSR1A & (1<<RXC1)));          //warten bis Zeichen fertig empfangen
       in = UDR1;                             //Zeichen in Variable ablegen

       if (in == 'e') {PORTB &=~(1<<PB0);}    //LED PB0 ein
       if (in == 'a') {PORTB |= (1<<PB0);}    //LED PB0 aus
}

int main(void)
{
       DDRB  |= (1<<DDB0);                    //PB0 OUTPUT
       PORTB |= (1<<PB0);                     //LED aus

       CLKPR = 0x80;
       CLKPR = 0x00;                          //CLK-Prescaler 1

       init_usart();                          //USART initialisieren
       sei();                                 //Interrupts global freigeben

       while(out_text[i]!='\0')               //Infotext ausgeben
       {
              while(!(UCSR1A & (1<<UDRE1)));  //warten bis Zeichen fertig gesendet wurde
              UDR1 = out_text[i];             //Zeichen senden
              i++;
       }

       while(1);                              //Endlosschleife wartet auf Zeicheneingabe
}
```

I2C

Features

- Simple Yet Powerful and Flexible Communication Interface, only two Bus Lines Needed
- Both Master and Slave Operation Supported
- Device can Operate as Transmitter or Receiver
- 7-bit Address Space Allows up to 128 Different Slave Addresses
- Multi-master Arbitration Support
- Up to 400 kHz Data Transfer Speed
- Slew-rate Limited Output Drivers
- Noise Suppression Circuitry Rejects Spikes on Bus Lines
- Fully Programmable Slave Address with General Call Support
- Address Recognition Causes Wake-up When AVR is in Sleep Mode

Register: TWBR, TWCR, TWSR, TWDR, TWAR, TWAMR

Pins: SCL (PD0)
SDA (PD1)

Figure 20-1. TWI Bus Interconnection

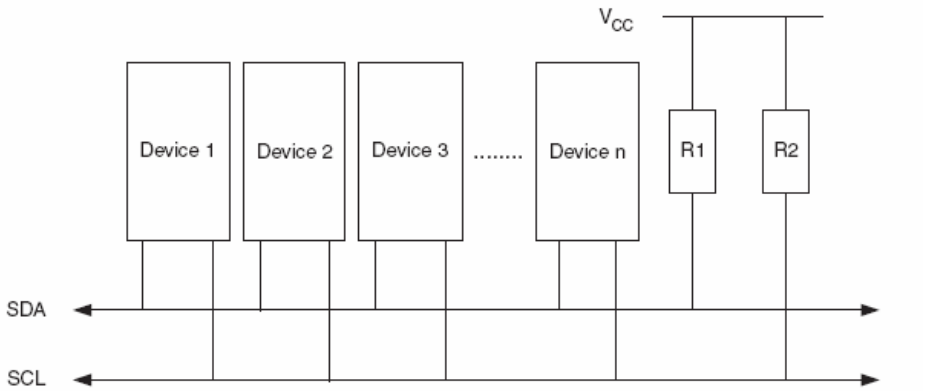


Figure 20-6. Typical Data Transmission

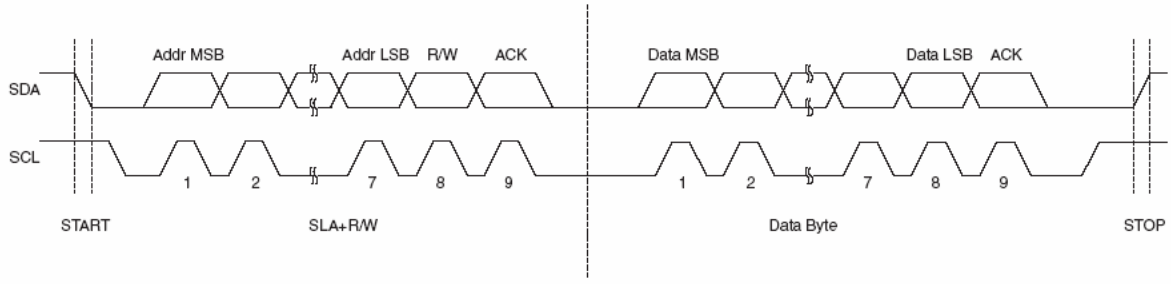
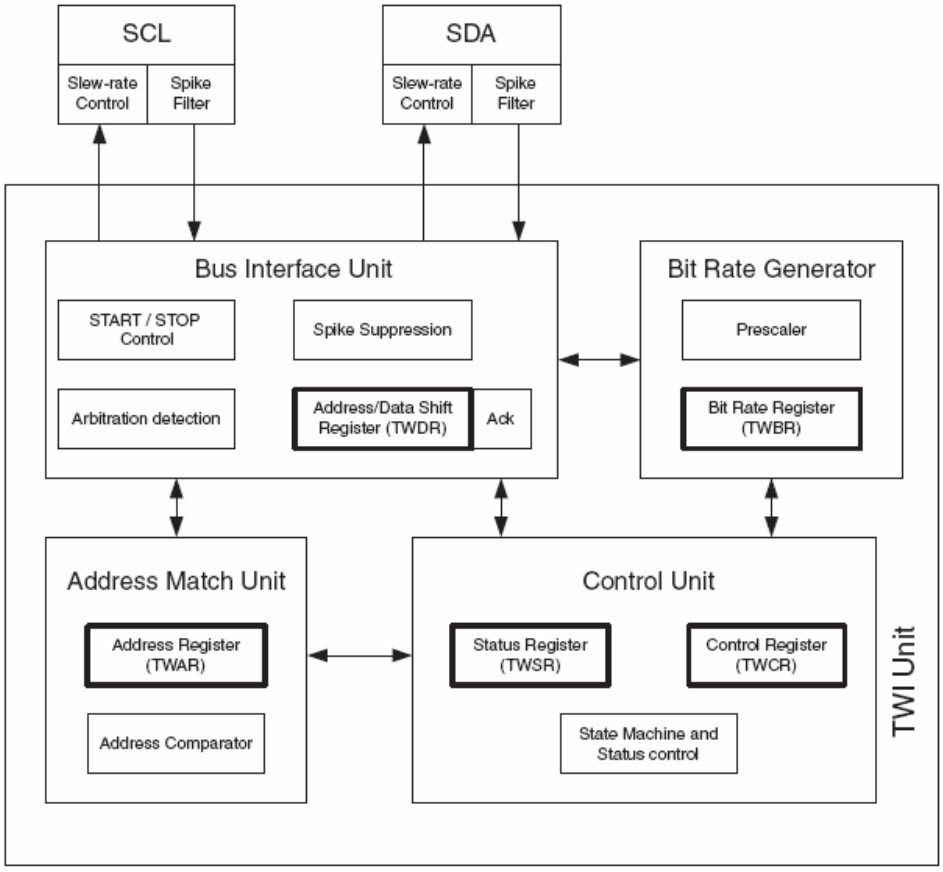


Figure 20-9. Overview of the TWI Module



$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \cdot 4^{TWPS}}$$

- TWBR = Value of the TWI Bit Rate Register.
- TWPS = Value of the prescaler bits in the TWI Status Register.

Note: TWBR should be 10 or higher if the TWI operates in Master mode. If TWBR is lower than 10, the Master may produce an incorrect output on SDA and SCL for the reminder of the byte. The problem occurs when operating the TWI in Master mode, sending Start + SLA + R/W to a Slave (a Slave does not need to be connected to the bus for the condition to happen).

Es muss mindestens sein: CPU Clock frequency ≥ 16 * SCL frequency

TWI Bit Rate Register – TWBR

Bit	7	6	5	4	3	2	1	0	
	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..0 – TWI Bit Rate Register**

TWBR selects the division factor for the bit rate generator. The bit rate generator is a frequency divider which generates the SCL clock frequency in the Master modes. See "Bit Rate Generator Unit" on page 229 for calculating bit rates.

TWI Status Register – TWSR

Bit	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	TWSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	1	1	1	1	1	0	0	0	

- **Bits 7..3 – TWS: TWI Status**

These 5 bits reflect the status of the TWI logic and the 2-wire Serial Bus. The different status codes are described later in this section. Note that the value read from TWSR contains both the 5-bit status value and the 2-bit prescaler value. The application designer should mask the prescaler bits to zero when checking the Status bits. This makes status checking independent of prescaler setting. This approach is used in this datasheet, unless otherwise noted.

- **Bit 2 – Res: Reserved Bit**

This bit is reserved and will always read as zero.

- **Bits 1..0 – TWPS: TWI Prescaler Bits**

These bits can be read and written, and control the bit rate prescaler.

Table 20-2. TWI Bit Rate Prescaler

TWPS1	TWPS0	Prescaler Value
0	0	1
0	1	4
1	0	16
1	1	64

To calculate bit rates, see "Bit Rate Generator Unit" on page 229. The value of TWPS1..0 is used in the equation.

TWI Control Register – TWCR

Bit	7	6	5	4	3	2	1	0	
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The TWCR is used to control the operation of the TWI. It is used to enable the TWI, to initiate a Master access by applying a START condition to the bus, to generate a Receiver acknowledge, to generate a stop condition, and to control halting of the bus while the data to be written to the bus are written to the TWDR. It also indicates a write collision if data is attempted written to TWDR while the register is inaccessible.

- **Bit 7 – TWINT: TWI Interrupt Flag**

This bit is set by hardware when the TWI has finished its current job and expects application software response. If the I-bit in SREG and TWIE in TWCR are set, the MCU will jump to the TWI Interrupt Vector. While the TWINT Flag is set, the SCL low period is stretched. The TWINT Flag must be cleared by software by writing a logic one to it. Note that this flag is not automatically cleared by hardware when executing the interrupt routine. Also note that clearing this flag starts the operation of the TWI, so all accesses to the TWI Address Register (TWAR), TWI Status Register (TWSR), and TWI Data Register (TWDR) must be complete before clearing this flag.

- **Bit 6 – TWEA: TWI Enable Acknowledge Bit**

The TWEA bit controls the generation of the acknowledge pulse. If the TWEA bit is written to one, the ACK pulse is generated on the TWI bus if the following conditions are met:

1. The device's own slave address has been received.
2. A general call has been received, while the TWGCE bit in the TWAR is set.
3. A data byte has been received in Master Receiver or Slave Receiver mode.

By writing the TWEA bit to zero, the device can be virtually disconnected from the 2-wire Serial Bus temporarily. Address recognition can then be resumed by writing the TWEA bit to one again.

- **Bit 5 – TWSTA: TWI START Condition Bit**

The application writes the TWSTA bit to one when it desires to become a Master on the 2-wire Serial Bus. The TWI hardware checks if the bus is available, and generates a START condition on the bus if it is free. However, if the bus is not free, the TWI waits until a STOP condition is detected, and then generates a new START condition to claim the bus Master status. TWSTA must be cleared by software when the START condition has been transmitted.

- **Bit 4 – TWSTO: TWI STOP Condition Bit**

Writing the TWSTO bit to one in Master mode will generate a STOP condition on the 2-wire Serial Bus. When the STOP condition is executed on the bus, the TWSTO bit is cleared automatically. In Slave mode, setting the TWSTO bit can be used to recover from an error condition. This will not generate a STOP condition, but the TWI returns to a well-defined unaddressed Slave mode and releases the SCL and SDA lines to a high impedance state.

- **Bit 3 – TWWC: TWI Write Collision Flag**

The TWWC bit is set when attempting to write to the TWI Data Register – TWDR when TWINT is low. This flag is cleared by writing the TWDR Register when TWINT is high.

- **Bit 2 – TWEN: TWI Enable Bit**

The TWEN bit enables TWI operation and activates the TWI interface. When TWEN is written to one, the TWI takes control over the I/O pins connected to the SCL and SDA pins, enabling the slew-rate limiters and spike filters. If this bit is written to zero, the TWI is switched off and all TWI transmissions are terminated, regardless of any ongoing operation.

- **Bit 1 – Res: Reserved Bit**

This bit is a reserved bit and will always read as zero.

- **Bit 0 – TWIE: TWI Interrupt Enable**

When this bit is written to one, and the I-bit in SREG is set, the TWI interrupt request will be activated for as long as the TWINT Flag is high.

TWI Data Register – TWDR

Bit	7	6	5	4	3	2	1	0	
	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	

In Transmit mode, TWDR contains the next byte to be transmitted. In Receive mode, the TWDR contains the last byte received. It is writable while the TWI is not in the process of shifting a byte. This occurs when the TWI Interrupt Flag (TWINT) is set by hardware. Note that the Data Register cannot be initialized by the user before the first interrupt occurs. The data in TWDR remains stable as long as TWINT is set. While data is shifted out, data on the bus is simultaneously shifted in. TWDR always contains the last byte present on the bus, except after a wake up from a sleep mode by the TWI interrupt. In this case, the contents of TWDR is undefined. In the case of a lost bus arbitration, no data is lost in the transition from Master to Slave. Handling of the ACK bit is controlled automatically by the TWI logic, the CPU cannot access the ACK bit directly.

- **Bits 7..0 – TWD: TWI Data Register**

These eight bits constitute the next data byte to be transmitted, or the latest data byte received on the 2-wire Serial Bus.

TWI (Slave) Address Register – TWAR

Bit	7	6	5	4	3	2	1	0	
	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWAR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	0	

The TWAR should be loaded with the 7-bit Slave address (in the seven most significant bits of TWAR) to which the TWI will respond when programmed as a Slave Transmitter or Receiver, and not needed in the Master modes. In multi master systems, TWAR must be set in masters which can be addressed as Slaves by other Masters.

The LSB of TWAR is used to enable recognition of the general call address (0x00). There is an associated address comparator that looks for the slave address (or general call address if enabled) in the received serial address. If a match is found, an interrupt request is generated.

- **Bits 7..1 – TWA: TWI (Slave) Address Register**

These seven bits constitute the slave address of the TWI unit.

- **Bit 0 – TWGCE: TWI General Call Recognition Enable Bit**

If set, this bit enables the recognition of a General Call given over the 2-wire Serial Bus.

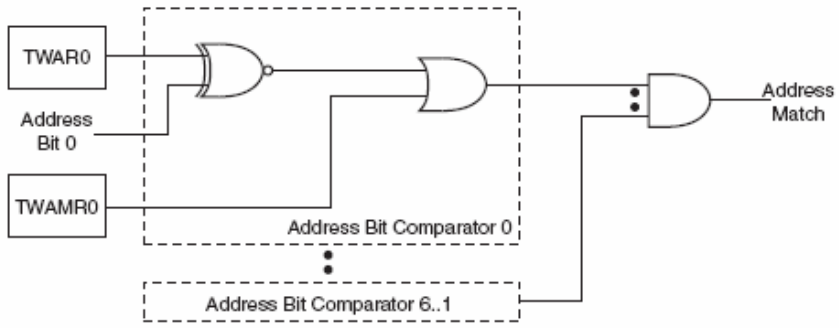
TWI (Slave) Address Mask Register – TWAMR

Bit	7	6	5	4	3	2	1	0	
	TWAM[6:0]							–	TWAMR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	
Initial Value	0	0	0	0	0	0	0	0	

- **Bits 7..1 – TWAM: TWI Address Mask**

The TWAMR can be loaded with a 7-bit Slave Address mask. Each of the bits in TWAMR can mask (disable) the corresponding address bit in the TWI Address Register (TWAR). If the mask bit is set to one then the address match logic ignores the compare between the incoming address bit and the corresponding bit in TWAR. Figure 20-10 shows the address match logic in detail.

Figure 20-10. TWI Address Match Logic, Block Diagram



- **Bit 0 – Res: Reserved Bit**

This bit is reserved and will always read as zero.

```
////////////////////////////////////////////////////////////////////////////////////////
//
// 32U4 + I2C ADC/DAC PCF8591                  f_CPU = 16 MHz
//                                             f_SCL = 100 kHz
// DAC gibt Sägezahn am Aout aus
// ADC wird nicht verwendet
//
//               f_CPU
// f_SCL = ---------------------  => TWBR=18 mit TWPS=1
//         16 + 2*TWBR * 4^TWPS
//
// Bem.: f_CPU muss min. 16*f_SCL sein
//       TWBR muss Wert > 10 haben
//
// PCF8591 Adr.: 1001 000  (A2=A1=A0=0)
//
////////////////////////////////////////////////////////////////////////////////////////

#include <avr/io.h>

int main(void)
{
        unsigned char i;

        CLKPR = 0x80;                                   //Interner CLOCK-Prescaler auf 1
        CLKPR = 0x00;                                   //damit f_CPU = 16 MHz

        TWBR = 18;                                      //TWBR=18, TWPS=1 im Reg. TWSR per default
                                                        //damit f_SCL = 100 kHz
        while(1)
        {
                TWCR = TWCR|(1<<TWINT)|(1<<TWSTA)|(1<<TWEN); //START
                while(!(TWCR&(1<<TWINT)));              //warten bis fertig

                TWDR = 0b10010000;                      //Adr. 1001 000W + Write (W=0)
                TWCR = (1<<TWINT)|(1<<TWEN);            //senden
                while(!(TWCR&(1<<TWINT)));              //warten bis fertig

                TWDR = 0x40;                            //2. Byte = Control Byte DAC-Mode
                TWCR = (1<<TWINT)|(1<<TWEN);            //Control Byte senden
                while(!(TWCR&(1<<TWINT)));              //warten bis fertig

                for(i=0;i<255;i++)
                {
                        TWDR = i;                       //Daten Byte
                        TWCR = (1<<TWINT)|(1<<TWEN);    //Daten Byte senden
                        while(!(TWCR&(1<<TWINT)));      //warten bis fertig
                }

                TWCR = TWCR|(1<<TWINT)|(1<<TWSTO)|(1<<TWEN); //STOP
        }

}
```

USB

Einfache, sofort lauffähige, leicht selbst modifizierbare Bsp. für Teensy 2.0 Board mit HID-Interface (kein Treiber nötig). Nur Interrupt Transfer.
http://www.pjrc.com/teensy/usb_debug_only.html

Oder ebenfalls einfaches Bsp. für Teensy 20. Board mit libusb-Treiber und/oder hidapi. Nutzt auch den Bulk Transfer. Inkl. Bsp. für PC-Seite für Qt-Creator:
http://www.weigu.lu/b/usb/

Auch für Win7 64-Bit und Win8, für alle 8-Bit USB-fähigen AVRs sowie für alle Transferarten mit LUFA. Inkl. sehr vielen Beispielen:
http://www.fourwalledcubicle.com/LUFA.php

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 120 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Througput at 20 MHz
- High Endurance Non-volatile Memory segments
  - 1K Bytes of In-System Self-programmable Flash program memory
  - 64 Bytes EEPROM
  - 64 Bytes Internal SRAM
  - Write/Erase cyles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C (see page 6)
  - Programming Lock for Self-Programming Flash & EEPROM Data Security
- Peripheral Features
  - One 8-bit Timer/Counter with Prescaler and Two PWM Channels
  - 4-channel, 10-bit ADC with Internal Voltage Reference
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - debugWIRE On-chip Debug System
  - In-System Programmable via SPI Port
  - External and Internal Interrupt Sources
  - Low Power Idle, ADC Noise Reduction, and Power-down Modes
  - Enhanced Power-on Reset Circuit
  - Programmable Brown-out Detection Circuit with Software Disable Function
  - Internal Calibrated Oscillator
- I/O and Packages
  - 8-pin PDIP/SOIC: Six Programmable I/O Lines
  - 20-pad MLF: Six Programmable I/O Lines
- Operating Voltage:
  - 1.8 - 5.5V
- Speed Grade:
  - 0 - 4 MHz @ 1.8 - 5.5V
  - 0 - 10 MHz @ 2.7 - 5.5V
  - 0 - 20 MHz @ 4.5 - 5.5V
- Industrial Temperature Range
- Low Power Consumption
  - Active Mode:
    - 190 µA at 1.8 V and 1 MHz
  - Idle Mode:
    - 24 µA at 1.8 V and 1 MHz

8-bit AVR® Microcontroller with 1K Bytes In-System Programmable Flash

ATtiny13A

Summary

Figure 1-1. Pinout ATtiny13A