

EPREUVE ECRITE

Ministère de l'Éducation nationale
et de la Formation professionnelle

EXAMEN DE FIN D'ÉTUDES SECONDAIRES TECHNIQUES

Régime de la formation de technicien

Division électrotechnique

Section: communication

BRANCHE : Microélectronique

SESSION :

2009

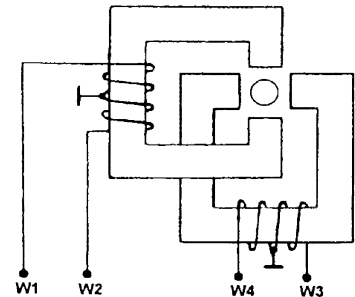
DATE :

DURÉE : 3 heures

Aufgabe 1 (3 + 3 + 8 = 14 Punkte)

Schrittmotor:

- Erklären Sie den Unterschied zwischen *Unipolarbetrieb* und *Bipolarbetrieb*.
- Nennen Sie alle Ihnen bekannten Schritarten und erklären Sie kurz ihre Eigenschaften.
- Es soll nun ein Schrittmotor angesteuert werden. Wählen Sie dazu eine beliebige Schritart aus und erstellen Sie die Tabelle mit den Steuerbytes.



Zur Bestimmung der Steuerbytewerte ist die Bitposition der Motoranschlüsse zu beachten

W4	W2	W3	W1
----	----	----	----

Der Schrittmotor wird an den Pins **PC4** - **PC7** von Port C angeschlossen, zusätzlich wird an **Bit 4** von Port A ein Schalter S gegen Masse geschaltet.

Nach dem Start des Programms soll sich der Schrittmotor je nach Stellung des Schalters S *im* Uhrzeigersinn oder *gegen* den Uhrzeigersinn drehen.

Schreiben Sie nun das Programm in Assembler für einen ATmega32-Kontroller und kommentieren Sie das Programm sinnvoll.

- Hinweise:**
- Es brauchen keine Standard-Definitionen oder -Initialisierungen aufgeschrieben zu werden, d.h. es sollen nur die Initialisierungen, die zum Betrieb des Schrittmotors benötigt werden, angegeben werden.
 - Zur korrekten Ansteuerung des Schrittmotors steht das Zeit-Unterprogramm W50ms, das eine Verzögerung von 50ms bewirkt, zur Verfügung!

Aufgabe 2 (6 + 4 = 10 Punkte)

Externe Interrupts:

Es soll nun ein Assembler-Programm für den ATmega32 erstellt werden, das bei einer steigenden Flanke am Interrupt-Eingang INT1 (PD3) ein Byte von Port C einliest und in eine Tabelle setzt.

Die Tabelle (RxTab) befindet sich im SRAM ab Adresse 0x0100 und hat eine Größe von 256 Byte.

- a)
- Nennen Sie alle Sonderfunktionsregister, die zur Programmierung des Interrupts benötigt werden und erklären Sie wie deren Bits korrekt zu setzen sind.
 - Schreiben Sie das Hauptprogramm, das u.a. auch alle notwendigen Initialisierungen durchführt. Kommentieren sie das Programm sehr ausführlich.

Hinweis: Es brauchen keine Standard-Definitionen oder -Initialisierungen aufgeschrieben zu werden.

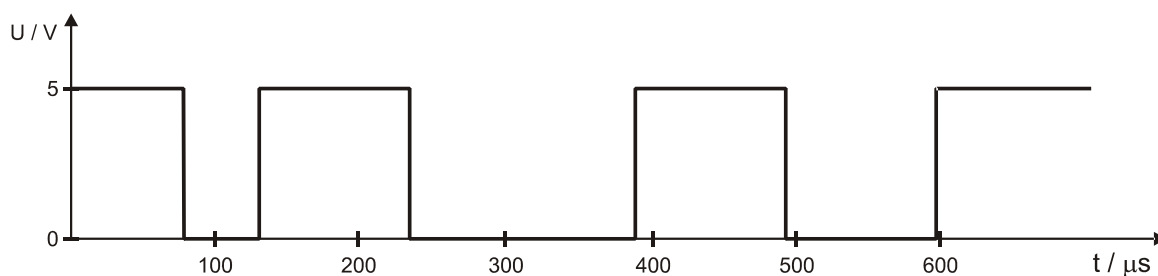
- b)
- Schreiben Sie jetzt die Interrupt-Behandlungsroutine, die die Daten einliest und in die Tabelle setzt. Kommentieren Sie auch diesen Programmteil sinnvoll.

Hinweis: Beachten Sie, dass eine Interrupt-Anforderung ein laufendes Programm zu jedem beliebigen Zeitpunkt unterbrechen kann!

Aufgabe 3 (4 + 4 + 6 + 4 + 4 + 5 = 27 Punkte)

Timer und serielle Schnittstelle:

- a)
- Gegeben ist folgendes Oszillogramm:



Die Abbildung zeigt einen Zeichenrahmen, wie er am Pin PD1 des ATmega32 gesendet wird.

- Um welches *Zeichen* handelt es sich?
 - Geben Sie die *Übertragungsparameter* der Schnittstelle an.
 - Berechnen Sie die *Übertragungsgeschwindigkeit*.
- b)
- Was ist eine *Datenflusskontrolle* und warum wird sie manchmal benötigt?
Nennen Sie zwei Möglichkeiten zur Datenflusskontrolle.

Über die serielle Schnittstelle des ATmega32 sollen 100 Mal pro Sekunde die Zustände an Port C übermittelt werden.

- c) Schreiben Sie dazu für den ATmega32 ein Assemblerprogramm, um *Timer 0* des Bausteins so zu programmieren, dass dieser 100 Mal pro Sekunde einen Interrupt auslöst und in die Interrupt-behandlungsroutine `isr_T0` verzweigt.

Berechnen Sie zuerst den *Teilerfaktor des Prescalers*.

Achtung! Taktfrequenz = **14,7456 MHz**)

Hinweis: Es brauchen keine Standard-Definitionen oder -Initialisierungen aufgeschrieben zu werden.

- d) Schreiben Sie nun die Assemblerzeilen um die serielle Schnittstelle für *19200 Baud* und *8E1* zu konfigurieren.

Hinweis: Das Steuern des Sendevorgangs soll mittels Polling erfolgen!










- e) Zeichnen Sie das *Flussdiagramm* der Interrupt-Behandlungsroutine (`isr_T0`) des Timers, in der die Zustände von Port C eingelesen und der seriellen Schnittstelle übergeben werden.
- f) Erstellen Sie das zum Flussdiagramm passende und sinnvoll dokumentierte Assemblerprogramm der Interrupt-Behandlungsroutine.

Aufgabe 4 (9 Punkte)

Analog-Digital-Wandler:

- Zeichnen Sie das Blockschaltbild des *Wägeverfahrens*.
- Geben Sie noch eine andere Bezeichnung für das Wägeverfahren an.
- Beschreiben Sie in Ihren eigenen Worten die Funktionsweise des Verfahrens.

A.1 ASCII-Tabelle

	Upper Bits	bin	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
Lower Bits		bin	hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0		NUL	DLE	space	0	@	P	`	p	Ç	É	á		Ł	⌌	α	≡	
		0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240		
0001	1		SOH	DC1	!	1	A	Q	a	q	ü	æ	í		⌐	⌌	ß	±	
		1	17	33	49	65	81	97	113	129	145	161	177	193	209	225	241		
0010	2		STX	DC2	"	2	B	R	b	r	é	Æ	ó		⌐	⌐	Γ	≥	
		2	18	34	50	66	82	98	114	130	146	162	178	194	210	226	242		
0011	3		ETX	DC3	#	3	C	S	c	s	â	ô	ú		⌐	⌌	π	≤	
		3	19	35	51	67	83	99	115	131	147	163	179	195	211	227	243		
0100	4		EOT	DC4	\$	4	D	T	d	t	ä	ö	ñ	⌐	—	Ł	Σ	∫	
		4	20	36	52	68	84	100	116	132	148	164	180	196	212	228	244		
0101	5		ENQ	NAK	%	5	E	U	e	u	à	ò	Ñ	⌐	⌐	ƒ	σ	∫	
		5	21	37	53	69	85	101	117	133	149	165	181	197	213	229	245		
0110	6		ACK	SYN	&	6	F	V	f	v	å	û	ª	⌐	⌐	⌐	μ	÷	
		6	22	38	54	70	86	102	118	134	150	166	182	198	214	230	246		
0111	7		BEL	ETB	'	7	G	W	g	w	ç	ù	º	⌐	⌐	⌐	τ	≈	
		7	23	39	55	71	87	103	119	135	151	167	183	199	215	231	247		
1000	8		BS	CAN	(8	H	X	h	x	ê	ÿ	¿	⌐	⌐	⌐	Φ	°	
		8	24	40	56	72	88	104	120	136	152	168	184	200	216	232	248		
1001	9		TAB	EM)	9	I	Y	i	y	ë	Ö	ƒ	⌐	⌐	⌐	⊙	•	
		9	25	41	57	73	89	105	121	137	153	169	185	201	217	233	249		
1010	A		LF	SUB	*	:	J	Z	j	z	è	Ü	ƒ	⌐	⌐	⌐	Ω	•	
		10	26	42	58	74	90	106	122	138	154	170	186	202	218	234	250		
1011	B		VT	ESC	+	;	K	[k	{	ï	ç	½	⌐	⌐		δ	√	
		11	27	43	59	75	91	107	123	139	155	171	187	203	219	235	251		
1100	C		FF	FS	,	<	L	\	l		î	£	¼	⌐	⌐		∞	n	
		12	28	44	60	76	92	108	124	140	156	172	188	204	220	236	252		
1101	D		CR	GS	-	=	M]	m	}	ì	¥	ı	⌐	=		φ	²	
		13	29	45	61	77	93	109	125	141	157	173	189	205	221	237	253		
1110	E		SO	RS	.	>	N	^	n	~	Ä	Œ	«	⌐	⌐		ε		
		14	30	46	62	78	94	110	126	142	158	174	190	206	222	238	254		
1111	F		SI	US	/	?	O	—	o	DEL	Å	f	»	⌐	⌐		∩		
		15	31	47	63	79	95	111	127	143	159	175	191	207	223	239	255		

ASCII CONTROL CHARACTER ABBREVIATIONS

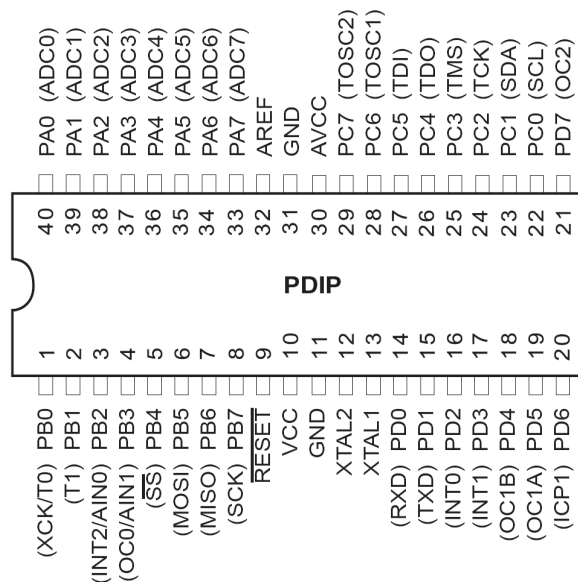
NUL	:	null	VT	:	vertical tabulation	SYN	:	synchronous idle
SOH	:	start of heading	FF	:	form feed	ETB	:	end of transmission block
STX	:	start of text	CR	:	carriage return	CAN	:	cancel
ETX	:	end of text	SO	:	shift out	EM	:	end of medium
EOT	:	end of transmission	SI	:	shift in	SUB	:	substitute
ENQ	:	enquiry	DLE	:	data link escape	ESC	:	escape
ACK	:	acknowledge	DC1	:	device control 1 / XON	FS	:	file separator
BEL	:	bell	DC2	:	device control 2	GS	:	group separator
BS	:	backspace	DC3	:	device control 3 / XOFF	RS	:	record separator / req. to send
TAB	:	horizontal tabulation	DC4	:	device control 4	US	:	unit separator
LF	:	linefeed	NAK	:	negative acknowledge	DEL	:	delete

A.2 Der SF-Registersatz

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
\$3F (\$5F)	SREG	I	T	H	S	V	N	Z	C	10
\$3E (\$5E)	SPH	–	–	–	–	SP11	SP10	SP9	SP8	12
\$3D (\$5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	12
\$3C (\$5C)	OCR0	Timer/Counter0 Output Compare Register								82
\$3B (\$5B)	GICR	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	47, 67
\$3A (\$5A)	GIFR	INTF1	INTF0	INTF2	–	–	–	–	–	68
\$39 (\$59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	82, 112, 130
\$38 (\$58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	83, 113, 130
\$37 (\$57)	SPMCR	SPMIE	RWWBSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	248
\$36 (\$56)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	177
\$35 (\$55)	MCUCR	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	32, 66
\$34 (\$54)	MCUCSR	JTD	ISC2	–	JTRF	WDRF	BORF	EXTRF	PORF	40, 67, 228
\$33 (\$53)	TCCR0	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	80
\$32 (\$52)	TCNT0	Timer/Counter0 (8 Bits)								82
\$31 ⁽¹⁾ (\$51) ⁽¹⁾	OSCCAL	Oscillator Calibration Register								30
	OCDR	On-Chip Debug Register								224
\$30 (\$50)	SFIO	ADTS2	ADTS1	ADTS0	–	ACME	PJD	PSR2	PSR10	56, 85, 131, 198, 218
\$2F (\$4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	107
\$2E (\$4E)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	110
\$2D (\$4D)	TCNT1H	Timer/Counter1 – Counter Register High Byte								111
\$2C (\$4C)	TCNT1L	Timer/Counter1 – Counter Register Low Byte								111
\$2B (\$4B)	OCR1AH	Timer/Counter1 – Output Compare Register A High Byte								111
\$2A (\$4A)	OCR1AL	Timer/Counter1 – Output Compare Register A Low Byte								111
\$29 (\$49)	OCR1BH	Timer/Counter1 – Output Compare Register B High Byte								111
\$28 (\$48)	OCR1BL	Timer/Counter1 – Output Compare Register B Low Byte								111
\$27 (\$47)	ICR1H	Timer/Counter1 – Input Capture Register High Byte								112
\$26 (\$46)	ICR1L	Timer/Counter1 – Input Capture Register Low Byte								112
\$25 (\$45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	125
\$24 (\$44)	TCNT2	Timer/Counter2 (8 Bits)								127
\$23 (\$43)	OCR2	Timer/Counter2 Output Compare Register								127
\$22 (\$42)	ASSR	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB	128
\$21 (\$41)	WDTCSR	–	–	–	WDTOE	WDE	WDP2	WDP1	WDP0	42
\$20 ⁽²⁾ (\$40) ⁽²⁾	UBRRH	URSEL	–	–	–	UBRR[11:8]				164
	UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	162
\$1F (\$3F)	EEARH	–	–	–	–	–	–	EEAR9	EEAR8	19
\$1E (\$3E)	EEARL	EEPROM Address Register Low Byte								19
\$1D (\$3D)	EEDR	EEPROM Data Register								19
\$1C (\$3C)	EEDR	–	–	–	–	EERIE	EEMWE	EWE	EERE	19
\$1B (\$3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	64
\$1A (\$3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	64
\$19 (\$39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	64
\$18 (\$38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	64
\$17 (\$37)	DDRB	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	64
\$16 (\$36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	65
\$15 (\$35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	65
\$14 (\$34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	65
\$13 (\$33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	65
\$12 (\$32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	65
\$11 (\$31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	65
\$10 (\$30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	65
\$0F (\$2F)	SPDR	SPI Data Register								138
\$0E (\$2E)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	138
\$0D (\$2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	136
\$0C (\$2C)	UDR	USART I/O Data Register								159
\$0B (\$2B)	UCSRA	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	160
\$0A (\$2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	161
\$09 (\$29)	UBRRL	USART Baud Rate Register Low Byte								164
\$08 (\$28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	199
\$07 (\$27)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	214
\$06 (\$26)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	216
\$05 (\$25)	ADCH	ADC Data Register High Byte								217
\$04 (\$24)	ADCL	ADC Data Register Low Byte								217
\$03 (\$23)	TWDR	Two-wire Serial Interface Data Register								179
\$02 (\$22)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	179
\$01 (\$21)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	178
\$00 (\$20)	TWBR	Two-wire Serial Interface Bit Rate Register								177

- Notes:
1. When the OCDEN Fuse is unprogrammed, the OSCCAL Register is always accessed on this address. Refer to the debug-ger specific documentation for details on how to use the OCDR Register.
 2. Refer to the USART description for details on how to access UBRRH and UCSRC.
 3. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
 4. 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 \$00 to \$1F only.

A.3 Pinbelegung



A.4 Interrupt-Vektor-Tabelle

Vektor-nummer	Adresse im Programmspeicher	symbolische Adresse	Quelle des Interrupts	verantwortlich für den Interrupt
1	0x000	keine	RESET	externer Pin, Power-On-Reset, Brown-Out-Reset,, ...
2	0x002	INT0addr	INT0	externer Interrupt-Eingang 0
3	0x004	INT1addr	INT1	externer Interrupt-Eingang 1
4	0x006	INT2addr	INT2	externer Interrupt-Eingang 2
5	0x008	OC2addr	TIMER2 COMP	Compare Match von Timer 2
6	0x00A	OVF2addr	TIMER2 OVF	Overflow Match von Timer 2
7	0x00C	ICP1addr	TIMER1 CAPT	Capture Event von Timer 1
8	0x00E	OC1Aaddr	TIMER1 COMPA	Compare Match A von Timer 1
9	0x010	OC1Baddr	TIMER1 COMPB	Compare Match B von Timer 1
10	0x012	OVF1addr	TIMER1 OVF	Overflow von Timer 1
11	0x014	OC0addr	TIMER0 COMP	Compare Match von Timer 0
12	0x016	OVF0addr	TIMER0 OVF	Overflow von Timer 0
13	0x018	SPIaddr	SPI, STC	Serielle Übertragung beendet
14	0x01A	URXCaddr	USART, RXC	USART, Empfangsregister voll
15	0x01C	UDREaddr	USART, UDRE	USRAT, UDR-Register leer
16	0x01E	UTXCaddr	USART, TXC	USART, Senderegister leer
17	0x020	ADCCaddr	ADC	AD-Wandlung vollständig
18	0x022	ERDYaddr	EE_RDY	EEPROM-Interface
19	0x024	ACIaddr	ANA_COMP	Analog-Komparator

20	0x026	TWIaddr	TWI	I ² C-Interface
21	0x028	SPMRaddr	SPM_RDY	Interface für Programmspeicher

A.5 Statusregister SREG

SREG = Status Register

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	
Startwert	0	0	0	0	0	0	0	0	

I Global Interrupt Flag

Das *Global Interrupt Enable/Disable-Flag I* wird vom Benutzer gesetzt („1“) oder zurückgesetzt („0“) um global Unterbrechungen zu erlauben bzw. zu verbieten (Befehle: SEI, CLI).

T Transfer-Flag

Das *Transfer-Flag T* erlaubt mit Hilfe der Befehle BLD und BST ein einzelnes Bit aus einem Arbeitsregister abzuspeichern (retten) und wieder zu laden (wiederherstellen). Es kann mit den Befehlen SET und CLT auch einfach gelöscht oder gesetzt werden.

H Halfcarry-Flag

Das *Halfcarry-Flag H* (oder *Auxiliary-Carry*) kennzeichnet einen Übertrag von Bit 3 auf Bit 4 und wird bei Berechnungen mit Nibbles (4 Bit) benötigt. Es kann mit den Befehlen SEH und CLH auch einfach gelöscht oder gesetzt werden.

S Sign-Flag

Das *Sign-Flag S* wird bei der Berechnung mit vorzeichenbehafteten Zahlen eingesetzt. Es entspricht der Exklusiv-Oder-Verknüpfung des Negative- und des Overflow-Flags: $S = N \text{ XOR } V$. Es kann mit den Befehlen SES und CLS auch einfach gelöscht oder gesetzt werden.

V Overflow-Flag

Das *Overflow-Flag V* zeigt einen Überlauf bei der Zweierkomplement-Berechnung, also bei vorzeichenbehafteten Zahlen an. Es kann mit den Befehlen SEV und CLV auch einfach gelöscht oder gesetzt werden.

N Negative-Flag

Das *Negative-Flag N* entspricht dem höchstwertigen Bit eines Ergebnisses. Bei 8 Bit ist $N = r7$ und bei 16 Bit ist $N = r15$. Es kann mit den Befehlen SEN und CLN auch einfach gelöscht oder gesetzt werden.

Z Zero-Flag

Das *Zero-Flag Z* wird auf „Eins“ gesetzt wenn das Ergebnis einer Operation Null ist. Es kann mit den Befehlen SEZ und CLZ auch einfach gelöscht oder gesetzt werden.

C Carry-Flag

Das *Carry-Flag C* wird „Eins“, wenn ein Übertrag an der höchsten Stelle entsteht. Es kann mit den Befehlen SEC und CLC auch einfach gelöscht oder gesetzt werden.

A.6 Ein- /Ausgabeports

DDRx = Data Direction Register PORTx

Bit	7	6	5	4	3	2	1	0	
	DDRx7	DDRx6	DDRx5	DDRx4	DDRx3	DDRx2	DDRx1	DDRx0	DDRx
Read / Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Startwert	0	0	0	0	0	0	0	0	

DDRxn Data Direction Register x Pin n

- 0** Der betreffende Pin ist als Eingang aktiv.
- 1** Der betreffende Pin ist als Ausgang beschaltet.

PORTx = PORTx Data Register

Bit	7	6	5	4	3	2	1	0	
	PORTx7	PORTx6	PORTx5	PORTx4	PORTx3	PORTx2	PORTx1	PORTx0	PORTx
Read / Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Startwert	0	0	0	0	0	0	0	0	

PORTxn PORTx Data Pin n

a) Der Pin ist als Ausgang konfiguriert DDRxn = 1

- 0** Der Pin liegt auf Masse.
- 1** Der Pin liegt auf V_{CC} .

b) Das Pin ist als Eingang konfiguriert DDRxn = 0

- 0** Der Pin ist hochohmig (TriState).
- 1** Ist das PUD-Bit (pull up disable) im Register SFIOR gelöscht (Startwert) so wird intern ein Pull Up-Widerstand gegen V_{CC} geschaltet. Bei gesetztem PUD-Bit ist der Ausgang hochohmig.

PINx = PORTx Input Pins Address

Bit	7	6	5	4	3	2	1	0	
	PINx7	PINx6	PINx5	PINx4	PINx3	PINx2	PINx1	PINx0	PINx
Read / Write	R	R	R	R	R	R	R	R	
Startwert	-	-	-	-	-	-	-	-	

PINxn PORTx Input Pin n

Unabhängig vom Datenrichtungsregister kann über die PINx Adresse der Zustand eines Pins eingelesen werden.

A.7 Interrupts

MCUCR = MCU Control Register

Bit	7	6	5	4	3	2	1	0	
	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read / Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Startwert	0	0	0	0	0	0	0	0	

ISCn Interrupt Sense Control n

ISC01 und ISC00 beziehen sich auf den Interrupt-Eingang INT0 und definieren, ob der Eingang zustandsgesteuert oder flankengesteuert ist (s. Tabelle unten), ISC11 und ISC10 beziehen sich auf den Eingang INT1.

ISCx1	ISCx0	Beschreibung
0	0	Ein Low-Pegel löst einen Interrupt aus.
0	1	Jede Pegeländerung löst einen Interrupt aus.
1	0	Eine fallende Flanke löst einen Interrupt aus.
1	1	Eine steigende Flanke löst einen Interrupt aus.

MCUCSR = MCU Control and Status Register

Bit	7	6	5	4	3	2	1	0	
	JTD	ISC2	-	JTRF	WDRF	BORF	EXTRF	PORF	MCUCSR
Read / Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Startwert	0	0	0	-	-	-	-	-	

ISC2 Interrupt Sense Control 2

Wird INT2 benutzt, so muss im Register MCUCSR das Bit ISC2 korrekt gesetzt werden. Es entscheidet darüber, ob der Controller auf eine fallende (ISC2 = 0) oder einer steigende Flanke (ISC2 = 1) reagieren soll.

GICR = General Interrupt Control Register

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	INT2	-	-	-	IVSEL	IVCE	GICR
Read / Write	R/W	R/W	R/W	R	R	R	R/W	R/W	
Startwert	0	0	0	0	0	0	0	0	

INTx External Interrupt Request n Enable

- 0** Der betreffende Interrupt ist nicht aktiviert.
- 1** Der betreffende Interrupt ist aktiviert, wenn ebenfalls das I-Bit in SREG gesetzt ist (Interrupts global erlaubt).
Ein Interrupt wird auch erkannt, wenn der betreffende Pin als Ausgang initialisiert wurde.

GIFR = General Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
	INTF1	INTF0	INTF2	-	-	-	-	-	GIFR
Read / Write	R/W	R/W	R/W	R	R	R	R	R	
Startwert	0	0	0	0	0	0	0	0	

INTFn External Interrupt Flag n

- 0** Es trat kein Interrupt auf. Das Flag wird automatisch gelöscht wenn die Interruptroutine aufgerufen wird. Es ist beim INT0 bzw. INT1 dauernd gelöscht, wenn die Interrupts auf Pegel reagieren sollen.
- 1** Es wurde ein Interrupt erkannt und dies wird im Flag gespeichert. Passiert dies innerhalb einer Interruptroutine (wo standardmäßig Interrupts global gesperrt sind), so wird das Interrupt nach dem Verlassen der Routine ausgeführt. Mit dem Schreiben einer „Eins“ (!) kann das Interrupt-Flag manuell gelöscht werden.

A.8 Timer

TCCR0 = Timer/Counter Control Register 0

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read / Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Startwert	0	0	0	0	0	0	0	0	

WGM0n Waveform Generation Mode (WGM01, WGM00)

Mit diesen zwei Bit wird der Operationsmodus des Timer festgelegt:

2¹ 2⁰ WGM0n

- 0 0** Normaler Modus
- 0 1** Phasenkorrekter PWM-Modus
- 1 0** CTC-Modus (clear timer on compare match)
- 1 1** Fast PWM-Modus

COM0n Compare Match Output Mode (COM01, COM00)

Mit diesen zwei Bit wird das Verhalten des Ausgangspin OC0 festgelegt. Je nach Modus ändert das Verhalten:

2¹ 2⁰ CM0n (Verhalten im normalen Modus)

- 0 0** OC0 abgeschaltet (normales Port-Pin)
- 0 1** Toggele OC0 bei Vergleichsübereinstimmung
- 1 0** Lösche OC0 bei Vergleichsübereinstimmung
- 1 1** Setze OC0 bei Vergleichsübereinstimmung

2¹ 2⁰ CM0n (Verhalten im Fast-PWM-Modus)

- 0 0** OC0 abgeschaltet (normales Port-Pin)
- 0 1** Reserviert
- 1 0** nicht-invertierender Fast-PWM-Modus
- 1 1** invertierender Fast-PWM-Modus

CS0n Clock Select Timer 0 (CS02, CS01, CS00)

Mit diesen drei Bit wird die Taktquelle für Timer 0 festgelegt.

2³ 2¹ 2⁰ CS0n (Taktquelle)

- 0 0 0** Timer Stopp (verbraucht keinen Strom, default nach RESET!)
- 0 0 1** Systemtakt (:1)
- 0 1 0** Systemtakt / 8
- 0 1 1** Systemtakt / 64
- 1 0 0** Systemtakt / 256
- 1 0 1** Systemtakt / 1024
- 1 1 0** externer Takt: fallende Flanke an T0 (PB0)
- 1 1 1** externer Takt: steigende Flanke an T0 (PB0)

FOC0 Force Output Compare

- 0** Dieses Bit wird hier nicht benötigt und sollte immer logisch „0“ sein.

TIMSK = Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Read / Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Startwert	0	0	0	0	0	0	0	0	

OCIE0 Timer/Counter 0 Output Compare Match Interrupt Enable

- 1** Das Setzen dieses Bits ermöglicht das Auslösen eines Interrupts in dem Moment, wo das OCF0-Flag (TIFR) gesetzt wird, also eine Übereinstimmung zwischen dem Zählregister TCNT0 und dem Vergleichsregister OCR0 stattgefunden hat, oder wenn das Flag manuell gesetzt wurde. Interrupts müssen dazu global frei gegeben sein (I = 1 im SREG mit SEI).
- 0** kein OCF0-Interrupt erlaubt.

TOIE0 Timer/Counter 0 Overflow Interrupt Enable

- 1** Das Setzen dieses Bit ermöglicht das Auslösen eines Interrupts in dem Moment, wo das TOV0-Flag (TIFR) gesetzt wird, also ein Überlauf auftrat oder wenn das Flag manuell gesetzt wurde. Interrupts müssen dazu global frei gegeben sein (I = 1 im SREG mit SEI).
- 0** kein TOV0-Interrupt erlaubt.

TIFR = Timer/Counter Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Read / Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Startwert	0	0	0	0	0	0	0	0	

OCF0 Output Compare Flag 0

- 0** Keine Übereinstimmung des Inhalts des Zählregister mit dem Inhalt des Vergleichsregister.
- 1** Bei Übereinstimmung (engl.: compare match) des Wertes im Zählregister TCNT0 mit dem Wert im Vergleichsregister OCR0 wird das Flag gesetzt.

TOV0 Timer/Counter 0 Overflow Flag

- 0** Kein Überlauf aufgetreten.
- 1** Tritt ein Überlauf des Zählregisters TCNT0 auf, so wird das Flag gesetzt.

TCNT0 = Timer/Counter Register 0

Bit	7	6	5	4	3	2	1	0	
	TCNT07	TCNT06	TCNT05	TCNT04	TCNT03	TCNT02	TCNT01	TCNT00	TCNT0
Read / Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Startwert	0	0	0	0	0	0	0	0	

OCR0 = Output Compare Register 0

Bit	7	6	5	4	3	2	1	0	
	OCR07	OCR06	OCR05	OCR04	OCR03	OCR02	OCR01	OCR00	OCR00
Read / Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Startwert	0	0	0	0	0	0	0	0	

A.9 Die serielle Schnittstelle

UCSRA = USART Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
Read / Write	R	R/W	R	R	R	R	R/W	R/W	
Startwert	0	0	1	0	0	0	0	0	

RXC USART Recieve Complete

- 0 Wenn sich keine Daten mehr im Empfangspuffer befinden oder wenn der Empfänger ausgeschaltet ist (Startwert).
- 1 Wird auf logisch „Eins“ gesetzt, wenn sich Daten im Empfangspuffer befinden. Zeigt also an, dass die Daten ausgelesen werden können.
Kann zusätzlich einen Interrupt auslösen (s. a. UCSRB).

TXC USART Transmit Complete

- 0 Es sind noch Daten im Schieberegister oder Datenregister vorhanden. Es kann noch kein neues Zeichen gesendet werden (Startwert).
- 1 Wird auf „Eins“ gesetzt, wenn alle Daten (gesamter Rahmen) aus dem Schieberegister ausgegeben worden sind und keine neuen Daten im UDR-Datenregister (Sendepuffer) vorliegen. Muss „*manuell*“ mit einer logischen „Eins“ (!) vor der Ausgabe gelöscht werden!
Kann zusätzlich einen Interrupt auslösen (s. a. UCSRB).

UDRE USART Data Register Empy

- 0 Datenregister UDR besetzt.
- 1 wird auf „Eins“ gesetzt, wenn das Datenregister UDR (Sendepuffer) leer ist und der USART somit bereit ist neue Daten anzunehmen (Startwert nach RESET!). Kann zusätzlich einen Interrupt auslösen (s. a. UCSRB).

FE Frame Error

- 0 kein Rahmenfehler.
- 1 Rahmenfehler, kein gültiges Stopbit erkannt.

DOR Data Overrun

- 0 kein Überlauffehler.
- 1 Überlauffehler; tritt auf wenn beide Pufferregister (RXB bzw. UDR) und das Schieberegister belegt sind, und dann ein gültiges Startbit erkannt wird.

PE Parity Error

- 0 kein Paritätsfehler
- 1 Paritätsfehler.

UCSRB = USART Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
Read / Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Startwert	0	0	0	0	0	0	0	0	

RXCIE RXC Interrupt Enable

- 0** kein RXC-Interrupt erlaubt.
- 1** Das Setzen dieses Bit ermöglicht das Auslösen eines Interrupts in dem Moment, wo das RXC-Flag (UCSRA) gesetzt wird, also neue Daten im Empfangspuffer vorhanden sind. Interrupts müssen dazu global frei gegeben sein (I = 1 im SREG mit dem Befehl SET).

TXCIE TXC Interrupt Enable

- 0** kein TXC-Interrupt erlaubt.
- 1** Das Setzen dieses Bit ermöglicht das Auslösen eines Interrupts in dem Moment, wo das TXC-Flag (UCSRA) gesetzt wird, also Schieberegister und UDR-Register (Sendepuffer) leer sind. Interrupts müssen dazu global frei gegeben sein.

UDRIE UDRE Interrupt Enable

- 0** kein UDRE-Interrupt erlaubt.
- 1** Das Setzen dieses Bit ermöglicht das Auslösen eines Interrupts in dem Moment, wo das UDRE-Flag (UCSRA) gesetzt wird, also das UDR-Datenregister (Sendepuffer) leer ist. Interrupts müssen dazu global frei gegeben sein

RXEN Receiver Enable

- 0** schaltet den Empfänger aus.
- 1** schaltet den Empfänger ein. Pin RxD (PD0) ist dann als Eingang reserviert (muss nicht extra initialisiert werden) und ist für andere Aktionen nicht mehr zugänglich.

TXEN Transmitter Enable

- 0** schaltet den Sender aus
- 1** schaltet den Sender ein. Pin TxD (PD1) ist als Ausgang reserviert (muss nicht extra initialisiert werden) und ist für andere Aktionen nicht mehr zugänglich.

UCSZ2 USART Character Size 2

siehe Register UCSRC

UCSRC = USART Control an Status Register C

Bit	7	6	5	4	3	2	1	0	
	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
Read / Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Startwert	1	0	1	0	0	1	1	0	

URSEL USART Register Select

Der Speicherplatz ist mit zwei Registern belegt.

- 0** das Register UBRRH wird ausgewählt. Beim Lesen von UBRRH ist das Bit logisch „Null“.
- 1** das Register UCSRC wird ausgewählt (default). Beim Lesen von UCSRC ist das Bit „Eins“.

UMSEL USART Mode Select

- 0** Asynchroner Modus
- 1** Synchroner Modus

UPM USART Parity Mode (UPM1, UPM0)

Ist die Parität eingeschaltet (gerade oder ungerade Parität) wird hardwaremäßig die Parität generiert und in den Zeichenrahmen integriert. Als Empfänger kontrolliert der USART die Parität. Ist ein Fehler aufgetreten, so wird das

PE-Flag (Parity Error) in UCSRA gesetzt.

2¹ 2⁰ UPM

0 0 keine Parität

0 1 reserviert

1 0 gerade Parität eingeschaltet

1 1 ungerade Parität eingeschaltet.

USBS USART Stop Bit Select

0 1 Stoppbit

1 2 Stoppbits

UCSZ USART Character Size (UCSZ2, UCSZ1, UCSZ0)

Mit diesen drei Bit wird die Anzahl der Datenbits (Zeichengröße) eingestellt. UCSZ2 befindet sich auf Bit 2 des UCSRB-Registers und wird nur benötigt wenn 9 Bit benutzt werden. Sonst kann dieses Bit unberücksichtigt bleiben, da sein Default-Wert „Null“ ist.

2² 2¹ 2⁰ UCSZ

0 0 0 5 Bit

0 0 1 6 Bit

0 1 0 7 Bit

0 1 1 8 Bit

1 0 0 reserviert

1 0 1 reserviert

1 1 0 reserviert

1 1 1 9 Bit

UCPOL USART Clock Polarity

Wird nur bei synchroner Datenübertragung benutzt

0 bei asynchroner Übertragung

UBRRH = USART Baud Rate Register High

Bit	7	6	5	4	3	2	1	0	
	URSEL	-	-	-	UBRR [11:8]				UCSRA
Read / Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
Startwert	0	0	0	0	0	0	0	0	

UBRRL = USART Baud Rate Register Low

Bit	7	6	5	4	3	2	1	0	
	UBRR [7:0]								UCSRA
Read / Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Startwert	0	0	0	0	0	0	0	0	

URSEL USART Register Select

Der Speicherplatz ist mit zwei Registern belegt.

- 0 das Register UBRRH wird ausgewählt. Beim Lesen von UBRRH ist das Bit logisch „Null“.
- 1 das Register UCSRC wird ausgewählt (Startwert). Beim Lesen von UCSRC ist das Bit „Eins“.

UBRR USART Baud Rate Register

In diesen 12 Bit befindet sich der Teiler aus dem sich die Baudrate berechnen lässt. Die Abtastung des Eingangssignals RxD erfolgt mit dem 16-fachen Übertragungstakt (Baudrate). Die Gleichung lautet:

$$Baudrate = \frac{Systemtakt}{16 \cdot (Teiler + 1)}$$

Umgekehrt lässt sich natürlich auch der Teiler bei erwünschter Baudrate errechnen:

$$Teiler = \frac{Systemtakt}{16 \cdot Baudrate} - 1$$

Die mit dem Teiler erreichte Baudrate entspricht nicht immer dem genauen Standardwert. Der Fehler (in Prozent) errechnet sich mit:

$$Fehler[\%] = \left(\frac{Baudrate}{Standard} - 1 \right) \cdot 100\%$$

Fehler unter 0,5% sollen angestrebt werden, da sonst die Fehlerrate bei der Übertragung zunimmt. Die kann zum Beispiel durch das Auswechseln des Quarzes erfolgen.

UDR = USART I/O Data Register

Bit	7	6	5	4	3	2	1	0	
	<div><div>RXB [7:0]</div><div>TXB [7:0]</div></div>								UCSRA
Read / Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Startwert	0	0	0	0	0	0	0	0	

TXB Transmit Data Buffer Register

Sendepuffer.

RXB Receive Data Buffer Register

Empfangspuffer.

A.10 Der Analog-Digital-Wandler

ADMUX = ADC Multiplexer Selection

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	
Startwert	0	0	0	0	0	0	0	0	

REFSx Reference Selection Bits (REFS1, REFS0)

Die Bits REFS0 und REFS1 wählen aus insgesamt 3 Möglichkeiten aus:

REFS1	REFS0	Beschreibung
0	0	Eine externe Referenzspannung liegt am Pin AREF, die interne Referenzspannungsquelle ist abgeschaltet.
0	1	Die Betriebsspannung des A/D-Wandlers an Pin AVCC wird als Referenzspannungsquelle genutzt. Die Spannung liegt am Pin AREF an dem sich ein Kondensator befinden muss.
1	0	Reserviert
1	1	Die interne Referenzspannungsquelle ist eingeschaltet und wird benutzt. Die Spannung liegt am Pin AREF an dem sich ein Kondensator befinden muss.

ADLAR ADC Left Adjust Result

Das Bit ADLAR entscheidet darüber, ob das 10-Bit-Ergebnis der Wandlung links- oder rechtsbündig im 16-Bit breiten Datenregister (ADCH und ADCL) liegt.

- 0 Die 8 niederwertigen Bits des Ergebnisses werden in ADCL abgelegt. Die verbleibenden 2 Bits des Ergebnisses werden im Register ADCH in den Bits 0 und 1 abgelegt. Die Positionen der Bits entsprechen dann den Wertigkeiten einer 16-Bit-Dualzahl.
- 1 Werden aber nur 8 Bit Auflösung benötigt, so sind die beiden niederwertigsten Bits 2^0 und 2^1 überflüssig und alle Bits müssten um 2 Positionen nach rechts geschoben werden, damit die Positionen der Ergebnisbits den Wertigkeiten einer 8-Bit-Dualzahl entsprechen. Die 8 werthöheren Bits stehen in ADCH.

MUXx Analog Channel and Gain Selection Bits

Fünf Steuerbits dienen dazu aus insgesamt 32 Möglichkeiten auszuwählen, um den Eingang des A/D-Wandlers mit einem der analogen Eingänge zu verbinden (s. Tabelle unten).

MUX 4..0	unsymmetrischer Eingang	nicht-invertierender Eingang	invertierender Eing.	Verstärkung
00000	ADC0			
00001	ADC1			
00010	ADC2			
00010	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			

MUX 4..0	unsymmetrischer Eingang	nicht-invertierender Eingang	invertierender Eingang	Verstärkung
01000		ADC0	ADC0	10x
01001		ADC1	ADC0	10x
01010		ADC0	ADC0	200x
01011		ADC1	ADC0	200x
01100		ADC2	ADC2	10x
01101		ADC3	ADC2	10x
01110		ADC2	ADC2	200x
01111		ADC3	ADC2	200x
10000		ADC0	ADC1	1x
10001		ADC1	ADC1	1x
10010		ADC2	ADC1	1x
10011		ADC3	ADC1	1x
10100		ADC4	ADC1	1x
10101		ADC5	ADC1	1x
10110		ADC6	ADC1	1x
10111		ADC7	ADC1	1x
11000		ADC0	ADC2	1x
11001		ADC1	ADC2	1x
11010		ADC2	ADC2	1x
11011		ADC3	ADC2	1x
11100		ADC4	ADC2	1x
11101		ADC5	ADC2	1x
11110	1,22 V			
11111	0 V (GND)			

ADCSRA = ADC Control and Status Register

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	
Startwert	0	0	0	-	-	-	-	-	

ADEN ADC Enable

- 0** Schaltet die A/D-Wandler-Baugruppe aus. Wird während einer laufenden Wandlung das Bit auf „Null“ gesetzt, so wird die Wandlung abgebrochen.
- 1** Wird auf logisch „Eins“ gesetzt, um die A/D-Wandler-Baugruppe einzuschalten.

ADSC ADC Start Conversion

- 0** Das Bit wird „Null“, wenn die A/D-Wandlung beendet ist. Das Rücksetzen dieses Bits (auf logisch „0“) hat keinen Einfluss.
- 1** Im Single Conversion Modus muss das Bit zum Auslösen jeder Wandlung explizit auf „1“ *gesetzt* werden, während

im Free Running Mode das Bit einmalig zum Auslösen der ersten Wandlung gesetzt wird. Das Bit darf zusammen (also zeitgleich) mit ADEN gesetzt werden.

ADATE ADC Auto Trigger Enable

- 0** Eine „Null“ beendet den Auto Trigger-Modus.
- 1** Wird dieses Bit auf „Eins“ gesetzt, so ist der *Auto Trigger-Modus* der Wandler-Baugruppe aktiviert. In diesem Fall wird eine Wandlung durch die steigende Flanke eines Trigger-Signals ausgelöst. Die Quelle des Trigger-Signals wird durch die Bits ADTS2 . . . ADTS0 des Sonderfunktionsregisters SFIOR definiert.

ADIF ADC Interrupt Flag

- 0** Es liegen keine aktuellen Wandlerdaten im Datenregister der A/D-Wandler-Baugruppe.
- 1** Dieses Bit wird auf „Eins“ gesetzt, wenn eine Wandlung beendet und das Datenregister (ADCH und ADCL) aktualisiert wurde.

ADIE ADC Interrupt Enable

- 0** Die Interrupt-Steuerung der Baugruppe ist abgeschaltet.
- 1** Ist dieses Bit und das I-Flag auf „Eins“ gesetzt, dann wird nach Beendigung einer A/D-Wandlung eine Unterbrechung ausgelöst.

ADPS ADC Prescaler Select Bits

Diese Bits legen den Teilerfaktor zwischen dem Systemtakt und dem Takt des A/D-Wandlers fest.

2² 2¹ 2⁰ Teilerfaktor

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

SFIOR = Special Function IO Register

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	-	ACME	PUD	PSR2	PSR10	SFIOR
Read / Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Startwert	0	0	0	0	0	0	0	0	

ADTS 2² 2¹ 2⁰ Trigger-Quelle

- 0 0 0** Free Running Mode
- 0 0 1** Analog Comparator
- 0 1 0** External Interrupt Request 0 (INT0)
- 0 1 1** Timer/Counter0 Compare Match
- 1 0 0** Timer/Counter0 Overflow
- 1 0 1** Timer/Counter1 Compare Match B
- 1 1 0** Timer/Counter1 Overflow
- 1 1 1** Timer/Counter1 Capture Event

A.11 Der Befehlssatz

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	$Rd \leftarrow Rd + Rr$	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	$Rd \leftarrow Rd + Rr + C$	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	$Rdh:Rdl \leftarrow Rdh:Rdl + K$	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	$Rd \leftarrow Rd - Rr$	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	$Rd \leftarrow Rd - K$	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	$Rd \leftarrow Rd - Rr - C$	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	$Rd \leftarrow Rd - K - C$	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	$Rdh:Rdl \leftarrow Rdh:Rdl - K$	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	$Rd \leftarrow Rd \bullet Rr$	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	$Rd \leftarrow Rd \bullet K$	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	$Rd \leftarrow Rd \vee Rr$	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	$Rd \leftarrow Rd \vee K$	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	$Rd \leftarrow Rd \oplus Rr$	Z,N,V	1
COM	Rd	One's Complement	$Rd \leftarrow \$FF - Rd$	Z,C,N,V	1
NEG	Rd	Two's Complement	$Rd \leftarrow \$00 - Rd$	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	$Rd \leftarrow Rd \vee K$	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	$Rd \leftarrow Rd \bullet (\$FF - K)$	Z,N,V	1
INC	Rd	Increment	$Rd \leftarrow Rd + 1$	Z,N,V	1
DEC	Rd	Decrement	$Rd \leftarrow Rd - 1$	Z,N,V	1
TST	Rd	Test for Zero or Minus	$Rd \leftarrow Rd \bullet Rd$	Z,N,V	1
CLR	Rd	Clear Register	$Rd \leftarrow Rd \oplus Rd$	Z,N,V	1
SER	Rd	Set Register	$Rd \leftarrow \$FF$	None	1
MUL	Rd, Rr	Multiply Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
MULS	Rd, Rr	Multiply Signed	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	$R1:R0 \leftarrow (Rd \times Rr) << 1$	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	$R1:R0 \leftarrow (Rd \times Rr) << 1$	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	$R1:R0 \leftarrow (Rd \times Rr) << 1$	Z,C	2
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	$PC \leftarrow PC + k + 1$	None	2
IJMP		Indirect Jump to (Z)	$PC \leftarrow Z$	None	2
JMP	k	Direct Jump	$PC \leftarrow k$	None	3
RCALL	k	Relative Subroutine Call	$PC \leftarrow PC + k + 1$	None	3
ICALL		Indirect Call to (Z)	$PC \leftarrow Z$	None	3
CALL	k	Direct Subroutine Call	$PC \leftarrow k$	None	4
RET		Subroutine Return	$PC \leftarrow \text{Stack}$	None	4
RETI		Interrupt Return	$PC \leftarrow \text{Stack}$	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	$\text{if } (Rd = Rr) PC \leftarrow PC + 2 \text{ or } 3$	None	1 / 2 / 3
CP	Rd,Rr	Compare	$Rd - Rr$	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	$Rd - Rr - C$	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	$Rd - K$	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	$\text{if } (Rr(b)=0) PC \leftarrow PC + 2 \text{ or } 3$	None	1 / 2 / 3
SBRs	Rr, b	Skip if Bit in Register is Set	$\text{if } (Rr(b)=1) PC \leftarrow PC + 2 \text{ or } 3$	None	1 / 2 / 3
SBIC	P, b	Skip if Bit in I/O Register Cleared	$\text{if } (P(b)=0) PC \leftarrow PC + 2 \text{ or } 3$	None	1 / 2 / 3
SBIS	P, b	Skip if Bit in I/O Register is Set	$\text{if } (P(b)=1) PC \leftarrow PC + 2 \text{ or } 3$	None	1 / 2 / 3
BRBS	s, k	Branch if Status Flag Set	$\text{if } (SREG(s) = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1 / 2
BRBC	s, k	Branch if Status Flag Cleared	$\text{if } (SREG(s) = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1 / 2
BREQ	k	Branch if Equal	$\text{if } (Z = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1 / 2
BRNE	k	Branch if Not Equal	$\text{if } (Z = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1 / 2
BRCS	k	Branch if Carry Set	$\text{if } (C = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1 / 2
BRCC	k	Branch if Carry Cleared	$\text{if } (C = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1 / 2
BRSH	k	Branch if Same or Higher	$\text{if } (C = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1 / 2
BRLO	k	Branch if Lower	$\text{if } (C = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1 / 2
BRMI	k	Branch if Minus	$\text{if } (N = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1 / 2
BRPL	k	Branch if Plus	$\text{if } (N = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1 / 2
BRGE	k	Branch if Greater or Equal, Signed	$\text{if } (N \oplus V = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1 / 2
BRLT	k	Branch if Less Than Zero, Signed	$\text{if } (N \oplus V = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1 / 2
BRHS	k	Branch if Half Carry Flag Set	$\text{if } (H = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1 / 2
BRHC	k	Branch if Half Carry Flag Cleared	$\text{if } (H = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1 / 2
BRts	k	Branch if T Flag Set	$\text{if } (T = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1 / 2
BRTc	k	Branch if T Flag Cleared	$\text{if } (T = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1 / 2
BRVS	k	Branch if Overflow Flag is Set	$\text{if } (V = 1) \text{ then } PC \leftarrow PC + k + 1$	None	1 / 2
BRVC	k	Branch if Overflow Flag is Cleared	$\text{if } (V = 0) \text{ then } PC \leftarrow PC + k + 1$	None	1 / 2

Mnemonics	Operands	Description	Operation	Flags	#Clocks
BRIE	k	Branch if Interrupt Enabled	If (I = 1) then PC ← PC + k + 1	None	1 / 2
BRID	k	Branch if Interrupt Disabled	If (I = 0) then PC ← PC + k + 1	None	1 / 2
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd, Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q, Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q, Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z+1	None	3
SPM		Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	Stack ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← Stack	None	2
BIT AND BIT-TEST INSTRUCTIONS					
SBI	P, b	Set Bit in I/O Register	I/O(P, b) ← 1	None	2
CBI	P, b	Clear Bit in I/O Register	I/O(P, b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z, C, N, V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z, C, N, V	1
ROL	Rd	Rotate Left Through Carry	Rd(0) ← C, Rd(n+1) ← Rd(n), C ← Rd(7)	Z, C, N, V	1
ROR	Rd	Rotate Right Through Carry	Rd(7) ← C, Rd(n) ← Rd(n+1), C ← Rd(0)	Z, C, N, V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z, C, N, V	1
SWAP	Rd	Swap Nibbles	Rd(3..0) ← Rd(7..4), Rd(7..4) ← Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow.	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
MCU CONTROL INSTRUCTIONS					
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For On-Chip Debug Only	None	N/A