

ÉPREUVE ÉCRITE

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: juin 2008

DATE: 22/05/08

DURÉE: 3 h

1. Unterprogramme (4 Punkte)

- a) Welche Initialisierung muss zwingend vorgenommen werden, damit ein Unterprogrammaufruf funktionieren kann? Erkläre kurz warum dies nötig ist. (2)
- b) Erkläre den Unterschied zwischen lokalen und globalen Variablen bei der Programmierung mit Assembler. (2)

2. Interrupts (8 Punkte)

- a) Im Register **GIFR** sollen Bit 6 und Bit 7 gelöscht werden, und Bit 5 soll gesetzt werden, ohne dass die anderen Bits verändert werden. Notiere den entsprechenden Assemblercode. (2)
- b) Welche drei Bedingungen müssen erfüllt sein, damit der Controller auf Unterbrechungen reagieren kann? (3)
- c) Erkläre ausführlich was passiert, wenn während dem Abarbeiten einer Interrupt-Routine erneut ein Interrupt erfolgt. (3)

3. Entprellung (6 Punkte)

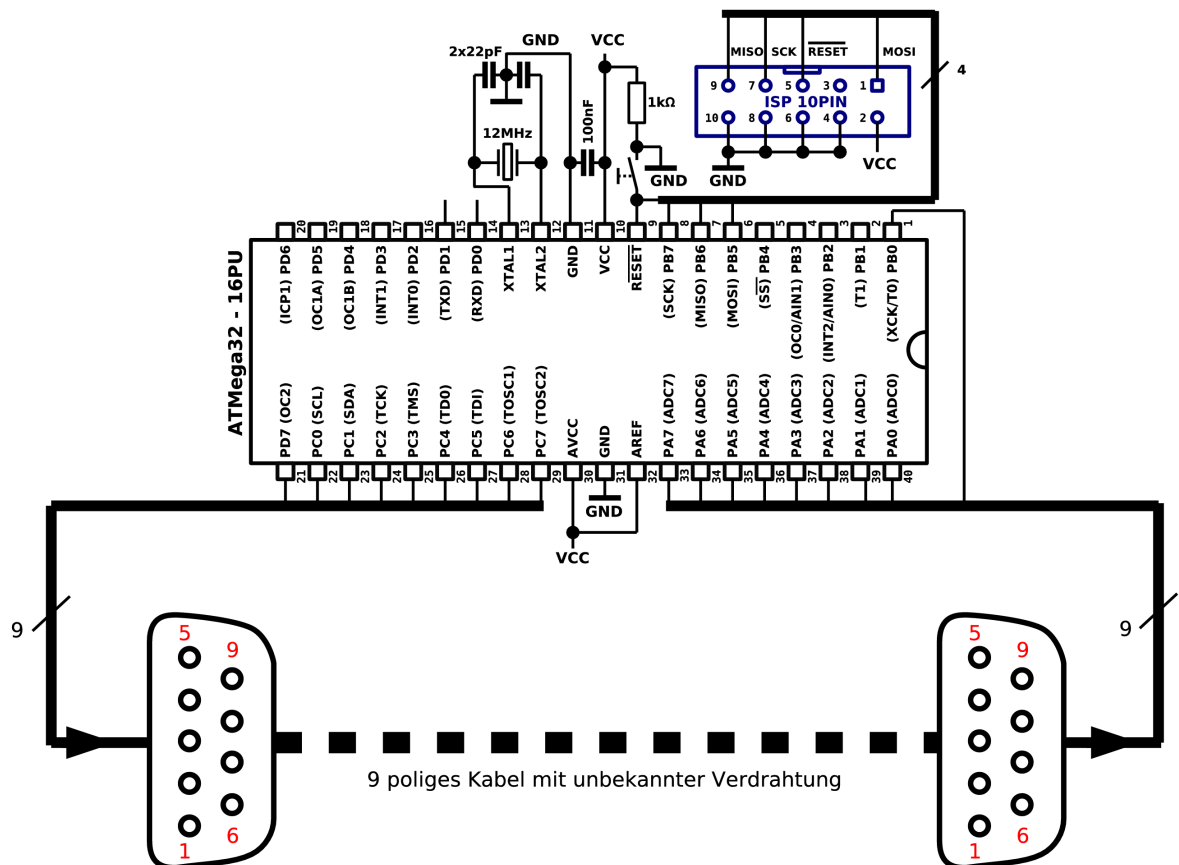
- a) Erkläre kurz anhand einer Skizze wieso Schalter entprellt werden müssen. (2)
- b) Erkläre ausführlich die softwaremäßige Entprellung mit einem Flussdiagramm (inklusive Text). (4)

4. Serielle Schnittstelle (7 Punkte)

- a) Zeichne die Zeitdiagramme (Zeichenrahmen) bei der seriellen Übertragung des Buchstaben 'g' (702) einmal am Pin PD0 des Mikrocontrollers und einmal am Pin 3 (TxD) der seriellen Schnittstelle. (4)
- b) Schreibe ein kleines Unterprogramm mit dem symbolischen Namen **RCVCHR** zum Empfang eines Zeichens mittels Polling.
Die Übergabevariable heißt Char (Register 20 wurde mit der Define-Anweisung im Hauptprogramm zugewiesen: **.DEF Char = r20**). Es soll keine Fehlerkontrolle erfolgen. (3)

5. Digitale Ein- und Ausgabe (19 Punkte)

Mit Hilfe eines ATmega32 soll ein Gerät konstruiert werden, das 9-polige serielle Kabel durchmessen und testen kann. Das Kabel wird, wie in der folgenden Skizze dargestellt mit dem Controller verbunden.



Ausgänge am Controller: PORTC0-PORTC7 und PORTD7

PORTC0 wird mit Pin 1 des seriellen Kabels verbunden, PORTC1 mit Pin2, PORTC2 mit Pin3 usw.. PORTD7 wird mit Pin 9 verbunden.

Eingänge am Controller: PORTA0-PORTA7 und PORTB0).

PORTA0 wird mit Pin 1 des seriellen Kabels verbunden, PORTA1 mit Pin2, PORTA2 mit Pin3 usw.. PORTB0 wird mit Pin 9 verbunden.

Die Pull-Up-Widerstände müssen für alle Eingänge gesetzt werden!

Funktionsweise: Jede einzelne Leitung wird am Kabelanfang nacheinander unter Spannung gesetzt. Die jeweiligen Zustände an den neun Kabelausgänge werden erfasst und abgespeichert.

Der Controller legt zuerst 5 Volt an Pin 1 des seriellen Kabels (PORTC0 = 1, PORTC1-7 = 0, PORTD7 = 0) und liest dann alle 9 Eingänge ein. Die Zustände der 9 Eingänge (9 Bit) werden in einer Tabelle im SRAM ab der Adresse **0x0400** nach folgendem Muster abgespeichert:

0x0411	Neuntes Wort (HByte)	TESTTAB
0x0410	Neuntes Wort (LByte)	
	
	
0x0401	Erstes Wort (HByte)	
0x0400	Erstes Wort (LByte)	

Zwischen der Aktivierung mit 5V und dem Abspeichern soll 1ms gewartet werden (ein fertiges Unterprogramm "W1ms" darf verwendet werden).

Da 9 Bit gespeichert werden müssen, wird ein ganzes Wort (2 Byte) zum Abspeichern benötigt. Der Zustand der unteren 8 Pin (PORTA) wird im LByte (Low Byte) auf der Adresse **0x0400** abgespeichert (Pin 1 entspricht Bit 0).

Der Zustand von Pin 9 wird alleine an der untersten Stelle (LSB) des HByte (High Byte) auf der Adresse **0x0401** abgespeichert.

Dieser Teil des Programms (Einlesen von PORTB0 und Abspeichern im HByte) soll als Unterprogramm realisiert werden.

Als nächstes legt der Controller 5 Volt an Pin 2 des seriellen Kabels (PORTC1 = 1, alle anderen Ausgänge = 0) und liest wieder alle 9 Eingänge ein. Dies wird insgesamt neun mal wiederholt bis alle Leitungen aktiviert waren.

- Notiere die Assembleranweisungen zur Speicherreservierung im SRAM. (1)
- Notiere die Assemblerzeilen zur Initialisierung der Ein- und Ausgänge und der SRAM-Tabelle. (7)
- Zeichne die Flussdiagramme zum Programm. Die Aktivierung der unteren 8 Pins soll in einer Schleife mit Hilfe eines Schiebefehls durchgeführt werden. (11)

6.Arbeiten mit Tabellen und Ausgabe über die serielle Schnittstelle (16 Punkte)

Das Programm aus der vorigen Aufgabe soll erweitert werden. Dazu soll die Tabelle die sich im SRAM ab **0x0400** (**TESTTAB**, 18 Werte) befindet mit der folgenden ersten Tabelle im Flash Speicher (**TABNM**) verglichen werden. Ist die SRAM-Tabelle mit der Flash-Tabelle identisch, so soll über die serielle Schnittstelle der Text der zweiten Flash-Tabelle (**TXTTAB**) ausgegeben werden (38400bit/s, 8N1). Der Controller ist mit 16MHz getaktet. Es soll ein Polling mit **UDRE** verwendet werden. Die Texttabelle schließt mit einem Nullbyte ab.

```
;-----  
;      Tabellen im Programmspeicher (Flash)  
;-----  
  
.ORG      0x1000                      ;Tabelle ab Adresse 0x1000  
TABNM:    .DB      0x01,0x00,0x02,0x00,0x04,0x00,0x08,0x00,0x10,0x00  
          .DB      0x20,0x00,0x40,0x00,0x80,0x00,0x00,0x01  
  
.ORG      0x1100                      ;Tabelle ab Adresse 0x1100  
TXTTAB:   .DB      "Vollstaendiges 1:1 Kabel!",10,13,0
```

- a) Notiere die Assemblerzeilen zur Initialisierung der seriellen Schnittstelle. (5)
- b) Schreibe das kommentierte Assemblerprogramm zum Vergleich der Tabellen und zur Ausgabe des Textes. (11)

A1 Befehlssatz

Instruction Set Summary

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	Rdi, K	Add Immediate to Word	$Rdh:Rdi \leftarrow Rdh:Rdi + K$	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	Rdi, K	Subtract Immediate from Word	$Rdh:Rdi \leftarrow Rdh:Rdi - K$	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	$Rd \leftarrow Rd \cdot Rr$	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	$Rd \leftarrow Rd \cdot K$	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	$Rd \leftarrow Rd \cdot (\$FF - K)$	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	$Rd \leftarrow Rd \cdot Rd$	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	$R1:R0 \leftarrow (Rd \times Rr) \lll 1$	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	$R1:R0 \leftarrow (Rd \times Rr) \lll 1$	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	$R1:R0 \leftarrow (Rd \times Rr) \lll 1$	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	if $(Rd = Rr)$ $PC \leftarrow PC + 2$ or 3	None	1 / 2 / 3
CP	Rd, Rr	Compare	$Rd - Rr$	Z, N, V, C, H	1
CPC	Rd, Rr	Compare with Carry	$Rd - Rr - C$	Z, N, V, C, H	1
CPI	Rd, K	Compare Register with Immediate	$Rd - K$	Z, N, V, C, H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if $(Rr(b)=0)$ $PC \leftarrow PC + 2$ or 3	None	1 / 2 / 3
SBRSC	Rr, b	Skip if Bit in Register is Set	if $(Rr(b)=1)$ $PC \leftarrow PC + 2$ or 3	None	1 / 2 / 3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if $(P(b)=0)$ $PC \leftarrow PC + 2$ or 3	None	1 / 2 / 3
SBIS	P, b	Skip if Bit in I/O Register is Set	if $(P(b)=1)$ $PC \leftarrow PC + 2$ or 3	None	1 / 2 / 3
BRBS	s, k	Branch if Status Flag Set	if $(SREG(s) = 1)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRBC	s, k	Branch if Status Flag Cleared	if $(SREG(s) = 0)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BREQ	k	Branch if Equal	if $(Z = 1)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRNE	k	Branch if Not Equal	if $(Z = 0)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRCS	k	Branch if Carry Set	if $(C = 1)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRCC	k	Branch if Carry Cleared	if $(C = 0)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRSH	k	Branch if Same or Higher	if $(C = 0)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRLO	k	Branch if Lower	if $(C = 1)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRMI	k	Branch if Minus	if $(N = 1)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRPL	k	Branch if Plus	if $(N = 0)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRGE	k	Branch if Greater or Equal, Signed	if $(N \oplus V = 0)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRLT	k	Branch if Less Than Zero, Signed	if $(N \oplus V = 1)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRHS	k	Branch if Half Carry Flag Set	if $(H = 1)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRHC	k	Branch if Half Carry Flag Cleared	if $(H = 0)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRTS	k	Branch if T Flag Set	if $(T = 1)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRTC	k	Branch if T Flag Cleared	if $(T = 0)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRVS	k	Branch if Overflow Flag is Set	if $(V = 1)$ then $PC \leftarrow PC + k + 1$	None	1 / 2
BRVC	k	Branch if Overflow Flag is Cleared	if $(V = 0)$ 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

Mnemonics	Operands	Description	Operation	Flags	#Clocks
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

A2 SF-Register

Register Summary

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	RWWSB	—	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)	SFIOR	ADTS2	ADTS1	ADTS0	—	ACME	PUD	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)	WDTCR	—	—	—	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)	EECR	—	—	—	—	EERIE	EEMWE	EEWE	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	ddb7	ddb6	ddb5	ddb4	ddb3	ddb2	ddb1	ddb0	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

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
\$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 debugger 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.

UCSRA = USART Control and Status Register A

Bit	7	6	5	4	3	2	1	0
UCSRA 0x0B	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM
Startwert	0	0	1	0	0	0	0	0
Read/Write	R	R/W	R	R	R	R	R/W	R/W

RXC *USART Receive Complete*

- 0** Wenn sich keine Daten mehr im Empfangspuffer befinden oder wenn der Empfänger ausgeschaltet ist (Startwert).
- 1** Wird auf Eins gesetzt, wenn sich Daten im Empfangspuffer befinden. Zeigt also an, dass die Daten ausgelesen werden können.
Kann zusätzlich einen Interrupt auslösen (siehe **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 wurden und keine neuen Daten im **UDR**-Datenregister (Sendepuffer) vorliegen. Muss manuell mit einer Eins! vor der Ausgabe gelöscht werden!
Kann zusätzlich einen Interrupt auslösen (siehe **UCSRB**).

UDRE *USART Data Register Empty*

- 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 (siehe **UCSRB**).

FE *Frame Error*

- 0** kein Rahmenfehler.
- 1** **Rahmenfehler**, kein gültiges Stoppbit 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
UCSRB 0x0A	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
Startwert	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W

RXCIE *RXC Interrupt Enable*

- 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 "**sei**").

0 kein **RXC**-Interrupt erlaubt.

TXCIE **TXC Interrupt Enable**

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 (**I**=1 im **SREG** mit

"**sei**")

0 kein **TXC**-Interrupt erlaubt.

UDRIE **UDRE Interrupt Enable**

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 (**I**=1 im **SREG** mit "**sei**")

0 kein **TXC**-Interrupt erlaubt.

RXEN **Receiver ENable**

1 **schaltet den Empfänger ein.** Pin **RxD (PORTD0)** ist dann als Eingang reserviert (muss nicht extra initialisiert werden) und ist für andere Aktionen nicht mehr zugänglich.

0 schaltet den Empfänger aus.

TXEN **Transmitter ENable**

1 **schaltet den Sender ein.** Pin **TxD (PORTD1)** ist als Ausgang reserviert (muss nicht extra initialisiert werden) und ist für andere Aktionen nicht mehr zugänglich.

0 schaltet den Sender aus.

UCSZ2 **USART Character Size 2**

siehe nächstes Register **UCSRC**

UCSRC = USART Control and Status Register C

Bit	7	6	5	4	3	2	1	0
UCSRC 0x20	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCP0L
Startwert	1	0	0	0	0	1	1	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

URSEL **USART Register SElect**

Der Speicherplatz ist mit zwei Registern belegt.

1 das Register **UCSRC** wird ausgewählt (default). Beim Lesen von **UCSRC** ist das Bit Eins.

0 das Register **UBRRH** wird ausgewählt. Beim Lesen von **UBRRH** ist das Bit Null.

UMSEL **USART Mode SElect**

1 **Synchroner Modus**

0 **Asynchroner Modus**

UPM **USART Parity Mode** **UPM1, UPM0**

Ist die Parität eingeschaltet (gerade oder ungerade Parität) wird hardwaremäßig die Parität generiert und in den Zeichenrahmen mit integriert. Als Empfänger kontrolliert der USART die Parität. Ist ein Fehler aufgetreten, so wird das **PE**-Flag (Parity Error) in **UCSRA** gesetzt.

- 00** keine Parität
- 01** reserviert
- 10** gerade Parität eingeschaltet
- 11** ungerade Parität eingeschaltet.

USBS **USART Stop Bit Select**

- 0** 1 Stoppbit
- 1** 2 Stoppbit

UCSZ **USART Character Size UCSZ2, UCSZ1, UCSZ0**

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

210 (UCSZ)

- 000** 5 Bit
- 001** 6 Bit
- 010** 7 Bit
- 011** 8 Bit
- 100** reserviert
- 101** reserviert
- 110** reserviert
- 111** 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
UBRRH 0x20	URSEL	-	-	-	UBRR 11	UBRR 10	UBRR 9	UBRR 8
Startwert	0	0	0	0	0	0	0	0
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W

UBRRL = USART Baud Rate Register Low

Bit	7	6	5	4	3	2	1	0
UBRRL 0x09	UBRR 7	UBRR 6	UBRR 5	UBRR 4	UBRR 3	UBRR 2	UBRR 1	UBRR 0
Startwert	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

URSEL **USART Register SElect**

Der Speicherplatz ist mit zwei Registern belegt.

- 1** das Register **UCSRC** wird **ausgewählt** (Startwert). Beim Lesen von **UCSRC** ist das Bit

Eins.

- 0** das Register **UBRRH** wird **ausgewählt**. Beim Lesen von **UBRRH** ist das Bit Null.

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 Formel 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:

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

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 oder durch Verdoppeln der Baudrate mit **U2X** in **UCSRA** (siehe Datenblatt).

UDR = USART I/O Data Register

Bit	7	6	5	4	3	2	1	0
UDR r/w 0x0C	RXB7 TXB7	RXB6 TXB6	RXB5 TXB5	RXB4 TXB4	RXB3 TXB3	RXB2 TXB2	RXB1 TXB1	RXB0 TXB0
Startwert	0	0	0	0	0	0	0	0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TXB **Transmit Data Buffer Register**
Sendepuffer.

RXB **Receive Data Buffer Register**
Empfangspuffer.

A3 ASCII-Tabelle

Dec.	Oct.	Hex	Binary	Value		Dec.	Oct.	Hex	Binary	Value
000	000	000	00000000	NUL	(Null char.)	064	100	040	01000000	@
001	001	001	00000001	SOH	(Start of Header)	065	101	041	01000001	A
002	002	002	00000010	STX	(Start of Text)	066	102	042	01000010	B
003	003	003	00000011	ETX	(End of Text)	067	103	043	01000011	C
004	004	004	00000100	EOT	(End of Transmission)	068	104	044	01000100	D
005	005	005	00000101	ENQ	(Enquiry)	069	105	045	01000101	E
006	006	006	00000110	ACK	(Acknowledgment)	070	106	046	01000110	F
007	007	007	00000111	BEL	(Bell)	071	107	047	01000111	G
008	010	008	00001000	BS	(Backspace)	072	110	048	01001000	H
009	011	009	00001001	HT	(Horizontal Tab)	073	111	049	01001001	I
010	012	00A	00001010	LF	(Line Feed)	074	112	04A	01001010	J
011	013	00B	00001011	VT	(Vertical Tab)	075	113	04B	01001011	K
012	014	00C	00001100	FF	(Form Feed)	076	114	04C	01001100	L
013	015	00D	00001101	CR	(Carriage Return)	077	115	04D	01001101	M
014	016	00E	00001110	SO	(Shift Out)	078	116	04E	01001110	N
015	017	00F	00001111	SI	(Shift In)	079	117	04F	01001111	O
016	020	010	00010000	DLE	(Data Link Escape)	080	120	050	01010000	P
017	021	011	00010001	DC1	(XON) (Device Control 1)	081	121	051	01010001	Q
018	022	012	00010010	DC2	(Device Control 2)	082	122	052	01010010	R
019	023	013	00010011	DC3	(XOFF) (Device Control 3)	083	123	053	01010011	S
020	024	014	00010100	DC4	(Device Control 4)	084	124	054	01010100	T
021	025	015	00010101	NAK	(Negative Acknowledgement)	085	125	055	01010101	U
022	026	016	00010110	SYN	(Synchronous Idle)	086	126	056	01010110	V
023	027	017	00010111	ETB	(End of Trans. Block)	087	127	057	01010111	W
024	030	018	00011000	CAN	(Cancel)	088	130	058	01011000	X
025	031	019	00011001	EM	(End of Medium)	089	131	059	01011001	Y
026	032	01A	00011010	SUB	(Substitute)	090	132	05A	01011010	Z
027	033	01B	00011011	ESC	(Escape)	091	133	05B	01011011	[
028	034	01C	00011100	FS	(File Separator)	092	134	05C	01011100	\
029	035	01D	00011101	GS	(Group Separator)	093	135	05D	01011101]
030	036	01E	00011110	RS	(Req to Send) (Rec Sep)	094	136	05E	01011110	^
031	037	01F	00011111	US	(Unit Separator)	095	137	05F	01011111	_
032	040	020	00100000	SP	(Space)	096	140	060	01100000	`
033	041	021	00100001	!		097	141	061	01100001	a
034	042	022	00100010	"		098	142	062	01100010	b
035	043	023	00100011	#		099	143	063	01100011	c
036	044	024	00100100	\$		100	144	064	01100100	d
037	045	025	00100101	%		101	145	065	01100101	e
038	046	026	00100110	&		102	146	066	01100110	f
039	047	027	00100111	'		103	147	067	01100111	g
040	050	028	00101000	(104	150	068	01101000	h
041	051	029	00101001)		105	151	069	01101001	i
042	052	02A	00101010	*		106	152	06A	01101010	j
043	053	02B	00101011	+		107	153	06B	01101011	k
044	054	02C	00101100	,		108	154	06C	01101100	l
045	055	02D	00101101	-		109	155	06D	01101101	m
046	056	02E	00101110	.		110	156	06E	01101110	n
047	057	02F	00101111	/		111	157	06F	01101111	o
048	060	030	00110000	0		112	160	070	01110000	p
049	061	031	00110001	1		113	161	071	01110001	q
050	062	032	00110010	2		114	162	072	01110010	r
051	063	033	00110011	3		115	163	073	01110011	s
052	064	034	00110100	4		116	164	074	01110100	t
053	065	035	00110101	5		117	165	075	01110101	u
054	066	036	00110110	6		118	166	076	01110110	v
055	067	037	00110111	7		119	167	077	01110111	w
056	070	038	00111000	8		120	170	078	01111000	x
057	071	039	00111001	9		121	171	079	01111001	y
058	072	03A	00111010	:		122	172	07A	01111010	z
059	073	03B	00111011	;		123	173	07B	01111011	{
060	074	03C	00111100	<		124	174	07C	01111100	
061	075	03D	00111101	=		125	175	07D	01111101	}
062	076	03E	00111110	>		126	176	07E	01111110	~
063	077	03F	00111111	?		127	177	07F	01111111	Nul