Mikrocontrollertechnik MC
SS 2012
Praktikumsaufgaben

Stand: 8. 5. 12

Versuch 1

Einführung in die C-Programmierung Atmel AVR.

1. Kennenlernen des grundsätzlichen Entwicklungsganges.
2. Darstellen von Zeit mittels Software.
3. Elementare Anwendungsprogrammierung.

Grundlagen:

- AVR Studio 4,
- GNU C-Compiler,
- Starterkit AVR STK 500.

Zum Starterkit siehe zusätzliche Kurzbeschreibung oder die Originaldokumentation der Fa. Atmel.

Grundkonfiguration: Port C mit den LEDs verbinden. Alle Ports auf Ausgang.

Aufgabe 1: Zeitdarstellung (1). Auf Port A abwechselnd 00H und FFH ausgeben. Kontrolle: mittels Oszilloskop. Wie lange dauert die kürzeste E-A-Schleife? (Im Versuch 2 werden wir die gleiche Aufgabe mittels Assemblerprogrammierung lösen.)

Aufgabe 2: Zeitdarstellung (2). Eine Funktion MILLISEC, die n Millisekunden darstellt, wobei der Parameter n eine 16-Bit-Binärzahl ist (hierzu brauchen wir zunächst eine Schleife die eine einzige Millisekunde darstellt). Funktionskontrolle wie bei Aufgabe 1.

Aufgabe 3: Lauflicht (1). Zyklische aufeinanderfolgende Erregung der LEDs in eine Richtung.

Aufgabe 4: Lauflicht (2). Zyklische aufeinanderfolgende Erregung der LEDs abwechselnd in beiden Richtungen.

Aufgabe 5: elementare Ein- und Ausgabe (1). Port D mit den Tasten verbinden und auf Eingabe programmieren. Jede Betätigung einer Taste muß dazu führen, daß die entsprechende LED leuchtet.

Aufgabe 6: elementare Ein- und Ausgabe (2). Port D mit den Tasten verbinden und auf Eingabe programmieren. Jede Betätigung einer Taste muß dazu führen, daß die entsprechende LED blinkt.

Aufgabe 7: Gebäudesystemtechnik (1). Ein- und Ausschalten des elektrischen Lichts. Mit einer der Tasten soll die gesamte LED-Reihe ein- und ausgeschaltet werden (erste Betätigung: ein, zweite Betätigung: aus usw.).

Aufgabe 8: Gebäudesystemtechnik (2). Jede LED soll durch Betätigen der jeweils benachbarten Taste ein- und ausgeschaltet werden, und zwar unabhängig von den anderen.

Aufgabe 9: Siebensegmentanzeigen. Wir verwenden die Siebenensegmentanzeigen 10a auf dem Pollin-Lehrgerät 10a (vgl. Digitaltechnik). Linke Anzeige an Port C, rechte Anzeige an Port B. Ports so umstellen (Richtungsregister), daß die Tasten abgefragt werden können.

a) Auf beiden Anzeigen gleichzeitig zyklisch modulo 10 zählen (0...9). Genug Zeit lassen, so daß der Zählvorhang visuell verfolgt werden kann.

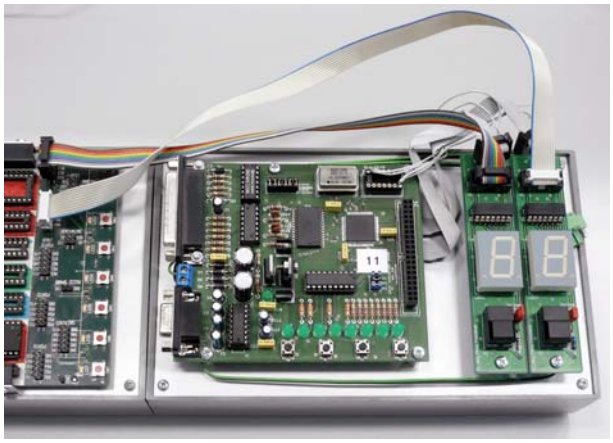
b) Zyklisch modulo 100 zählen (0...99).

c) Handzähler (zum Zählen von Autos, Schafen, BVB-Fans usw.). Rechte Taste zum Zählen (+ 1), linke Taste zum Löschen. Bei 99 mit Zählen aufhören.

d) Wie b),aber zyklsich von 1 bis 49 zählen (Funktionskontrolle für Lottozahlen).

e) Ziehen von Lottozahlen. So lange schnell zwischen 1 und 49 zählen, wie die rechte Taste niedergehalten wird.

7	6	5	4	3	2	1	0
KEY#	G#	F#	E#	D#	C#	B#	A#



Versuch 2

Einführung in die Assemblerprogrammierung Atmel AVR. Elementare Anwendungsprogrammierung. Wir bearbeiten nur Aufgaben, die keine Kenntnis von Schnittstellen, E-A-Einrichtungen usw. voraussetzen. (Das kommt in der Vorlesung und in Versuch 3.)

1. Darstellen von Zeit mittels Software.
2. Elementare Programmschleifen. Veranschaulichung mittels XY-Adapter und Oszilloskop.
3. Kurzausbildung Musik. Mit Software Sound herstellen.

Versuchsanordnung:

- Startertkit Atmel STK 500.
- Peripherie: XY-Adapter 09a, Universaladapter 10b, Lautsprecher.
- Programmiersprache: Assembler (Atmel Developer Studio).

Aufgabe 1: Zeitdarstellung (1). Auf Port A abwechselnd 00H und FFH ausgeben. Kontrolle: mittels Oszilloskop. Wie lange dauert die kürzeste E-A-Schleife?

Aufgabe 2: Zeitdarstellung (2). Nutzung des Unterprogramms MILLISEC, das n Millisekunden darstellt, wobei der Parameter n eine 16-Bit-Binärzahl ist. Funktionskontrolle wie bei Aufgabe 1.

Aufgabe 3: Lauflicht. Zyklische aufeinanderfolgende Erregung der LEDs in eine Richtung. LEDs an Port C. Her können wir die Programme der Übungsaufgaben übernehmen. Wir müssen aber die Zeitverzögerung einbauen (beispielsweise 200 ms).

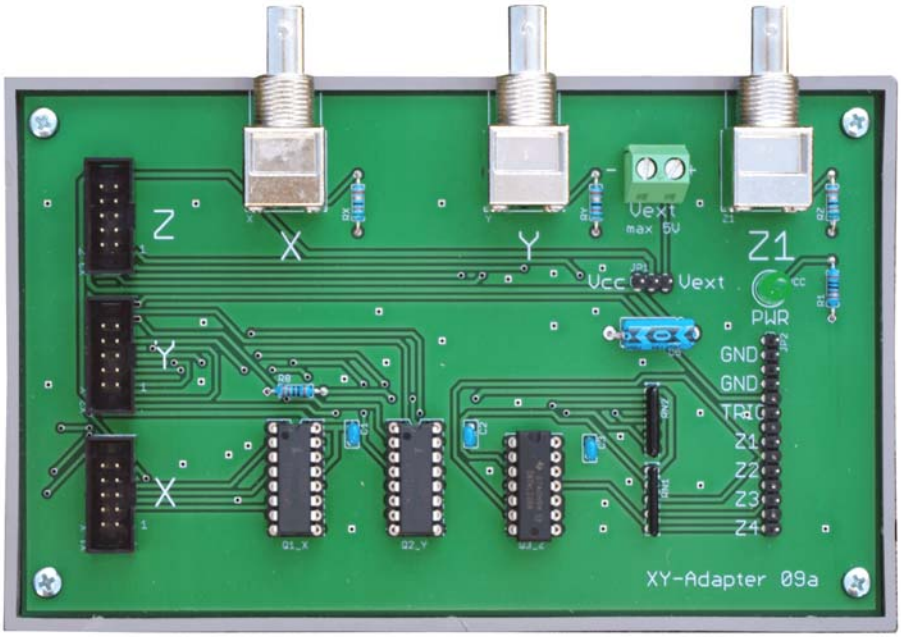
Aufgabe 4: Wir schließen den XY-Adapter an die Ports A und C an sowie die Tasten an Port D. Auf dem Oszilloskop soll ein Quadrat dargestellt werden.

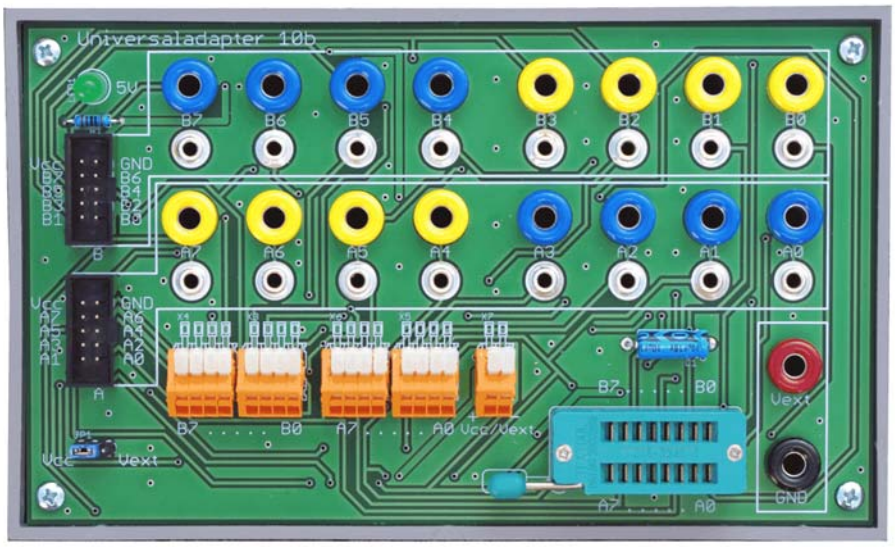
Aufgabe 5: mit zwei Tasten soll das Quadrat vergrößert oder verkleinert werden.

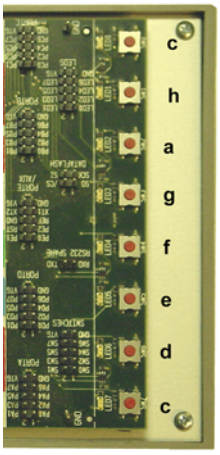
Aufgabe 6: mit zwei weiteren Tasten soll das Quadrat über den Bildschirm bewegt werden – und zwar einfach und schmucklos, ohne auf die Ränder zu achten. Wer will, kann auch noch den Bildrand darstellen.

Aufgabe 7. Kurzausbildung Musik / Musikinstrument. Der XY-Adapter wird abgebaut. Port A wird mit einem Port des Universaladapters verbunden. Der Lautsprecher wird an eine beliebige Bitposition von Port A sowie an VCC angeschlossen (Kabel mit Klinkenstecker).. Die Tasten werden zyklisch abgefragt. Eine betätigte Taste veranlaßt die Abgabe eines Tonsignals (aktiv Low; Port A schaltet zyklisch zwischen FFH und 00H um). Das Tonsignal ist eine symmetrische Rechteckwelle (Tastverhältnis 50:50) mit jeweils bestimmter Frequenz (gemäß chromatischer Tonleiter (s. nachstehend unter Kurzausbildung Musik; Frequenzangaben in Hz). Wenn kein Ton abgegeben wird, Ausgänge auf High (damit der angeschlossene Lautsprecher keinen Gleichstrom zieht).

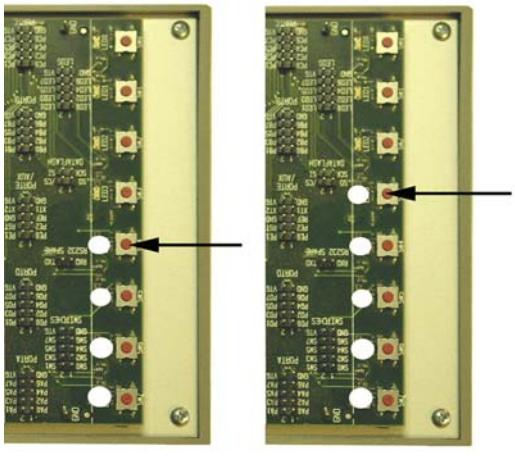
Aufgabe 8: Wenn noch Zeit ist: Richtige Noten spielen (Tonhöhe und Dauer) oder Morsezeichen ausgeben (nach Wahl).

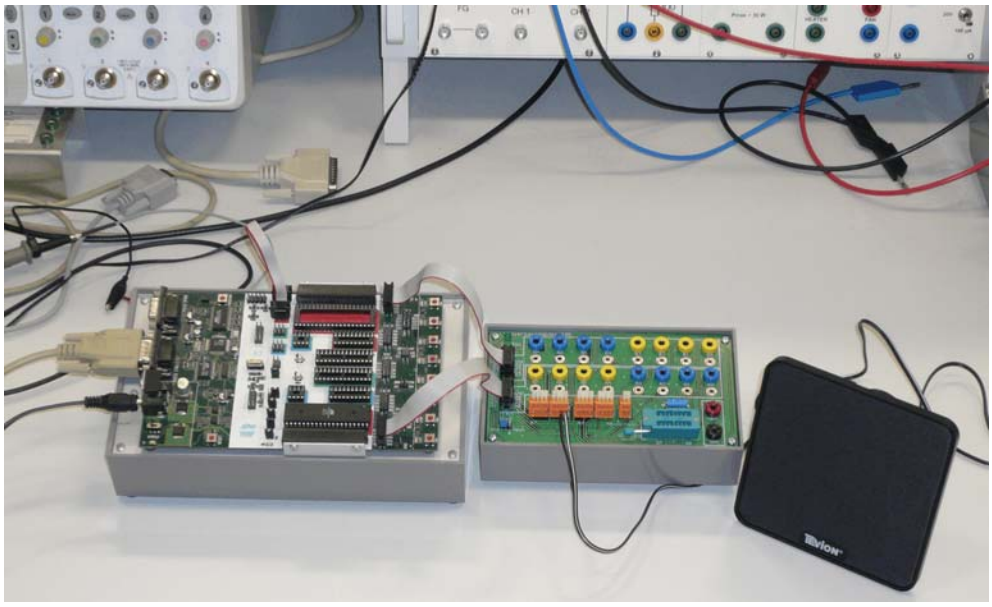






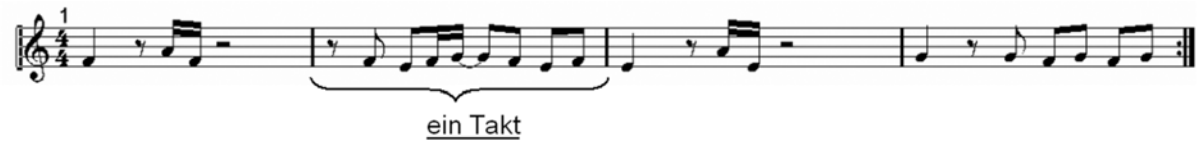
	f [Hz]	$t_{p/2}$ [µs]
c	525,25	952
h	493,88	1012
a	440	1136
g	392	1276
f	349,23	1432
e	329,63	1517
d	293,67	1703
c	261,63	1911





Kurzausbildung Musik

Musik ist nicht einfach nur mit Geräusch verbunden, sondern sie reizt zum gleichnishaften Anschauen der dionysischen Allgemeinheit (F. Nietzsche); sie ist "... darin von allen anderen Künsten verschieden, daß sie nicht Abbild der Erscheinung, oder richtiger, der adäquaten Objectität des Willens, sondern unmittelbar Abbild des Willens selbst ist und als zu allem Physischen der Welt das Metaphysische, zu aller Erscheinung das Ding an sich darstellt." (A. Schopenhauer). Wir fangen allerdings erst einmal ganz klein an ...



Eine Taktlänge entspricht 2s unterteilt in 4 Schläge á 500ms bei einem Tempo von 120 bmp

Noten					Pausen					Tempo: 120 bpm
𝅝	=	ganze Note	=	4/4 = 2000 ms	𝄻	=	ganze Pause	=	4/4 =	2000 ms
𝅗𝅥	=	halbe Note	=	2/4 = 1000 ms	𝄼	=	halbe Pause	=	2/4 =	1000 ms
♩	=	viertel Note	=	1/4 = 500 ms	𝄽	=	viertel Pause	=	4/4 =	500 ms
♪	=	achtel Note	=	1/8 = 250 ms	𝄾	=	achtel Pause	=	4/4 =	250 ms
𝅘𝅥𝅯	=	sechzehntel Note	=	1/16 = 125 ms	𝄿	=	sechzehntel Pause	=	4/4 =	125 ms

andere Zeichen:

= Wiederholungszeichen

Note	**Frequenz (Hz)**	**Halbperiode (µs)**	**$^{1}/_{16}$ Note (125 ms) entspricht ... Perioden**
c	525,25	952	66
h	493,88	1012	62
a	440	1136	55
g	392	1276	49
f	349,23	1432	44
e	329,63	1517	41
d	293,67	1703	37
c	261,63	1911	33

Versuch 3

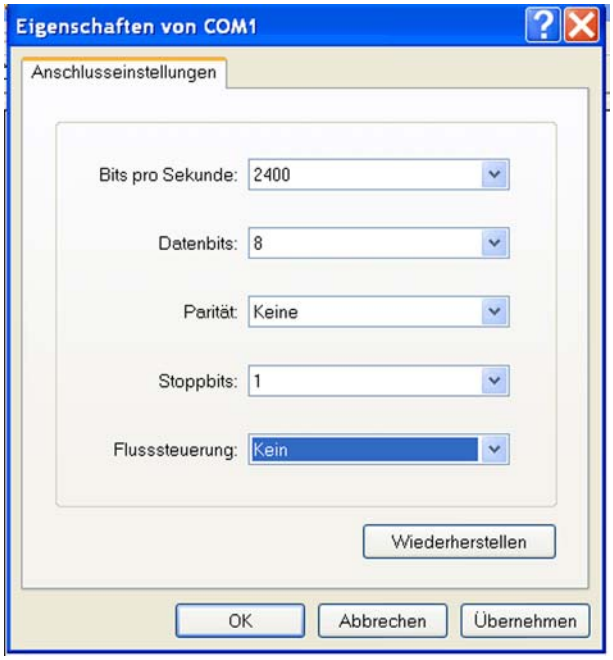
Schnittstellen und periphere Einrichtungen des Atmel AVR. Einführung in die Anwendungsprogrammierung unter einschluß von Interruptserviceroutinen (ISRs).

1. Inbetriebnahme der seriellen Schnittstelle mittels Hyperterminal.
2. Inbetriebnahme einer LCD-Punktmatrixanzeige.
3. Inbetriebnahme des A-D-Wandlers.

Versuchsanordnung:
- Startertkit Atmel STK 500 mit Atmel ATMega 16.
- Peripherie: PC mit Hyperterminal, LCD-Anzeige 09 mit Punktmatrix-LCD 2 * 16, Universaladapter 10b.
- Programmiersprache: Assembler (Atmel Developer Studio).

Aufgabe 1: Inbetriebnahme der seriellen Schnittstelle mittels Hyperterminal
Die zweite serielle Schnittstelle des Starterkits mit dem PC verbinden. Auf dem Starterkit die Bits 1 und 0 des Ports D mit der seriellen Schnittstelle verbinden (2adriges Kabel). Hyperterminal aufrufen.

Verbindungsdaten: 2400 Bits/s, 8 Datenbits, kein Paritätsbit, 1 Stopbit (8, N, 1), kein Handshaking (“Flusssteuerung”).



Versuchsschritte:

1. Die Schnittstelle im AVR initialisieren. Baudrateneinstellung = 103.

2. Ein einzelnes Zeichen ausgeben.

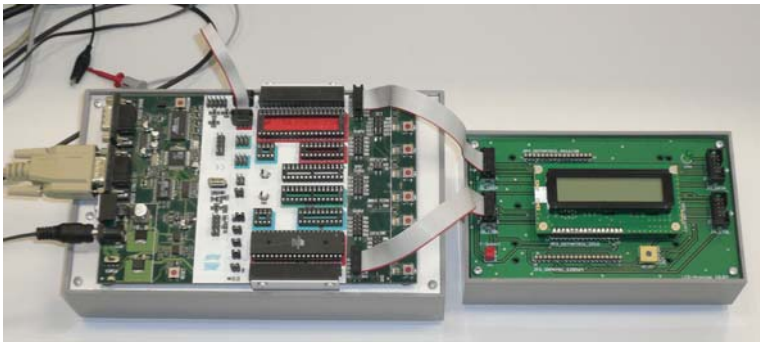
3. Einen Text (im Flash gespeicherte Zeichenkette) ausgeben.

4. Interruptserviceroutine zum Empfangen von Zeichen. Das jeweils empfangene Zeichen über Port C auf die LEDs des Starterkits ausgeben. Zuvor Port C mit den LEDs verbinden.

5. Die empfangenen Zeichen zum PC zurücksenden, so daß sie auf dem Bildschirm so erscheinen, als wären sie an Ort und Stelle eingetippt worden (Echofunktion).

6. Die Echofunktion auf die Unterstützungder Enter-Taste (CR = 13) und der Backspace-Taste (BS = 08) erweitern. Echo bei Enter: Rücklauf und neue Zeile (CR LF = 13 10), bei Backspace: Löschen des letzten Zeichens und Zurücksetzen des Cursors (BS SP BS = 08 32 08). Die Zeichenausgabe zum Port C entfernen.

Aufgabe 2: LCD-Punktmatrixanzeige
Die LCD-Anzeige 09 anschließen. Daten an Port C, Steuersignale an Port B. Einzelheiten s. Kurzausbildung LCD-Display.



Versuchsschritte:

1. LCD initialisieren. Wir arbeiten mit Zeitsteuerung (Schreiben und Ausführungszeit abwarten; kein Zurücklesen). Ein einzelnes Zeichen darstellen (Probe dafür, daß die Initialisierung geklappt hat).

2. Einen einzeiligen Festtext darstellen.

3. Einen zweizeiligen Festtext darstellen.

4. Die Anzeige zyklisch umlaufen lassen (Verschiebung).

5. Die Interruptserviceroutine (ISR) der seriellen Schnittstelle so erweitern, daß die empfangenen Zeichen auf der LCD-Anzeige dargestellt werden (einfachste Funktion; noch ohne Schönheiten).

6. Die ISR auf einen selbsttätigen Wrap Around erweitern (Übergang in die zweite Zeile, wenn Ende der ersten Zeile erreicht, Übergang an den Anfang, wenn Ende der 2. Zeile erreicht). Falls noch Zeit ist: die Tasten Backspace und Enter vernünftig auswerten (Rückschritt, Anfang der neuen Zeile), so daß sich die Anzeige wie ein kleines Terminal verhält.

Aufgabe 3: A-D-Wandler
LCD-Anzeige abbauen. Port A an den Universaladapter 10b anschließen. Die LEDs des Starterkits mit Port C verbinden. Voltcraft-Netzgerät auf 0 V drehen. Minuspol an Masse, Pluspol an Port A0. Digitalvoltmeter anschließen.

Grundsätzlich: Nicht über 5 V gehen!!!

Nachsehen (AVR-Programmer), ob die Referenzspannung auf 4 V (genau: 4,096 V) steht. Ggf. nachstellen.

1. A-D-Wandler initialisieren. Wir arbeiten im Abfragebetrieb und nutzen nur die höherwertigen 8 Bits.

2. Die Spannung an Port A0 zyklisch wandeln. Den binären Wert auf die LEDs ausgeben. Am Voltcraft-Netzteil die Spannung von 0 bis auf max. 5 V hochdrehen. An markanten Punkten kontrollieren (Digitalvoltmeter). Wenn die Referenzspannung 4 V ist, welche Bitmuster müssen dann bei 4 V, 2 V, 1 V, 0,5 V erscheinen? Ggf. den Windows-Rechner benutzen, um Dezimal und Binär ineinander umzurechnen.

3. Die binären Werte ins Hexadezimale wandeln und an das Hyperterminal ausgeben. Nach jeder 2stelligen Hexadezimalzahl ein Leerzeichen (32). Zyklsich wandlen und nur dann ausgeben,w enn sich die Anzeige geändert hat. Besonders Ehrgeizige können sich ja an einer Wandlung ins Dezimale versuchen...

Zusatzaufgabe: Zähler/Zeitgeber. Wir verwenden den Counter/Timer 0 und probieren verschiedene Betriebsarten aus:

a) Rechteckwellenerzeugung mit Compare Output Mode.

b) Pulsweitenmodulation im einafchen Modus (Fast PWM Mode).

c) Pulsweitenmodulation im phasenrichtigen PWM-Modus.

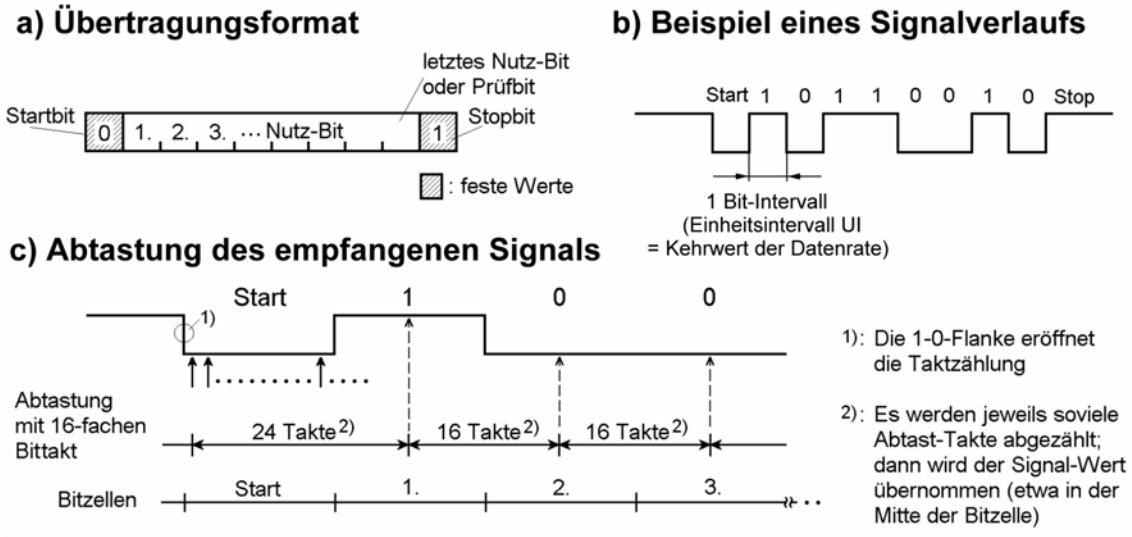
Ausgabe über Pin OC1 = Port B3. Richtungsregister (DDR) auf Ausgabe stellen!

Für die Zählweite definieren wir ein Register, das anfänglich auf einen mittleren Wert (128) geladen wird. Um die Einstellung zu verändern, nutzen wir das Hyperterminal über die serielle Schnittstelle und die Interruptserviceroutine. Z. B. Zeichen 0 = aus (Wert 0) , + = beschleunigen, – = verringern.

Weitere Demonstrationen: (1) Drehzahlsteuerung, (2) Helligkeitssteuerung der LEDs, (3) Nutzung als D-A-Wandler.

Kurzausbildung serielle Schnittstelle

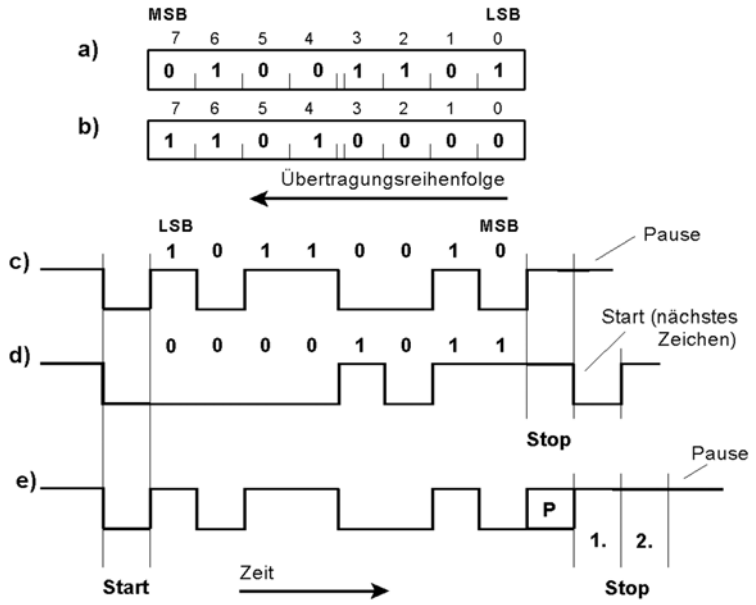
Die serielle Schnittstelle ist ein bitserielles asynchrones Interface. Je Richtung ist eine Datenleitung vorgesehen. Es werden einzelne Zeichen übertragen (Start-Stop-Verfahren). Der Übertragung liegt ein festes Zeitraster (Übertragungsrate) zugrunde. Die Zeichen werden mit zusätzlichen Start- und Stopbits voneinander abgegrenzt. Erkennt der Empfänger ein Startbit, so beginnt er, den ankommenden Datenstrom mit seinem Takt abzutasten. Das Abtasten endet mit dem Empfang des (bzw. der) Stopbits. Danach wartet der Empfänger auf das nächste Startbit. Codierung der Bitfolgen: NRZ. Übertragungsreihenfolge: von der niedrigstwertigen zur höchstwertigen Bitposition (LSB => MSB).



Der Ruhezustand wird durch einen Eins-Pegel signalisiert. Die Übertragung eines Zeichens beginnt mit einem Nullbit (Startbit). Der erste Eins-Null-Übergang – aus dem Ruhezustand heraus – veranlaßt den Empfänger, mit dem Abtasten des ankommenden Signals zu beginnen. Jede Bitzelle wird mehrmals abgetastet, beispielsweise mit einem Takt, der die 16fache Frequenz des Bittaktes hat. Trifft der erste Abtastimpuls auf den Eins-Null-Übergang, so hat man nach weiteren 24 solchen Impulsen ziemlich sicher die Mitte der nachfolgenden Bitzelle getroffen (diese enthält das erste Nutz-Bit des übertragenen Zeichens). Mit jeweils 16 Abtastimpulsen Abstand werden dann die weiteren Bitzellen näherungsweise in der Mitte abgetastet. Sind alle Zeichenbits (und ggf. ein zusätzliches Paritätsbit) übertragen worden, wird ein Einsbit als Endekennung (Stopbit) erwartet (kommt keine 1, liegt ein Fehler vor (Fachbegriff: Framing Error). Daraufhin gelangt der Empfänger in den Ruhezustand und erwartet das nächste Startbit. Es gibt auch Übertragungsformate mit 2 oder mit 1½ Stopbits ("1½ Bits" bedeutet, daß der High-Pegel wenigstens 1½ Bitzellen lang anliegen muß).

Gängige Übertragungsraten (in Bits/s):

50	75	110	134,5	150	200	300
600	1200	1800	2000	2400	3600	4800
7200	9600	14400	19200	38400	57600	115200



a), b) Zwei zu übertragende Bytes (Beispiele). Die Übertragung beginnt stets mit dem niedrigstwertigen Bit (LSB).

c) Übertragung von Byte a). 10-Bit-Format. Keine Parität, 1 Stopbit. Pause nach Übertragung des Zeichens.

d) Übertragung von Byte b). 10-Bit-Format. Keine Parität, 1 Stopbit. Nach dem Stopbit folgt sofort das Startbit des nächsten Zeichens (schnellste Übertragungsfolge).

e) Übertragung von Byte a). 12-Bit-Format. Paritätsbit (P), 2 Stopbits. Pause nach Übertragung des Zeichens. Wert des Paritätsbits P hängt von programmseitiger Einstellung im UART ab. Im Beispiel werden 4 Einsen übertragen. Deshalb ist P bei gerader Parität = 0, bei ungerader = 1.

Die serielle Schnittstelle im ATMega16

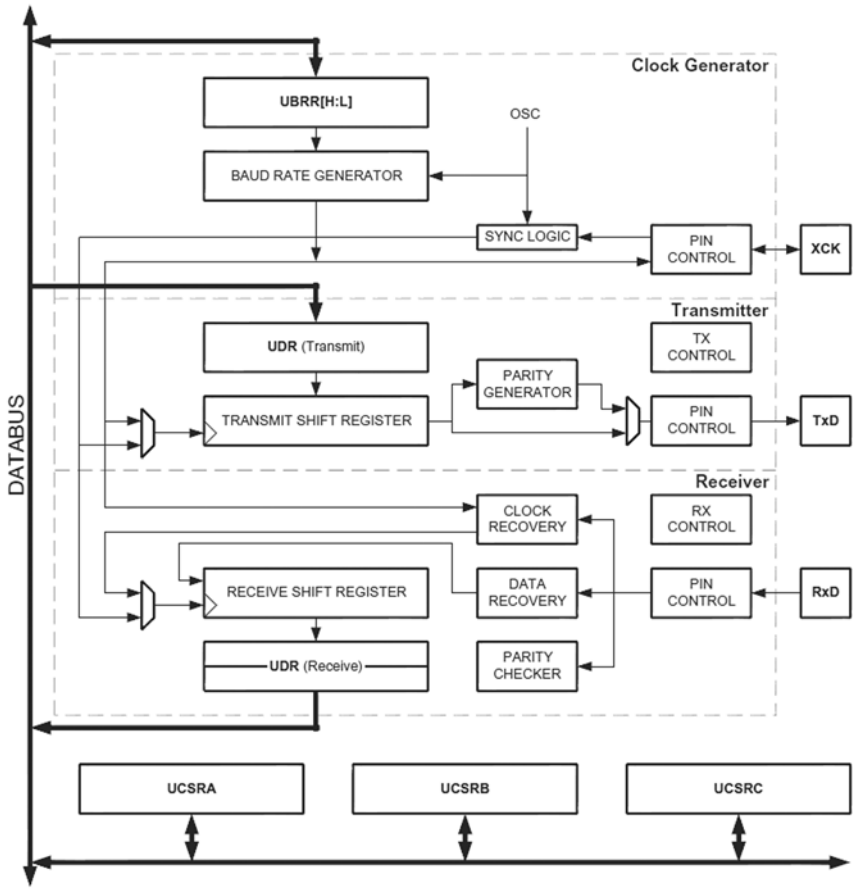
Wir beschränken uns auf die einfachsten Betriebsarten. Manche Einstellungen ergeben sich bereits beim Rücksetzen.

Programmseitige Steuerung:

a) Einstellung der Baudrate: UART Baud Rate Register UBRRH und UBRRL.
b) Betriebsartensteuerung und Zustandsabfrage: UART Control and Status Register A, B, C (UCSRA, UCSRB, UCSRC).
c) Datenbytes eintragen oder abholen: USART I/O Data Register UDR.

Wir arbeiten mit folgenden Einstellungen:

- Baudrate 2400,
- 8 Datenbits,
- kein Paritätsbit,
- 1 Stopbit,
- Senden über Abfrage,
- Empfangen über Unterbrechung.



(Bildquelle: Atmel.)

Baudrateneinstellung:
Die Baudrate ergibt sich gemäß Formel:.

$$BAUD = \frac{f_C}{16 \cdot (UBRR + 1)}$$

$$UBRR = \frac{f_C}{16 \cdot BAUD} - 1$$

f_C = Taktfrequenz.

Typische UBRR-Werte können aus Tabellen im Datenbuch entnommen werden. UBRR ist als Binärzahl von 12 Bits Länge einzustellen; 4 Bits in UBRRH, 8 Bits in UBRRL.

Für 2400 Baud (= Bits/s) brauchen wir nur das niederwertige Byte. Wert bei 4 MHz Takt = 103.

UART Baud Rate Register Low (UBRRL)

7	6	5	4	3	2	1	0
UBRRL7							UBRRL0

UART Status and Control Register A (USRA)

7	6	5	4	3	2	1	0
RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM

- RXC: Zeichen empfangen.
- TXC: Zeichen gesendet.
- UDRE: Datenregister leer.
- FE: Stopbit des empfangenen Zeichens = 0 (Framing Error).
- DOR: Überlauf. Ein empfangenes Byte ist noch anhängig (nicht abgeholt worden), und es kommt schon eine neues (Data Overrun).
- PE: Paritätsfehler.
- U2X: Doppelte Übertragungsgeschwindigkeit.
- MPCM: Multiprozessor-Kommunikationsmodus. Auswertung des 9. Datenbits als Adresse.

Wir fragen TXC ab, um zu erfahren, ob das Sendedatenregister frei ist oder nicht. Wurde TXC als gesetzt vorgefunden, schreiben wir eine Eins in diese Bitposition, um es zu löschen.

UART Control and Status Register B (UCSRB)

7	6	5	4	3	2	1	0
RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8

- RXCIE: Interruptauslösung nach Empfang eines Zeichens.
- TXCIE: Interruptauslösung nach Senden eines Zeichens.
- UDRIE: Interruptauslösung, wenn Datenregister leer (beim Senden).
- RXEN: Empfang ein- und ausschalten. 0 = ausgeschaltet, 1 = eingeschaltet.
- TXEN: Sendebetrieb ein- und ausschalten. 0 = ausgeschaltet, 1 = eingeschaltet.
- UCSZ2: eine Bitstelle der Zeichenlängeneinstellung.
- RXB8: das 9. Bit des empfangenen Zeichens.
- TXB8: das 9. Bit des zu sendenden Zeichens.

Wir setzen: RXCIE, RXEN und TXEN, also UCSRB = 98H.

Das UART Control and Status Register C (UCSRC) ist schon vom Rücksetzen her richtig eingestellt.

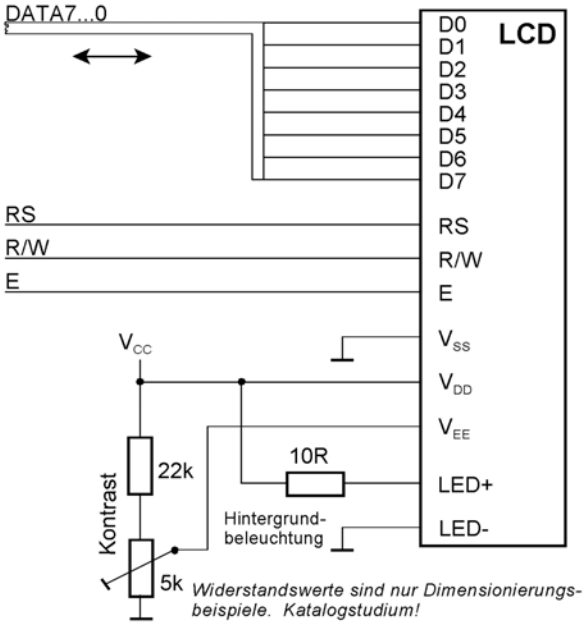
UART I/O Data Register (UDR)

7	6	5	4	3	2	1	0
D7							D0

Es sind zwei Register unter einer Adresse. Das Schreiben (Ausgabe) betrifft das Sendedatenregister, das Lesen (Eingabe) das Empfangsdatenregister.

Kurzausbildung LCD-Display

Interface:



Datenbus: Port C

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0

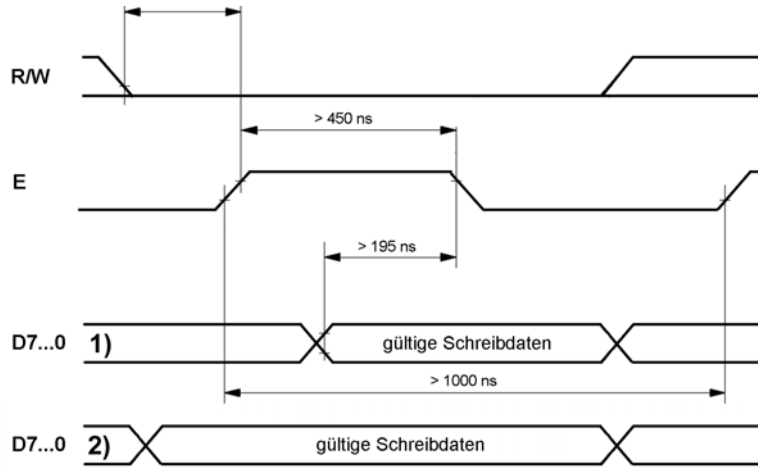
Steuersignale: Port B

7	6	5	4	3	2	1	0
–	–	–	–	–	E1	R/W	RS

E: LCD-Erlaubniseingang (Enable, Strobe). 0 = kein LCD-Zugriff, 1 = LCD-Zugriff.
R/W: LCD-Zugriffssteuerung. 0 = Schreiben, 1 = Lesen.
RS: LCD-Registerauswahl: 0 = LCD-Steuerregister, 1 = LCD-Datenregister.

Der typische Ablauf eines Schreibzugriffs:

1. RS je nach Zugriff einstellen. R/W auf 0.
2. Schreibdaten auf den Bus legen.
3. von Schritt 1 an müssen wenigstens 140 ns vergangen sein. E-Impuls erzeugen (mindestens 450 ns).
4. Ggf. Schreibdaten vom Bus nehmen und Bus freigeben.
5. Beginn des nächsten Zugriffs: der nächste E-Impuls darf frühestens 1 µs nach Schritt 3 ausgelöst werden.



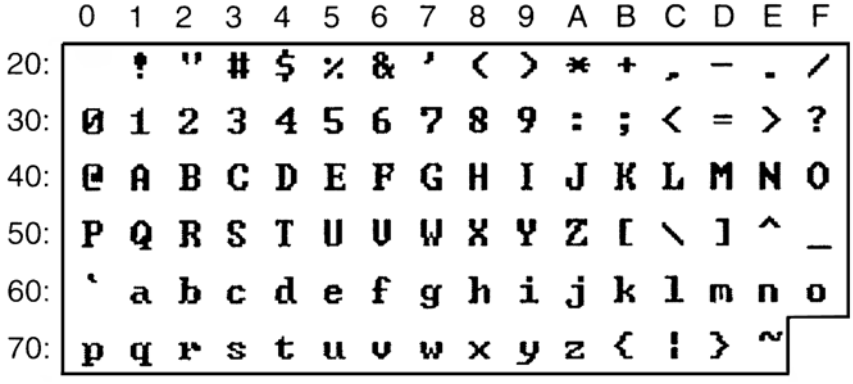
Die Zeichendarstellung wird durch Folgen entsprechender Kommandos aufgebaut. Nach dem Senden eines Kommandos ist die Anzeige zunächst mit dessen Ausführung beschäftigt und kann kein weiteres Kommando annehmen (Besetztzustand). Wir implementieren folgenden Ablauf:

- Das erste Kommando übertragen.
- Lange genug warten (gemäß maximaler Ausführungszeit (s. Kommandoübersicht)).
- Das zweite Kommando übertragen.
- Lange genug warten usw.

Initialisierung:
Datenbits und Steuersignale auf Ausgabe. Nach dem Rücksetzen Kommandos gemäß folgender Tabelle übertragen:

Kommando	Steuerl.		Datenbyte								Anmerkungen
	RS	R/W	7	6	5	4	3	2	1	0	
Function Set	0	0	0	0	1	1	1	0	0	0	8-Bit-Betrieb. 2 Zeilen, Zeichenraster 5 • 8
Clear Display	0	0	0	0	0	0	0	0	0	1	Datenspeicher löschen. Cursor auf erste Zeichenposition (links (oben))
Display On/Off Control	0	0	0	0	0	0	1	1	1	1	Anzeige ein, Cursordarstellung ein, Cursor blinken
Entry Mode Set	0	0	0	0	0	0	0	1	1	0	Cursoradresse zählt aufwärts (Autoincrement); kein Schieben

Zeichentabelle (ASCII):



Datenspeicheradressierung:

Anzeige	1. Zeile	2. Zeile	3. Zeile	4. Zeile
2 • 16	00H...0FH	40H...4FH		
2 • 20	00H...13H	40H...53H		
2 • 24	00H...17H	40H...57H		
2 • 40	00H...27H	40H...67H		
4 • 16	00H...0FH	40H...4FH	10H...1FH	50H...5FH
4 • 20	00H...13H	40H...53H	14H...27H	54H...67H

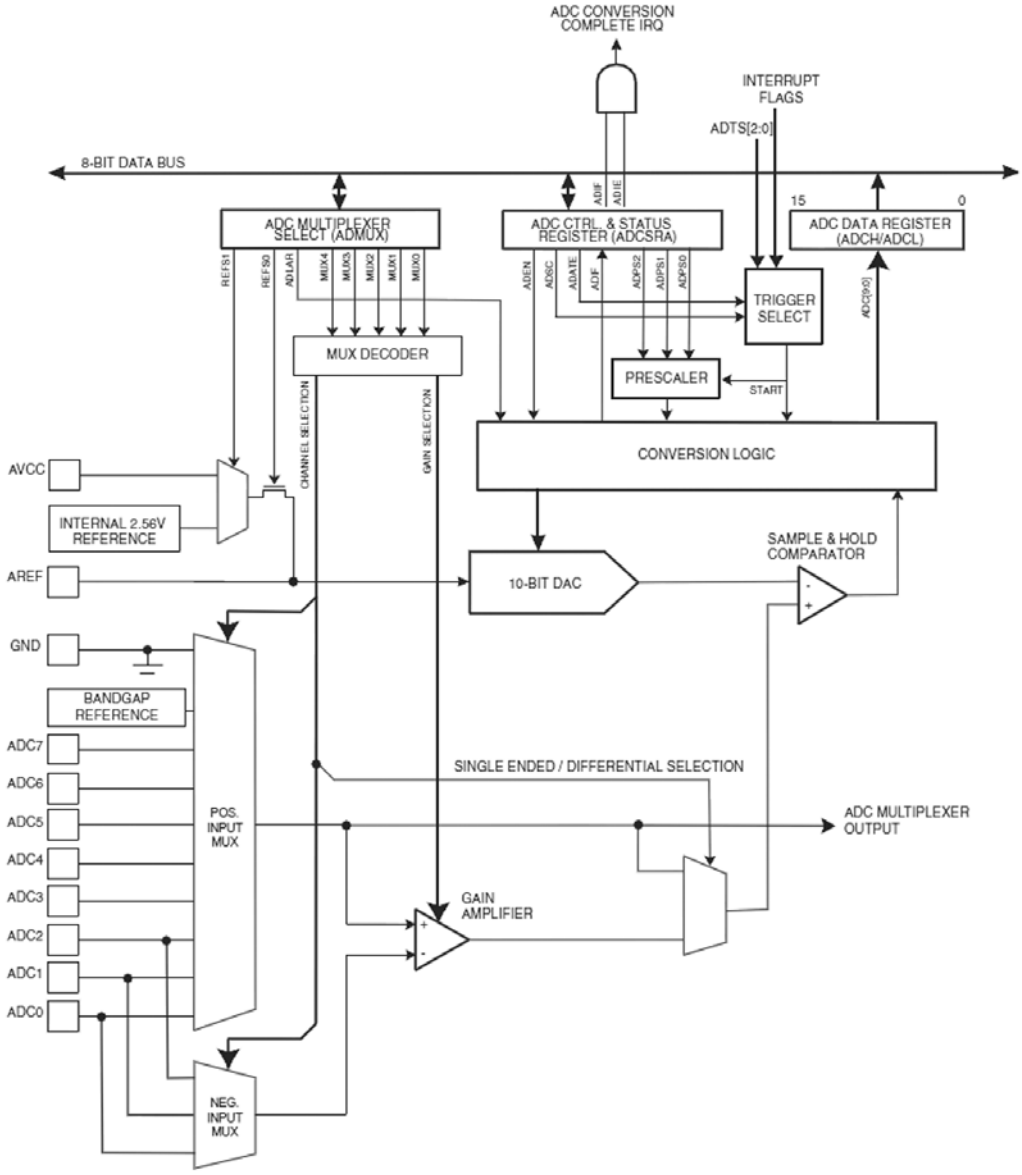
Es gibt keinen automatischen Übergang von Zeile zu Zeile. Vor dem Eintragen in eine bestimmte Zeile jeweils Datenspeicheradresse setzen (Kommando *DD RAM Adrs Set*).

Kommandoübersicht:

Kommando	Steuerl.		Datenbyte								Beschreibung	max. Ausführungszeit[2)]
	RS	R/W	7	6	5	4	3	2	1	0		
Clear Display	0	0	0	0	0	0	0	0	0	1	gesamte Anzeige löschen. Datenspeicheradresse auf Null	1,52 / 1,64 ms
Return Home	0	0	0	0	0	0	0	0	1	*[1)]	Datenspeicheradresse auf Null. Anzeige an Originalposition (keine Verschiebung). Datenspeicherinhalt bleibt erhalten	1,52 / 1,64 ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Einstellung der Bewegungsrichtung des Cursors (I/D). Wahl zwischen Cursorbewegung und Verschieben der Anzeige (S)	37 / 40 µs
Display On/Off Control	0	0	0	0	0	0	1	D	C	B	Anzeige ein/aus (D), Cursor ein/aus (C), Blinken an Cursorposition ein/aus (B)	37 / 40 µs
Cursor/Display Shift	0	0	0	0	0	1	S/C	R/L	*[1)]	*[1)]	Cursorbewegung und Verschieben der Anzeige	37 / 40 µs
Function Set	0	0	0	0	1	DL	N	F	*[1)]	*[1)]	Einstellung der Zugriffsbreite (DL), der Zeilenzahl (N) und des Zeichensatzes (F)	37 / 40 µs
CG RAM Adrs Set	0	0	0	1	Zeichengeneratoradresse						Adresse des ladbaren Zeichengenerators einstellen	37 / 40 µs
DD RAM Adrs Set	0	0	1	Datenspeicheradresse							Datenspeicheradresse einstellen	37 / 40 µs
Busy Flag / Adrs Read	0	1	BF	Adreßzähler							Busy-Bit (BF) und aktuelle Adreßzählerbelegung lesen	-
CG RAM / DD RAM Data Write	1	0	zu schreibendes Byte								Schreiben in ausgewählten Speicher	37 / 40 µs
CG RAM / DD RAM Data Write	1	1	gelesenes Byte								Lesen des ausgewählten Speichers	37 / 40 µs

1): bedeutungslos (don‘t care); 2): 44780U / herkömmliche Ausführungen

Kurzausbildung A-D-Wandler



(Bildquelle: Atmel.)

Merkmale:

- Auflösung: 10 Bits,
- integrale Nichtlinearität: 0,5 LSB,
- absolute Genauigkeit: ± 2 LSB,
- höchste Abtastrate bei maximaler Auflösung: 15 kHz,
- Eingangsspannungsbereich: 0V bis Referenzspannung (AREF),
- Referenzspannung (AREF): 2 V bis AV_{CC},
- Wirkprinzip: schrittweise Annäherung (sukzessive Approximation),
- Betriebsarten: Einzelumsetzung (Single Conversion) oder kontinuierlicher Betrieb (Free Running),
- Anzahl der Eingänge: 8 positive Spannungen gegen Masse (programmseitig wählbar). Die Eingänge sind alternativ auch für Differenzsignale nutzbar (siehe Handbuch).

Programmseitige Steuerung:

a) Eingangsauswahl: ADC Multiplexer Select Register: ADMUX.
b) Betriebsartensteuerung und Auslösung: ADC Control and Status Register ADCSR.
c) Abholen der Digitalwerte: ADC Data Register ADCL und ADCH.

ADC Multiplexer Select Register (ADMUX)

7	6	5	4	3	2	1	0
REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

- REFS1, 0: Referezspannungsauswahl.
- ADLAR: Ergebnis linksbündig liefern (ADC Left Adjust Result).
- MUX4...MUX0: Auswahladresse. Belegung 0H wählt Eingang ADC 0 aus, Belegung 1H Eingang ADC1 usw.

Wir arbeiten mit externer Referenzspannung (vom Starterkit) und linksbündigem Ergebnis. Zum Auswählen der Bits A7...A0 brauchen wir nur MUX2...0. Initialsierung: 20H.

ADC Control and Status Register A (ADCSRA)

7	6	5	4	3	2	1	0
ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

- ADEN: Analog-Digital-Wandler ein- und ausschalten. 0 = ausgeschaltet, 1 = eingeschaltet.
- ADSC: Start der Abtastung.
- ADATE: automatische Trigerauslösung über Triggersignal.
- ADIF: Flagbit. Kennzeichnet, daß die Wandlung beendet und ggf. eine Interrupanforderung anhängig ist.
- ADIE: Interruptauslösung nach Wandlung.
- ADPS2...ADPS0: Auswahl des Abtasttaktes (über Vorteiler (Prescaler)).

Wir arbeiten mit Abfrage und 64 Takten Wandlungsintervall. ADEN = 1, ADPS2...0 = 6. Also ADCSRA = 86H.

ADPS2...0	**Abtastintervall**	**ADPS2...0**	**Abtastintervall**
0	2 Takte ($f_S = f_C : 2$)	4	16 Takte ($f_S = f_C : 16$)
1	2 Takte ($f_S = f_C : 2$)	5	32 Takte ($f_S = f_C : 32$)
2	4 Takte ($f_S = f_C : 4$)	6	64 Takte ($f_S = f_C : 64$)
3	8 Takte ($f_S = f_C : 8$)	7	128 Takte ($f_S = f_C : 128$)

f_S = Abtastfrequenz; f_C : Taktfrequenz

ADC Data Register (ADCH, ADCL). Ergebnis rechtsbündig (ADLAR = 0)

Register	7	6	5	4	3	2	1	0
ADCH	-	-	-	-	-	.	ADC9	ADC8
ADCL	ADC7							ADC0

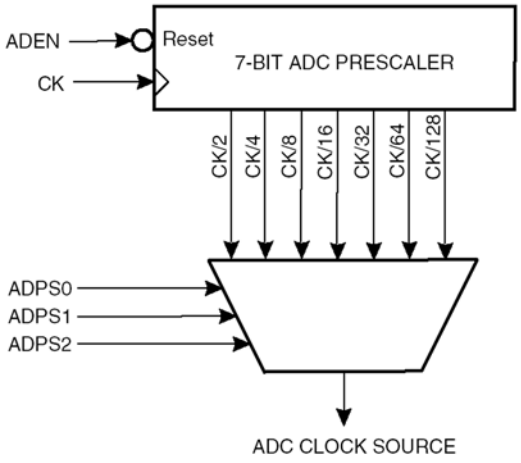
ADC Data Register (ADCH, ADCL). Ergebnis linksbündig (ADLAR = 1)

Register	7	6	5	4	3	2	1	0
ADCH	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2
ADCL	ADC1	ADC0	_	_	_	_	_	_

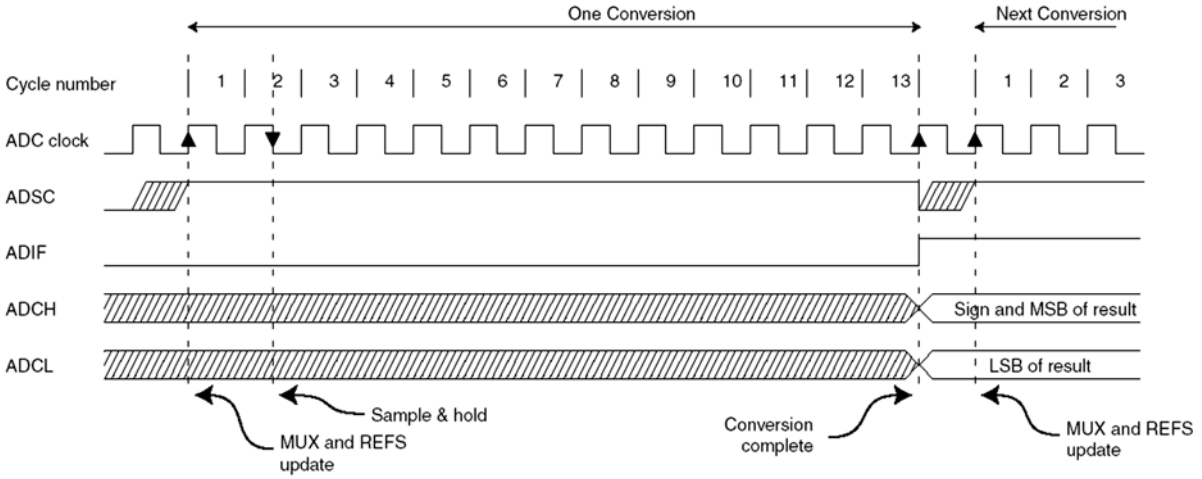
Ein Wandlungsablauf:

1. Wandlung starten. ADSC in ADCSRA auf 1.
2. Warten, bis ADIF in ADCSRA = 1 ist.
3. ADCH abholen (8-Bit-Ergebnis).
4. Eine Eins in ADIF schreiben, um das Flagbit zu löschen.

Der Prescaler bestimmt, wie lange eine Wandlung dauert. Je länger, desto genauer.



Der Wandlungsablauf:



(Bildquellen: Atmel.)