Praktikum Mikrocontrollertechnik SS 2015
Versuch 2

Stand: 8. 6. 2015

Elementare Anwendungs- und Schnittstellenprogrammierung in C unter Einschluß von Interruptserviceroutinen (ISRs).

1. Inbetriebnahme der seriellen Schnittstelle mittels Terminalprogramm.
2. Inbetriebnahme einer LCD-Punktmatrixanzeige.
3. Inbetriebnahme des Analog-Digital-Wandlers.

Versuchsanordnung:
Startertkit Atmel STK 500 mit Atmel ATMega 16. Peripherie: PC mit Terminalprogramm, LCD-Anzeige 09 mit Punktmatrix-LCD 2 • 16, Universaladapter 10b.

Die C-Programmierung vorbereiten:

- Das Projekt des ersten Versuchs aufrufen oder eine neues C-Projekt einrichten (s. die Anleitung zum ersten Versuch).
- Das Programm **V2C_15_template.c** in das Editorfenster des Projekts kopieren.
- Die Datei **V2C_15_examples.c** im Editor öffnen, um ggf. Programmbeispiele übernehmen zu können.
- Optimierung ausschalten (“O0").

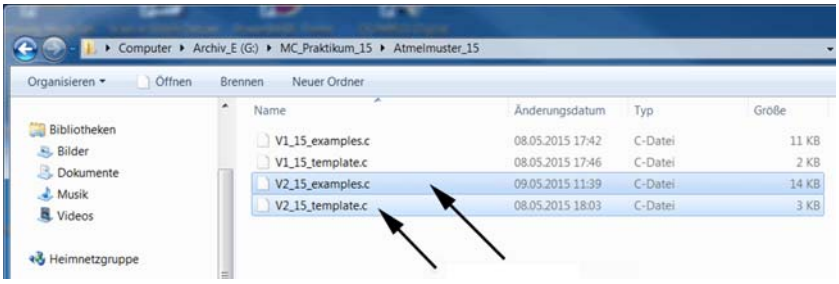


Abb. 1 Musterdateien für die C-Programmierung.

Aufgabe 1: Serielle Schnittstelle.

Kurzausbildung serielle Schnittstelle
Die serielle Schnittstelle ist ein bitserielles asynchrones Interface. Je Richtung ist eine Datenleitung vorgesehen (Duplexbetrieb). Es werden einzelne Zeichen übertragen (Start-Stop-Verfahren). Der Übertragung liegt ein festes Zeitraster zugrunde (Übertragungsrate in Bits/s oder Baud). 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).

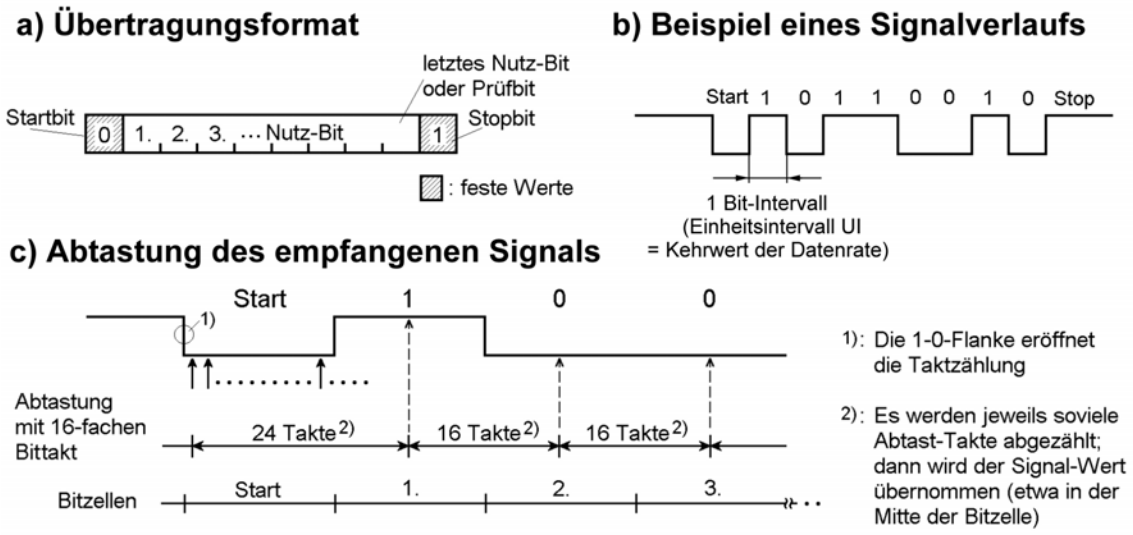


Abb. 2 Das Übertragungsprinzip.

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

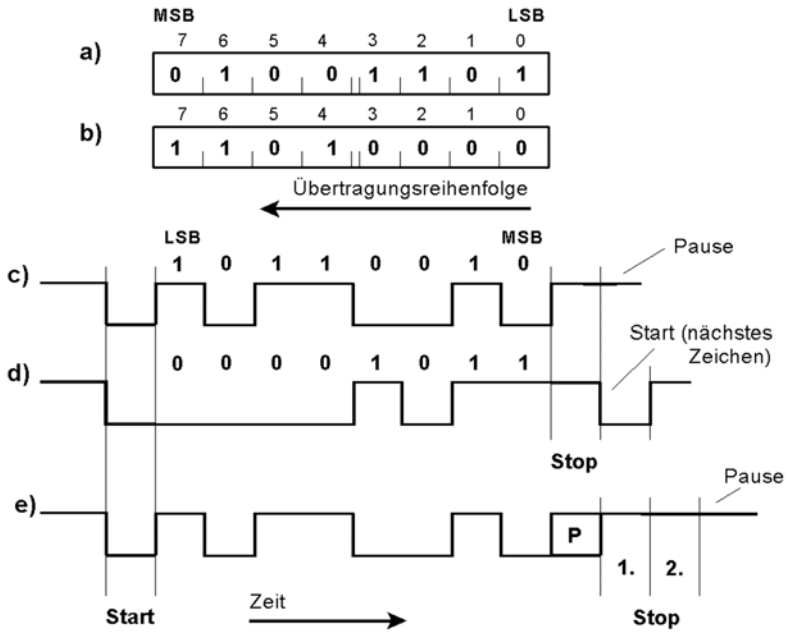
Die serielle Schnittstelle im ATMega16
Wir beschränken uns auf die einfachsten Betriebsarten. Manche Einstellungen ergeben sich bereits beim Rücksetzen.

Programmseitige Steuerung:

a) Einstellung der Baudrate: USART 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.



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 USART ab. Im Beispiel werden 4 Einsen übertragen. Deshalb ist P bei gerader Parität = 0, bei ungerader = 1.

Abb. 3 Übertragungsbeispiele.

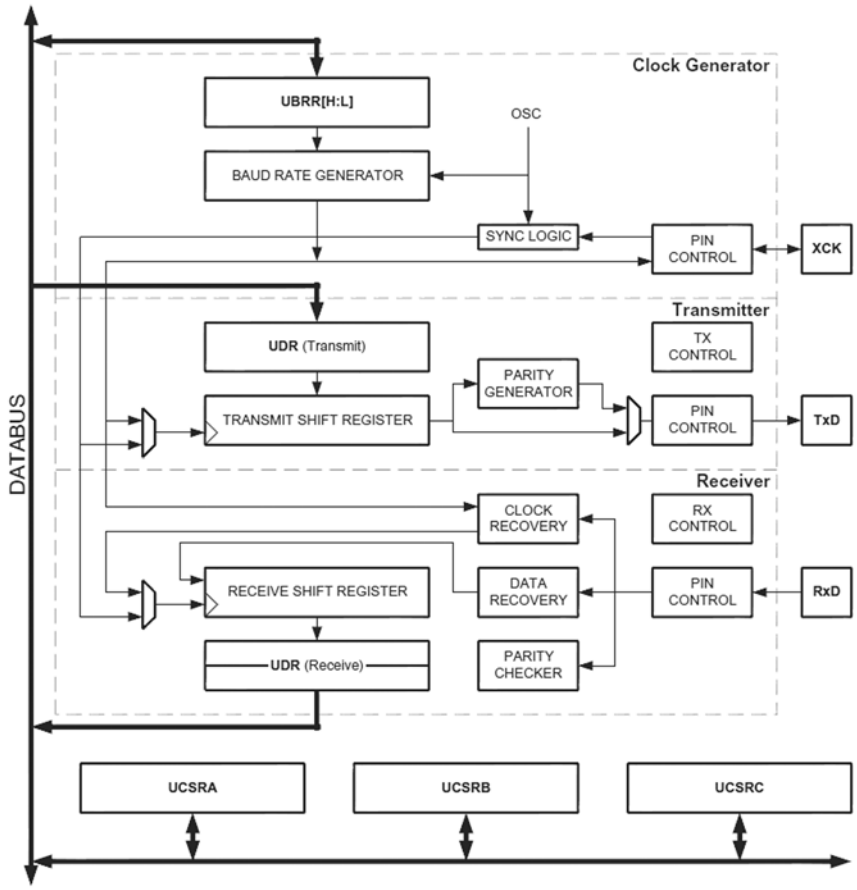
Baudrateneinstellung:
Die Baudrate ergibt sich gemäß Formel:.

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

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

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 bei f_C = 4 MHz nur das niederwertige Byte. Wert = 103.



(Bildquelle: Atmel.)

Abb. 4 Blockschaltbild der Schnittstellenhardware. USART = Universal Synchronous and Asynchronous serial Receiver/Transmitter.

USART Baud Rate Register Low (UBRRL)

7	6	5	4	3	2	1	0
UBRRL7							UBRRL0

USART Status and Control Register A (UCSRA)

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 UDRE ab, um zu erfahren, ob das Sendedatenregister frei ist oder nicht. UDRE wird automatisch gelöscht wenn ein Datenbyte in das Datenregister geschrieben wird.

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

USART 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.

Versuchsaufbau:

1. Die serielle Erprobungsschnittstelle des Starterkits mit Port D, Bits 0 und 1, verbinden (Brückenkabel).
2. Das zweite Schnittstellenkabel des PCs ans Starterkit anschließen.
3. Terminalprogramm aufrufen. Es ein ganz einfaches Programm, an dem nichts einzustellen ist.
4. Den COM-Port auswählen. Es ist entweder COM-Port 1 oder COM-Port 2. Wenn es nicht klappt, mit dem jeweils anderen COM-Port probieren oder das andere Schnittstellenkabel anschließen.
5. Programm eingeben und zum Laufen bringen.

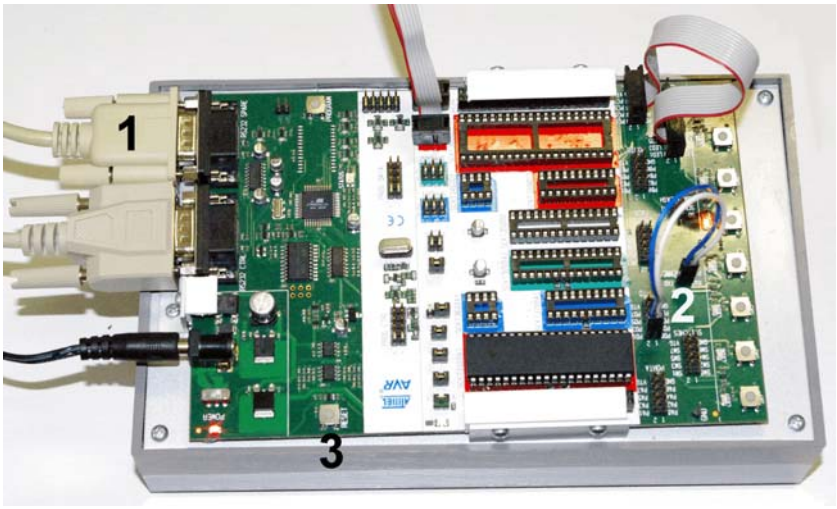


Abb. 5 So wird die serielle Schnittstelle angeschlossen. 1 - Kabel zum PC; 2 - Verbindung des Treiberschaltkreises mit Port D (Signale PD0 und PD1); 3 - RESET-Taste.

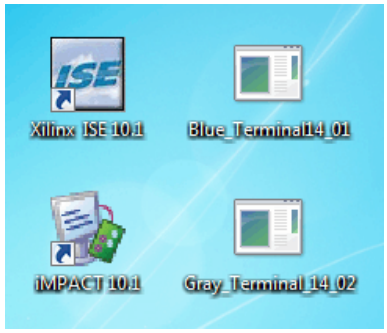


Abb. 6 Ein Terminalprogramm auswählen.

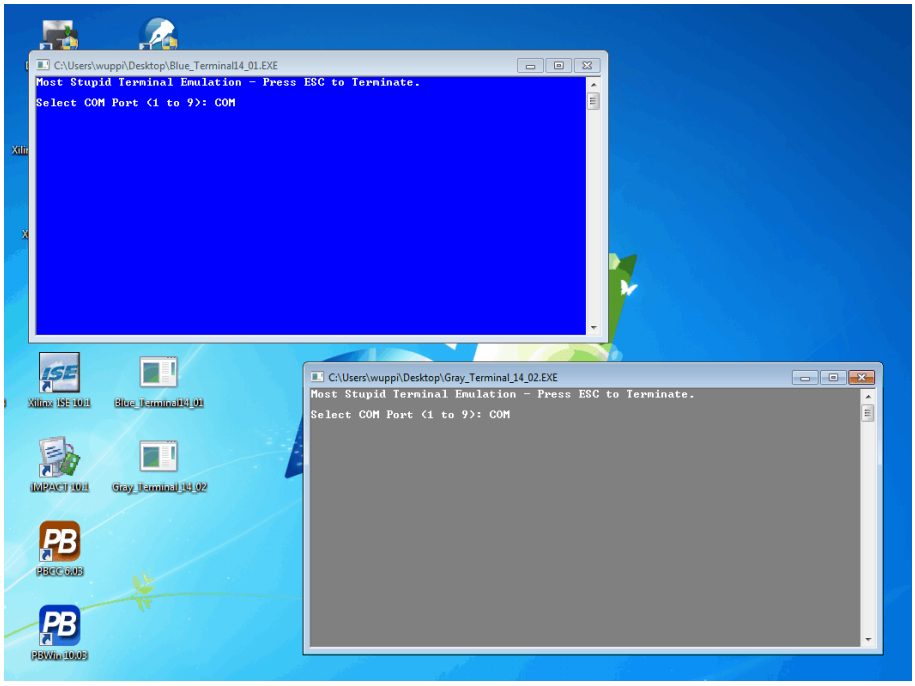


Abb. 7 Die Wahl ist reine Geschmackssache. Die Nutzung erklärt sich von selbst.

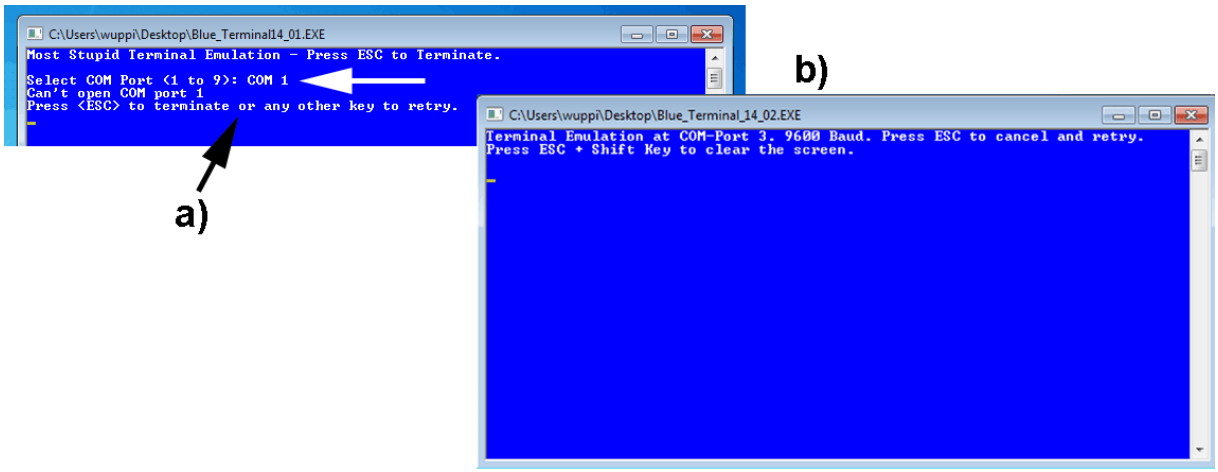


Abb. 8 Zur Auswahl des COM-Ports die Nummer (1 oder 2) eingeben (Pfeil). a) erscheint, wenn es nicht klappt, b) wenn alles in Ordnung ist.

Eine kleine Fehlersuchhilfe
Wenn im Terminalfenster nichts zu sehen ist:

- Richtiges Schnittstellenkabel angesteckt?
- Schnittstelle auf dem Starterkit richtig angeschlossen?
- Ggf. nachmessen (Oszilloskop).

Abb. 9 Ein Praxistip zwischendurch: Welches Kabel gehört zu welchem COM-Port? Das kann man mit einem Kurzschlußstecker experimentell herausfinden. Die Signale TX und RX (Pins 2und 3) sind hier miteinander verbunden (echte Profis schaffen es auch mit einem Schraubenzieher oder einer aufgebogenen Büroklammer...). Gehört das Kabel zum ausgewählten COM-Port, werden Tastatureingaben auf dem Terminalfenster angezeigt (Echofunktion). Kurzschlußstecker kann bei Bedarf ausgeliehen werden.

Bedienhinweise:
- Den Bildschirm löschen: Die Umschalttaste (SHIFT) niederhalten und ESC betätigen.
- Einen COM-Port zuweisen: ESC betätigen.
- Das laufende Programm beenden: ESC zweimal betätigen.

Zu den Grundlagen: Registerbits einstellen und abfragen:
C ist uralt und keine Sprache zum Programmieren von Steuerungsalgorithmen im allgemeinen und Mikrocontrollern im besonderen. Von Grund auf ist nichts eingebaut. Was es gibt, sind Behelfslösungen oder Zusätze.

Die einfachste Lösung:
Wir schauen uns die Registerbelegung im Datenbuch an und überlegen, wie das Bitmuster aussehen muß. Um es einzustellen, wird es als hexadezimale oder binäre Konstante zugewiesen.

```
UCSRB = 0b10011000;
```

Einzelne Bits setzen: bitweise ODER-Verknüpfung mit einer Konstanten, die an den betreffenden Bitpositionen Einsen enthält und sonst Nullen.

```
UCSRB = (UCSRB | 0b01000000); // setzt Bit 6
```

Einzelne Bits löschen: bitweise UND-Verknüpfung mit einer Konstanten, die an den betreffenden Bitpositionen Nullen enthält und sonst Einsen.

```
UCSRB = (UCSRB & 0b10111111); // löscht Bit 6
```

Einzelne Bits umschalten: bitweise XOR-Verknüpfung mit einer Konstanten, die an den betreffenden Bitpositionen Einsen enthält und sonst Nullen.

```
UCSRB = (UCSRB ^ 0b01000000); // schaltet Bit 6 um
```

Um ein Bit abzufragen, programmieren wir eine bitweise UND-Verknüpfung mit einer Konstanten, die an der betreffenden Bitposition eine Eins enthält. Ist das Bit gelöscht, so ist das Ergebnis gleich Null, ist das Bit gesetzt, so ist das Ergebnis ungleich Null.

```
if ((UCSRA & 0b00100000))        // wenn Bit 5 in UCSRA gesetzt*

if (!(UCSRA & 0b00100000))       // wenn Bit 5 in UCSRA gelöscht
```

*: *Hinweis:* Die beiden äußeren Klammern gehören zur **if**-Anweisung, die beiden inneren sind notwendig, um den Compiler zu veranlassen, die UND-Verknüpfung wirklich bitweise auszuführen. Ohne sie würde er nämlich die kompletten Variablen als logische Werte interpretieren und konjunktiv verknüpfen (Wert = 0 ergibt logisch 0, Wert $\neq$ 0 ergibt logisch 1).

Wie es oft gemacht wird:
Die Registerbits haben Namen. Sie stehen in den Atmel-Handbüchern. Die Include-Dateien von Atmel geben die Positionen dieser Bits an. Dies ermöglicht es, den Preprocessor des C-Compilers auszunutzen, um die konstanten Bitmuster herzustellen. Es werden Einsen an die betreffenden Positionen verschoben:

```
1 << UDRE;    // Beim Atmega16 ist UDRE das Bit Nr. 5 im Register UCSRA.
              // Der Preprocessor verschiebt die Eins um 5 Bits,
              // so daß sie auf Bitposition 5 zu liegen kommt.
```

Geht es um mehrere Bits auf einmal, werden solche verschobenen Einzelbits mit ODER-Verknüpfungen zusammengefaßt:

```
UCSRB = (1 << TXEN | 1 << RXEN);
```

Eine weitere Alternative:
Die Einzelbitfunktionen verwenden, die in manchen Compilern angeboten werden (auch in dem Compiler, den wir verwenden). Dann hängt man aber vom betreffenden Compiler ab. Auch ist die Unterstützung gelegentlich eingeschränkt. Deshalb nutzen wir diese Funktionen hier nicht.

1. Die Schnittstelle im AVR initialisieren und ausprobieren.
Das anfängliche Projektprogramm (als Kopie von V2_15_template.c) zum Laufen bringen. Es gibt ein einziges Zeichen aus (zu Probe, ob die Initialisierung geklappt hat). Ggf. die RESET-Taste am Starterkit betätigen, um das Senden erneut auszulösen. Die Unterbrechungsbehandlung (Interrupt Service Routine ISR) bewirkt zweierlei:

1. Das jeweils empfangene Zeichen wird über Port C auf die LEDs des Starterkits ausgegeben. Mit einigen Zeichen probieren und nachsehen (ASCII-Tabelle am Ende dieser Versuchsanleitung), ob es stimmt.

2. Das Zeichen wird zum Terminal zurückgesendet und somit dort angezeigt (Echofunktion). Auf dem Bildschirm erscheinen die Zeichen so, als wären sie an Ort und Stelle eingetippt worden.

Welche Zeichen können angezeigt werden? – Terminalcodes sind eine Wissenschaft für sich (s. die einschlägigen Handbücher und Hilfedateien). Ohne weiteres verfügbar sind nur das lateinische Alphabet (ohne Umlaute und Eszett), die Ziffern und die übliche Satzzeichen – mit anderen Worten, der Zeichensatz des sog. ASCII-Codes. Codetabelle am Ende dieser Versuchsanleitung.

Hinweis zur Interruptserviceroutine:
Wie der Interruptmechanismus wirklich funktioniert, kann man nur dann kennenlernen, wenn man sich mit den Wirkprinzipien des Prozessors vertraut macht und die Maschine zu Fuß programmiert

(Assemblerprogrammierung). Hier bleibt uns hingegen nur ein Dienst nach Vorschrift übrig – es muß einfach so übernommen werden, wie es vorgestanzt ist. Das C-Programm muß einen einzigen Assemblerbefehl enthalten, nämlich den, der den Interruptmechanismus scharfmacht. In dem C-Compiler, den wir verwenden, wird leider das Einfügen von Assemblerbefehlen so unterstützt, daß sich einem die Fußnägel aufbiegen...

2. Einen Text (Zeichenkette) ausgeben.
Die Zeichenkette wird mit 00H abgeschlossen. Das Zeichenkettenhandling in C ist lausig – wir können wirklich nichts dafür... Wir brauchen ein Array, das die zu behandelnde Zeichenkette aufnimmt. Füllen des Arrays: mit Funktion strcopy.

Programmbeispiel: 1.

3. Verschönerung der Ausgabe
Nach Ausgabe der Zeichenkette soll der Cursor des Terminals an den Anfang der folgenden Zeile gesetzt werden. Hierzu müssen wir zwei Zeichen senden: CR (Carriage Return = Wagenrücklauf) und LF (Line Feed = Zeilenvoschub). Diese Bezeichnungen stammen noch aus der Zeit der Fernschreibtechnik...

Programmbeispiel: 2.

4. Die Echofunktion auf die Unterstützung der Enter-und der Backspace-Taste (BS = 08) erweitern.
Die zu unterstützenden Tastencodes: Enter-Taste (CR) = 13; Backspace-Taste (BS) = 08. Die Zeichenausgabe zum Port C entfernen.

- Echo bei Enter: Rücklauf und neue Zeile (CR LF = 13 10). Die CR-Taste wirkt wie die Wagenrücklauftaste einer herkömmlichen Schreibmaschine.
- Echo bei Backspace: Löschen des letzten Zeichens und Zurücksetzen des Cursors (Folge BS – SP – BS = 08 32 08). Die Backspace-Taste setzt den Cursor um einen Zeichenposition zurück und löscht dieses Zeichen, so daß an dieser Stelle neu eingegeben werden kann. (Schreibmaschine: Rückschritt- bzw. Korrekturtaste).

Programmbeispiel: 3.

Aufgabe 2: LCD-Punktmatrixanzeige.

Die LCD-Anzeige 09 anschließen. Daten an Port C, Steuersignale an Port B.

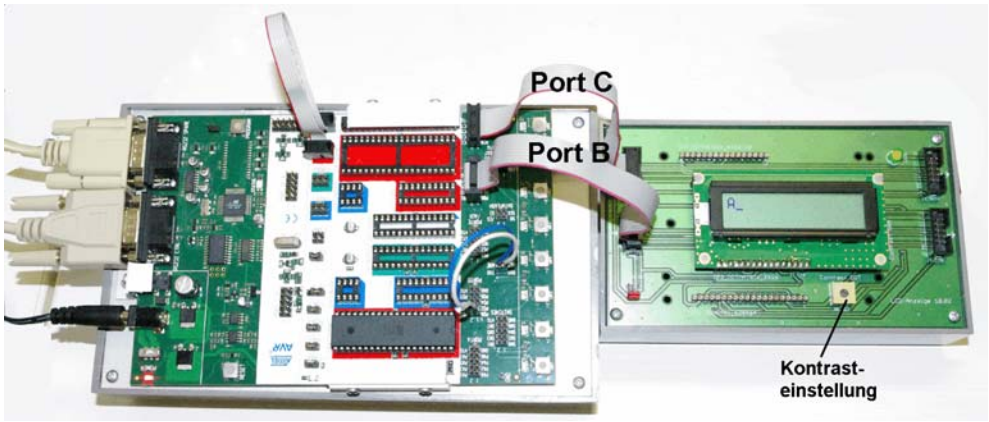


Abb. 10 Versuchsaufbau mit LCD-Anzeige. Die serielle Schnittstelle bleibt angeschlossen.

Kurzausbildung LCD-Punktmatrixanzeige (Dotmatrix Display)
Punktmatrixanzeigen dienen zum Darstellen von Zeichen. Eine Anzeige des Typs X • Y kann X Zeilen zu jeweils Y Zeichen darstellen. In unserem Versuchaufbau ist die Anzeigeeinheit mit einem Anzeigemodul 2 • 16 bestückt.

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.

Es ist nicht zwingend erforderlich, Lesezugriffe zu unterstützen. Schreibzugriffe genügen vollauf. Wir beschränken uns deshalb auf diese Betriebsweise.

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. Er muß mindestens 450 ns lang sein.
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.

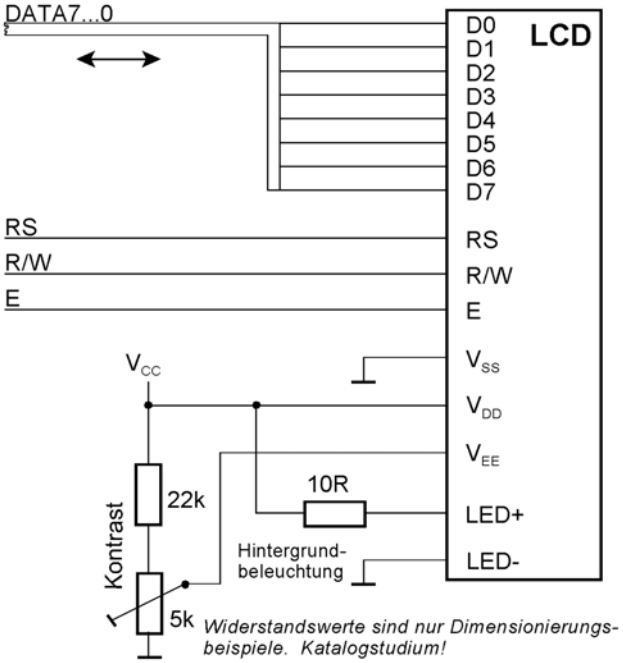


Abb. 11 Die Schnittstelle einer typischen Punktmatrixanzeige. 8-Bit-Datenbus, drei Steuersignale.

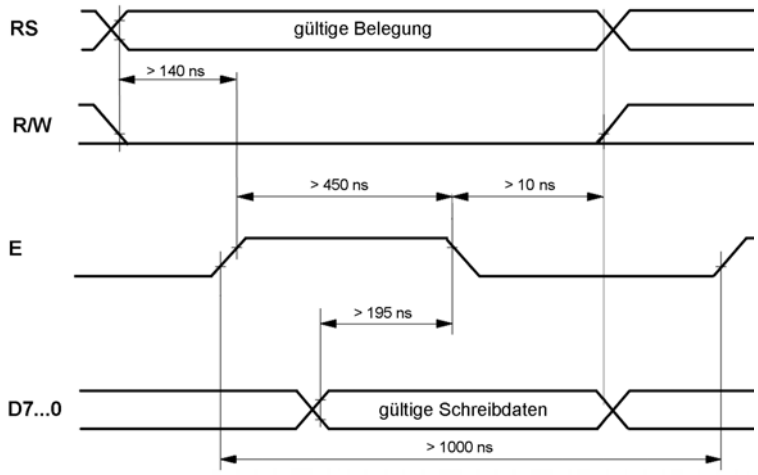


Abb. 12 Ein Schreibzyklus.

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 laut Kommandoübersicht).
- Das zweite Kommando übertragen.
- Lange genug warten usw.

Darstellbare Zeichen:
Es gibt verschiedene Zeichensätze. Zu den Einzelheiten siehe das Datenmaterial der Anzeigemodule. Grundsätzlich verfügbar sind – wie beim Terminal – nur die Zeichen des ASCII-Codes. Weitere Darstellversuche (Umlaute, Eszett, griechisch, kyrillisch usw.) sind ohne Datenblattstudium zwecklos.

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

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		

Anzeige	1. Zeile	2. Zeile	3. Zeile	4. Zeile
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. Aus-führungs-zeit[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.

1. LCD initialisieren und ausprobieren.

Wir arbeiten mit Zeitsteuerung (Schreiben und Ausführungszeit abwarten; kein Zurücklesen). Ein einzelnes Zeichen darstellen (zur Probe, ob die Initialisierung geklappt hat). Zeichen muß links oben erscheinen, rechts daneben muß der Cursor blinken. Ggf. Kontrast nachstellen (Trimmpotentiometer).

Programmbeispiel: 4.

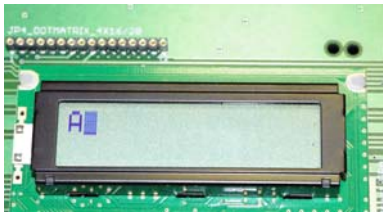


Abb. 13 Die erste Zeichenanzeige.

2. Einen einzeiligen Text darstellen.
Wir richten eine Zeichenkette als Array ein und übertragen Zeichen für Zeichen zur LCD-Anzeige. Der Ablauf entspricht dem Senden über die serielle Schnittstelle (nur andere Transportroutine).

Programmbeispiel: 5.

3. Einen zweizeiligen Text darstellen.
ImGrunde nichts besonderes. Vor dem Senden der zweiten Zeichenkette müssen wir den Cursor auf den Anfang der zweiten Zeile setzen.

Programmbeispiel: 6.

4. Die Anzeige zyklisch umlaufen lassen (Laufschrift).
Hierzu nutzen wir das Verschiebekommando der LCD-Anzeige. Es wird in der Endlosschleife ausgeführt.

Programmbeispiel: 7.

So besonders sieht es nicht aus. Das liegt daran, daß der gesamte Anzeigepufferinhalt verschoben wird. Es sind immer 80 Zeichen, die umlaufen, bei einer zweizeiligen Anzeige also 2 • 40. Sie können versuchen, eigene Texte[1] zu gestalten, die unter dieser Bedingung einigermaßen ansehnlich sind.

5. Die Interruptserviceroutine (ISR) der seriellen Schnittstelle erweitern.
Und zwar so, daß die empfangenen Zeichen auf der LCD-Anzeige dargestellt werden. Zunächst probieren wir es einfach und schmucklos.

Programmbeispiel: 8.

6. Die ISR auf selbsttätigen Wrap Around erweitern.
Das heißt: Übergang in die zweite Zeile, wenn Ende der ersten Zeile erreicht, Übergang an den Anfang, wenn Ende der 2. Zeile erreicht. Die eingebaute Adreßzählung der LCD-Anzeige unterstützt das nicht. Deshalb definieren wir zwei globale Variabe x und y und zählen die Zeichenadresse selbst mit. x = Adresse des Zeichens in der Zeile (Spaltenadresse; Bereich 0...15). y = Adresse der Zeile (Zeilenadresse; Bereich 0, 1). Das Mitzählen erledigen wir in einer neuen Funktion enter, die in die ISR eingebaut wird. Da es nur zwei Zeilen sind, wird die Zeilenumschaltung sehr einfach. Hier wollen wir uns mit einer Primitivlösung begnügen.

Programmbeispiel: 9.

7. Die Tasten ENTER und BACKSPACE vernünftig unterstützen.
Die Anzeige soll sich wie ein kleines Terminal verhalten. ENTER führt an den Anfang der nächsten Zeile. BACKSPACE bewegt den Cursor um eine Position nach links (Rückschritt). Sowohl die bisherige als auch die neue Zeichenposition werden gelöscht. Löschen heißt Eintragen eines Leerzeichens (SPACE

1: Aber bitte nach Möglichkeit nicht allzu primitive, ordinäre usw.

= 20H). Beide Funktionen sollen mit Wrap Around wirken. Wir unterstützen erst einmal die ENTER-Funktion allein (einfacher) und nehmen dann BACKSPACE hinzu. Die Rückwärtsbewegung des Cursors erledigen wir mit einer neuen Funktion backspace. Dann läuft alles fast genauso ab wie beim Terminal.

Programmbeispiele: 10 und 11.

Aufgabe 3: Der Analog-Digital-Wandler (ADC).

Kurzausbildung A-D-Wandler

Analog-Digital-Wandler (Analog-to-Digital Converters, ADCs) setzen analoge (kontinuierliche) Signalverläufe in Folgen einzelner (diskreter) Wertangaben um, die üblicherweise binär codiert sind.

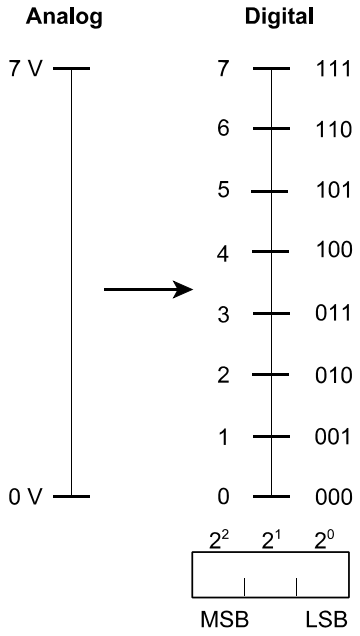


Abb. 14 Analoge und digitale Signalwerte. Hier sind die digitalen Werte als vorzeichenlose Binärzahlen codiert, die 3 Bits lang sind. Deshalb kann es höchstens $2^3 = 8$ verschiedene digitale Signalwerte geben. Jeder der digitalen Werte von 0 bis 7 entspricht einem analogen Signalpegel. Im Beispiel ist die Zuordnung besonders einfach: 0 = 0 V, 1 = 1 V, 2 = 2 V usw.

Wir beschränken uns auf die einfachste Nutzungsweise und verzichten auf jegliche Behandlung von Spitzfindigkeiten.

Die Referenzspannung (VREF)

Der Wandler liefert digitale Werte, die natürliche Binärzahlen sind. Mit n Bits kann man jeweils einen von 2^n möglichen Werten darstellen. Diese Werte sind im Grunde Verhältnisangaben, die sich auf die sog. Referenzspannung beziehen. Die Referenzspannung ist eine konstante Spannung, die dem Wandler zugeführt wird. Von der Genauigkeit dieses Spannungswertes hängt die Genauigkeit der Wandlung entscheidend ab.

Die Auflösung (Resolution)

Die Auflösung betrifft die geringste Wertänderung auf der digitalen Seite, mit anderen Worten: den Unterschied zwischen zwei unmittelbar benachbarten Digitalwerten. Ein Wandler, der für n Bits ausgelegt ist, kann mit 2^n Binärwerten (Codes) arbeiten, und zwar typischerweise im Bereich von 0 bis $2^n–1$. Dabei gibt es $2^n–1$ Wertübergänge.

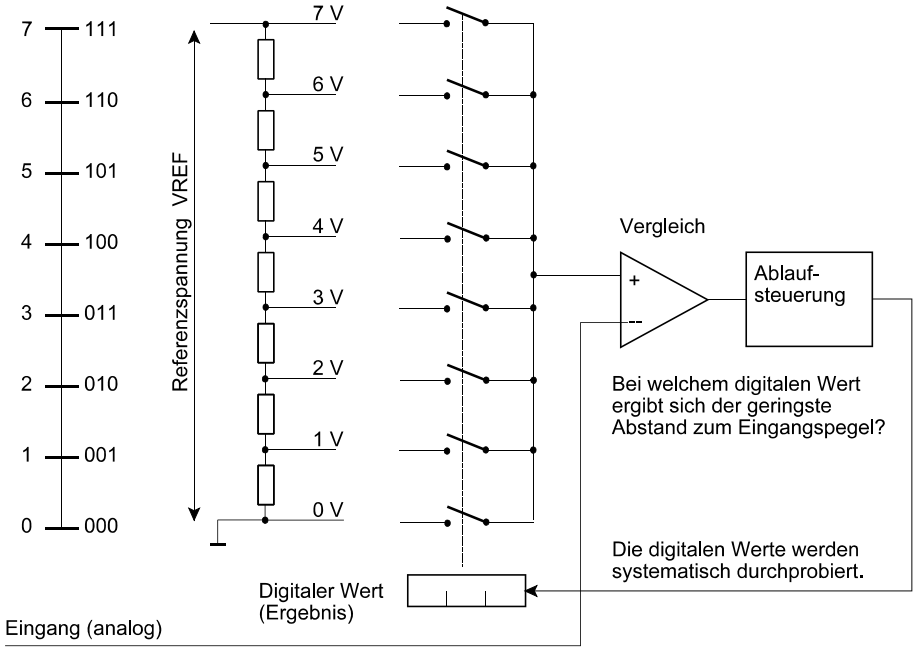


Abb. 15 Das Prinzip der Analog-Digital-Wandlung.Das analoge Eingangssignal wird mit festen Pegeln verglichen, die den Binärzahlen entsprechen. Die einfachste Modellvorstellung: Es werden alle digitalen Werte durchprobiert. Als Wandlungsergebnis gilt der Wert, dessen zugeordneter Pegel dem Eingangssignal am nächsten kommt.

Zwei wichtige Fachbegriffe: LSB und FSR:

- LSB = Least Significant Bit = die niedrigstwertige Bitposition.
- FS = Full Scale. Der jeweilige Höchstwert. Der analoge Höchstwert (Analog FS) entspricht typischerweise der Referenzspannung.
- FSR = Full Scale Range = der gesamte Wertebereich.

Es sind zwei Wertebereiche zu unterscheiden:

- der analoge (Analog FSR): der Bereich von der geringsten bis zur höchsten analogen Spannung (kontinuierlich).
- der digitale (Digital FSR): der Bereich aller Digitalwerte (diskret). Mit n Bits kann man 2^n Werte darstellen. Die einfachste Codierung: als natürliche (vorzeichenlose) Binärzahl. Wertebereich einer Binärzahl aus n Bits: von 0 bis 2^n-1 (000...00 bis 111...11 oder 00...0H bis FF...FH).

Die Auflösung wird als Wert der niedrigstwertigen Bitposition (LSB) angegeben. Sie wird auf den gesamten Wertebereich (FSR) bezogen. 1 LSB entspricht der Spannungsänderung, die eine Änderung in der niedrigstwertigen Bitposition hervorruft.

Jeder Binärwert codiert im Grunde dasVerhältnis zwischen der jeweiligen (analogen) Spannung und der festen Bezugsspannung (Referenzspannung). Es liegt nahe, folgende Entsprechungen anzusetzen:

Analoge Spannung	**Binärwert**
0 V	0
Referenzspannung (VREF)	Maximum (FF...FH)

Was ist ein LSB?
Ein n-Bit-Wandler unterteilt den gesamten Wertebereich in 2^n–1 gleichgroße Intervalle. Deshalb liegt diese Definition nahe:

$$1\ \text{LSB} = \frac{\text{FS}}{2^n - 1}$$

Der Nenner dieses Bruchs ist aber ein krummer Wert. Um bequemer rechnen zu können, nutzt man die Tatsache aus, daß es ohnehin nicht möglich ist, den Eingangsspannungsbereich bis zur Gleichheit mit der Referenzspannung auszunutzen. Die maximal zulässige analoge Eingangsspannung muß etwas unterhalb der Referenzspannung liegen, sonst ist das Wandlungsergebnis nicht genau genug. Auf der digitalen Seite ist der kleinste Unterschied ein LSB. Somit kann man ansetzen:

- Maximale analoge Eingangsspannung = Referenzspannung – 1 LSB.
- Der binäre Höchstwert (FF...FH) ergibt sich, wenn die maximale analoge Eingangsspannung anliegt.
- Die Referenzspannung (VREF) ist um 1 LSB höher.
- Die Referenzspannung (VREF = Analog FS) entspricht einem fiktiven binären Wert 2^n (1 Bit mehr, als der Wandler tatsächlich hat).
- Der tatsächliche binäre Höchstwert (Digital FS; FF...FH) ist gleich Analog FS – 1 LSB.

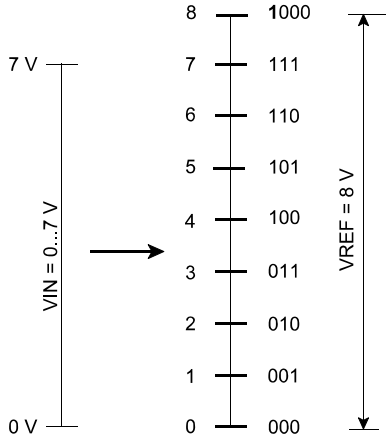


Abb. 16 Das Beispiel veranschaulicht einen 3-Bit-Wandler. Die Auflösung beträgt 1 V, die Referenzspannung 2^3 = 8 V. Die 8 möglichen Meßwerte entsprechen 8 Spannungswerten von 0 bis 7 V.

Somit gilt:

$$1\ \text{LSB} = \frac{\text{Analog FS}}{2^n} = \frac{\text{VREF}}{2^n}$$

Typische Auflösungen:

Bits	Binärwerte	Auflösung in % vom Endwert (% Full Scale)	Wert der niedrigstwertigen Bitposition (LSB)		
			Bereich = 20 V	Bereich = 5 V	Bereich = 2 V
8	256	0,39%	78.1mV	19.5mV	7.81mV
10	1024	0,098%	19.5mV	4.88mV	1.95mV
12	4096	0,0244%	4.88mV	1.22mV	488µV
14	16 384	0,0061% = 61 ppm	1.22mV	305µV	122µV
16	65 536	0,0015% = 15 ppm	305µV	76.3µV	30.5µV
18	262 144	0,000381% = 3,8 ppm	76.3µV	19.1µV	7.63µV
20	1 048 576	0,000095% = 0,95 ppm	19.1µV	4.78µV	1.91µV

Brauchbare Ergebnisse gewinnen
Unser Ziel ist, ein Digitalvoltmeter zu bauen, das tatsächliche Spannungswerte anzeigt. Die Binärzahlen vom Wandler beziehen sich aber auf die Referenzspannung. Also müssen wir umrechnen.

Die Referenzspannung (VREF) entspricht einem (fiktiven) maximalen Binärwert (Digital Maximum Scale DMS). Welche Spannung gibt dann ein beliebiger binärer Meßwert (Digital Measured Value DM) an? Das ist mittels Dreisatzrechnung zu lösen:

$$\mathrm{DMS} = \mathrm{VREF}$$
$$\mathrm{DM} = x$$

$$\frac{\mathrm{DMS}}{\mathrm{DM}} = \frac{\mathrm{VREF}}{x}$$

$$x = \frac{\mathrm{VREF}}{\mathrm{DMS}} \cdot \mathrm{DM}$$

VREF : DMS ist eine Konstante, die mit dem binären Meßwert zu multiplizieren ist.

Praktisches Rechnen:
Es kommt auf die Stellen nach dem Komma an. Wir können aber nur mit ganzen Zahlen rechnen. Deshalb müssen wir alles mit einem Skalierungsfaktor multiplizieren, der die Stellen, mit denen wir weiterarbeiten wollen (z. B. zwecks Anzeige), in den Bereich der ganzen Zahlen bringt. Wo das Komma stehen müßte, merken wir uns. Wir berücksichtigen es ggf. beim Formatieren der Anzeige.

Rechnerleichterung:
DMS ist üblicherweise eine Zweierpotenz (2^n). Besonders bequem läßt es sich dann rechnen, wenn auch die Referenzspannung einen Wert hat, der (entsprechend skaliert) einer Zweierpotenz entspricht. Typische Beispiele: 1,024 V; 2,048 V; 4,096 V; 8,192 V. Damit wird bei entsprechender Skalierung der Bruch VREF : DMS zu einer ganzen Zahl mit wenigen Stellen[2].

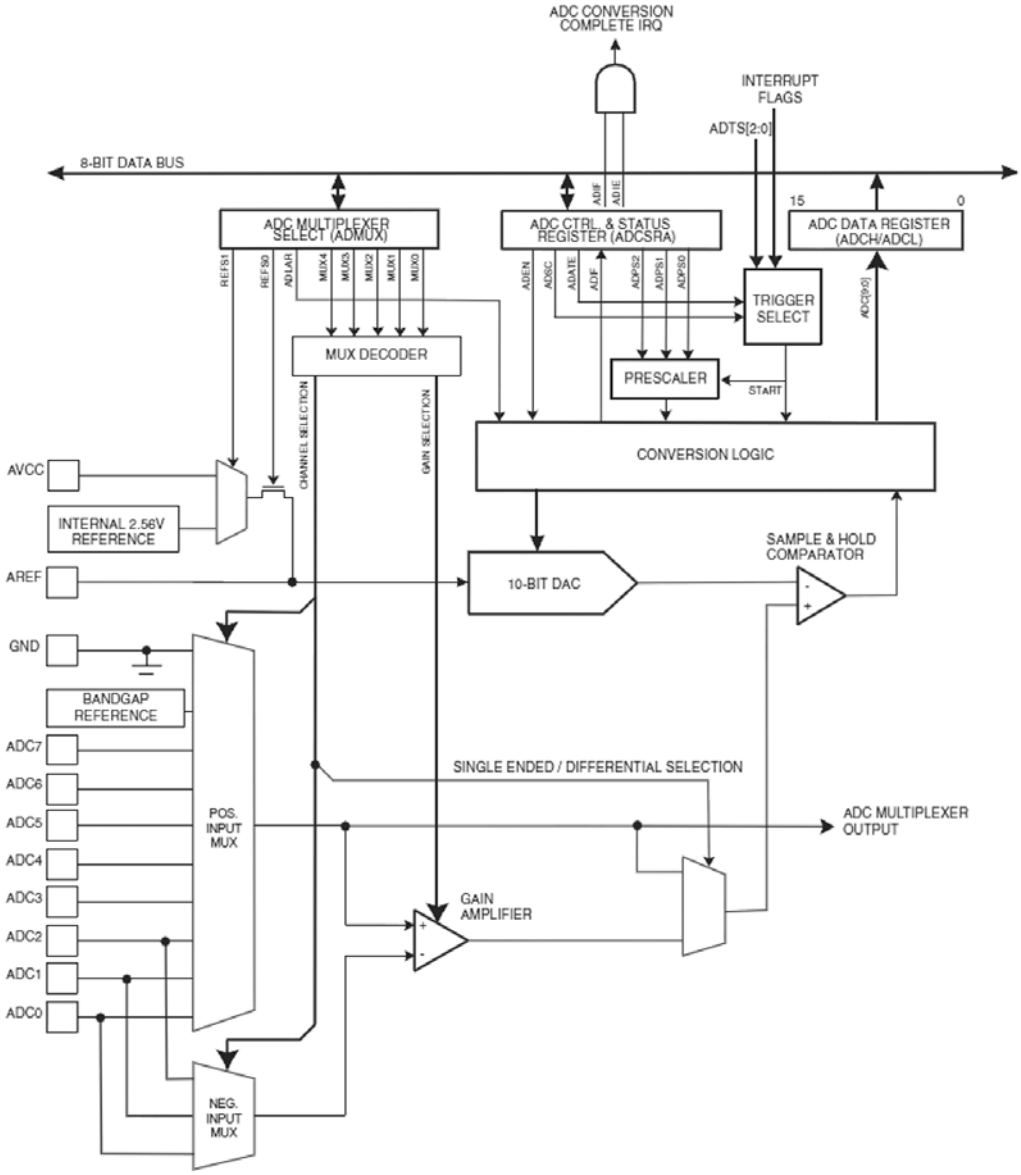
Die Praxis:

- Der Wandler im Atmel AVR liefert 10 Bits.
- Wir wählen eine Referenzspannung von 4,096 V.
- 1 LSB = 4,096 V : 1024 = 0,004 V.
- Um eine ganze Zahl zu erhalten, multiplizieren wir mit 1000. Das ergibt den Wert 4 bzw. 4 mV.

Spannungsanzeige in mV = gemessener Binärwert • 4 mV.

Der höchste Binärwert ist 1023. 1023 • 4 mV = 4092 mV. Wir können direkt mit den mV rechnen. Bei Bedarf setzen wir das Komma (in der Anzeige) zwischen die 4. und die 3. Stelle.

2: So daß man beim Rechnen z. B. mit 16 Bits auskommt und keine 32 Bits braucht. Der günstigste Fall: es ist auch eine Zweierpotenz. Dann kann man das Multiplizieren durch Linksverschieben ersetzten.



(Bildquelle: Atmel.)

Merkmale:

- Auflösung: 10 Bits,
- integrale Nichtlinearität: 0,5 LSB,
- absolute Genauigkeit: $\pm$ 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). Unser Analogsignal kommt über PortA, Bitposition 0.

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

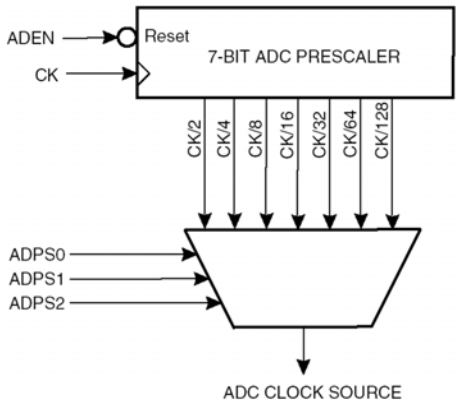
ADC Data Register (ADCH, ADCL). Ergebnis linksbündig (ADLAR = 1)

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

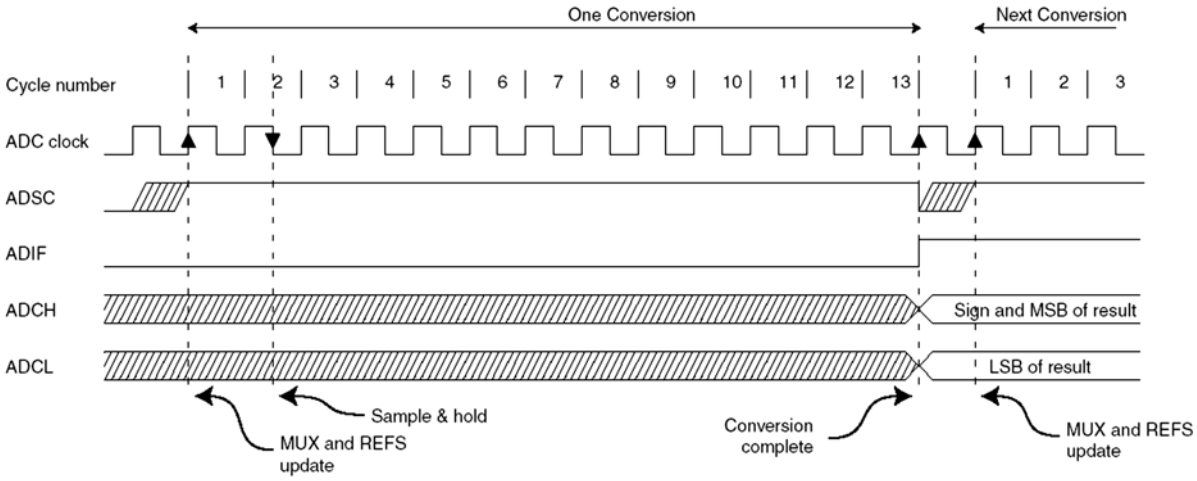
Ein Wandlungsablauf:

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.)

Versuchsanordnung:

- Port A des Starterkits mit Port A des Universaladapters 10b (der Bunten Kuh) verbinden.
- Masse und VCC mit den beiden äußeren Anschlüssen des Potentiometers im 19"-Einbaurahmen verbinden.
- Den Schleifer des Potentiometers mit Bit 0, Port A verbinden. Das eingebaute Digitalmultimeter mit Masse und dem Schleifer des Potentiometers verbinden.
- Port C mit den LEDs des Starterkits verbinden. Die LCD-Anzeige wird zunächst nicht genutzt.

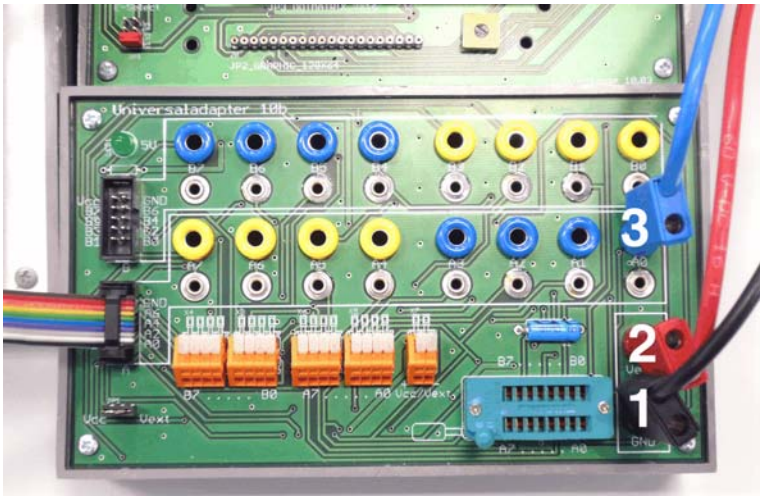


Abb. 17 Analogsignalanschluß über Universaladapter 10b (genannt die Bunte Kuh). 1 - Masse; 2 - Betriebsspannung; 3 - Analogeingang auf Pin A0. Das Bandkabel links führt zu Port A des Starterkits.

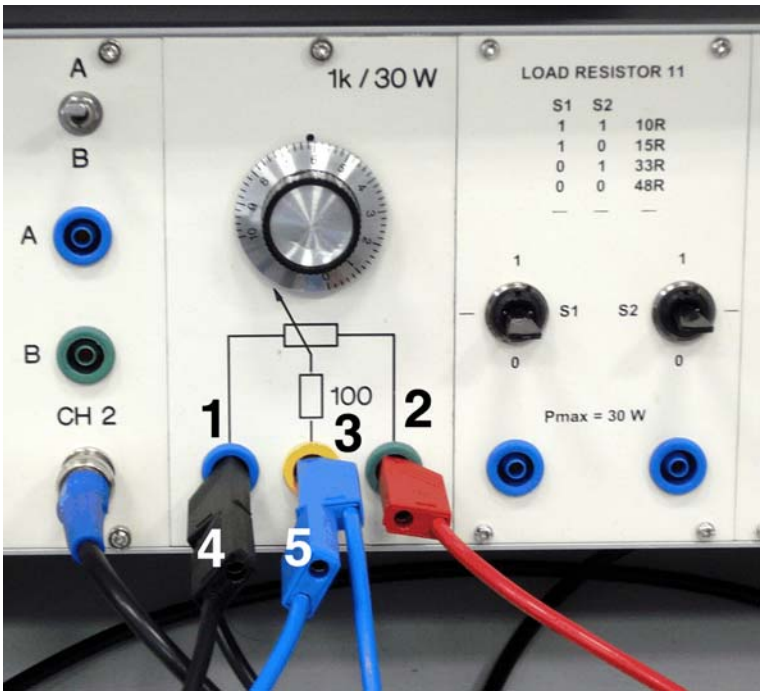


Abb. 18 Prüfsignalerzeugung mittels Potentiometer. 4 - Masse zum Digitalmultimeter; 5 - Eingangsspannung zum Digitalmultimeter.

1. Den Wandler initialisieren. Zyklische Binäranzeige über die LEDs (Funktionsprobe).
Referenzspannung (Starterkit) auf 5 V. Wir nutzen nur die höchstwertigen 8 Bits des Wandldungsergebnisses. Sie werden direkt auf den LEDs dargestellt. Das Potentiometer vom linken zum rechten Anschlag langsam durchdrehen. Dann wieder zurück. Die Anzeige des Digitalmultimeters beachten. Wann schalten die einzelnen Bitpositionen? Die höchstwertige (Bit 7) muß ungefähr in der Mitte umschalten (bei 2, 5V). Bit 6 muß bei jeder Erhöhung / Verminderung um 1,25 V umschalten usw.

Programmbeispiel: 11.

2. Digitalvoltmeter mit LCD-Anzeige.

Port C wieder mit der LCD-Anzeige verbinden. Referenzspannung (Starterkit) auf 4,1 V (als Annäherung an 4,096 V). Dann genügt es, die Binärzahl vom Wandler mit 4 zu multiplizieren, um den Spannungswert in mV zu erhalten. Hier nutzen wir alle 10 Bits des Wandlers aus. Der höchste Ausgabewert ist dann 4 • 1023. Dazu genügen 16 Bits. Die Wandlung ins Dezimale kennen wir vom Versuch 1 her. Ausprobieren: am Potentiometer langsam drehen und die LCD-Anzeige mit der Anzeige des Digitalmultimeters vergleichen. Spannung nicht über 4 V. So richtig genau ist es offensichtlich nicht[3] ...

Programmbeispiel: 12.

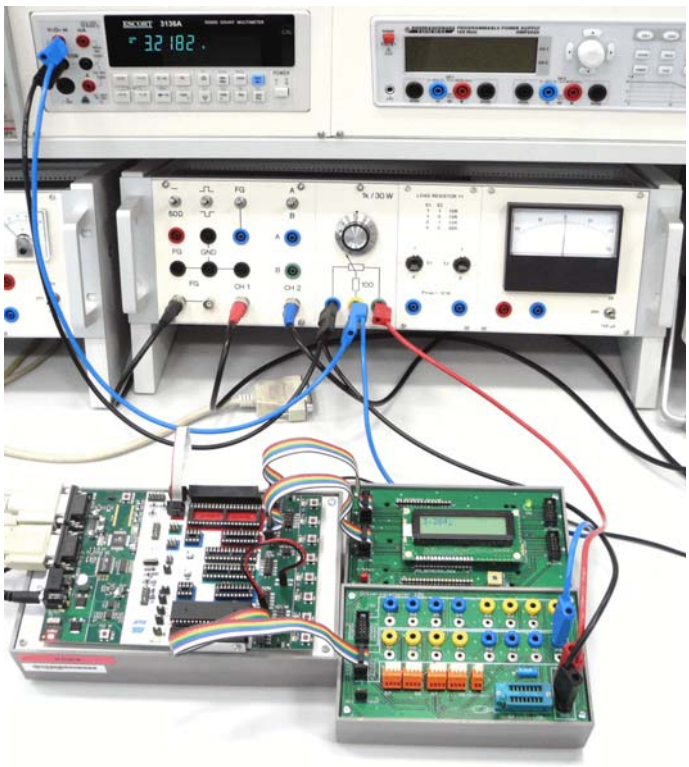


Abb. 19 Digitalvoltmeter mit LCD-Anzeige.

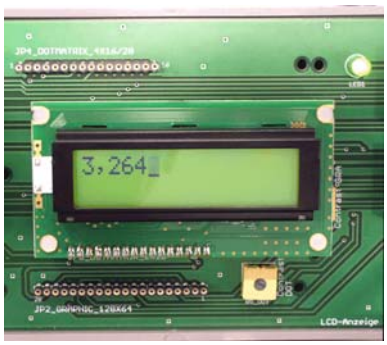


Abb. 20 Kontrolle mittels Digitalmultimeter. Daß es schon in der dritten Stelle nicht mehr übereinstimmt, wollen wir vorerst hinnehmen – jeder hat mal ganz einfach angefangen...

3: Das Genauigkeitsproblem ist eine Wissenschaft für sich. Es gibt eine Vielzahl von Einflußgrößen. Zu den wichtigsten gehört die Referenzspannung. Wer will, kann sie nachmessen (Pin 32 des Prozessors oder AREF-Jumper des Starterkits).

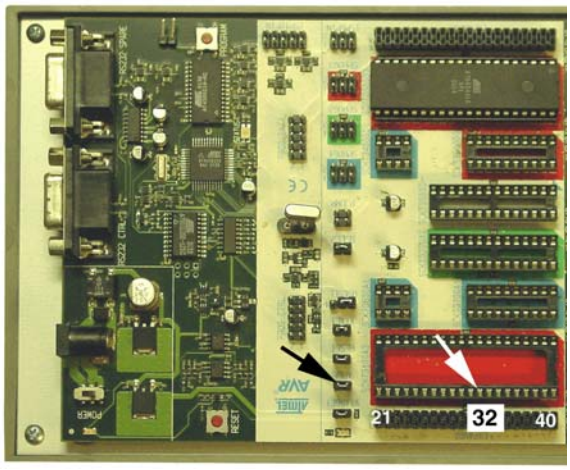


Abb. 21 Wo die Referenzspannung (hier: AREF) nachgemessen werden kann (gegen Masse...).

Zusatzaufgaben (selbständig zu lösen):

1. Meßwertausgabe aufs Terminal. Nach jeder Ausgabe eine neue Zeile. Nur dann ausgeben, wenn sich der digitale Meßwert geändert hat, denn sonst wird nur der Terminalbildschirm vollgeschrieben.

2. Auf eine andere Referenzspannung ändern, z. B. auf 5 V oder auf den Wert, den Sie tatsächlich gemessen haben. Wie müssen Sie das Rechenergebnis skalieren, um alle wichtigen Stellen in den Bereich der ganzen Zahlen zu überführen? Genügen dafür 16 Bits?

3. Digitalvoltmeter mit Siebensegmentanzeige.

Die LCD-Anzeige und die Verbindung zur seriellen Schnittstelle abbauen. Die Siebensegmentanzeige anschließen. Weil nur drei Ports frei verfügbar sind, können wir nur drei Stellen ausnutzen. Die Einerstelle (der Millivolts) müssen wir weglassen. Tausenderstelle an Port D, Hunderterstelle an Port B, Zehnerstelle an Port C. Ggf. den Dezimalpunkt sichtbar machen (Jumper der Tausenderstelle abziehen). Im Programm müssen die Initalisierungen der seriellen Schnittstelle und der LCD-Anzeige entfernt werden.

Programmbeispiel: 13.

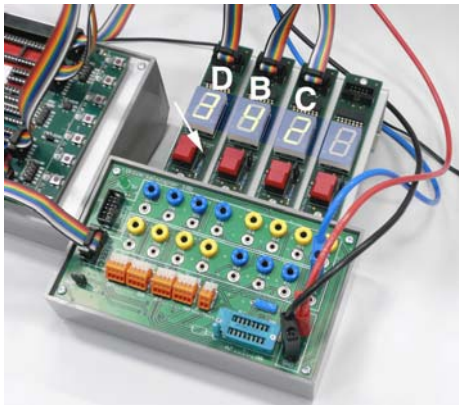


Abb. 22 Die Siebensegmentanzeige anschließen. Der Pfeil zeigt auf den Jumper der höchstwertigen Stelle. Abziehen, um den Dezimalpunkt anzuzeigen.

Die ASCII-Codetabelle:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
20:		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
30:	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40:	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50:	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
60:	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70:	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

Ablesebeispiele: “1" = 0x31; “A” = 0x41; “t” = 0x74.