Praktikum Mikrocontrollertechnik SS 2014
Versuch 2

Stand: 5. 5. 2014

Einführung in die C- und Assemblerprogrammierung Atmel AVR.

Aufgaben:

1. Anwendungsprogrammierung in C. Schwierigkeitsgrad etwas höher als in Versuch 1.
2. Grundlagen der Assemblerprogrammierung kennenlernen.

C-Programmierung

- Das Projekt des ersten Versuchs aufrufen oder eine neues C-Projekt einrichten (s. die Anleitung zum ersten Versuch).
- Das Programm **V2C_14_template.c** in das Editorfenster des Projekts kopieren.
- Die Datei **V2C_14_examples.c** im Editor öffnen, um ggf. Programmbeispiele übernehmen zu können.
- Optimierung ausschalten (“O0").

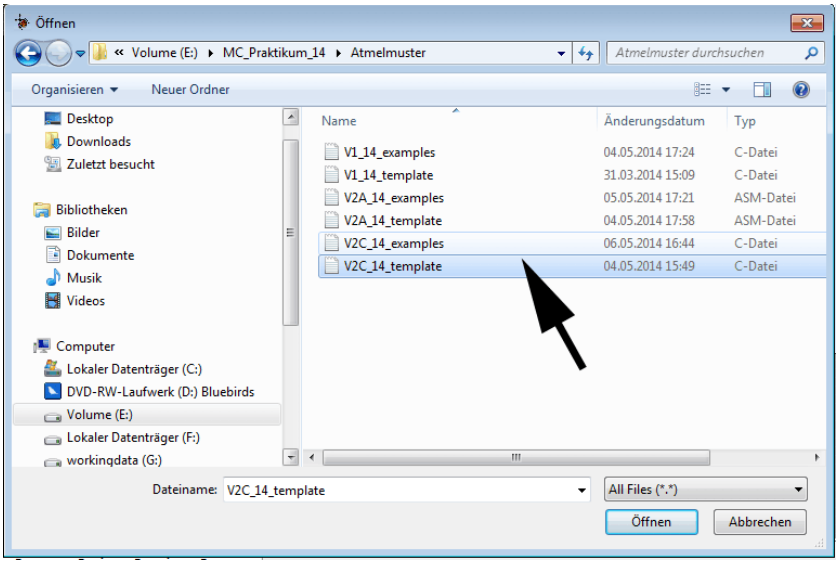


Abb. 1 Musterdateien für die C-Programmierung.

Aufgabe C1: Zweistellige Siebensegmentanzeige.

Die Siebensegmentanzeige 09/13 anschließen. Wir verwenden zwei Anzeigemodule, die wir mit den Ports B und C verbinden.

Jeder der Ports:

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

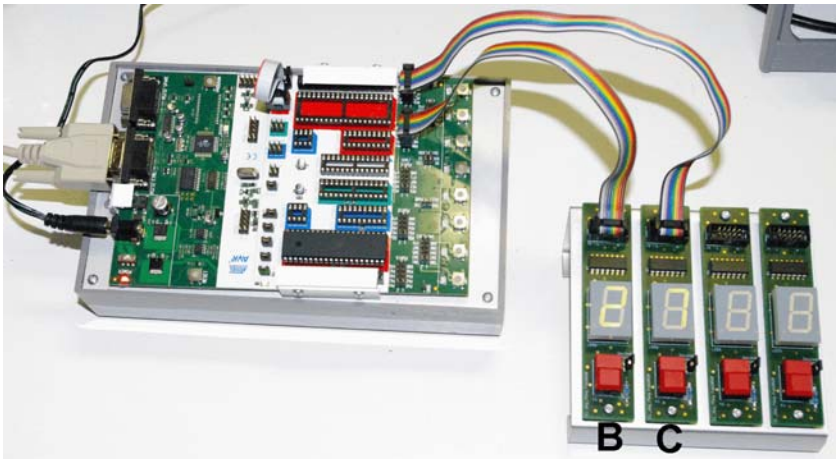


Abb. 2 Starterkit mit Siebensegmentanzeige. LEDs und Tasten sind aktiv Low.

Jeder der Ports:

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

1. Zyklisch von 0 bis 99 zählen.
Genügend Zeitverzögerung zwischen den Zählschritten vorsehen (z. B. 200 ms), so daß der Zählvorgang visuell verfolgt werden kann. Worauf es besonders ankommt, ist die Wandlung vom Binären (der Zählvariablen in der Schleife) ins Dezimale und von da aus in die Segmentbitmuster. Wir können hierzu die Grundrechenarten ausnutzen; Tricksen müssen wir nicht. Ein deutliches Beispiel für die Vorteile einer "höheren" Programmiersprache...

Programmbeispiel: 1.

2. Handzähler (zum Zählen von Autos, Schafen, BVB-Fans usw.).
Rechte Taste zum Zählen (+ 1), linke Taste zum Löschen. Bei 99 mit Zählen aufhören. Probeweise die Entprellung weglassen (auskommentieren). Die Tasten auf den LED-Anzeigen prellen typischerweise gut und lassen so den Prelleffekt deutlich erkennen.

Programmbeispiel: 2.

3. Zyklisch von 1 bis 49 zählen (Lottozahlen).
Genügend Zeitverzögerung zwischen den Zählschritten vorsehen (z. B. 200 ms), so daß der Zählvorgang visuell verfolgt werden kann.

Programmbeispiel: 3.

4. Lottozahlen ziehen.
Schnell zählen, solange die rechte Taste gedrückt ist. Bei losgelassener Taste hält der Zähler an.

Programmbeispiel: 4.

Aufgabe C2: Tageszeituhr mit vierstelliger Siebensegmentanzeige.

Alle vier Siebensegment-Anzeigemodule an das Starterkit anschließen.

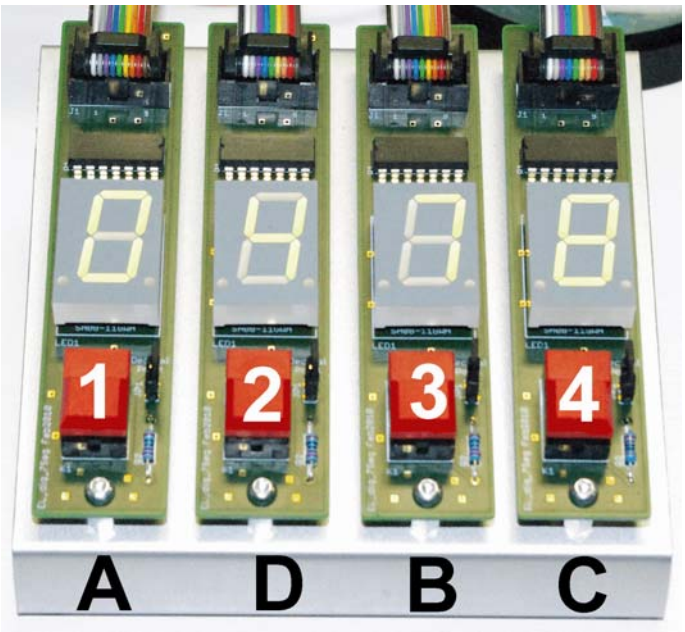


Abb. 3 So werden die Anzeigemodule angeschlossen.

1. Tageszeituhr (1). Elementare Zeitzählung von 00:00 bis 23:59.

Erst einmal das Zählen an sich ausprobieren. Die Uhrzeit hat es in sich... Wir wollen hier binär zählen und zwecks Anzeige ins Dezimale wandeln. Eine solche Uhr schaltet alle Minute weiter. Der Zähltakt ist also die Minute. Der Tag hat 1440 Minuten. Also modulo 1440 zählen. 16 Bits genügen.

Wenn wir einen Minutenwert durch 60 teilen, erhalten wir als Quotienten die Stunde und als Rest die verbleibenden Minuten. Beide Werte sind in jeweils zweistellige Dezimalzahlen zu wandeln. Hierzu werden sie durch 10 geteilt. Der Quotient ergibt die höherwertige Dezimalstelle, der Rest die niederwertige.

Nun können wir nicht wirklich mit einem Minutentakt arbeiten, denn dann würde es viel zu lange dauern, den Zählablauf zu beobachten. Deshalb verwenden wir zunächst einen Takt von 100 ms.

Programmbeispiel: 5.

2. Tageszeituhr (2). Wählbarer Zähltakt.

Jetzt soll der Zähltakt über die Tasten unter den Siebensegmentanzeigen ausgewählt werden. Um jeweilige Taste nicht dauernd niederhalten zu müssen, übernehmen wir die Zähltaktauswahl in eine Merkvariable **status**, die wir neu definieren. Zudem müssen die Bits 7 der E-A-Ports Eingänge werden.

- Taste 1: Zählen mit 1 ms.
- Taste 2: Zählen mit 10 ms.
- Taste 3: Zählen mit 100 ms.
- Taste 4: Zählen mit 1 s.

Die Statusvariable wird auf einfache und schmucklose Weise gesetzt und dann mit einer **switch**-Anweisung ausgewertet.

Sehen Sie nach, wie lang das Maschinenprogramm ist.

Programmbeispiel: 6.

3. Stoppuhr (1). Überprüfen der Zählfunktion.
Die Stoppuhr soll von 00.00 bis 99.99 Sekunden zählen, und zwar mit einer Auflösung von 1/100 s. Um die Zählabläufe überprüfen zu können, behalten wir die Zähltaktauswahl der vorigen Aufgabe bei, stellen aber die Anzeige auf reine Dezimalzählung um (alle Stellen von 0...9).

Programmbeispiel: 7.

4. Stoppuhr (2).
Taste 1 dient als Löschtaste, Taste 4 als Start- und Stoptaste. Der Zähltakt dauert jetzt 1/100 s = 10 ms. Das Programm muß alles allein leisten: den Zähltakt herstellen, zählen, wandeln, die Tasten abfragen und die Grundfunktionen einer Stoppuhr erledigen (laufen, anhalten, löschen). Der Zustandsautoman ist ein bewährtes Prinzip, eine solche Aufgabenvielfalt zu beherrschen. Wir verwenden folgende Zustandscodierung:

- 0 = Grundzustand. Ruhe
- 1 = Taste 4 wurde betätigt. Uhr zählt.
- 2 = Uhr zählt. Taste 4 wurde losgelassen.
- 3 = Uhr steht. Taste 4 wurde betätigt. Übergang nach 0, wenn Taste wieder losgelassen.
- 9 = Zählerstand 99.99 = Zählende. Wenn Taste 1 betätigt (Löschen), dann Übergang nach 10.
- 10 = Löschtaste betätigt. Löschen. Wenn beide Tasten losgelassen, Übergang nach 0.

Programmbeispiel: 8.

Das Programmbeispiel veranschaulicht, daß ein solcher Zustandsautomat auf überschaubare, reguläre Weise mit einer **switch**-Anweisung ausprogrammiert werden kann.

Assemblerprogrammierung

- Ein neues Projekt einrichten, aber diesmal als Assemblerprogramm.
- Das Programm **V2A_14_template.asm** in das Editorfenster des Projekts kopieren.
- Die Datei **V2A_14_examples.asm** im Editor öffnen, um ggf. Programmbeispiele übernehmen zu können.

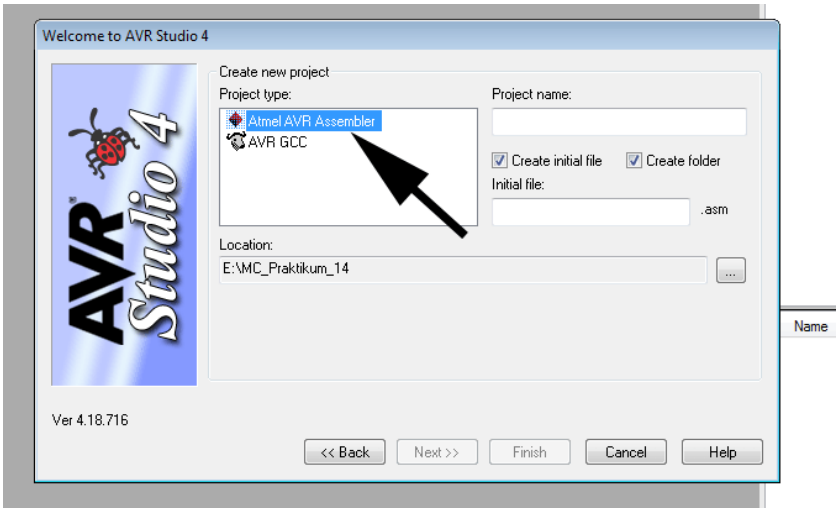


Abb. 4 So beginnt das Einrichten eines Asssemblerprojekts. Dann geht's weiter wie im Versuch 1.

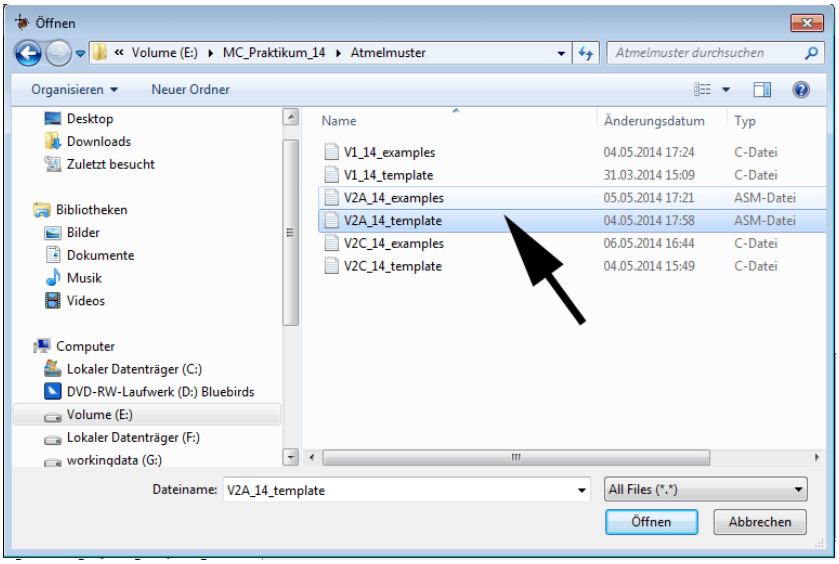


Abb. 5 Musterdateien für die Assemblerprogrammierung.

Selbst programmieren

Zum Aufbau eines typischen Assemblerprogramms s. Abb. 6. Der Assembler achtet nicht auf die Groß- und Kleinschreibung. PORTA = porta = PoRtA usw. Kommentare werden mit einem Semikolon eingeleitet. Es sind aber auch die Kommentarsymbolik von C zulässig (//; /*...*/). Syntaktisch korrekte Befehlsbezeichner (Mnemonics) werden blau dargestellt, Kommentare grün.

Aufgabe A1: Siebensegmentanzeige.

Die Siebensegmentanzeige 09/13 bleibt so angeschlossen, wie sie ist.

1. Zählen in einer Ziffernstelle.

Wir zählen zyklisch von 0 bis 9 und zeigen die jeweilige Ziffer über Port A an. Die Umsetzung der Binärzahl in das zugehörige Siebensegment-Bitmuster erledigen wir über eine Tabelle im Flash. Die Tabellenzugriffe erfolgen mit LPM-Befehlen, denen eine Adreßrechung vorausgeht. Es muß alles bis ins Kleinste ausprogrammiert werden...

Programmbeispiel: 1.

2. In allen vier Ziffernstellen dezimal zählen (von 0000 bis 9999).

Die Maschine hat keine Divisionsbefehle. Somit können wir nicht auf einfache Weise Binärzahlen ins Dezimale wandeln. Also zählen wir stellenweise dezimal. Für jede Dezimalstelle nutzen wir ein Register. Zunächst werden alle Registerinhalte in Siebensegmentmuster gewandelt. Es lohnt sich, hierfür ein Unterprogramm zu schreiben. **showsegs** erhält die Binärzahl in r16 und gibt das Bitmuster wieder in r16 zurück.

Es wird erst die niedrigstwertige Stelle gezählt. Hat sie den Wert 9, so wird sie im nächsten Zähldurchlauf wieder zu 0, und die nachfolgende Stelle wird weitergezählt. Sinngemäß geht es mit den übrigen Stellen weiter (Löschen beim Zählen über 9 hinaus und Zählen in der nächste Stelle).

Programmbeispiel: 2.

```
.include "m16def.inc"
```
Die fertigen Funktionen und Definitionen

```
.CSEG
```
Das Programmsegment (= Flash)

```
.ORG 0x0000
```
Beginn an Adresse 0

```
jmp anfang
jmp extint0
jmp extint1
jmp tim2comp
jmp tim2ovf
jmp tim1capt
jmp tim1compa
jmp timicompb
jmp tim1ovf
jmp tim0ovf
jmp spistc
jmp usartrxc
jmp usartudre
jmp usarttxc
jmp adcrdy
jmp eeready
jmp anacomp
jmp twi
jmp extint2
jmp tim0comp
jmp spmrdy
```
Die Interruptvektortabelle

Initialisierung:

```
anfang:
    ldi     r16,low(RAMEND)
    out     SPL,r16
    ldi     r16,high(RAMEND)
    out     SPH,r16
```
1. Stackpointer

```
ldi r16,0xff
out porta,r16
out portb,r16
out portc,r16
out portd,r16
out ddra,r16
out ddrb,r16
out ddrc,r16
out ddrd,r16
```
2. E-A-Ports

Die Anwendung (Grundsteuerschleife)

Texte und Konstanten

Die Unterbrechungsbehandler:

```
extint0:

extint1:

tim2comp:
```
usw.

```
twi:

extint2:

tim0comp:

spmrdy:
```

Allgemeine Unterprogramme

Abb. 6 Der Aufbau eines typischen Assemblerprogramms.

3. Mit wählbarem Zähltakt zählen.
Die Tastenbits müssen Eingänge werden. Die Tastenabfrage erfolgt mit Bitabfragebefehlen (**sbis**). Das Ganze ist etwas trickreich. Damit der Ablauf schön aussieht, verwenden wir Unterprogramme, um die Zeitwerte zu setzen.

Programmbeispiel: 3.

Aufgabe A2: Tageszeituhr.

Es ist von 00:00 bis 23:59 zu zählen. Die Zähltaktauswahl behalten wir bei. Die Zählung muß stellenweise ausprogrammiert werden. In der niederwertigen Minutenstelle zählen wir modulo 10, in der höherwertigen modulo 6. In der niederwertigen Stundenstelle wird modulo 10 gezählt, aber auch die höherwertige Stundenstelle ausgewertet, um den Übergang auf den Wert 24 zu erkennen. Wurde dieser Wert erkennt, so wird der gesamte Zähler gelöscht (Übergang von 23:59 nach 00:00).

Sehen Sie nach, wie lang das Maschinenprogramm ist. Vergleichen Sie diesen Wert mit dem vom C-Programmbeispiel 6.

Programmbeispiel: 4.

Aufgabe A3: Musik.

Die Siebensegmentanzeige wird abgebaut. Die Tasten des Starterkits werden mit Port D verbunden. An Bitposition 0 von Port C wird – über einen Universaladapter 10b (die Bunte Kuh) ein Lautsprecher angeschlossen.

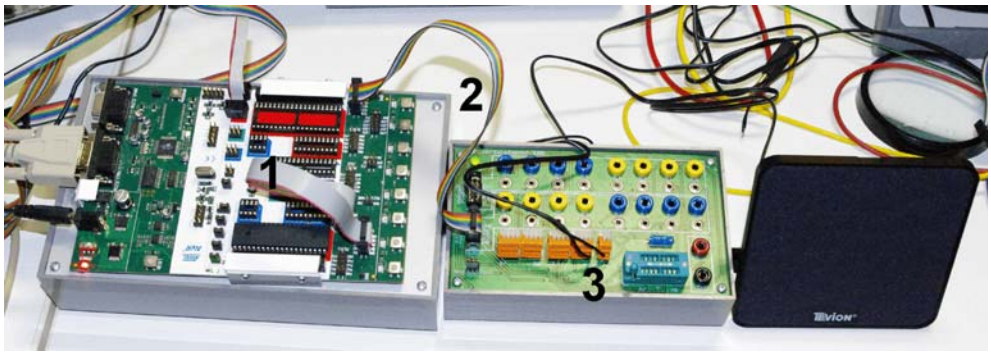


Abb. 7 Das Musikinstrument (1). 1 - Tasten des Starerkits an Port D. 2 - Port C Starterkit an Port A Bunte Kuh. 3 - Lautsprecheranschluß.

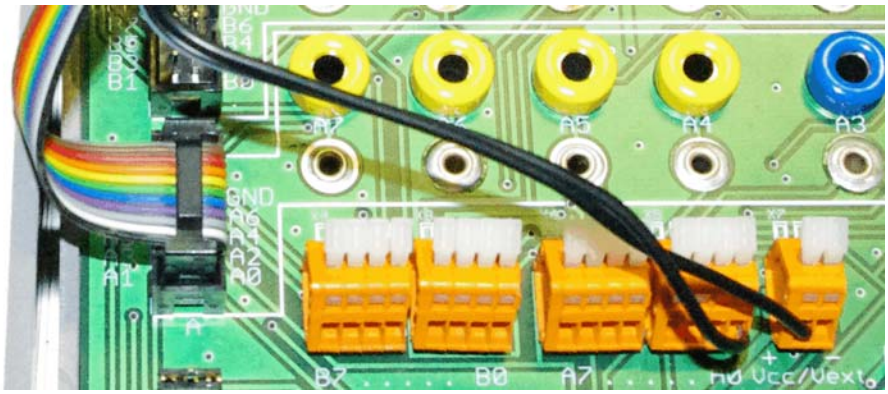


Abb. 8 Das Musikinstrument (2). Der Lautsprecheranschluß. Von VCC nach Bit 0.

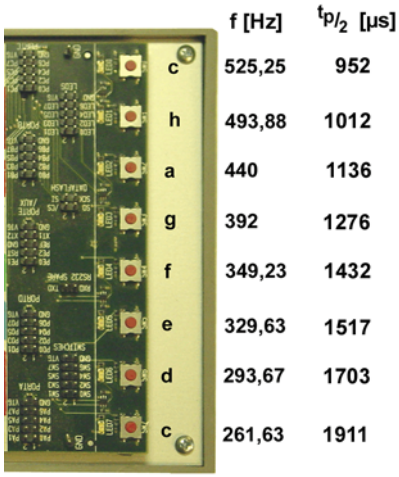
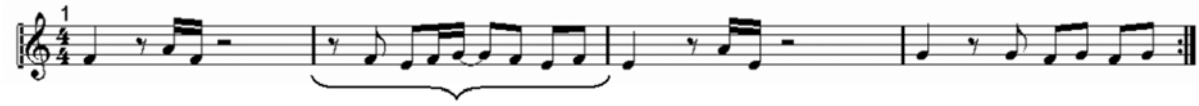


Abb. 9 Das Musikinstrument (3). So wirken die Tasten.

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

Abb. 10 Was man von Musik wissen sollte (1).

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

Tabelle 1 Was man von Musik wissen sollte (2). Die chromatische Tonleiter.

1. Musikinstrument (1).
Die Tasten werden zyklisch abgefragt. Eine betätigte Taste veranlaßt die Abgabe eines Tonsignals (Bit 0, Port C), das über den Lautsprecher hörbar wird. Das Tonsignal ist eine symmetrische Rechteckwelle (Tastverhältnis 50:50) mit jeweils bestimmter Frequenz (gemäß chromatischer Tonleiter). Wenn kein Ton abgegeben wird, Ausgänge auf High schalten, damit der Lautsprecher keinen Gleichstrom zieht.

Das Unterprogramm **play** stellt eine Impuslperiode dar, indem es die Mikrosekunden einer Halbperiode laut Tabelle auszählt. Solange eine Taste gedrückt ist, wird es immer wieder aufgerufen. Um uns das Eingeben der Zahlen zu erleichtern, schreiben wir einen Makro **setsound**.

Programmbeispiel: 5.

2. Musikinstrument (2).
Das Musikinstruemnt soll jetzt Noten mit konstanter Dauer spielen. Jeder Tastendruck soll bewirken, daß der Impulszug einer 16tel-Note abgegeben wird, gefolgt von einer ebenso langen Pause (125 ms). Wieviele Perioden jeweils zu spielen sind, steht in Tabelle 1. Wir können uns die Werte aber auch vom Assembler ausrechnen lassen (Makro **setsound)**.

Programmbeispiel: 6.

3. Musik automatisch erzeugen.
Eine Zeichenkette im Flash bestimmt, was gespielt wird. Jedes Zeichen erzeugt eine 16tel-Note oder eine entsprechende Pause. Der Code ist einfach. Noten = c, h, a, g, f, e, d, k; Pause = p, Ende der Zeichenkette = 0 (00H). Die Zeichen werden nacheinander mit LPM-Befehlen gelesen. Längere Töne oder Pausen ergeben sich durch entsprechend viele aufeinanderfolgende Zeichen. Viel Erfolg beim Komponieren...

Programmbeispiel: 7.