Betrieb der RTX-03x AD-Karten unter Linux

zusammengestellt von:

Olaf Schultz
Buxtehuder Straße 57
21073 Harburg

Stand: 29. März 1999

Dieser Text beschreibt die Inbetriebname der AD-Wandlerkarten RTX-03A (12 bit) und RTX-03B (14 bit) von "Real Time Industrial Ltd." unter Linux. Diese Karten sind in Deutschland eher unter den AD-Wandlerkarten von Conrad Electronic mit den Bestellnummern 976121 für 119,95 DM und 976148 für 219,95 DM bekannt.

Vielleicht spinnen andere den hier ausgelegten Faden weiter und tragen ebenso wie Michael Müller (e-mail: moeller.heinemann@t-online.de), der mich durch seine Temperaturerfassung mit der 12-Bit-Karte zur Eigenintiative angestachelt hat, konstruktiv zum Inhalt und Struktur des Textes bei. Da ich nur dummdreistanwendender Maschinenbauer bin, sind im elektro- und programmtechnischen Abschnitt sicherlich einige zumindest unsaubere Formulierungen enthalten.

Ein ganz besonderes Lob an Jens Eike Jedsau (FH-Bielefeld) (email: jesau@warpserv.fh-bielefeld.de) für die Entwicklung des Treibers. Zusammen mit Robert Weis (email: weis@kelvin.aphys.uni-heidelberg.de), der eine vergleichbare Karte einsetzt, wurden von uns dreien der Treiber getestet und die maximale Wandlungsrate rausgekitzelt.

Nomenklatur

Symbol	Einheit	Bedeutung
f	Hz	Frequenz
I	A	Strom
P	Nm/s=VA	Leistung
Q	-	Gütefaktor
R	V/A=Ω	Widerstand
R_a	V/A	Außenwiderstand
R_i	V/A	Innenwiderstand
n	-	Auflösung (Bit)
t	s	Zeit
T	s	Periodenzeit
U	V	Spannung
U_{eff}	V	Effektivwert der Spannung
$\lvert\bar{U}\rvert$	V	Gleichrichtwert der Spannung
$\hat{U}$	V	Spitzen/ Amplitudenwert der Spannung

Inhaltsverzeichnis

Autor: Olaf Schultz
Buxtehuder Straße 57
21073 Hamburg-Harburg
Tel.: 040-77 51 43
WWW: http://www.tu-harburg.de/~kt2os/privat/
WWW: http://experte.kt2.tu-harburg.de:8015
E-Mail: O.Schultz@tu-harburg.de

Stand: 29. März 1999

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Auch mit LinuX kann man Meßtechnik betreiben. Die Datenerfassung läuft normalerweise mit AD-Karten ab. Das Programmieren dieser Karten ist nicht immer einfach und es gibt erst wenige Karten, die von LinuX untersützt werden. Die Kartenhersteller sehen derzeit keinen Markt in LinuX, so daß die LinuX-Gemeinde auf Eigeninitiative angewiesen ist. Neben dem weiteren Manko des Preises (es ist ohne weiteres möglich mehrere Tausend DM für eine Karte auszugeben) ist die Hardwareprogrammierung ein Stolperstein. National Instruments hat alleine in die universellen Treiber für M$-OS einige Mannjahre gesteckt. Das kann hier vom Autor nicht geleistet werden. Darum schildert dieser Text die einfache und schmerzlose Inbetriebnahme der Conrad-AD-Wandlerkarten für derzeit 130 bzw. 210 DM.

Für die Programmierung und Anwendung unter Linux sind die von Conrad angebotenen Programme (965782 für 39,95 DM inkl. Win32s) und DLLs (965774 für 19,95 DM) nicht notwendig. Bei mir hat sich das RTX-View unter WfW 3.11 als ziemlich unstabil erwiesen, außerdem kann es maximal alle 100 ms Werte abholen.[1] Unter LinuX habe ich auf bis zu 5,x kHz Summenabtastrate gebracht.

[1]Was erwartet man auch anderes von M$-Ware.

2 Hardware

Die Karten wandelt mit dem Verfahren der sukzessiven Approximation (Wägevefahren) (vergl. Bild 1). Hierbei wird die zu bestimmende Spannung (Eingangssignal) mit einer Spannung (aus dem DA-Wandler) in einem hier als Komperator verwendetem Operationsverstärker LM311 (OP-AMP) verglichen. Beim Meßbeginn wird das Sukzessive Approximationsregister auf Null gesetzt. Anschließend wird das höchste Bit gesetzt und geprüft, ob die Eingangsspannung $U_e > U(Z)$ ist. Ist dies der Fall, so bleibt es gesetzt, andernfalls wird es auf 0 zurückgesetzt. Der Vorgang wiederholt sich in absteigender Folge so lange, bis alle Bits bestimmt sind. Der Nachteil des Verfahrens ist, daß für jedes Bit ein Takt notwendig ist. Hier heist das, daß die von Conrad angegeben Wandlungsraten bei weitem nicht zu erzielen sind. Der 14-Bit-DA-Wandler hat eine Wandlungszeit von 2 μs, als AD-Konverter sind das dann mindestens (14+1)×2 μs=30 μs oder 33 kHz Summenabtastrate ohne Overhead zum Nullsetzen etc.

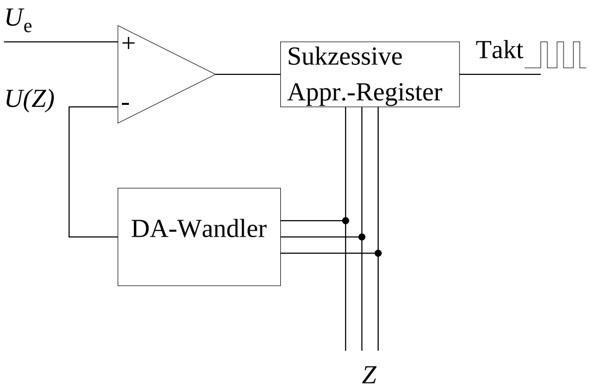

Bild 1: Ablaufplan der sukzessiven Approximation

Der Nachteil dieses preiswerten Verfahrens ist ein enormer Overhead auf der Programmseite, sprich Rechnerbelastung.

Der Ausgang der Karte ist eine 25-polige SUB-D-Buchse (female). An Ihr liegen die 16 AD-Kanäle, beide DA-Kanäle, drei Masse-Leitungen, +5 V, +12 V, -5 V und -12 V.

2.1 RTX-03A

Diese Karte verfügt nur über die Spannungsbereiche -8,5–8,5 V und 0–8,5 V im DA und AD-Bereich. Des weiteren ist sie nur mit einem DA-Wandler versehen.

2.2 RTX-03B

Als Wandler dient der DA-Konverter MP7614JN von EXAR. Dieser weist eine Wandlungszeit von typ. 2 μs auf.

Der zweite DA-Kanal ist optional, d.h. die Karte wird mit einem Kanal geliefert und kann durch Nachrüsten eines MP7614JN auf zwei DA-Kanäle aufgerüstet werden.

Die Karte kann über Jumper auf folgende Wandlungsbereiche eingestellt werden: 0–8,5 V, 0–5 V, 0–2,5 V, -8,5–8,5 V, -5–5 V und -2,5–2,5 V. Diese Spannungen sind jeweils für die AD und die beiden DA-Kanäle getrennt einstellbar.

2.3 Resourcen

Die Karten können auf folgende Adressen geschaltet werden: 200h–209h, 210h–219h, 220–229h . . . 2B0h–2B9h.

Die Belegung der Register der Karten sind in Tabelle 1 wiedergegeben.

Tabelle 1: Adressenbelegung der Karte

Adresse	Bedeutung
Port+0	Kanalwahl
Port+1	AD Low-Byte
Port+2	AD High-Byte
Port+3	Wandlung anstoßen
Port+4	AD readf
Port+5	AD reads
Port+6	DA0 Low-Byte
Port+7	DA0 High-Byte
Port+8	DA1 Low-Byte
Port+9	DA1 High-Byte

Der zweite DA-Kanal mit Port+8 und Port+9 sind nur auf der RTX-03B ansprechbar.

3 Programmierung

Um aus den beiden ausgelesenen Bytes, Low (L) und High (H) einen Wandlungswert W zu generieren wird folgende Rechnung angestellt.

$$W_{\mathrm{n\ Bit}} = \left(H - 2^{n-8} \mathrm{INT} \left(\frac{B}{2^{n-8}} \right) \right) 256 + L \tag{1}$$

Dies kommt dem Kappen der höchstwertigen Bits im High-Byte gleich. Dies kann u.a. auch durch eine bitweise AND-Verknüpfung mit 0x0F für den 12-Bit bzw. 0x3F für den 14-Bit-Wandler auf das High-Byte erreicht werden.

Um die gewandelte Spannung zu bestimmen wird entsprechend eine der folgenden linearen Gleichungen berechnet:

$$U_{\mathrm{bipolar},\ n\ \mathrm{bit}} = U_{\mathrm{max.}} \left(\frac{W_{n\ \mathrm{Bit}}}{2^{n-1}} - 1 \right) \tag{2}$$

$$U_{\mathrm{unipolar},\ n\ \mathrm{bit}} = U_{\mathrm{max.}} \frac{W_{n\ \mathrm{Bit}}}{2^{n}} \tag{3}$$

3.1 Ohne Treiber

Hierzu ziehe man das Programm von Michael Müller oder meines heran. Es ist eigentlich gnadelos einfach, läßt ich allerdings nur als root ausführen. Gut, das mag verschmerzbar sein, da Meßrechner i.d.R. eh Daddelrechner und keine Server sind (oh, jetzt brüllen gleich die Sicherheitsfanatiker).

3.2 Treiber (Modular)

Die Sicherheitsfanatiker können aber auch zufriedengestellt werden. Dank Jens Eike Jesau aus Bielefeld (Adresse siehe in der Einleitung) existiert ein Treiber als Kernelmodul. Nun können sich die Studenten in den Laboren austoben und haben trotzdem ihre Grenzen.

Aktuelle Versionen des Treibers sind auf dem anonymous-ftp-Server `experte.kt2.tu-harburg.de` im Verzeichnis `/pub/LinuxProgramme/Hardware/RTX03/` zu finden

4 Anwendungen

So eine Karte brüllt geradezu nach Anwendungen.

4.1 Thermometer

Michael Müller verwendet die Karten als 16-Kanal-Thermometer. Er hat sie gemäß Bild 2 beschaltet.

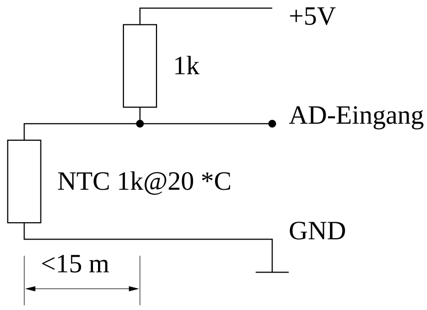

Bild 2: Thermometerbeschaltung

Das entprechende Listing ist im Anhang C wiedergegeben.

Michael meinte schon, daß die Genauigkeit des Thermometers verbessert werden könne, indem die Versorgung von +5 V auf +8,5 V hochgezogen wird. Da die Karte jedoch nicht 8,5 V zur Verfügung stellt, muß man +12 V benutzen. Wenn ich mir die Widerstandwerte angucke, so sollte man auch damit im normalen Temperaturbereich nicht in kritische Wandlungsbereiche kommen. Habe leider bisher keine Empfindlichkeiten von NTCs gefunden. Und Conrad hat auch nur welche ab 5 kΩ@25 °C .

4.2 Effektivspannungsmeßgerät

Hier wird das vorerst nur für Niederspannungen vorgenommen. Bei Hochspannungen sieht das schon etwas anders aus.

Normale Multimeter messen nur den Gleichrichtwert $|\bar{U}|$ und nicht den Effektivwert (trms) U_{eff}. Effektivwert-Meßgeräte kosten i.d.R. mehr als eine AD-Karte und ein bißchen Elektronik. Und Effektivwertgleichrichter krepieren bei langen Periodendauern an hohen Fehlern, da der Innenwiderstand R_i der Kondensatoren sich massiv bemerkbar macht. Was liegt da näher als ....

$$U_{\text{eff}} = \sqrt{\frac{1}{T}\int_0^T u^2 \, \mathrm{d}T} \tag{4}$$

$$|\bar{U}| = \frac{1}{T} \int_0^T |u| \, \mathrm{d}T \qquad (5)$$

4.3 Effektivleistungsmeßgerät

Der Schritt vom Effektivspannungsmeßgerät zum Effektivleistungsmeßgerät ist prinzipiell sehr klein. Hier kommt erschwerend hinzu, daß die Kanäle immer eine gleiche Masse haben und keine differentiellen AD-Wandler zu Verfügung stehen.

Weiterhin hat der Autor der Karte bisher maximal 5 kHz Summenabtastrate aus dem Rücken geleiert. Beim Netzwechselstrom mit 50 Hz kann das schon deutliche Abtastverschiebungen hervorrufen... und damit Meßfehler. Schaun'n mer mal.

5 Zusammenfassung

Mit den oben vorgestellten Zusammenhängen ist es jetzt möglich eine preiswerte AD/DA-Karte unter Linux anzusprechen. Die Software ist zwar noch nicht das gelbe vom Ei, für meßtechische Zwecke sollte es jedoch alle mal reichen. Und wer eine Skalierung im Programmquellcode nicht vornehmen will: Dafür gibt es schließlich noch `awk` und `tcl`. Ich persönlich werde demnächst dem Siemens Kompensomat (eigentlich ein schöner 2-Kanal x-t Schreiber mit einem minimalen Papierverbrauch von 30 mm/h) wieder in die Ecke stellen können und via Fernwartung meinen Reibprüfstand überwachen können.

Vielleicht steige ich jetzt auch noch in die Programmierung der PC226[2] von Amplicon hinter.

[2] 12-Bit AD-Karte mit 16 Kanälen bis zu echten 750 kHz aus der derzeigten 1200 DM-Klasse, läuft derzeit nur unter DOS.

A Mitgeliefertes Basic-Programm

Da die von Conrad mitgelieferte schriftliche Dokumentation für die Programmierung mehr als mangelhaft ist, folgt hier das mitgelieferte Basic-Listing. Erst damit ist m. E. die Programmierung der Karte für hardwareunbeleckte Personen verständlich.

```
1  REM
2  REM
3  REM CARD NO. : RTX-03B
4  REM COMPANY  : REAL TIME PRODUCT
5  REM DESIGN BY: MR. ROBIN SHEN
6  REM
7  REM
10 CLS:PORT=512
20 LOCATE 5,18:PRINT "14 BIT AD-DA CONVERTION CARD"
30 LOCATE 6,18:PRINT "============================"
40 LOCATE 9,20:PRINT "1, D/A-0 CONVERTION DEMO"
45 LOCATE 11,20:PRINT "2, D/A-1 CONVERTION DEMO"
50 LOCATE 13,20:PRINT "3, A/D CONVERTION DEMO"
60 A$=INKEY$:IF A$="" THEN 60
70 IF A$="1" THEN 200
75 IF A$="2" THEN 800
80 IF A$="3" THEN 400
90 GOTO 10
200 CLS
202 LOCATE 5,15:PRINT"D/A-0 CONVERTION DEMO"
204 LOCATE 7,15:PRINT "OUTPUT WAVEFORM FROM D/A PORT"
206 LOCATE 9,15:PRINT "PRESS ANY KEY RETURN MENU"
210 OUT PORT +6,0
220 FOR I= 0 TO 63
230 OUT PORT+7,I
240 NEXT I
250 A$=INKEY$:IF A$="" THEN 210
260 GOTO 10
400 CLS
410 FOR CHANNEL=0 TO 15
420 GOSUB 550
430 B=INP(PORT+2)
440 C=INP(PORT+1)
450 D=(B-64*(INT(B/64)))*256+C
460 PRINT "   CHANNEL= ";CHANNEL,"DATE= ";D
470 NEXT CHANNEL
480 PRINT:PRINT :PRINT
490 GOTO 410
550 OUT PORT+3,0
560 OUT PORT+0,CHANNEL
570 FOR I=1 TO 8:A=INP(PORT+4):NEXT I
580 FOR I=1 TO 8:A=INP(PORT+5):NEXT I
590 RETURN
800 CLS
```

```
802 LOCATE 5,15:PRINT"D/A-1 CONVERTION DEMO"
804 LOCATE 7,15:PRINT "OUTPUT WAVEFORM FROM D/A PORT"
806 LOCATE 9,15:PRINT "PRESS ANY KEY RETURN MENU"
810 OUT PORT +8,0
820 FOR I= 0 TO 63
830 OUT PORT+9,I
840 NEXT I
850 A$=INKEY$:IF A$="" THEN 800
860 GOTO 10
```

B Mitgelieferter Assemblercode

```
; CARD NAME : RTX-03B 14 ADDA A/D Convert
; LICENCE   : REAL TIME INDUSTRIAL CO., LTD.
; AUTHOR BY : MR. ROBIN SHEN
; SOFTWARE  : Assembly Language 01/07/96
;
; ************************************************
; Define string character
;
DATASEG SEGMENT ; data segment
ADTIL_1 DB '  Channel =  $'
ADTIL_2 DB '  Date =  $'
DATASEG ENDS ; end of data segment
;
; ************************************************
PROGRAM SEGMENT ; code segment
;
; ------------------------------------------------
MAIN PROC FAR ; main part of program
ASSUME CS: PROGRAM, DS: DATASEG
;
START: ; starting execution address
; Set up stack for return
PUSH DS  ; save DS on stack
SUB  AX,AX ; set AX to 0
PUSH AX  ; put AX to stack
KEYBOD: MOV  AH,0BH ; check keyboard input funct
INT  21H ; call DOS
CMP  AL,00H ; is keyin ?
JZ   KEYBOD ; yes, keyin !
CALL CRLF ; skip a line
AD14: MOV  DX,0200H ; set DX port address
MOV  CL,00H ; CL = channel number
NXCHL: CALL LPBACK ; loop back 8 times
PUSH DX  ; save DX
MOV  DL,02H ; get high byte address
IN   AL,DX ; get high 6 bit data
AND  AL,3FH ; mask high 2 bit
MOV  AH,AL ; move to high byte in AX
DEC  DL  ; get low byte address
IN   AL,DX ; get low 8 bit data
;
; Now A/D convert data in AX register
;
PUSH AX  ; store AX
PUSH CX  ; store CX
MOV  AX,DATASEG ; set data to work area
MOV  DS,AX ;
MOV  DX,OFFSET ADTIL_1
MOV  AH,09H ; display function
INT  21H ; call DOS
POP  CX  ; pop CX
MOV  BX,CX ; channel number to BX
CALL BIN2DEC ; call display BX routine
PUSH CX  ; store CX
MOV  AX,DATASEG ; set data to work area
MOV  DS,AX ;
MOV  DX,OFFSET ADTIL_2
MOV  AH,09H
INT  21H ; call DOS
POP  CX  ; pop CX
POP  AX  ; pop AX
MOV  BX,AX ; A/D data to BX
CALL BIN2DEC ; call display BX routine
CALL CRLF ; carriage return & linefeed
CALL DELAY ; delay CX = FFFF
POP  DX  ; pop DX
INC  CL  ; next channel
CMP  CL,10H ; is 15 channel ?
CALL DELAY ; delay CX = FFFF
JNZ  NXCHL ; yes !
CALL CRLF ; carriage return & linefeed
CALL CRLF ; carriage return & linefeed
JMP  AD14 ; next routine
RET ; return to DOS
MAIN ENDP ; end of main porcedure
;
; --------------------------------------------------
; Loop back 8 times subroutine
;
LPBACK PROC NEAR ; define procedure
PUSH DX  ; save DX
PUSH CX  ; save CX
MOV  AL,00H ; clear AX
MOV  DL,03H ; REM SET DX ADDRESS &H203
OUT  DX,AL ; clear A/D register &h203
DEC  DL  ; decrement DX
DEC  DL  ; decrement DX
DEC  DL  ; decrement DX
MOV  AL,CL ; channel number into AL
; MOV  DL,00H ; rem A/D CHANNEL NUMBER &H200
OUT  DX,AL ; select A/D channel
MOV  DL,04H ; REM select low byte 8 bit
MOV  CX,08H ;  loop back 8 times
LOWBYTE: ;   for low byte
IN   AL,DX ;
LOOP LOWBYTE ;
MOV  DL,05H ; REM select high byte 6 bit
MOV  CX,08H ;  loop back 8 times
HIBYTE:  ;   for high byte
IN   AL,DX ;
LOOP HIBYTE ;
POP  CX  ; pop CX
POP  DX  ; pop DX
RET ; return form LPBACK
CALL DELAY ; delay CX = FFFF
LPBACK ENDP ; end of porcedure
;
; --------------------------------------------------
; Convert binary in BX to decimal in screen
;
BIN2DEC PROC NEAR ; define porcedure
PUSH CX  ; save CX
MOV  CX,10000D ; divide by 10000
CALL DEC_DIV
MOV  CX,1000D ; divide by 1000
CALL DEC_DIV
MOV  CX,100D ; divide by 100
CALL DEC_DIV
MOV  CX,10D ; divide by 10
CALL DEC_DIV
MOV  CX,1D ; divide by 1
CALL DEC_DIV
POP  CX  ; pop CX
RET ; return form BIN2DEC
;
; -------------------------------------------------
; Sub-subroutine to divide number in BX by
;   number in CX, print in screen
;
DEC_DIV PROC NEAR ; define procedure
MOV  AX,BX ; number high half
MOV  DX,00H ; zero out low half
DIV  CX  ; divide by CX
MOV  BX,DX ; remainder into BX
MOV  DL,AL ; quotient into DL
ADD  DL,30H ; convert to ASCII
MOV  AH,02H ; display function
INT  21H ; call DOS
RET ; return from DEC_DIV
DEC_DIV ENDP ; end of porcedure
;
; -------------------------------------------------
;
BIN2DEC ENDP ; end of porcedure BIN2DEC
;
; -------------------------------------------------
; Carriage return & Linefeed
;
CRLF PROC NEAR
PUSH CX  ; save CX
MOV  DL,0DH ; carriage return
MOV  AH,02H ; display function
INT  21H ; call DOS
MOV  DL,0AH ; linefeed
MOV  AH,02H ; display function
INT  21H ; call DOS
POP  CX  ; pop CX
RET ; return from CRLF
CRLF ENDP ; end of procedure
;
; -------------------------------------------------
; Delay subroutine, delay time CX = FFFF
```

```
;
DELAY PROC NEAR ; define procedure
PUSH CX  ; save CX
MOV  CX,9FFFH ; define delay CX = FFFFH
NXDEL1: LOOP NXDEL1 ; is zero ?
POP  CX  ; pop CX
RET ; return form delay
PUSH CX  ; save CX
MOV  CX,9FFFH ; define delay CX = FFFFH
NXDEL2: LOOP NXDEL2 ; is zero ?
POP  CX  ; pop CX
RET ; return form delay
DELAY ENDP ; end of procedure
;
; --------------------------------------------------
;
PROGRAM ENDS ; end of code segment
;
; **************************************************
;
END START ; end of main porgram
;
```

C Erstcode von Michael Müller

Es folgt das erste Listing von Michael Müller. Hiermit betreibt er die Karte gemäß Abschnitt 4.1 als Thermometer.

```
/*
 * ad.c: very simple example of port I/O
 *
 * This code reads the 16 inputs of a connected 12-bit io-card
 * Compile with ‘gcc -O2 -lm -o adwandler adwandler.daemon.c‘
 */

#include <stdio.h>
#include <unistd.h>
#include <asm/io.h>
#include <math.h>
#include <time.h>

#define datei "/var/log/temperatur"
#define refvoltage 2407.0 /* entspricht 5V bei 12 bit bis 8,5V */
#define vara 158 /* empirisch ermittelt-NTC abhaengig */
#define varb 3567.221 /* empirisch ermittelt-NTC abhaengig */
#define BASEPORT 0x290 /* ad1 */
#define AD_CHANNEL (BASEPORT + 0)
#define AD_LOWBYTE (BASEPORT + 1)
#define AD_HIGHBYTE (BASEPORT + 2)
#define AD_SETZERO (BASEPORT + 3)
#define AD_readf (BASEPORT + 4)
#define AD_reads (BASEPORT + 5)
#define DA_LOWBYTE (BASEPORT + 6)
#define DA_HIGHBYTE (BASEPORT + 7)
#define LOOPS 1 /* Anzahl der Messungen pro Output */
#define OFFSET 0 /* 0 Volt Fusspunkt */


int main()
{
  double temperatur;
  double temper;
  float resistor;
  float resist;
  int highbyte;
  int lowbyte;
  int channel;
  int i;
  int a;
  int k;
  FILE *protokoll;
  time_t sekunden;
```

```
  /* get access to the ports */
  if (ioperm(BASEPORT,8,1)) {perror("ioperm");exit(1);}

  protokoll = fopen(datei, "a");

  while ( 5 > 3 ) /* dumme Bedingung, ich weiss */
  {
    sekunden = time((time_t *) 0);
    fprintf(protokoll, "%lu", sekunden);
    /* Auslesen der 16 Kanaele */
    for (channel = 0; channel < 16; channel++)
    {
outb(0,AD_SETZERO); /* noch ungeklaert aber notwendig */
outb(channel,AD_CHANNEL); /* Anwahl des Kanals */
a = 0;
/* for (k = 0; k < LOOPS; k++) */
{
outb(channel,AD_CHANNEL);
/* outb(0,AD_CHANNEL); */
for (i = 0; i < 7; i++) /* 7bits digitalisieren low */
{
a = inb(AD_readf);
usleep(1000); /* verzoeg. fuer den Wandler */
}
for (i = 0; i < 7; i++) /* 7bits digitalisieren high sind
zwar nur 4 muss aber sein */
{
a = inb(AD_reads);
usleep(1000); /* verzoeg. fuer den Wandler */
}
highbyte = (inb(AD_HIGHBYTE) & 0x0f);
lowbyte = inb(AD_LOWBYTE);
}
a = highbyte * 256 + lowbyte + OFFSET;
resist = (refvoltage / a) - 1;
resistor = 1000 / resist; /* bestimmen des NTC Widerstands */
temperatur = varb / log(resistor * vara) - 273;

fprintf(protokoll, "%7.2f", temperatur);
usleep(1000);
    }
    fprintf(protokoll, "\n");
    fflush(protokoll);
    sleep(55); /* ergibt ca 1 minute pro Messreihe */
  }
  /* Alles ab hier ist eigentlich quatsch, da es nie ausgefuehrt wird, da oben
     eine endlosschleife ist. */

  /* We don't need the ports anymore */
  if (ioperm(BASEPORT,3,0)) {perror("ioperm");exit(1);}

  exit(0);
```

```
}

/* end of adwandler.daemon.c */
```

Literatur

[Dub16] Beitz, W.; Küttner, H.-H.: Dubbel, Taschenbuch für den Maschinenbau, Springer-Verlag, Berlin-Heidelberg-New York, 1987, 16. Auflage. *Zwar kein Taschenbuch und viel hier Unwichtiges d'rinn, aber trotzdem in einigen Abschnitten zur Übersichtsgewinnung empfehlenswert.*

Index