wird eine Hardcopy-Routine für den hochauflösenden Grafikmode des C16 gestartet. Diese Routine greift auf den Grafikbildspeicher zu, der im Adreßbereich von 2000h bis 3F3F liegt. Die Escape-Sequenzen, die den Drucker auf Grafik schalten, sind für den Epson 80/85 ausgelegt und müssen für nicht Epson-kompatible Drucker angepaßt werden. Der Ausdruck erfolgt mit einer Punktdichte von 72 Pkt./Zoll (Plotmodus), womit eine proportionsgerechte Darstellung erreicht wird. Nicht alle Drucker ermöglichen diese Punktdichte, so daß eventuell Verzerrungen in Kauf genommen werden müssen.

Die Escape-Sequenz zur Einstellung des Zeilenvorschubs steht ab Adresse 117B, die Escape-Sequenz zur Definition der Bildpunktdichte und der Anzahl der Grafikbytes steht ab 117E, jeweils in umgekehrter Reihenfolge.

## Stückliste

**Halbleiter:**

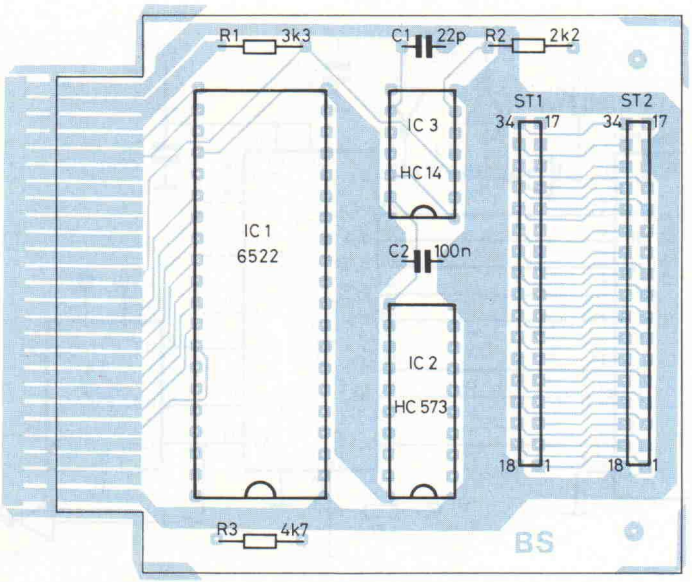
| | |
|---|---|
| IC1 | 6522 |
| IC2 | 74HC573 o. 74HCT573 |
| IC3 | 74HC14 o. 74HCT14 |

**Kondensatoren**

| | |
|---|---|
| C1 | 22pF |
| C2 | 100nF |

**Widerstände**

| | |
|---|---|
| R1 | 3k3 |
| R2 | 2k2 |
| R3 | 4k7 |
| ST1 | 2 × 34polige |
| ST2 | Pfostenstiftleisten |




***Literatur***

Fritz Schäfer: Das große C16-Buch, Kingsoft

```
                        1   ****************************************
                        2   *                                      *
                        3   * C16/C116-DRUCKERTREIBER FUER DRUCKER  *
                        4   * MIT PARALLELSCHNITTSTELLE            *
                        5   *                                      *
                        6   * (C) HANS ULRICH SCHUBERT 1986        *
                        7   *                                      *
                        8   * STAND: 30.06.86                      *
                        9   *                                      *
                       10   ****************************************
                       11   *
                       12   * BENUTZTE KERNAL-ROUTINEN
                       13   CLOSE    =    $EE5D      ;DATEI SCHLIESSEN
                       14   CLSNUM   =    $EEED      ;GERAETENR. IN X HOLEN
                       15   CHKOUT   =    $ED60      ;AUSGABEKANAL OEFFNEN
                       16   CHKOPN   =    $EEE8      ;PRUEFEN OB DATEI OFFEN
                       17   CHKNUM   =    $EEF8      ;GERAETENR. IN A HOLEN
                       18   CHROUT   =    $EC4B      ;ZEICHENAUSGABE
                       19   * VERAENDERTE SYSTEMVEKTOREN
                       20   ICLOSE   =    $031A      ;ZEIGER AUF CLOSE
                       21   ICKOUT   =    $031E      ;ZEIGER AUF CHKOUT
                       22   IBSOUT   =    $0324      ;ZEIGER AUF CHROUT
                       23   * SONSTIGE BENUTZTE SYSTEMADRESSEN
                       24   FAT      =    $0513      ;GERAETENUMMERN
                       25   DFLTO    =    $99        ;AUSGABEGERAET
                       26   TXTTAB   =    $2B        ;ANFANG BASIC TEXT
                       27   VARTAB   =    $2D        ;ANFANG BASIC VARIABLE
                       28   ARYTAB   =    $2F        ;ANFANG BASIC FELDER
                       29   STREND   =    $31        ;ENDE BASIC FELDER
                       30   * LOKAL BENUTZTE ZEROPAGE-ADRESSEN
                       31   LSWTCH   =    $D8        ;KLEINSCHREIBSWITCH
                       32   DIROUT   =    $D9        ;DIREKTAUSGABE
                       33   BSPADR   =    $DA        ;BILDSPEICHERADRESSE
                       34   * BASIS I/O-ADRESSE DES VIA 6522
                       35   IOBAS    =    $FD00
                       36   * GERAETENUMMER DES DRUCKERS
                       37   PPRINT   =    $10        ;DEZIMAL "16"
                       38   *
                       39            ORG. $1000
                       40   *
1000: 00               41            BRK
1001: 00               42            BRK
1002: 00               43            BRK
                       44   * EINSPRUNGADRESSEN NACH DEM LADEN
1003: 4C 0C 10         45            JMP  INSTAL     ;SYS 4099:INSTALLATION
1006: 4C E1 10         46            JMP  HRDCPY     ;SYS 4102:HARDCOPY
1009: 4C 9D 10         47            JMP  PROUT1     ;SYS 4105:DIREKTAUSGABE
100C: A9 00            48   INSTAL   LDA  #$00
100E: 85 D8            49            STA  LSWTCH
1010: A9 0A            50            LDA  #$0A       ;6522 PORT A ...
1012: 8D 0C FD         51            STA  IOBAS+12
1015: A9 FF            52            LDA  #$FF       ;... AUSGABE ...
1017: 8D 03 FD         53            STA  IOBAS+3    ;... IM PULSMODE
                       54   * SYSTEMVEKTOREN "UMBIEGEN"
101A: A9 C4            55            LDA  #<LCLOSE
101C: 8D 1A 03         56            STA  ICLOSE
101F: A9 10            57            LDA  #>LCLOSE
1021: 8D 1B 03         58            STA  ICLOSE+1
1024: A9 AB            59            LDA  #<LCKOUT
1026: 8D 1E 03         60            STA  ICKOUT
1029: A9 10            61            LDA  #>LCKOUT
102B: 8D 1F 03         62            STA  ICKOUT+1
102E: A9 54            63            LDA  #<LBSOUT
1030: 8D 24 03         64            STA  IBSOUT
1033: A9 10            65            LDA  #>LBSOUT
1035: 8D 25 03         66            STA  IBSOUT+1
                       67   *ANFANGSADRESSE FUER BASIC NEU SETZEN
1038: A9 12            68            LDA  #$12
103A: 85 2C            69            STA  TXTTAB+1
103C: 85 2E            70            STA  VARTAB+1
103E: 85 30            71            STA  ARYTAB+1
1040: 85 32            72            STA  STREND+1
                       73   * INSTALLATIONSMELDUNG
1042: A0 00            74            LDY  #$00
1044: B9 8B 11         75   TXTOUT   LDA  TEXT,Y
1047: C9 04            76            CMP  #$04       ;BIS "EOT" AUSGEBEN
1049: F0 07            77            BEQ  ENDINS
104B: 20 4B EC         78            JSR  CHROUT
104E: C8               79            INY
104F: 4C 44 10         80            JMP  TXTOUT
1052: 60               81   ENDINS   RTS
1053: EA               82            NOP
                       83   *** LOKALE ZEICHENAUSGABE-ROUTINE ***
1054: 48               84   LBSOUT   PHA
1055: A5 99            85            LDA  DFLTO      ;AUSGABEGERAET...
1057: C9 10            86            CMP  #PPRINT    ;PARALLEL PRINTER ?
1059: F0 04            87            BEQ  LBOUT1
                       88
105B: 68               89            PLA
105C: 4C 4B EC         90            JMP  CHROUT     ;KERNAL ZEICHENAUSGABE
                       91   * SONDERFAELLE ABFRAGEN
105F: 68               92   LBOUT1   PLA
1060: C9 0D            93            CMP  #$0D       ;"CARRIAGE RETURN" ?
1062: F0 1D            94            BEQ  AUTOLF
1064: C9 0E            95            CMP  #$0E       ;"SWITCH TO LOWER CASE"?
1066: F0 1E            96            BEQ  LOCASE
1068: C9 8E            97            CMP  #$8E       ;"SWITCH TO UPPER CASE"?
106A: F0 20            98            BEQ  UPCASE
106C: C9 1D            99            CMP  #$1D       ;"PSEUDOTABULATOR"?
106E: D0 02           100            BNE  TSTCAS
1070: A9 20           101            LDA  #$20       ;MIT "BLANK" ERSETZEN
                      102   * FUER "A"... "Z" GGF. KLEINSCHREIBUNGSCODE BILDEN
1072: C9 5B           103   TSTCAS   CMP  #$5B       ;"Z"?
1074: B0 06           104            BCS  CODOUT
1076: C9 41           105            CMP  #$41       ;"A"?
1078: 90 02           106            BCC  CODOUT
107A: 05 D8           107            ORA  LSWTCH
107C: 20 9F 10        108   CODOUT   JSR  PRTOUT
107F: 18              109            CLC
1080: 60              110            RTS
```

```
1081: 20 91 10 111  AUTOLF   JSR  CRLF
1084: 18       112           CLC
1085: 60       113           RTS
1086: A9 20    114  LOCASE   LDA  #$20       ;KLEINSCHREIBSWITCH...
1088: 85 D8    115  SETCAS   STA  LSWTCH     ;SETZEN
108A: 18       116           CLC
108B: 60       117           RTS
108C: A9 00    118  UPCASE   LDA  #$00       ;SWITCH LOESCHEN
108E: F0 F8    119           BEQ  SETCAS
1090: EA       120           NOP
               121  *** CARRIAGE RETURN/LINE FEED-ROUTINE ***
1091: A9 0D    122  CRLF     LDA  #$0D
1093: 20 9F 10 123           JSR  PRTOUT
1096: A9 0A    124           LDA  #$0A
1098: 20 9F 10 125           JSR  PRTOUT
109B: 60       126           RTS
109C: EA       127           NOP
               128  *** DRUCKERAUSGABE-ROUTINE ***
109D: A5 D9    129  PROUT1   LDA  DIROUT     ;DIREKTAUSGABE
109F: 8D 01 FD 130  PRTOUT   STA  IOBAS+1
10A2: AD 0D FD 131           LDA  IOBAS+13
10A5: 29 02    132           AND  #$02
10A7: F0 F9    133           BEQ  PRTOUT+3   ;WARTEN AUF "ACKNOWLEDGE"
10A9: 60       134           RTS
10AA: EA       135           NOP
               136  *** LOKALE CHKOUT-ROUTINE ***
10AB: 8A       137  LCKOUT   TXA
10AC: 48       138           PHA
10AD: 20 E8 EE 139           JSR  CHKOPN     ;DATEI OFFEN ?
10B0: D0 0C    140           BNE  KRNCHK
10B2: 20 F8 EE 141           JSR  CHKNUM     ;GERAETENUMMER HOLEN
10B5: C9 10    142           CMP  #PPRINT    ;PARALLEL PRINTER?
10B7: D0 05    143           BNE  KRNCHK
10B9: 85 99    144           STA  DFLTO      ;JA: NEUES AUSGABEGERAET
10BB: 68       145           PLA
10BC: 18       146           CLC
10BD: 60       147           RTS
10BE: 68       148  KRNCHK   PLA
10BF: AA       149           TAX
10C0: 4C 60 ED 150           JMP  CHKOUT     ;KERNAL CHKOUT-ROUTINE
10C3: EA       151           NOP
               152  *** LOKALE CLOSE-ROUTINE ***
10C4: 48       153  LCLOSE   PHA
10C5: 20 ED EE 154           JSR  CLSNUM     ;GERAETENR. IN X HOLEN
10C8: BD 13 05 155           LDA  FAT,X
10CB: C9 10    156           CMP  #PPRINT    ;PARALLEL PRINTER ?
10CD: D0 04    157           BNE  KRNCLS
10CF: A9 03    158           LDA  #$03       ;JA: BILDSCHIRM WIRD...
10D1: 85 99    159           STA  DFLTO      ;WIEDER AUSGABEGERAET
10D3: 68       160  KRNCLS   PLA
10D4: 4C 5D EE 161           JMP  CLOSE      ;KERNAL CLOSE-ROUTINE
10D7: EA       162           NOP
10D8: EA       163           NOP
10D9: EA       164           NOP
10DA: EA       165           NOP
10DB: EA       166           NOP
10DC: EA       167           NOP
10DD: EA       168           NOP
10DE: EA       169           NOP
10DF: EA       170           NOP
10E0: EA       171           NOP
               172  *** HIRES-HARDCOPY-ROUTINE ***
10E1: 20 33 11 173  HRDCPY   JSR  INITHC     ;DRUCKER INITIALISIEREN
10E4: A9 00    174           LDA  #$00
10E6: 85 DA    175           STA  BSPADR
10E8: A9 20    176           LDA  #$20
10EA: 85 DB    177           STA  BSPADR+1
10EC: A2 19    178           LDX  #$19       ;25 BILDREIHEN
10EE: 20 52 11 179  ONEROW   JSR  NEWROW     ;BILDREIHE EINLEITEN
10F1: A0 28    180           LDY  #$28       ;40 BILDPOSITIONEN
10F3: 20 01 11 181  ONEBP    JSR  BPOS       ;BILDPOSITION ZUM DRUCKER
10F6: 88       182           DEY             ;ALLE BILDPOSITIONEN ...
10F7: D0 FA    183           BNE  ONEBP      ;DIESER BILDREIHE GEDRUCKT?
10F9: CA       184           DEX             ;JA: NAECHSTE BILDREIHE
10FA: D0 F2    185           BNE  ONEROW     ;BIS 25 BILDREIHEN GEDRUCKT
10FC: 20 43 11 186           JSR  RELEAS     ;DRUCKER NORMIEREN
10FF: 60       187           RTS
1100: EA       188           NOP
               189  *** BILDPOSITION UMSETZEN UND DRUCKEN ***
1101: 8A       190  BPOS     TXA
1102: 48       191           PHA
1103: 98       192           TYA
1104: 48       193           PHA
1105: A2 00    194           LDX  #$00       ;BITPOSITION-ZAEHLER
1107: A0 00    195  NXTPOS   LDY  #$00       ;BYTE-ZAEHLER
1109: 38       196  NXTBYT   SEC             ;FALLS MSB GESETZT
110A: B1 DA    197           LDA  (BSPADR),Y
110C: 30 01    198           BMI  ROTATE
110E: 18       199           CLC             ;WENN MSB=0 WAR
110F: 2A       200  ROTATE   ROL             ;CY NACH LSB, MSB NACH CY
1110: 91 DA    201           STA  (BSPADR),Y
1112: 3E 83 11 202           ROL  BUFFER,X   ;CY IN DEN BUFFER
1115: C8       203           INY             ;NAECHSTES BYTE
1116: C0 08    204           CPY  #$08       ;8 BYTES EINMAL ROTIERT?
1118: D0 EF    205           BNE  NXTBYT
111A: E8       206           INX             ;NAECHSTE BITPOSITION
111B: E0 08    207           CPX  #$08       ;ACHTMAL ROTIERT ?
111D: D0 E8    208           BNE  NXTPOS
111F: 20 6C 11 209           JSR  PRTBUF     ;BUFFER ZUM DRUCKER
               210  * ADRESSE DER NAECHSTEN BILDPOSITION ERRECHNEN
1122: A5 DA    211           LDA  BSPADR
1124: 18       212           CLC
1125: 69 08    213           ADC  #$08
1127: 90 02    214           BCC  JUSTLO
1129: E6 DB    215           INC  BSPADR+1
112B: 85 DA    216  JUSTLO   STA  BSPADR
112D: 68       217           PLA
112E: A8       218           TAY
112F: 68       219           PLA
1130: AA       220           TAX
1131: 60       221           RTS
1132: EA       222           NOP
               223  *** DRUCKER-INITIALISIERUNG FUER HARDCOPY ***
1133: 20 43 11 224  INITHC   JSR  RELEAS
1136: A0 02    225           LDY  #$02
1138: B9 7B 11 226  ESC1LP   LDA  ESC1,Y     ;ZEILENABSTAND
113B: 20 9F 10 227           JSR  PRTOUT
113E: 88       228           DEY
113F: 10 F7    229           BPL  ESC1LP
1141: 60       230           RTS
1142: EA       231           NOP
               232  *** DRUCKER NORMIEREN ***
1143: A9 1B    233  RELEAS   LDA  #$1B
1145: 20 9F 10 234           JSR  PRTOUT
1148: A9 40    235           LDA  #$40
114A: 20 9F 10 236           JSR  PRTOUT
114D: 20 91 10 237           JSR  CRLF
1150: 60       238           RTS
1151: EA       239           NOP
               240  *** NEUE BILDREIHE EINLEITEN ***
1152: 20 91 10 241  NEWROW   JSR  CRLF
1155: A0 11    242           LDY  #$11       ;17 "BLANKS"
1157: A9 20    243  NXTBLK   LDA  #$20
1159: 20 9F 10 244           JSR  PRTOUT
115C: 88       245           DEY
115D: D0 F8    246           BNE  NXTBLK
115F: A0 04    247           LDY  #$04
1161: B9 7E 11 248  ESC2LP   LDA  ESC2,Y     ;BIT IMAGE MODUS
1164: 20 9F 10 249           JSR  PRTOUT
1167: 88       250           DEY
1168: 10 F7    251           BPL  ESC2LP
116A: 60       252           RTS
116B: EA       253           NOP
               254  *** BUFFER AN DRUCKER AUSGEBEN ***
116C: A0 00    255  PRTBUF   LDY  #$00
116E: B9 83 11 256  BUFLP    LDA  BUFFER,Y
1171: 20 9F 10 257           JSR  PRTOUT
1174: C8       258           INY
1175: C0 08    259           CPY  #$08
1177: D0 F5    260           BNE  BUFLP
1179: 60       261           RTS
117A: EA       262           NOP
               263  *** ESCAPE-SEQUENZEN ***
117B: 18 33 1B 264  ESC1     HEX  18,33,1B
117E: 01 40 05
1181: 2A 1B    265  ESC2     HEX  01,40,05,2A,1B
               266  *** BUFFERBEREICH ***
1183: 00 00 00
1186: 00 00 00
1189: 00 00    267  BUFFER   HEX  00,00,00,00,00,00,00,00
               268  *** INSTALLATIONSMELDUNG ***
118B: 13 93 0D
118E: 20 20 12
1191: 20       269  TEXT     HEX  13,93,0D,20,20,12,20
1192: 50 41 52
1195: 41 4C 4C
1198: 45 4C 20 270           HEX  50,41,52,41,4C,4C,45,4C,20
119B: 50 52 49
119E: 4E 54 45
11A1: 52 20    271           HEX  50,52,49,4E,54,45,52,20
11A3: 44 52 49
11A6: 56 45 52
11A9: 20       272           HEX  44,52,49,56,45,52,20
11AA: 49 4E 53
11AD: 54 41 4C
11B0: 4C 45 44 273           HEX  49,4E,53,54,41,4C,4C,45,44
11B3: 20 20 0D
11B6: 20 20 12
11B9: 20       274           HEX  20,20,0D,20,20,12,20
11BA: 52 45 4D
11BD: 45 4D 42
11C0: 45 52 3A 275           HEX  52,45,4D,45,4D,42,45,52,3A
11C3: 20 50 52
11C6: 49 4E 54
11C9: 45 52    276           HEX  20,50,52,49,4E,54,45,52
11CB: 20 43 48
11CE: 41 4E 4E
11D1: 45 4C    277           HEX  20,43,48,41,4E,4E,45,4C
11D3: 20 49 53
11D6: 20 23 31
11D9: 36       278           HEX  20,49,53,20,23,31,36
11DA: 20 20 20
11DD: 0D 04    279           HEX  20,20,20,0D,04
```

**Der Centronics-Treiber bindet sich selbst ins Betriebssystem ein – außerdem enthält er noch eine Hardcopy-Routine für die hochauflösende Grafik von C16 und C116.**

# Löschsperre

## Speichererhaltender Reset bei den CPCs

Sönke Marsch

**Zwar läßt sich über einen zusätzlichen Reset-Taster ein irrlaufender CPC jederzeit zur Vernunft bringen, doch dabei geht leider jeglicher Speicherinhalt verloren, was unter Umständen den Verlust wertvoller Daten bedeuten kann. Mit einer cleveren Reset-Logik kann man den störenden Löscheffekt aber verhindern.**

Allerdings muß man dabei hoffen, daß der amoklaufende Computer die Daten oder Programme nicht bereits zerstört hat, denn dann ist Hopfen und Malz verloren. Falls aber die Systemvariablen des BASIC-Interpreters noch vorhanden sind, braucht man keine neue Initialisierung. Es genügt, wenn man in den Ready-Modus springt.

Für die Initialisierung des BASIC-ROM nach einem Reset sorgt das Betriebssystem im Kernel. Die verantwortliche Routine liegt ab Adresse 77h im ROM (das ROM wird ja nach einem Reset grundsätzlich eingeblendet). Nun könnte man dem mit einem eigenen, veränderten ROM (256er EPROM) abhelfen. Das führt allerdings auf die bekannten rechtlichen Probleme und ist auch nicht ganz einfach (ROM meistens eingelötet, EPROM-Brenner erforderlich et cetera).

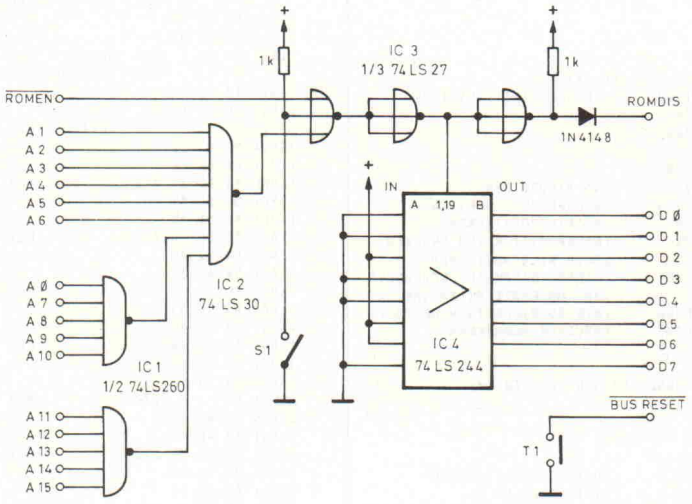
### Sich einmischen . . .

Statt dessen kann eine kleine Umblendschaltung einen Eingriff in den Rechner überflüssig machen. Ab Adresse 7Dh wird mit dem Befehl LD HL,C00Bh die Einsprungsadresse des BASIC-ROM geladen. Die Schaltung filtert die Adresse 7Eh heraus und blendet das Betriebssystem-ROM aus, gleichzeitig wird ein Bustreiber aktiviert. Dieser hat an seinen Dateneingängen (für den 464) den Wert 64h (binär 01100100) fest verdrahtet. Diesen Wert liest nun die CPU statt 0Bh ein, so daß das HL-Register den geänderten Wert C064h erhält. Beim anschließenden 'FAR CALL' springt das Betriebssystem direkt zum Ready-Modus.

Bei den CPCs 664/6128 liegt die Kernel-Routine auf den gleichen Adressen, nur der Ready-Modus beginnt hier bei C058h. Folglich muß man hier 58h (binär 01011000) fest verdrahten.

Die Schaltung selbst gestaltet sich einfach. IC1 und IC2 filtern die Adresse 7Eh aus. Mit IC3 wird das ROM ausgeblendet und der Bustreiber aktiviert.

Wer Strom sparen möchte, sollte (neben HCT-Chips) bei der festen Verdrahtung mit einem Pull-up-Widerstand (etwa 4k7) an Plus gehen. Um das Ganze noch etwas komfortabler zu machen, wurde noch ein Schalter eingefügt. Mit diesem läßt sich der gesamte Umschaltmechanismus abschalten.

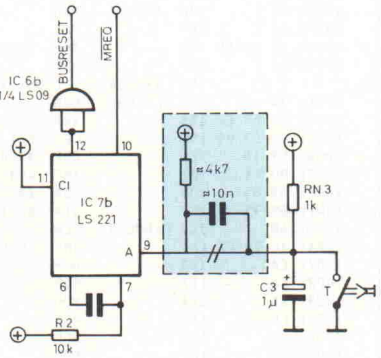



**Der Adreßdekoder erkennt die Adresse 7Eh, blendet das ROM aus und legt 64h auf den Datenbus.**

### . . . und Erholung gewähren

Der abgebildete Reset-Taster ist hier nur eine Primitiv-Lösung, da er keinerlei Rücksicht auf das CPU-Timing und den Refresh der dynamischen Speicher nimmt. Wird die CPU gerade in einem Schreibzyklus von einem Reset 'überrascht', nimmt der aktuell adressierte Speicher eventuell Unsinn auf. Bei einer außer Kontrolle geratenen CPU muß man aber eh mit dem einen oder anderen zerstörten Byte rechnen, so daß es auf dieses eine Byte nicht unbedingt ankommt, und vielleicht hat man ja Glück . . .

Auf den Refresh kann der Benutzer aber selbst einwirken. Drückt er den Reset-Taster zu lange, so muß er eben mit umgekippten Speicherzellen rechnen, da in dieser Zeit keinerlei Refresh der dynamischen Speicherzellen stattfindet. Heutzutage halten diese Chips aber ganz schön lange durch (oftmals länger als eine Sekunde). Bei einem kurzen Tastendruck ist das Risiko also gering.

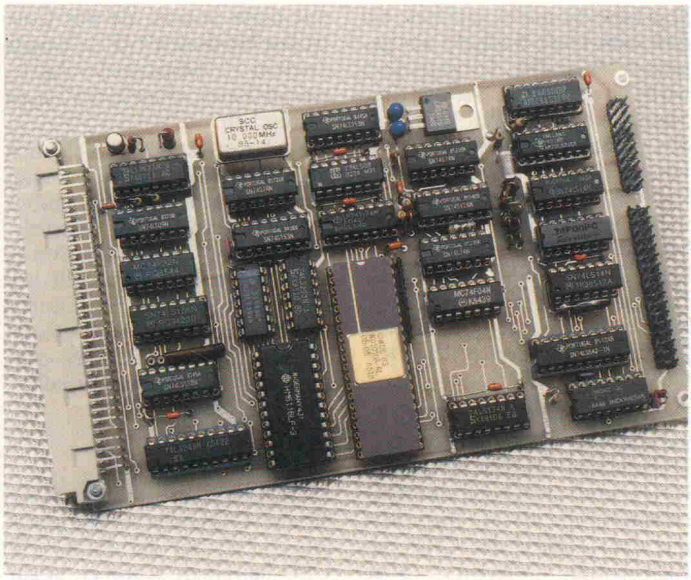



**Einen völlig sicheren Reset erreicht man durch diese Schaltung, die leicht abgewandelt auf dem c't-ECB-Adapter vorhanden ist.**

Einen wesentlich sicheren Reset kann man durch eine Reset-Logik erreichen, wie sie 'im Prinzip' auf dem c't-ECB-Anschluß vorhanden ist (timing-gerecht über ein Monoflop). Allerdings hat man auch hier leider noch eine refresh-lose Zeit während der Druckdauer. Mit einer kleinen Änderung läßt sich das Problem aber beheben.

Damit ist dann der Reset 'niet- und nagelfest'. Auf das LS09 kann man normalerweise verzichten. Es ist nur nötig, falls von irgendwelchen Peripherie-Geräten ebenfalls ein Reset ausgelöst werden soll.

Wer mit dem ECB-Adapter arbeitet, kann die Umblendschaltung bedenkenlos auf einer ECB-Bus-Platine unterbringen, da der Adapter die Signale ROMDIS und ROMEN für die Datenrichtung berücksichtigt und ROMDIS sogar über den Bus geführt werden kann.



# Hart, schnell und sicher

## Hard-Disk-Controller mit ST-506-Schnittstelle für ECB-Bus-Rechner

### Teil 2

Andreas Zippel

**Nachdem im ersten Teil der Artikelreihe Theorie und Grundlagen dominierten, kommen jetzt endlich die Praktiker zu ihrem Recht. Dieser Teil beschert ihnen die Schaltungsbeschreibung nebst Anleitung zur Inbetriebnahme und Testprogrammen. Bis auf die Einbindungen in Betriebssysteme ist damit das gesamte Rüstzeug vorhanden, das man braucht, um die Funktionsfähigkeit des Systems Rechner-Controller-Laufwerk zu gewährleisten.**

Ehe es zu spät ist, gleich ein paar Worte an die Leser, die sich ausschließlich für Stückliste, Schaltplan und Bestückungsplan interessieren, um dann umgehend dem Lötkolben zu huldigen: Lesen Sie dieses eine Mal den Artikel vorher. So können Sie sich nämlich die Mühe sparen, diverse Bauteile zum Abgleich des Controllers wieder auslöten zu müssen.

Und vorab noch einen kleinen Euphorie-Dämpfer: Ein Festplatten-Controller stellt schon einige Ansprüche an das Können des Selbstbauers, und auch ein minimaler 'Meßpark' muß im Zugriff sein. Es handelt sich also aus unserer Sicht ganz eindeutig nicht um ein Anfänger-Projekt, und wir werden auch bei der folgenden Beschreibung Grundkenntnisse im 'Computer-Eigenbau' voraussetzen. Andererseits: Was kann außer der Erzeugung einiger Kilo Computerschrott schon Schlimmes passieren... Wer nicht wagt, der nicht gewinnt.

## Adressaten

In etlichen Belangen unterscheidet sich die c't-HDC-Karte nicht von anderen ECB-I/O-Karten wie etwa Floppy-Controllern, so daß wir hier auf Standard-Baugruppen nicht weiter eingehen.

Eigentlich gehört auch die Adreßdekodierung in diese Standard-Kategorie, hier sind allerdings einige Besonderheiten zu konstatieren, da der Betrieb der Controller-Karte an 8-Bit-Rechnern (etwa PROF-80) ebenso wie an Computern mit 16 Bit breitem Datenbus (c't86) erfolgen soll.

Konkret muß dem Betrieb am c't86 in folgenden Punkten Rechnung getragen werden:

Beim Zugriff auf ungerade Portadressen transportiert die 8086-CPU die Daten über ihren oberen Datenbus, also D8 bis D15. Dieser Datenbus ist beim normalen ECB-Bus (64polig) jedoch nicht vorhanden. Um nun keine zwei unterschiedlichen Bestückungen und diverse Extralogik für jeden Bus auf der HDC-Karte unterbringen zu müssen, wird die Adreßdekodierung durch Steckbrücken für den 16-Bit-Betrieb etwas modifiziert (Tabelle 1).

Dabei werden alle HDC-Ports auf gerade I/O-Adressen gelegt, so daß stets nur der untere Datenbus des 8086-Prozessors benutzt werden kann. Dieses Verfahren hat allerdings zur Folge, daß die Adreßlage der einzelnen Ports im 8- und im 16-Bit-Betrieb deutlich voneinander abweicht (Tabelle 2), was man aber sehr einfach in der Treiber- und Test-Software berücksichtigen kann.

Des weiteren wird beim c't86 nicht die ECB-Bus-Leitung 10a, sondern 28a für das Wait-Signal verwendet, weswegen diese Leitung ebenfalls an den Kollektor des Transistors T1 geführt wird.

## Die harten Sachen

Etwas ungewöhnlich und damit erklärungsbedürftig sind eigentlich nur drei Schaltungsaspekte. Zum einen wird aus der 12-V-Versorgung des Rechners eine zweite Betriebsspannung von 5V abgeleitet. Damit soll sichergestellt werden, daß die 'neuralgischen' Schaltungsteile (alles, was mit dem pingelig genauen Abgleich des Datenseparators – wir kommen noch darauf – zu tun hat) bestmöglich stabil und sauber versorgt sind.

Zum anderen ist es interessant, das Wechselspiel zwischen Controller-Sektorpuffer und Sektorpuffer-Rechner zu kennen. Und schließlich verdient der Aufbau des Datenseparators Beachtung, da mit dessen ordnungsgemäßer Funktion die ganze Massenspeicherei steht und fällt.

Zum Zusammenspiel CPU-Controller einige grundlegende Dinge vorweg. Der Sektorpuffer ist als FIFO organisiert. Da der WD1010 im Wechsel mit der CPU Zugriff auf den internen Bus haben muß, befindet sich auch eine kleine Busarbitration im Steuerteil der Platine.

## First In, First Out

Einige 'Tricks', die der Controller unterstützt, ermöglichen es, das 'ganz normale' RAM-IC 6116 in ein FIFO mit genau der gerade benötigten (Sektor-) Länge zu verwandeln. Beteiligt an der erforderlichen Logik sind die ICs 7 bis 10 und 17 (siehe

| BR1 | Basis-Adresse 8-Bit-ECB | Basis-Adresse 16-Bit-ECB |
|---|---|---|
| 1-8 | 80h | C0h |
| 2-8 | 88h | - |
| 3-8 | 90h | D0h |
| 4-8 | 98h | - |
| 5-8 | A0h | E0h |
| 6-8 | A8h | - |
| 7-8 | B0h | F0h |
| J1 | 2-3 | 1-2 |
| J2 | 2-3 | 1-2 |
| J3 | 2-3 | 1-2 |
| J4 | 1-2 | 2-3 |

Tabelle 1. Mit der Brücke BR1 legt man die Basisadresse der c't-HDC-Karte im I/O-Bereich des Host-Rechners fest. Einige Basis-Adressen lassen sich im 16-Bit-Betrieb nicht erreichen, auch ist die Lage der HDC-Register zur Basis-Adresse im 16-Bit-Betrieb anders als im 8-Bit-Betrieb (siehe Tabelle 2).

| Register | Offset-Adresse 8-Bit-ECB | 16-Bit-ECB |
|---|---|---|
| Datenregister | 00h | 00h |
| Error-/WP-Register | 01h | 08h |
| Sektoranzahl | 02h | 02h |
| Sektor#/GAP3 | 03h | 0Ah |
| Zylinder (Low-Byte) | 04h | 04h |
| Zylinder (High-Byte) | 05h | 0Ch |
| SDH-Register | 06h | 06h |
| Status-/Kommando-Reg. | 07h | 0Eh |

Tabelle 2. Für den 8-Bit- und den 16-Bit-Betrieb ergeben sich durch die Vertauschung von Adreßleitung A0 gegen A3 unterschiedliche Lagen der HDC-Register bezogen auf die eingestellte Basisadresse.

Schaltplan). Die Adressen für den Speicher-Chip IC 7 (Sektorpuffer) generiert ein rücksetzbarer Zähler bestehend aus IC 8 und 9. Die erwähnte Unterstützung des WD1010 besteht nun darin, daß er den Adreßzähler stets zum richtigen Zeitpunkt zurücksetzt. Dazu später noch Genaueres.

Sowohl der Controller-Chip als auch die CPU müssen die Möglichkeit haben, den Sektorpuffer zu füllen oder zu leeren. Zunächst werden die Machenschaften der CPU betrachtet. Sie greift über die Adresse des Datenports auf das RAM zu, wobei nach jedem Zugriff der Adreßzähler (per Buf-Clk) inkrementiert wird.

Wenn nicht noch weitere Logik im Spiel wäre, könnte man den Speicherinhalt ohne weiteres mehrmals hintereinander auslesen. Man müßte nur 2048mal zugreifen und wäre wieder am Anfang dieses Ringpuffers. Damit ist bereits die erste Anforderung (nach einem FIFO nämlich) erfüllt. Da der Zähler nur in eine Richtung läuft, kann man von einer Anzahl Daten, die ins RAM geschrieben wurden, bei einem nachfolgenden Lesevorgang das zuerst geschriebene Byte (first in) auch zuerst wieder lesen (first out).

Es ist aber unglücklich, immer 2048 Zugriffe machen zu müssen, wenn die Sektorlängen 128, 256, 512 oder 1024 Byte betragen. Hier kommt IC 17 ins Spiel. Es liegt parallel zum SDH-Register des Controllers und speichert wie dieses die Sektorlänge, die Drive-Select- und die Head-Select-Informationen.

Die vier möglichen Sektorlängen sind als 2-Bit-Information in IC 17 gespeichert. Über diese zwei Bits wird der Multiplexer IC 10 so gesteuert, daß er beim Erreichen des entsprechenden Zählerstandes die Leitung BRDY zum WD1010 aktiviert. Damit wird dem Controller kundgetan, daß der Sektorpuffer geleert (beim Lesen) beziehungsweise gefüllt wurde (beim Schreiben). Daraus, daß das BRDY-Signal per Hardware aus dem Adreßzählerstand ermittelt wird, erklärt sich auch, warum stets die im SDH-Register vereinbarte Anzahl Bytes transferiert werden muß: der Controller erkennt sonst kein Transfer-Ende.

Sowie BRDY aktiv wird (steigende Flanke), setzt der WD1010 sein DRQ-Bit zurück und signalisiert damit der CPU, daß zunächst keine weiteren Zugriffe auf den Sektorpuffer erlaubt sind. Damit ist der Datentransfer zwischen Sektorpuffer und CPU abgeschlossen, und wenn es sich um eine Lese-Operation der CPU gehandelt hat, dann hatte der WD1010 ja bereits Daten von der Platte gelesen und hat somit bis zum nächsten Kommando erstmal frei.

## Immer nur einer

Hat die CPU jedoch ein Schreibkommando abgesetzt, so zeigt der Controller-Chip durch Setzen des DRQ-Bits an, daß er den Adreßzähler zurückgesetzt hat und die CPU den Sektorpuffer beschreiben darf. Wenn der Controller jetzt ein BRDY erhält, heißt das, daß alle notwendigen Daten zum Schreiben auf die Platte im Sektorpuffer stehen.

Mit Hilfe der Leitung $\overline{BCS}$ schaltet der Controller nun den Bus für Zugriffe von außen ab. Gleichzeitig wird dadurch IC 6 in folgender Weise aktiv: Es legt auf Bit 7 des Datenbusses zum Host das Signal L-BCS, also den durch IC 11 invertierten und gelatchten Zustand des Controller-Signals $\overline{BCS}$.

Das bewirkt nichts anderes, als daß dem Host-Rechner ein gesetztes Busy-Bit vorgegaukelt wird, wenn er jetzt auf das Statusregister zugreift. Während der Controller die Karte quasi von der Außenwelt abhängt, kommt die CPU nämlich auch nicht ans Statusregister heran beziehungsweise würde zufällige Buszustände auswerten. Damit bildet das simulierte BUSY den einzigen definierten Kontakt des Controllers zur CPU.

Da Controller-Chip und CPU asynchron, also mit unterschiedlichen Taktfrequenzen arbeiten, muß auch noch eine Synchronisation dieses Busy-Bits mit dem Zugriff der CPU berücksichtigt werden (über IC 11), damit die CPU im Moment der Bus-Freigabe durch den WD1010 nicht Unfug einliest.

Nachdem der Controller $\overline{BCS}$ aktiviert hat, gibt er wieder einen Puls auf der Leitung $\overline{BCR}$ aus und setzt den Adreßzähler zurück. Für jedes Byte, das er transferiert, sei es zur Winchester oder in den Sektorpuffer, erzeugt er mittels seiner Schreib-/Lesesignale ($\overline{IWE}$, $\overline{IRE}$, steigende Flanke) über IC 5, IC 4 und IC 18 einen Buf-Clk-Impuls und zählt die Speicheradresse hoch.

Die nette Eigenschaft des Controllers, vor jedem Setzen des DRQ-Bits und nach Erhalt eines BRDY den Zähler zurückzusetzen, erleichtert den Aufbau des maßgeschneiderten FIFO beträchtlich. Die Handbücher von Western Digital lassen sich daher auch lang und breit über dieses Feature aus.

Es gibt natürlich noch andere Möglichkeiten, den Sektorpuffer zu realisieren, zum Beispiel mit speziellen FIFO-Bausteinen. Das hat allerdings gleich zwei Riesennachteile. Zum einen sind fertige FIFOs teuer, und zwar um so mehr, je mehr Speicherkapazität sie haben. Zum andern läßt sich die Möglichkeit der Fehlerkorrektur beim Einsatz des WD2010 damit nicht nutzen, da der Controller bei diesen FIFOs keinen zweiten Adressierdurchgang starten kann, ohne deren Inhalt zu löschen.

Aus all dem ist hoffentlich auch klar geworden, daß sich immer nur ein Sektor im Sektorpuffer befinden kann, also beispielsweise kein Track-Buffering möglich ist. Weiterhin heißt das, daß der Sektorpuffer immer vollständig gefüllt oder geleert werden muß, damit alles ordnungsgemäß zusammenspielt.

## Full Speed

Manch einer möchte den Winchester-Controller sicher gerne am c't 180 mit der HD64180-CPU betreiben. Kann er. Sogar mit der vollen Geschwindigkeit der chip-internen DMA. Denn das Datenregister (also der Sektorpuffer) wird nicht durch das Wait-Signal gebremst. Man muß lediglich die Zugriffszeiten des RAMs einhalten.

Das Wait-Signal wird nur beim Zugriff auf das Taskfile erzeugt (das sind die WD1010-internen Register, siehe erster Teil des Artikels). Aber dazu wird man kaum eine DMA einsetzen, zumal zum Beispiel Bausteine wie die Z80-DMA keine Einzel-Byte-Zugriffe ermöglichen.

Tabelle 3 liefert ein bißchen konkretes Zahlenmaterial zur Orientierung. Die rechnerisch ermittelten Werte gelten zwar für unterschiedliche Sektorlängen, jedoch stets für den Transfer von 1024 Bytes. Bei der DMA wird der Continuous Mode (7 Takte pro Byte plus Overhead) zugrunde gelegt, bei der Z80-CPU Blocktransfer mit INIR/OTIR.

Man erkennt aus dieser Tabelle, daß der Einsatz einer DMA einiges 'bringen' kann. Hierzu noch ein Hinweis: Die Karte liefert kein DATA-READY-Signal, mit dessen Hilfe der DMA-Transfer direkt per Hardware gestartet werden könnte. Das ist aber auch nur selten nötig, denn die CPU kann ja (wie ohne DMA auch) das

| | 1024 | 512 | 256 | 128 |
|---|---|---|---|---|
| 3 MHz Z80-DMA | 2,5 ms | 2,6 ms | 2,9 ms | 3,4 ms |
| 4 MHz Z80-DMA | 1,9 ms | 2,0 ms | 2,2 ms | 2,6 ms |
| 6 MHz Z80-CPU | 3,6 ms | 3,6 ms | 3,6 ms | 3,6 ms |

Tabelle 3. Zeiten für den Transfer von 1024 Bytes mit DMA und CPU bei unterschiedlichen Sektorlängen. Da es keine 6-MHz-DMAs gibt, werden diese in 6-MHz-Systemen meistens mit halbemTakt gefahren.

DRQ-Bit im Status-Register des WD1010 testen und dann die DMA per Software starten. Der Overhead, der durch dieses Polling entsteht, ist nicht sehr groß und könnte eigentlich nur bei Multitasking-Systemen die Systemleistung merkbar bremsen.

Es soll allerdings nicht verschwiegen werden, daß der WD1010 synchron zum Signal DRQ den Ausgang BDRQ (Pin 36) für genau solche Zwecke hat. Laut Datenbuch sollte man damit wunderbar DMA-Chips starten können, jedoch haben wir keine praktischen Untersuchungen durchgeführt und wissen daher nicht, ob es irgendwelche 'Sonderfälle' (vielleicht doch nicht synchron zu DRQ) oder Timing-Probleme gibt.

## Datentrennung

Nun zum eigentlichen 'Kasus Knusus', dem Datenseparator. Er ist sicherlich einer der Hauptgründe dafür, daß so manche Selbstbauversuche gescheitert oder gar nicht erst angegangen worden sind. Nur die Ruhe, unsere Schaltung läuft ohne Probleme!

Zentrales Element ist eine PLL-Schaltung, die in der Lage sein muß, auf die Datenrate von 5 MBit/s zu synchronisieren. Rein digital realisierte PLLs, wie es sie für Floppy-Controller mittlerweile als Single-Chip-Lösung gibt, müssen mit Vielfachen der resultierenden VCO-Frequenz betrieben werden. Ist das bei Floppies mit ihrer 'niedrigen' Bit-Rate recht einfach, kommt man bei Hard-Disks in Größenordnungen von 50 MHz. Dafür gibt es noch keine 'Komplett-Chips'.

Da Festplatten jedoch eine wesentlich stabilere Drehzahl als beispielsweise Floppy-Laufwerke haben, ist der Phasenjitter sehr gering, so daß eine PLL mit analog angesteuertem VCO ohne Probleme eingesetzt werden kann.

Die folgende Beschreibung zeigt nur die Funktionsblöcke auf. Wer sich genauer mit dieser Materie befassen will, findet eine recht genaue Darstellung mit Berechnungsbeispielen etwa im Datenbuch zum Separator-Chip für Hard-Disks HDC 9226 (der im c't-HDC nicht zum Einsatz kommt).

IC 19 bildet mit den beiden D-Flipflops einen Phasendetektor, der das Pump-Up- und Pump-Down-Signal zum Steuern der Frequenz des VCO (IC 20) liefert. Das verzögerte READ-DATA-Signal (RD, abgegriffen an Pin 8 von IC 27) erreicht den Phasendetektor erst rund 50 Nanosekunden nach dem Eintreffen des unverzögerten READ-DATA-Impulses (RD unverz., Pin 8 von IC 26).

Mit R3 und R24 wird die Offset-Spannung des VCO in die Mitte seines Aussteuerungsbereiches gelegt. C3, C2 und R8 stellen eine Art Tiefpaß mit zwei Eckfrequenzen dar, der die digitalen Ausgangssignale von IC 19 glättet und dadurch in die analoge Steuerspannung für den VCO wandelt.

Mit R5 und R6 erfolgt der genaue Frequenzabgleich des VCO auf die Ruhefrequenzlage von 10 MHz. IC 26 legt zu allen Zeiten, in denen nicht gelesen wird, das 10-MHz-Referenzsignal an die PLL. Damit lassen sich Drifterscheinungen unterdrücken, und man erreicht eine schnellere Synchronisation beim Lesen.

## Endlich löten

Das Bestücken und Löten der Controller-Platine ist (verglichen mit vollgepfropften RAM-Karten) recht einfach. Es empfiehlt sich, die kostbareren ICs zu sockeln und beim Erwerb der Fassungen nicht zu geizen. Unsere Redaktions-Testmuster wurden vollständig mit recht preiswerten Sockeln (gedrehte Kontakte) bestückt, aber eigentlich sollte man zu Fassungen mit massiver vergoldeten Kontakten greifen, als wir es taten (AUGAT oder ähnliches).

Das direkte Einlöten von ICs ist natürlich die beste Kontaktiermöglichkeit und auch weitaus billiger als die Verwendung hochwertiger IC-Fassungen. Sollte allerdings eine Fehlersuche fällig werden, ist es mit schnellem IC-Austausch natürlich Essig. Hier muß jeder Leser selbst sein Budget gegen seine Arbeitszeit aufwiegen.

IC 19 muß auf jeden Fall mit einem Sockel versehen werden, da es zum Abgleich entfernt werden muß. Aber auch IC 13 (WD1010) sollte man nicht einlöten. Und die Bauteile C1, R6 und R5 dürfen noch nicht eingesetzt werden (erst nach dem Abgleich!).

Zunächst werden die Bussteuerung und der Sektorpuffer bestückt. Dazu folgende Bauteile einsetzen: IC 1 bis IC 11 sowie IC 18, R15 bis R17 und C14. Die Abblockkondensatoren (sie haben alle einen Wert von 0,1 µF) sollten jetzt ebenfalls eingelötet werden. Wenn man nun die 5-V-Betriebsspannung (über ein Labor-Netzgerät) an die Karte legt, sollte die Stromaufnahme der Karte rund 150 mA betragen (der 12-V-Zweig ist noch nicht in Betrieb, da IC 22 noch nicht bestückt wurde). Wenn der Strom erheblich darüber liegt, empfiehlt sich eine Suche nach ungewollten Lötbrücken.

Wenn soweit alles klar ist, kann die Karte dem Rechner-System zugemutet werden (ohne Betriebsspannung auf den Rechner-Bus stecken, versteht sich). Läuft Ihr Rechner nach dieser Aktion immer noch vorschriftsmäßig, so ist der erste Teil der Schlacht gewonnen.

## Computer Aided Testing

Denn jetzt hat man die Möglichkeit, die Karte mit Hilfe des Rechners zu testen. Es ist kein Luxus, diese Tests auch wirklich zu machen, denn Freund Murphy schlägt natürlich am liebsten da zu, wo es teuer wird und ordentlich weh tut.

Die beiden reinen Testprogramme HDCTEST und HDCSC sind ebenso wie der (auch zum Testen benötigte) Formatter in Turbo-Pascal geschrieben. Da die meisten Definitionen und Prozeduren in allen drei Programmen benötigt werden, wurden diese in gemeinschaftlich genutzte Include-Dateien verlagert.

| Laufwerksparameter | Beispielwert (Wertebereich) | | Bedeutung |
|---|---|---|---|
| Anzahl_Koepfe | 4 | (2,4,8) | Anzahl der Köpfe |
| Anzahl_Zylinder | 611 | (1...1024) | Anzahl der Zylinder |
| Sektoren_pro_Spur | 9 | | Anzahl der Sektoren/Spur |
| Sektor_Länge | 1024 | (128, 256 512,1024) | Anzahl der Bytes/Sektor |
| Speedvar | 3 | (abhg. v. Laufw.-Typ) | Angabe der Drehzahlabweichung direkt in Prozent |
| PRECOMP_Spur | 128 | (abhg. v. Laufw.-Typ) | Spur, ab der Precompensation nötig wird |

Besondere Parameter:

| | | | |
|---|---|---|---|
| track_length | 10416 | | Länge einer Spur in Bytes |

Im Programm HDCFORM.PAS wird geprüft, ob das gewählte Format überhaupt auf eine Spur paßt. Es wird dabei der formatabhängige Platzbedarf mit der Spurlänge track_length verglichen. Dieser Wert läßt sich aus der Rotationsgeschwindigkeit der Platte sowie der maximalen Transferrate ermitteln.

$$\text{track\_length} = \frac{\text{Transferrate (5 MBit/s)}}{\text{Drehzahl (3600 U/min) x 8 Bits/Byte}} = \frac{\text{625 000 Byte/s}}{\text{60 U/s}} = 10416$$

Der Wert von track_length müßte korrigiert werden, wenn Ihnen ein Laufwerk mit leicht abweichender Drehzahl in die Hände fällt (die es "eigentlich" aber gar nicht gibt...)

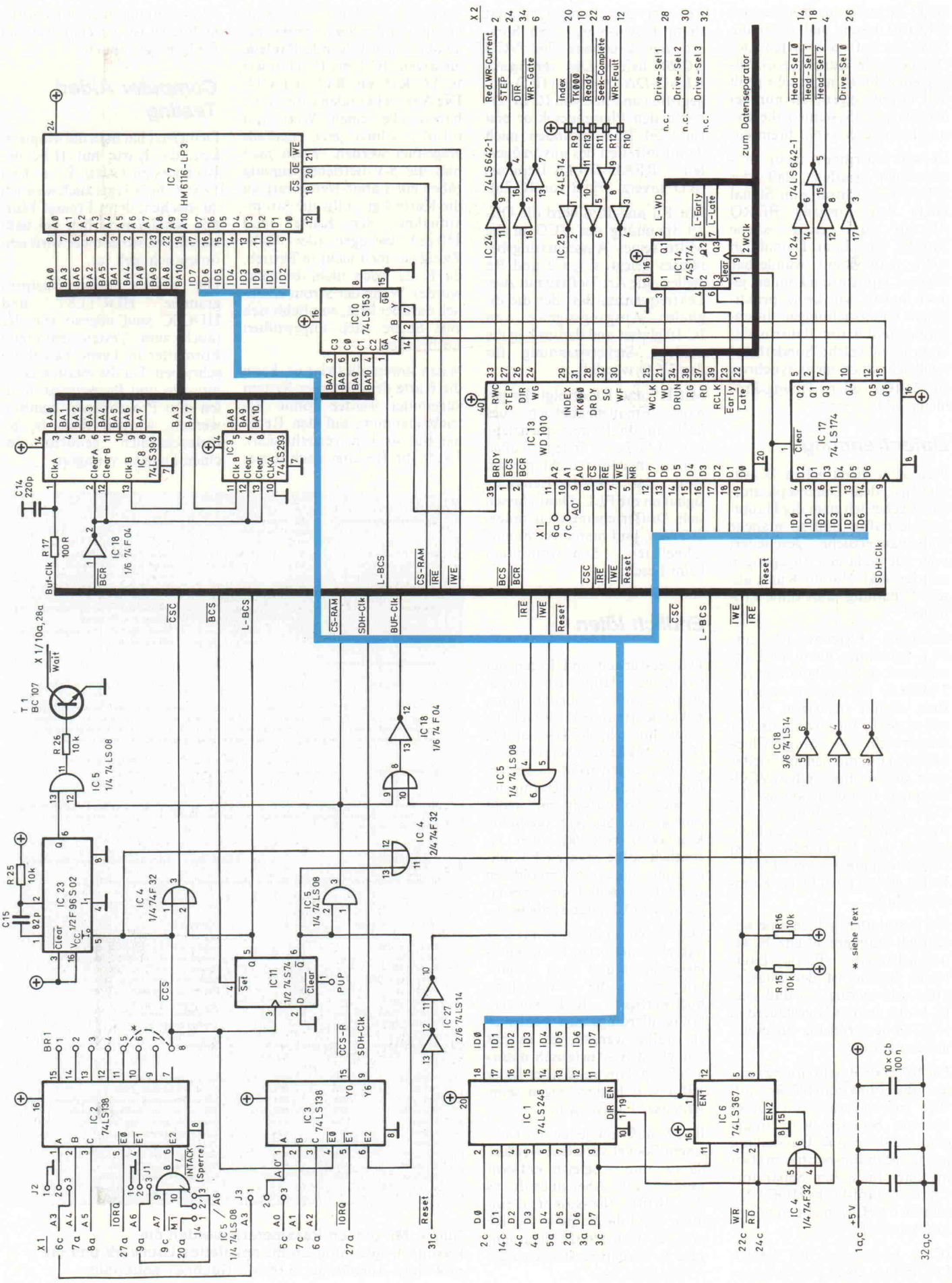
| | | | |
|---|---|---|---|
| Interleaving | 2 | (s. c't 8/86) | physikalischer Sektorversatz |

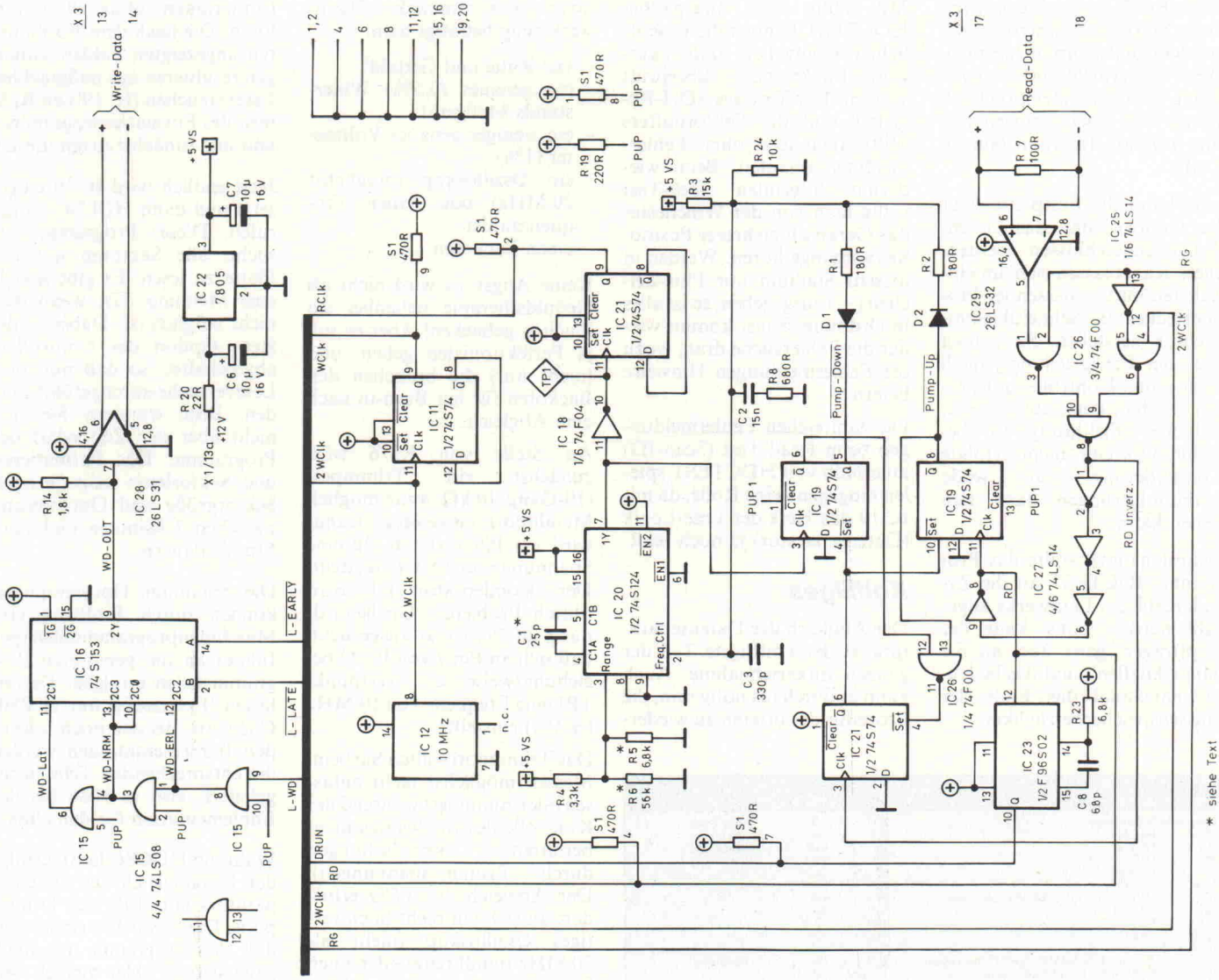
Dieser Parameter wird beim Aufruf des Formatierprogrammes erfragt.

| WD1010-Kommando | Wert | Bedeutung |
|---|---|---|
| Restore_CMD | $10 | Köpfe auf Zylinder 0 fahren |
| Seek_CMD | $70 | Köpfe positionieren (explizit) |
| Read_CMD | $20 | Sektor lesen (mit Retry) |
| Read_Fast_CMD | $21 | Sektor lesen (ohne Retry) |
| Write_CMD | $30 | Sektor schreiben (mit Retry) |
| Scanid_CMD | $40 | ID lesen (mit Retry) |
| Scanid_Fast_CMD | $41 | ID lesen (ohne Retry) |
| Format_CMD | $50 | Formatieren |

| Portadressen | Beispielwert c't86 | Beispielwert PROF-80 | Bedeutung |
|---|---|---|---|
| hdcdata | $E0 | $90 | Daten-Register (R/W) |
| hdcerr | $E8 | $91 | Error-Register (RO) |
| hdcwpc | $E8 | $91 | Write-Precompensation-Register (WO) |
| hdcscnt | $E2 | $92 | Sektor-Anzahl-Register (R/W) |
| hdcsek | $EA | $93 | Sektor-Nummer-Register/GAP3 (R/W) |
| hdccyl | $E4 | $94 | Zylinder Low-Byte (R/W) |
| hdccyh | $EC | $95 | Zylinder High-Byte (R/W) |
| hdcsdh | $E6 | $96 | SDH-Register (R/W) |
| hdcstat | $EE | $97 | Status-Register (RO) |
| hdccmd | $EE | $97 | Kommando-Register (WO) |

**Bild 1. Mit diesen Parametern werden die Pascal-Programme an Ihr spezielles Laufwerk und die gewählte Adreßlage in Ihrem Rechner angepaßt.**


IC 7 HM6116-LP3
IC 10 74LS153
IC 24 74LS642-1
IC 25 74LS14
IC 14 74S174
IC 13 WD 1010
IC 17 74LS174
IC 24 74LS642-1
IC 8 74LS393
IC 9 74LS293
IC 18 1/6 74F04
IC 5 1/4 74LS08
IC 4 1/4 74F32
IC 23 1/2 F96S02
IC 11 1/2 74S74
IC 27 2/6 74LS14
IC 18 3/6 74LS14
IC 2 74LS138
IC 3 74LS138
IC 1 74LS245
IC 6 74LS367
IC 5 1/4 74LS08
IC 4 1/4 74F32
T 1 BC 107
X 1/10a,28a
Wait
INTACK (Sperre)
Buf-Clk
SDH-Clk
L-BCS
CS-RAM
BUF-Clk
Reset
zum Datenseparator
Red.WR-Current
STEP
DIR
WR-Gate
Index
TK000
Ready
Seek-Complete
WR-Fault
Drive-Sel 1
Drive-Sel 2
Drive-Sel 3
Drive-Sel 0
Head-Sel 0
Head-Sel 1
Head-Sel 2
L-WD
L-Early
L-Late
2 WClk
n.c.
* siehe Text
+5 V
10 x Cb 100 n




**Auf dem linken Schaltbild finden Sie die Schnittstelle zum ECB-Bus, den Sektorpuffer und den Controller-Chip samt Steuerbus. Oben den Datenseparator nebst Schreib-/Leselogik.**

Bitte beachten Sie, daß alle Programme darauf eingerichtet sind, nur Laufwerke mit einer Kopfanzahl, die direkt einer Zweierpotenz entspricht (also 2, 4, 8), korrekt zu bedienen. Um Laufwerke mit abweichender Kopfanzahl zu bedienen, sind daher einige Änderungen (bei den Maskierungen) vorzunehmen.

Die Anpassung an das vorhandene Laufwerk muß in HDCDEF.INC erfolgen. Die einzelnen Parameter sind nochmals gesondert in Bild 1 aufgeführt und erklärt. Es sollte auch Nicht-Pascal-Freaks leichtfallen, daraus die entsprechenden Assembler-Files zu kreieren. Der Vorteil von Pascal liegt hier in der einfachen Übertragbarkeit von einer CPU auf eine andere. Somit dürften sich auch c't86- und c't68k-Besitzer ohne Probleme ihre Test-Utilities stricken können.

Mit Hilfe des Programmes HDCTEST kann jetzt eine Prüfung des Sektorpuffers und damit der Adressierung erfolgen. Dazu wird Pin 1 der Fassung von IC 13 (nicht bestückt) über einen 1-kΩ-Widerstand auf + 5V (Pin 40) gelegt, Pin 9 von IC 1 mit 3k3 auf Masse. Nach Aufruf des Programmes HDCTEST sollten beide Speicher-Tests fehlerfrei vonstatten gehen. Dabei wird das gesamte RAM – also der volle 2-KByte-Bereich – geprüft, da ja der WD1010 nicht bestückt ist und somit kein Rücksetzen der Zähler möglich ist.

Während des nachfolgenden Seek-Tests hängt sich das Programm wegen des fehlenden Controller-Chips auf, und aus demselben Grund sollte das Programm auch den Wert im SDH-Register monieren. Andere Fehler, die im ersten Teil des Programmes angezeigt werden, deuten auf weiterhin versteckte Lötbrücken oder möglicherweise defekte ICs hin.

Wenn auch diese Hürde erfolgreich genommen wurde, wird es langsam ernst. Jetzt werden die restlichen ICs und Bauteile (bis auf IC 13, IC 19, C1, R6 und R5, die kommen noch später dran!) bestückt. Die Stromaufnahme beträgt dann rund 500 mA im 5-V-Zweig und knapp 100 mA bei 12V. Jetzt sollten die Pegel an allen Pins der Fassung für IC 13 überprüft werden. Für offene Leitungen, also Ausgänge des WD1010, kann man mit Hilfe eines 680-Ω-Widerstandes 0- und 1-Pegel simulieren. Die Spannungen sollten dabei bei knapp 5V ('1') beziehungsweise 0V ('0') liegen.

Ist soweit alles in Ordnung, kommt endlich IC 13 an seinen Platz (Stromaufnahme jetzt etwa 700 mA bei 5V). Klingt alles etwas umständlich, aber bei einem Preis von zur Zeit noch deutlich über 100 Märkern für den WD1010 kann ein wenig Vorsicht wohl nicht schaden – oder sehen Sie gerne 100-Mark-Scheine brennen?

Bevor ein Laufwerk angeschlossen wird, ruft man sicherheitshalber noch einmal HDCTEST auf. Das SDH-Register darf nun nicht mehr Ursache einer Fehlermeldung sein, und das Programm sollte auch den Seek-Test mit einer ordnungsgemäßen Fehlermeldung (Restore failed, Error-Code = 4) ohne Absturz überstehen. Beim

'Disk-Read-Test' sollte die Meldung 'Status = 01, Error = 04' auf dem Bildschirm erscheinen. Diese Fehlermeldungen zeugen (auch wenn es widersprüchlich klingt) von einer – soweit bislang prüfbar – funktionsfähigen Karte.

Jetzt kann die Festplatte über das Control- und das Daten-Kabel angeschlossen werden. Diese Kabel lassen sich am einfachsten mittels passender Pfostenstecker in Schneidklemm-Technik herstellen. Sie sollten nach der Fertigstellung jedoch noch einer Kontrolle unterzogen werden, denn aus eigener, leidvoller Erfahrung ist bekannt, was eine nicht erfolgte Kontaktierung für wilde Schlußfolgerungen nach sich ziehen kann.

Spätestens jetzt sollte das Programm HDCTEST an die Zylinderzahl des Laufwerks angepaßt werden. Sonst kann der Kopfträger 'ganz übel an die Bande knallen', und das bedeutet manchmal das Ende der Massenspeicherherrlichkeit.

Mit Hilfe des angepaßten HDCTEST können die wesentlichen Controller- und Laufwerks-Funktionen überprüft werden. Der Test des SDH-Registers und des Sektorpuffers sollte auch jetzt ohne Fehlermeldung ablaufen. Beim wiederum folgenden Seek-Test sollte man von der Winchester das Geräusch mehrerer Positioniervorgänge hören. Werden in diesem Stadium nur Plus-Zeichen (+) ausgegeben, so ist alles in Ordnung. Sonst kommt wieder die Fehlersuche dran, wozu die Fehlermeldungen Hinweise liefern.

Die zahlreichen Fehlermeldungen beim Read-Test (Scan-ID) innerhalb von HDCTEST spielen hingegen keine Rolle, da mit IC 19 das Herz der Lese-Logik (Datenseparator) ja noch fehlt.

## Kniffliges

Der Abgleich des Datenseparators ist der kniffligste Teil der ganzen Inbetriebnahme. Auch kann es durchaus nötig sein, die Prozedur des öfteren zu wiederholen. Als minimales Handwerkszeug benötigt man:

- viel Ruhe und Geduld!
- ein genaues (0,5%) Widerstands-Meßgerät
- ein weniger genaues Voltmeter (1%)
- ein Oszilloskop (möglichst 20 MHz) oder einen Frequenzzähler
- einen Backofen

Keine Angst, es wird nicht als Geduldstherapie nebenbei ein Kuchen gebacken! Aber es soll ja Perfektionisten geben, und (nicht nur) die brauchen den Backofen für ein Burn-in nach dem Abgleich.

An Stelle von R5/6 wird zunächst ein Trimmpoti (10-Gang, 10 kΩ, wenn möglich Metallfilm) eingelötet. Damit wird am Pin 3 von IC 20 eine Spannung von 2,5 V eingestellt. Der Kondensator C1 wird (durch Probieren, Größenordnung 20-25 pF) so ausgesucht, daß sich an Pin 7 von IC 20 beziehungsweise am Testpunkt TP1 eine Frequenz von 10 MHz (± 5%) einstellt.

Das Trimmpoti sollten Sie beim Messen möglichst nicht anfassen oder zumindest während der Kontrolle des Meßwertes nicht berühren (Verfälschungen durch Brumm-Spannungen). Der Abgleich des VCO erfordert an sich ein recht hochwertiges Oszilloskop (mehr als 20 MHz Bandbreite) oder einen Frequenzzähler.

Man kann allerdings auch mit einem 'minderwertigeren' Oszilloskop sehr gute Erfolge erzielen, indem man sich der Lissajousschen Figuren erinnert und bedient. Als Referenztakt verwendet man dazu den Ausgang des 10-MHz-Taktgenerators auf der Platine. Allerdings sollte man unbedingt darauf achten, daß die Figur stabil ist, also nicht 'wackelt' oder 'umkippt', denn dann hat man unter Umständen nur die Oberwelle einer wesentlich niedrigeren Frequenz erwischt.

Für den weiteren Verlauf des Abgleichs (Sie sind noch lange nicht fertig) muß die Festplatte formatiert werden (IC 19 immer noch nicht bestücken!). Dazu dient das Utility-Programm HDCFORM, das ebenfalls an das Laufwerk und die Adreßlage des Controllers angepaßt werden muß (HDCDEF.INC). Nach dem Formatieren sind selbstverständlich alle alten Informationen auf der Platte verloren. Die nach dem Formatieren angezeigten Fehlermeldungen resultieren aus mißglückten Leseversuchen (IC 19!) im Rahmen des Formatierprogrammes und sind zunächst zu ignorieren.

Jetzt endlich wird IC 19 eingesetzt und dann HDCSC aufgerufen. Dieses Programm versucht, alle Sektoren auf der Platte zu lesen. Es gibt jeweils eine Meldung aus, wenn dies nicht möglich ist. Dabei ist die Retry-Option des Controllers abgeschaltet, so daß nur zwei Leseversuche durchgeführt werden. Bitte wundern Sie sich nicht über den Zeitbedarf der Programme: Das Formatieren und Sektorlesen kann je nach Sektorgröße und Optimierung zwischen 7 Minuten und einer Stunde dauern.

Die genannten Optimierungen können durch Einfügen von Maschinenprogrammteilen (per Inline) an die genannten Programmstellen erfolgen. Derzeit liegen 'Ersatzteile' nur in Z80-Code vor. Bei den noch folgenden Implementationen werden die entsprechenden Teile nachgeliefert, also in 8086 bei der Implementation für den c't86.

Während HDCSC läuft, erfolgt der Feinabgleich des Datenseparators mit Hilfe des Trimmpotis. Dieses wird so eingestellt, daß das Programm nur noch ganz selten Fehler meldet. Mit einigen Fehlern werden Sie vermutlich zunächst leben müssen. Wenn Sie später im normalen Betrieb mit der Retry-Option arbeiten, werden diese Fehler aufgefangen, sofern die Platte keine Beschädigungen aufweist beziehungsweise die Formatierung nicht fehlerhaft ist.

Läßt sich durch das Abgleichen kein Zustand herbeiführen, bei dem die weitaus überwiegende Mehrzahl der Sektoren ordnungsgemäß gelesen werden kann, ist eventuell die Schreiblogik oder die PLL des Controllers nicht in Ordnung (hoffentlich nicht die Platte). Geht es, so sollte man versuchen, die Mitte des fehlerfreien Bereiches einzustellen.

Dann muß man das Poti vorsichtig (nicht den Wert verstellen) auslöten und den eingestellten Widerstand bestimmen. Wie Sie aus dem Schaltbild ersehen können, liegen die Widerstände R5 und R6 parallel. So ist es

### Stückliste

**Integrierte Schaltungen**

| | | |
|---|---|---|
| IC1 | 74LS245 | |
| IC2,3 | 74LS138 | |
| IC4 | 74F32 | |
| IC5,15 | 74LS08 | |
| IC6 | 74LS367 | |
| IC7 | 6116 L-3 o. ä. (2K x 8-RAM, low power) | |
| IC8 | 74LS393 | |
| IC9 | 74LS293 | |
| IC10 | 74LS153 | |
| IC11,19,21 | 74S74 | |
| IC12 | Quarzoszillator 10 MHz (DIL-14-kompatibel) | |
| IC13 | WD1010-05 (Hard-Disk-Controller) | |
| IC14 | 74S174 | |
| IC16 | 74S153 | |
| IC17 | 74LS174 | |
| IC18 | 74F04 | |
| IC20 | 74S124 | |
| IC22 | 7805 (Spannungsstabilisator 5V / 1A) | |
| IC23 | F 96S02 (Fairchild) o. AM 26S02 (AMD) | |
| IC24 | 74LS642-1 ("-1": doppelter Treiberstrom) | |
| IC25,27 | 74LS14 | |
| IC26 | 74F00 | |
| IC28 | AM 26LS31, AM 36LS31 (RS-422-Treiber) | |
| IC29 | AM 26LS32, AM 36LS32 (RS-422-Empfänger) | |

**Diskrete Halbleiter**

| | | |
|---|---|---|
| T1 | BC 107 o. ä. (NPN-Transistor) | |
| D1,2 | 1N4148 o. ä. | |

**Widerstände**

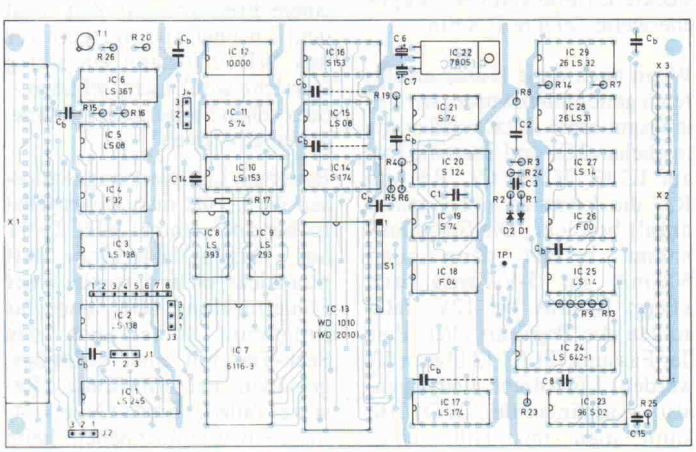
| | | |
|---|---|---|
| R1,2,9 bis 13 | 180R | 7 |
| R3 | 15k | 1 (Metallfilm) |
| R4 | 3k3 | 1 (Metallfilm) |
| R5 | 6k8* | 1 (Metallfilm) |
| R6 | 56k* | 1 (Metallfilm) |
| R7,17 | 100 R | 2 |
| R8 | 680R | 1 |
| R14 | 1k8 | 1 |
| R15,16 | 10k | 2 |
| R19 | 220R | 1 |
| R20 | 22R | 1 |
| R23 | 6k8 | 1 |
| R24 | 10k | 1 (Metallfilm) |
| R25,26 | 10k | 2 |
| S1 | Widerstands-Array (SIL, 9polig, 8 x 470 R) | |

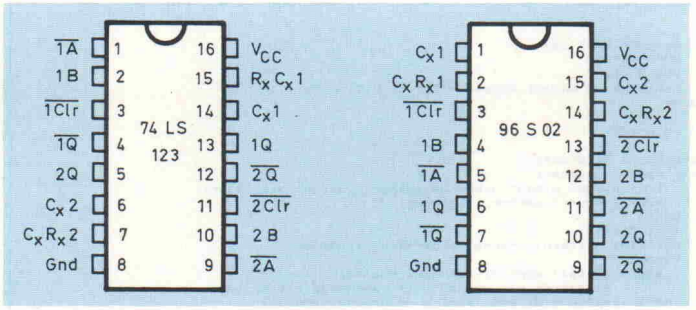
**Kondensatoren**

| | | |
|---|---|---|
| Cb | 10 x 100nF | (keramisch, RM 5) |
| C1 | 25pF | (keramisch, RM 2,5) |
| C2 | 15nF | (Wima, MKS, RM 7,5) |
| C3 | 330pF | (keramisch, RM 2,5) |
| C6,7 | 10µF/15V | (Tantal, RM 2,5) |
| C8 | 68pF | (keramisch, RM 2,5) |
| C14 | 220pF | (keramisch, RM 2,5) |
| C15 | 82pF | (keramisch, RM 2,5) |

**Steckverbinder**

| | |
|---|---|
| X1 | 64polige VG-Steckerleiste |
| X2 | Wrap-Pfosten (34polig, zweireihig) |
| X3 | Wrap-Pfosten (20polig, zweireihig) |
| BR1 | Wrap-Pfosten (8polig, einreihig) |
| J1 bis J4 | Wrap-Pfosten (3polig, einreihig, 4x) |






Die Chips unterscheiden sich folgendermaßen in ihrem logischen Verhalten:

| 74 LS 123 | | | | 96 S 02 | | | |
|---|---|---|---|---|---|---|---|
| Clr | $\bar{A}$ | B | Q | Clr | $\bar{A}$ | B | Q |
| H | L | ∫ | Puls | H | H | ⎍ | Puls |
| H | ⎍ | H | Puls | H | ⎎ | L | Puls |

**Bild 2. Wenn Sie den 96S02 durch einen 74LS123 ersetzen wollen, dann müssen Sie sich einen Sockeladapter gemäß der angegebenen Anschlußbilder erstellen und die Logik-Abweichung korrigieren.**

etwas einfacher, durch Kombination dieser zwei Widerstände (bitte wie auch R4 nur Metallfilm!) den ausgemessenen Wert auf ein Prozent genau nachzubilden.

Wer kein so 'hochprozentiges' Widerstandsmeßgerät zu Hause hat, kann sich auch behelfen. Die meisten Meßgeräte haben nämlich eine gute Wiederholgenauigkeit, sind also relativ, nicht absolut, recht genau. Dadurch reicht es vielfach aus, wenn Sie die Widerstandskombination bestmöglich an den Meßwert vom Trimmpoti annähern.

## Ab in den Ofen

Nicht nur die angesprochenen Perfektionisten sollten die Karte jetzt in den Backofen legen und bei etwa 80°C eine Stunde lang garen lassen. Denn sonst blüht Ihnen unter Umständen die 'same procedure' alle paar Monate. Anschließende Tests mit HDCSC sollten keine oder nur ganz vereinzelt Fehler melden. Ansonsten muß die gesamte Prozedur wiederholt werden.

Ein keineswegs scherzhaft gemeinter Tip am Rande: Ein Burn-in in einem Mikrowellenherd könnte problematisch sein, da hier prinzipbedingt eine völlig ungleichmäßige Erwärmung (abhängig vom spezifischen Widerstand der Materialien) zu erwarten ist. Ausprobiert haben wir es zwar nicht, es steht jedoch zu vermuten, daß dadurch vor allem die ICs einer inneren Zerreißprobe (im Sinne des Wortes) unterzogen werden.

Wenn alles klargegangen ist, steht dem Einbinden eines Hard-Disk-Treibers in das BIOS des Betriebssystems nichts mehr im Wege. Diesem Thema werden wir einen eigenen Beitrag in der nächsten c't widmen, wobei zunächst die Einbindung in das Betriebssystem CP/M 3.0 vorgeführt wird.

Vor dem Gebrauch im 'echten' Arbeitseinsatz sollte die Hard-Disk nochmals formatiert und getestet werden. Dann empfiehlt es sich, das Formatierprogramm im Schrank zu versenken und zu vergessen.

## Einkaufsbummel

Manchen Leser wird beim Betrachten der Stückliste ein leichter Frust überkommen: Viele ICs stammen aus der S-(wie Schottky)-Familie der TTL-Sippschaft, einige aus dem F-(wie 'Fast')-Zweig. Also alles nicht gerade das, was man so zu Hause 'rumliegen hat.

Natürlich sind hier gewisse Variationen möglich, so könnte hier und da auch ein AS- oder ALS-Typ den Ansprüchen genügen (positive Rückmeldungen – bitte erst nach längerer Erprobung – werden von der Redaktion gern weitergegeben), aber die fehlerfreie Lauffähigkeit der Karte haben wir bislang nur mit der angegebenen Bestückung verifiziert.

Bevor Sie aber wahllos drauflosexperimentieren, noch ein paar Tips: IC 19 muß ein S-Typ sein, da andere ICs nicht in der Lage sind, den nötigen Strom zu liefern. IC 15 dient als Verzögerungsleitung für die Precompensation. Hier also bitte nicht zu sehr drehen, ALS-Typen zum Beispiel haben höhere Schaltgeschwindigkeiten. Auch für IC 27 (LS 14) sollte man keine ICs mit abweichenden Laufzeiten verwenden, da die damit aufgebaute Verzögerungsleitung maßgeblich den Fangbereich der PLL bestimmt.

Ein Ausweichtyp für den F96S02 von Fairchild ist der AM26S02 von AMD. Zugegeben, die Beschaffung dieses ICs ist etwas schwierig, aber wir sind guter Hoffnung, daß er bald auch im Einzelhandel auftaucht. Wenn es gar nicht anders geht, ist auch eine Kompromißlösung mit einem 74LS123 möglich, der wegen des anderen Pin-out jedoch nur über einen Adaptersockel angeschlossen werden kann.

Bild 2 zeigt Ihnen, wie die Pins aufeinander abzubilden sind, wobei auch unbedingt noch der kleine Logik-Unterschied berücksichtigt werden muß. Obwohl wir diese Variante erfolgreich ausprobiert haben, sollte sie nicht die Dauerlösung sein.

Zum einen weist der 74er eine um 10 bis 15 Nanosekunden höhere Gatterlaufzeit auf, so daß es nötig sein kann, ihn 'zwei Inverter früher' aus der Verzögerungsleitung (IC 27) auszukoppeln.

Viel unglücklicher ist jedoch, daß der 74LS123 flankengetriggert ist, der 96S02 jedoch pulsgetriggert. Ersterer könnte also leichter auf Spikes ansprechen.

Die anderen Chips sind durchaus gängig, nur beim 74LS642-1 sollte man auf die '-1' achten. Denn das bedeutet, daß dieser Chip die doppelte Treiberleistung gegenüber seinen Kollegen ohne '-1' aufbietet.

Im nächsten Teil des Beitrages werden wir uns, wie erwähnt, der Treiber-Software und deren Einbindung in das Betriebssystem CP/M 3.0 widmen.

### *Literatur*


Storage Management Handbook, Western Digital

Inside the PC (Sonderheft der BYTE, ca. Mai 85).

Components Databook 83/84, ZILOG

HDC 9226 Application Notes, Standard Microsystems Corporation


```
program hdcsc;
const
{$i hdcdef.inc}

type
   str10 = string[10];
   wrkstring = string[80];

var
   i          : integer;
   stp        : boolean;
   ch         : char;
   Gap3_Len   : integer;
   Sekt_Size  : integer;
   Buffer     : array[1..1024] of Byte;
   maxtrack   : integer;
   maxsek     : integer;


{$i hdcrout.inc}

begin
  clrscr;
  writeln('Abgleich Routine fuer den Winchester Controller [V 0.1]');
  writeln;
  HDC_Init;
  writeln('Der Test kann mit Tastendruck beendet werden !');
  writeln;
  writeln('Es werden nur fehlerhafte Sektoren mit einer Bildschirm-');
  writeln('ausgabe quittiert: >nn< nn=Statusregister. Erfolgt keine');
  writeln('Ausgabe, so ist die Hard-Disk fehlerfrei.              ');
  writeln;
  maxsek := Anzahl_Zylinder * Sektoren_pro_Spur * Anzahl_Koepfe;
  writeln('Anzahl der Sektoren = ',maxsek);
  writeln;
  writeln(' Testing all Sectors ... ');
  i := 0;
  repeat
    readsec(i);                  { Lese einen Sektor }
    i := i + 1;
    stp := keypressed;
  until (i >= maxsek) or stp;
  if stp then begin
    read(kbd,ch);
    writeln;
    writeln('Programm abgebrochen.');
  end;
  HDC_Deselect;
end.
```

**Das Programm HDCSC wird zum Abgleich des Datenseparators benötigt. Dabei werden alle Sektoren der Platte gelesen.**

```
program hdformat;

const
{$i hdcdef.inc}    { Laufwerksdefinitionen }

    track length = 10416; { Laenge einer Spur in Bytes }
    CR           = ##0d;
    ESC          = ##1b;

type
    wrkstring = string[80];
    str10     = string[10];

var
    Gap3_Len  : integer;
    Sekt_Size : integer;
    Buffer    : array[1..1024] of Byte;
    maxtrack  : integer;
    maxsek    : integer;
    i,inter   : integer;
    ch        : char;

{$i hdcrout.inc} { Controller-Schnittstellen-Routinen }

{ Berechne die Gap3-Laenge aus Interleaving, Drehzahl-      }
{ variation und Sektorgroesse                               }

procedure Compute_Gap3_Len;
var k : integer;
begin
  if inter = 1 then k := 25 else k := 0;
  Gap3_Len := round((2 * Speedvar * Sektor_Laenge) / 100) + k;
  writeln;writeln;
  writeln(' Gap3-Laenge = ',Gap3_Len);
end;

{ Pruefe, ob das gewaehlte Format (mit all seinen Gaps)     }
{ auf die Spur passt                                        }

procedure Checkspace;
var k1,k2 : integer;
begin
  k1 := 43;   { all konstant bytes per Sector }
  k2 := Gap3_Len;
  k1 := k1 + Sektor_Laenge + k2;
  k1 := Sektoren_pro_Spur * k1;
  if k1 > Track_Length then
    abort('Fehler : gewaehltes Format ist zu lang fuer eine Spur !');
end;

{ Loesche den Buffer zu $E5 (Filler-Byte)                   }

procedure Clear_Buffer;
var i : integer;
begin
  for i:=1 to Sektor_Laenge do buffer[i] := $e5;
end;

{ Setze den Buffer zum Formatieren gemaess dem gewaehlten   }
{ Interleave                                                }

procedure Setup_Buffer;
var first,Sek_Num,i : integer;
begin
  writeln;
  repeat
    write(' Interleaving Faktor eingeben [1..',Sektoren_pro_Spur,'] : ');
    readln(inter);
  until ((inter > 0) and (inter <= Sektoren_pro_Spur));
  for i:=1 to (2*Sektor_Laenge)+1 do buffer[i] := 0;
  first := 0;
  Sek_Num := 0;
  for i:=0 to Sektoren_pro_Spur-1 do begin
    Buffer[2*Sek_Num+2] := lo(i);
    Sek_Num := Sek_Num + Inter;
    if Sek_Num >= Sektoren_pro_Spur then begin
      first := first + 1;
      Sek_Num := first;
    end;
  end;
  writeln;
  for i:=1 to (Sektoren Pro Spur) do begin
    write(buffer[(2*i)]:2,' ');
  end;
end;

{ Formatiere eine Spur (Abbruch nur mit ^S^C)               }

procedure Formatiere_Spur(Spur : integer);
var Zylinder,Kopf : integer;
    tmp           : integer;
begin
  tmp := Gap3_Len - 3;
  if tmp < 0 then tmp := 0;
  Zylinder  := Spur div Anzahl_Koepfe;
  Kopf      := Spur mod Anzahl_Koepfe;
  write(chr($0D),' Zylinder = ',Zylinder:4,' Kopf = ',(Kopf+1):1);
  Update_Task_Register(Zylinder,Kopf,tmp);
  Set_Sectorcount(Sektoren_pro_Spur);
  HDC_Format;
  Send_Data_to_HDC;
  Test_Status(true);
end;

{ Fuelle eine Spur mit $E5 und lese sie wieder aus          }

procedure Fuelle_Sektoren(Spur : integer);
var Zylinder,Kopf,Sektor : integer;
begin
  Zylinder  := Spur div Anzahl_Koepfe;
  Kopf      := Spur mod Anzahl_Koepfe;
  write(chr($0D),' Zylinder = ',Zylinder:4,' Kopf = ',(Kopf+1):1);
  for Sektor := 0 to Sektoren_pro_Spur-1 do begin
    Update_Task_Register(Zylinder,Kopf,Sektor);
    Set_Sectorcount(0);
    HDC_Write;
    Test_Status(false);
    HDC_Read;
    Test_Status(false);
  end;
end;

{ Initialisiere alle Parameter zum Formatieren              }

procedure Init_Format;
begin
  Setup_Buffer;
  Compute_Gap3_Len;
  checkspace;
  HDC_Init;
  maxtrack := Anzahl_Koepfe * Anzahl_Zylinder;
end;

{ no coment...                                              }

procedure DoFormat;
var spur : integer;
    kopf_u,kopf_o,spur_u,spur_o,sektor_u,sektor_o:integer;
    kopf_1,spur_1,sektor_1:integer;
  begin
  Init_Format;
  writeln; writeln(' Formatierbereichs-Eingabe');
  repeat
    begin write(' Kopf (1-',Anzahl_Koepfe:1,') : ');
          read(kopf_u); write('-'); read(kopf_o); writeln; end;
  until (kopf_u > 0) and (kopf_o <= Anzahl_Koepfe);
  repeat
    begin write(' Spur (0-',Anzahl_Zylinder:3,') : ');
          read(spur_u); write('-'); read(spur_o); writeln; end;
  until (spur_u >= 0) and (spur_o <= Anzahl_Zylinder);
  writeln;
  writeln(' Das Formatieren beginnt : ');
  for spur_1:=spur_u to spur_o do
     begin
     for kopf_1:=kopf_u-1 to kopf_o-1 do
        begin
        spur:=(Kopf_1 and (Anzahl_Koepfe-1)) or (spur_1 * Anzahl_Koepfe);
        Formatiere Spur(Spur);
        end;
     end;
  writeln;
  writeln(' Disk ist formatiert . ');
  writeln;
  writeln(' Die Sektoren werden jetzt mit 0E5H gefuellt : ');
  Clear_Buffer;
  for spur_1:=spur_o downto spur_u do
     begin
     for Kopf_1:=Kopf_o-1 downto Kopf_u-1 do
        begin
        spur:=(Kopf_1 and (Anzahl_Koepfe-1)) or (spur_1 * Anzahl_Koepfe);
        Fuelle_Sektoren(Spur);
        end;
     end;
  writeln;writeln;
  writeln('Formatierung beendet');
end;

{ ************ M A I N ****************************** }
begin
  clrscr;
  writeln('Hard-Disk-Formatter [V 1.0] (C) A.Zippel, A.Zinser');
  writeln;
  writeln('A C H T U N G :');
  writeln('mit dem Formatieren gehen alle Daten auf der Hard-Disk verloren !');
  writeln('Ist dies unerwuenscht,so kann jetzt mit <ESC> beendet werden.');
  writeln('Der Formatiervorgang selbst kann nur mit ^S^C abgebrochen werden!');
  writeln('Bitte Kaffee aufsetzen. (Das Formatieren dauert...)');
  writeln;
  writeln(' Mit <CR> wird das Formatieren gestartet.');
  writeln;
  write(' <ESC> oder <CR> jetzt eingeben : ');
  if keypressed then read(kbd,ch);
  read(kbd,ch);
  writeln;
  if ch = CR then begin
    DoFormat;
  end else begin
    writeln; writeln('Formatieren wurde abgebrochen !');
  end;
  HDC_Deselect;
end.
```

**Das Formatierprogramm ist wie die anderen Programme an die Systemparameter anzupassen, der Interleave-Faktor wird erst beim Programmstart erfragt.**

```
{$C-,U-,r-,a-}
program hdctest; {test des Winchestercontrollers }
const
{$i hdcdef.inc} { Definitionen fuer die Winchester }

type
    str10 = string[10];
    wrkstring = string[80];

var
    hdcok     : Boolean;
    ch        : char;
    Gap3_Len  : integer;
    Sekt_Size : integer;
    Buffer    : array[1..1024] of Byte;
    maxtrack  : integer;
    maxsek    : integer;

{$i hdcrout.inc}  { I/O-Routinen }

{ Gebe 1 Byte als Hex-Zahl aus }

procedure WriteHex(b : byte);
var b1 : byte;
  procedure WriteNibble(b : byte);
  begin
    b := b + $30;
    if (b > $39) then b:= b + 7;
    write(chr(b));
  end;
begin
  b1 := b shr 4;   WriteNibble(b1);
  b1 := b and $0F; WriteNibble(b1);
end;

{ Gebe b:integer als n-stellige Dualzahl aus }
```

```
procedure dispbit(b,n : integer);
var i,t : integer;
begin
  t := 1;
  for i:=1 to n do begin
    t := b shr (n-i);
    if (t and 1) = 1 then write('1') else write('0');
  end;
end;

{ Warte auf Tastatureingabe und verarbeite diese entsprechend }
{ W = Pause, G = Continue                                     }

procedure waitkey;
var ch1,ch2 : char;
begin
  if keypressed then begin
    read(kbd,ch1);
    if upcase(ch1) = 'W' then begin
      repeat
        read(kbd,ch2);
      until (upcase(ch2) = 'G');
    end;
  end;
end;

{ erster Speichertest. Schreibt den gesamten Speicher mit einem }
{ Pattern voll und vergleicht den gesamten Speicher auf diesen  }
{ Wert. Der Wert wird von 0 bis $FF wiederholt und laeuft       }
{ solange, bis eine Taste gedrueckt wird.                       }

function checkpatram(seklen : integer) : boolean;
var i,d : integer;
    s,ok,f : boolean;
    pat : byte;
    ch  : char;
begin
  write(' Testing Buffer-Ram with Pattern ... ');
  ok := true;
  f := true;
  pat := 0;
  repeat
    for i:=1 to seklen do port[hdcdata] := pat;
    s := true;
    for i:=1 to seklen do begin
      d := port[hdcdata];
      s := s and (d = Pat);
      if not s then ok := s;
    end;
    if not s then begin
      if f then begin writeln;f := false;end;
      write('Buffer Error : Pattern = ');WriteHex(pat);
      write(' = ');dispbit(pat,8);write(' -> Data = ');
      WriteHex(d);write(' = ');dispbit(d,8);writeln;
      waitkey;
    end;
    pat := pat + 1;
  until (pat>255) or (not s) or keypressed;
  if keypressed then read(kbd,ch);
  if ok then writeln('Buffer is OK') else writeln('Buffer is not OK');
  checkpatram := ok;
end;

{ Zweiter Speichertest. In jedes Byte des Speichers wird ein    }
{ anderer Wert geschrieben. Nach dem Schreiben wird jedes Byte  }
{ gelesen und auf Richtigkeit geprueft. Der Test ist nach       }
{ einem Durchlauf beendet.                                      }

function checkadrram(seklen : integer): boolean;
var i : integer;
    d : byte;
    ok,first : boolean;

begin
  writeln;
  write(' Testing Buffer-Ram with Adress-Data ... ');
  for i:=1 to seklen do port[hdcdata] := lo(i);
  ok := true;
  for i:=1 to seklen do begin
    d := port[hdcdata];
    ok := ok and (d = lo(i));
    if (d <> lo(i)) then begin
      if first then begin
        writeln;
        first := false;
      end;
      write(' Adress = ');writehex(hi(i));writehex(lo(i));
      write(' Data = ');writehex(lo(i));write(' -> ');writehex(lo(d));
      writeln;
    end;
  end;
  if ok then writeln('Buffer is OK') else writeln('Buffer is not OK');
  checkadrram := ok;
end;

{ Setze das SDH-Register. Es werden S = Sektorlaenge (verschluesselt) }
{ D = Drive-Select und H = Head-Nummer gesetzt                        }

procedure setsdh(drv,head,seklen : integer);
var i,s : integer;
begin
  i := (head and (Anzahl Koepfe-1));
  i := i or ((drv and $03) shl 3);
  case seklen of
    128 : s := $60;
    256 : s := $00;
    512 : s := $20;
    1024 : s := $40;
    else begin
      writeln(' Internal Error : Sektorlength incorrect');
    end;
  end; { of case }
  i := i or s or 8;
  port[hdcsdh] := i;
  if port[hdcsdh] <> i then begin
    writeln;
    write('Error : SDH-Register incorrect : ',i:3,' --> ');
    writehex(port[hdcsdh]);
    writeln;
    end
  else
    begin
    writeln;
    writeln(' SDH-Register ok ');
    writeln;
    end;
end;
```

```
{ Seek-Test. Es wird 10-mal der letzte Zylinder (Anzahl Zylinder)  }
{ auf der Hard-Disk angefahren und danach wieder ueber Restore die }
{ Spur 0. Der Test laesst sich durch beliebige Eingabe vorzeitig   }
{ abbrechen.                                                       }

function seektest(nt : integer): boolean;
var i,j,k : integer;
    ok : boolean;
    st : byte;
    ch : char;
begin
 writeln;
 write(' Seek-Test (10 RESTORE-SEEKs) ... ');
 k := 0;
 repeat
  k := k + 1;
  HDC_Wait;
  HDC_Home;
  HDC_Wait;
  st := port[hdcstat];
  if ((st and 1) <> 0) then begin
    ok := false;
    st := port[hdcerr];
    writeln;
    writeln('Restore failed, Error Code = ',st:2);
  end else begin
    port[hdccyl] := lo(nt);
    port[hdccyh] := hi(nt);
    port[hdccmd] := Seek CMD;
    HDC_Wait;
    st := port[hdcstat];
    if ((st and 1) <> 0) then begin
      st := port[hdcerr];
      writeln;
      writeln('Seek failed, Error Code = ',st:2);
      ok := false;
    end else begin
      write('+');
      st := port[hdcstat];
      if ((st and 1) <> 0) then begin
        write('Seek to Track 0 failed, Error Code = ');writehex(st);writeln;
        ok := false;
      end else ok := true;
    end;
  end;
 until (k>10) or keypressed;
 if keypressed then read(kbd,ch);
 if ok then writeln(' Seek-Test OK');
 seektest := ok;
end;

{ Disk-Read-Test. In diesem Test wird das ID-Feld der ueber tr     }
{ angesteuerten Spur gelesen. Der Test laesst sich mit beliebiger  }
{ Eingabe abbrechen.                                               }

function scanid(tr: integer):boolean;
var i,j : integer;
    ok  : boolean;
begin
  ok := true;
  writeln;
  write(' Disk-Read-Test ... ');
  port[hdccyl] := lo(tr);
  port[hdccyh] := hi(tr);
  port[hdcsek] := 1;
  port[hdccmd] := Seek CMD;
  j:=0;
  while (not keypressed) and (j=0) do begin
    HDC_Wait;
    port[hdccmd] := Scanid Fast CMD;
    i := port[hdcstat];
    if (i and 1) = 1 then begin
      HDC_WAIT;
      j := port[hdcerr];
      ok := false;
    end else begin
      j := 0;
    end;
  end;
  if ok then
     writeln(' Disk-Read-Test ok')
  else
     begin
     writeln(' Error: Disk-Read-Test');
     writeln;
     write(' Error = ');writehex(j);writeln;
     i := port[hdcstat];
     write(' Status = ');writehex(i);writeln;
     i := port[hdccyh]*256 + port[hdccyl];
     writeln(' Track = ',i);
     i := port[hdcsek];
     writeln(' Sektor = ',i);
     end;
  scanid := ok;
end;

{ ****************** M A I N ********************************* }

begin
  clrscr;
  writeln('Test Routinen fuer den Winchester Controller [V 0.1]');
  writeln;
  HDC Init;
  writeln('Alle Tests koennen mit einem Tastendruck beendet werden !');
  writeln;
  setsdh(0,0,Sektor Laenge);
  hdcok := checkpatram(2048);
  hdcok := hdcok and checkadrram(2048);
  hdcok := hdcok and seektest(Anzahl Zylinder);
  hdcok := hdcok and scanid(10);
  writeln;
  if hdcok then writeln('Controller ist OK und soweit funktionsbereit')
           else writeln('HDC-Karte fehlerhaft / nicht funktionsbereit');
  writeln;
  writeln('Ende der Test-Routinen');
  HDC deselect;
end.
```

**Das Testprogramm HDCTEST wird mehrere Male während der Inbetriebnahme gebraucht. Es testet das SDH-Register, prüft ausgiebig den Sektor-Puffer und veranstaltet einen Seek- und einen Read-Test.**

```
{ Gebe Fehlermeldung aus und beende den Programmlauf          }

procedure abort(s : wrkstring);
begin
  gotoxy(1,23);
  writeln(s);
  halt;
end;

{ Warte, bis das BUSY-Bit des Statusregisters inaktiv wird }

procedure HDC_Wait;
begin
  while ((port[hdcstat] and $80) <> 0) do;
end;

{ Warte, bis das DRQ-Bit des Statusregisters aktiv wird     }

procedure HDC_DRQ;
begin
  while ((port[hdcstat] and $8) = 0) do;
end;

{ Lese solange Daten vom Datenregister hdcdata, bis das     }
{ DRQ-Bit im Statusregister inaktiv wird                    }

procedure HDC_Clear_DRQ;
var bt : byte;
begin
  while ((port[hdcstat] and $8) <> 0) do
    begin bt := port[hdcdata] end;
end;

{ Teste das Statusregister auf Fehler (Bit 0 = 1) und       }
{ breche das Programm ab, falls abortflag = true            }
{ ansonsten erfolgt Fehlermeldung in der Form >nn<          }

procedure Test_Status(abortflg : boolean);
var error : integer;
    s     : str10;
begin
  if ((port[hdcstat] and $01) <> 0) then
    begin
    error := port[hdcerr];
    if abortflg then
      begin
      str(error:4,s);
      abort('Fehler : HDC meldet Fehler :'+s);
      end
    else
      begin
      write(' >',error,'< ');
      end;
    end;
end;

{ Restore Drive (Koepfe auf Zylinder 0)                     }

procedure HDC_Home;
begin
  port[hdccmd] := Restore_CMD;
  HDC_Wait;
end;

{ Setze Formatiercommando ab                                }

procedure HDC_Format;
begin
  HDC_Wait;
  port[hdccmd] := Format_CMD;
end;

{ Setze Lesekommando (ohne Retry) ab                        }

procedure HDC_Read_Noretry;
begin
  HDC_Wait;
  port[hdccmd] := Read_Fast_CMD;
end;

{ Setze das Task-File ab                                    }

procedure Update_Task_Register(Cylinder,Head,Sector : integer);
begin
  HDC_Wait;
  port[hdcsdh] := lo(Sekt_Size or Head or $08);
  port[hdccyl] := lo(Cylinder);
  port[hdccyh] := hi(Cylinder);
  port[hdcsek] := lo(Sector);
end;

{ Setze den Sektor-Zaehler                                  }

procedure Set_SectorCount(cnt : integer);
begin
  port[hdcscnt] := lo(cnt);
end;

{ Lese den Sektor-Buffer aus (siehe auch Text)              }

procedure Get_Data_From_HDC;
var i : integer;
begin
  HDC_Wait;
  HDC_DRQ;
  for i:=1 to Sektor_Laenge do buffer[i] := port[hdcdata];
end;

{ Schreibe einen Sektor in den Sektor-Buffer (siehe Text)   }

procedure Send_Data_to_HDC;
var i : integer;
begin
  HDC_DRQ;
  for i:=1 to Sektor_Laenge do port[hdcdata] := buffer[i];
  HDC_Wait;
end;

{ Lese einen Sektor der HD                                  }

procedure HDC_Read;
begin
  HDC_Wait;
  port[hdccmd] := Read_CMD;
  Get_Data_From_HDC;
end;
```

```
{ Schreibe einen Sektor auf die HD                          }

procedure HDC_Write;
begin
  HDC_Wait;
  port[hdccmd] := Write_CMD;
  Send_Data_to_HDC;
end;

{ Initialisiere die Harddisk. Es werden die Variable        }
{ Sektor_Size und das Write_Precomp-Register gesetzt.       }
{ Der dummy-Zugriff dient nur zu Testzwecken.               }

procedure HDC_Init;
var dummy : integer;
begin
  dummy := port[hdcstat];
  HDC_Wait;
  case Sektor_Laenge of
     128 : Sekt_Size := $60;
     256 : Sekt_Size := $00;
     512 : Sekt_Size := $20;
    1024 : Sekt_Size := $40;
    else abort('Interner Fehler : falsche Sektorlaenge !');
  end;
  port[hdcwpc] := PRECOMP_Spur div 4;
  Update_Task_Register(0,0,0);
  HDC_Home;
end;

{ Lese einen Sektor (ohne Retry)                            }
{ abssec ist die absolute Sektornummer auf der Harddisk     }
{ (phys.Kopf * phys.Zylinder * phys.Sektor)                 }

procedure readsec(abssec : integer);
var i : integer;
    Kopf,Zylinder,Sektor : integer;
begin
  i        := abssec div Sektoren_pro_Spur;
  Zylinder := i div Anzahl_Koepfe;
  Kopf     := i mod Anzahl_Koepfe;
  Sektor   := abssec mod Sektoren_Pro_Spur;
  Update_Task_Register(Zylinder,Kopf,Sektor);
  HDC_Read_Noretry;
  HDC_Wait;
  Get_Data_From_HDC;
  Test_Status(false);
end;

{ Setze das Drive-Select-Signal zurueck                     }

procedure HDC_Deselect;
begin
  port[hdcsdh] := $00;
end;
```

**Die Include-Datei HDCROUT.INC enthält die meistgebrauchten I/O-Prozeduren für alle drei Testprogramme.**

```
procedure Get_Data_From_HDC;
var i : integer;
begin
  HDC_Wait;
  HDC_DRQ;
  for i:=0 to (Sektor_Laenge div 128) do
      inline( $2A / buffer / $01 / hdcdata / $B0 / $ED / $B2 );
end;

procedure Send_Data_to_HDC;
var i : integer;
begin
  HDC_DRQ;
  for i:=0 to (Sektor_Laenge div 128) do
      inline($2A / buffer / $01 / hdcdata / $B0 / $ED / $B3 );
  HDC_Wait;
end;
```

**Zur Vermeidung einer Coffein-Vergiftung empfiehlt es sich, die beiden Pascal-Schleifen in HDCROUT.INC durch INLINEs zu ersetzen. Hier die Version für Z80-Rechner.**

```
Anzahl_Koepfe     = 4;
Anzahl_Zylinder   = 611;
Sektoren_pro_Spur = 9;
Sektor_Laenge     = 1024;
Speedvar          = 1;
PRECOMP_Spur      = 128;

Restore_CMD       = $10;
Seek_CMD          = $70;
Read_CMD          = $20;
Read_Fast_Cmd     = $21;
Write_CMD         = $30;
Scanid_CMD        = $40;
```

```
Scanid_Fast_CMD   = $41;
Format_CMD        = $50;

hdcdata           = $90;
hdcerr            = $91;
hdcwpc            = $91;
hdcscnt           = $92;
hdcsek            = $93;
hdccyl            = $94;
hdccyh            = $95;
hdcsdh            = $96;
hdcstat           = $97;
hdccmd            = $97;
```

**In HDCDEF.INC sind alle laufwerks- und rechnerspezifischen Definitionen versammelt. Wie diese im Einzelfall zu wählen sind, ist gesondert in Bild 1 dargestellt.**