Harry Siebert

AIM-65 treibt Drucker

Hier wird beschrieben, wie man einen Normalpapier-Drucker mit größerer Schriftbreite als der im AIM-65 bzw. PC-100 bereits eingebaute Thermodrucker an diese beiden Computer anschließt. Um den User-Port (VIA 6522) nicht damit zu belegen, dient ein PIA-Baustein mit eigener Adressendecodierung als Parallel-Schnittstelle.

Als dem Verfasser nach einem halben Jahr das Aufkleben der Thermodrucker-protokolle zu dumm wurde, beschloß er, seinen PC-100 um einen „echten" Drukker zu erweitern.
Die Wahl fiel schließlich auf den Microline 80 (Vertrieb: F. Percom, München), einen Matrixdrucker der 2000-DM-Klasse – übrigens nach Herstellerangabe Centronics-schnittstellenkompatibel.
Da der Anschluß solcher Luxusdrucker am PC-100 nicht vorgesehen ist, hier zunächst die Anschlußhardware (*Bild 1*).
Die 8-Bit-Breite der Datenschnittstelle läßt den PIA-Baustein 6520 als den Chip der Wahl erscheinen, zumal ja die Übertragung per Handshake erfolgen soll. Leider stellt sich aber bald heraus, daß sich die als statische Signalquelle angelegte Busy-Leitung jeder einfachen Verarbeitung mit einer flankengesteuerten CA-Leitung widersetzt, so daß dafür eine Datenleitung (PA7) abgezweigt werden muß. Wo nun aber das achte Datenbit – das zwar „nur" für Nicht-ASCII-Grafiksymbole verwendet wird, auf das man aber dennoch ungern verzichtet – hernehmen? Die B-Hälfte des 6520 deswegen anzureißen, wäre doch zu verschwenderisch, deshalb wurde auf den Trick mit der Doppeladresse zurückgegriffen: Der PIA 6520 wird nur über zwei Adreß- (A0, A1) und eine CS-Leitung angesteuert, das CS-Signal jedoch aus A4...A15 decodiert (andere PIAs belegen 16 Adressen), was dazu führt, daß ein und dasselbe Register (z. B. DRA) durch zwei Adressen (z. B. A020 und A024) angesprochen wird. Das Problem läßt sich also auf eineinhalb zusätzliche TTL-Bausteine reduzieren: A2 und A3 liefern über einen (halben) 2-Bit-Decoder (74139) die Preset- und Reset-Signale für ein Flipflop 7470, dessen Ausgang Q als Datenbit 7 agiert. Ansonsten ist die Schaltung trivial und läßt sich problemlos in Fädeltechnik auf einer Lochrasterplatine bewerkstelligen.
Dank der äußerst sinnigen Adressendecodierung wird in AIM-65 und PC-100 ein ganzes KByte Speicher (von A000 bis A3FF) für die ganze 16 Bytes „lange" User-VIA (Baustein Z1, 6522) verbraten. Dieser Fehler läßt sich aber leicht beheben, indem man den I/O-Raum – wie in *Bild 2* beispielsweise – selbst decodiert. Damit gewinnt man mühelos Platz für 15 weitere I/O-Bausteine. Die User-VIA muß dann allerdings auch mit ausgelagert werden, was aber dank gesockelter Montage problemlos ist. Sie wird dann mit $\overline{CS}$ an den Ausgang Q_0 des 74154 angeschlossen. Dabei bietet sich auch gleich die Gelegenheit, eine anständige Applikationssteckbuchse (Fernschreiber-Buchse 25polig o. ä.) anzubringen. Unser Druckerinterface ist dann mittels Q_2 anzusteuern.
Nun ist aber die schönste Hardware nur der halbe Weg. Mindestens genauso wichtig ist eine effiziente Software (*Bild 3*). Ein Zugang zur Druckroutine sollte auf vier Ebenen möglich sein:

1. Im Kommandomodus (Monitor) zur Voreinstellung von Zeilenlänge, Zeichendichte u. a.
2. Von Basic aus über die USR-Funktion, um irgendwelche Texte, die man mittels POKE irgendwo im Speicher komponiert hat (das ist der Graphikzeichen wegen ohnehin nötig; die Verwendung des DILINK-Vektors nützt hier nicht viel), direkt und auf einen Rutsch auszuwerfen.

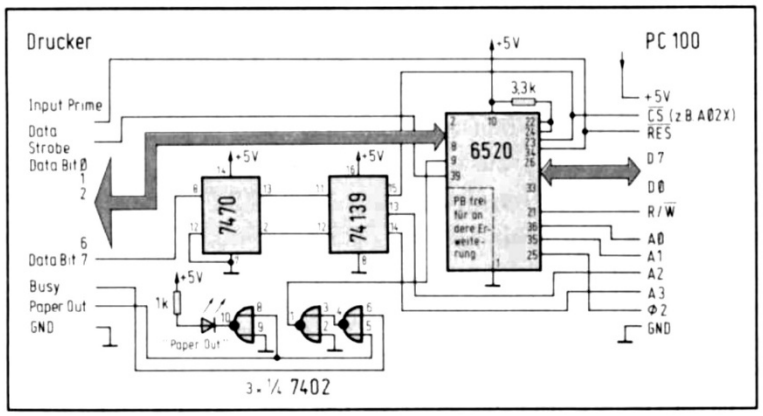

Bild 1. Statt den User-Port für den Drucker zu verschwenden, ist es sinnvoll, einen zusätzlichen PIA-Baustein als Schnittstelle vorzusehen

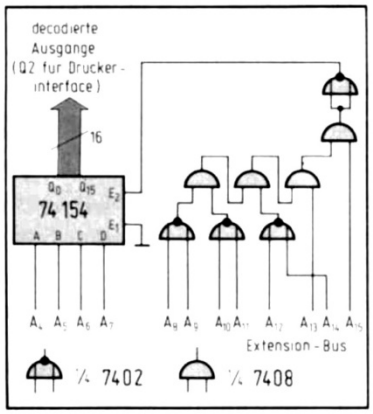

Bild 2. So kann man den I/O-Adressenraum von AIM-65 und PC-100 „sauber" decodieren

3. Im Editor über die U-Option beim List-Kommando.
4. Und schließlich von anderen Maschinenprogrammen aus als handliches Unterprogramm.

Diese Anwendungsvielfalt erfordert einen geschickt modularisierten Aufbau: Im Kern eine Routine PRINT, die ein im Akku übergebenes Byte direkt an den Drucker weitergibt (für Punkt 4); als Überbau hierfür ein PRINC getaufter Einstieg für den UOUT-Vektor, wo auch gleich zur Initialisierungsroutine PINIT verzweigt wird, wenn das Carry-Bit=0 den Erstaufruf markiert (Punkt 3). PRINT und PINIT werden schließlich von PLIST verwendet, um – z. B. von Basic aus – Listen auszudrucken, deren Startadresse in TOP abgelegt ist und die mit NUL (hex 00) beendet sein müssen.

Last not least gibt's dann noch ein Dialogmodul, das es ermöglicht, einzelne Bytes, die im Hexcode erfragt werden, direkt an ein Output-Device (hier LP) abzuschicken und dabei gleich den UOUT-Vektor auf den aktuellen Ausgabekanal einzurichten; ein Anknüpfpunkt für weitere Geräte wie Lochstreifenstanzer, Akustikkoppler, zweiter Drucker usw. ist dabei eingearbeitet.

```
0000          ;DIALOG ZUR DEVICE-DEFINITION
0000                *=$200
0200 TOP      0000  .WOR 0
0202                *=$0C00
0C00 T1       4445  .BYT 'DEVICE='    ;DIALOGTEXTE
0C07          464F  .BYT 'FORM?'
0C0C          4845  .BYT 'HEXCODE='
0C14
0C14          ;ROUTINE ZUR DIALOGTEXTAUSGABE
0C14 LOUT     2044EB JSR $EB44    ;CLEAR DISP
0C17 L10      B9000C LDA T1,Y     ;READ CHAR
0C1A          207AE9 JSR $E97A    ;OUT DISP
0C1D          C8     INY
0C1E          CA     DEX
0C1F          D0F6   BNE L10
0C21          60     RTS
0C22          ;EINSTIEG FUER F3-TASTE
0C22 F3       A207   LDX #7
0C24          A000   LDY #0
0C26          20140C JSR LOUT
0C29          2073E9 JSR $E973    ;DEVICE-CODE
0C2C          2073E9 JSR $E973    ;2-STELLIG
0C2F          C950   CMP #'P ;"LP"?
0C31          D02E   BNE ON1
0C33          A205   LDX #5
0C35          A007   LDY #7
0C37          20140C JSR LOUT     ;"FORMAT?"
0C3A          2073E9 JSR $E973
0C3D          C959   CMP #'Y ;FORM ?
0C3F          D012   BNE EX1      ;NEIN, INIT
0C41 FORM     A208   LDX #8       ;JA,DIALOG
0C43          A00C   LDY #12
0C45          20140C JSR LOUT     ;"HEXCODE"
0C48          205DEA JSR $EA5D    ;2CHAR=1BYTE
0C4B          B013   BCS EX2      ;FEHLER
0C4D          207D0C JSR PRINT    ;ANDRUCK
0C50          4C410C JMP FORM
0C53 EX1      A962   LDA #<PRINC
0C55          8D0A01 STA $10A
0C58          A90C   LDA #>PRINC
0C5A          8D0B01 STA $10B     ;UOUT-VEKTOR
0C5D          206D0C JSR PINIT
0C60 EX2      60     RTS
0C61 ON1      00     BRK          ;EXTENSION
0C62          ;PRINT-ROUTINE (VIA "U")
0C62 PRINC    9009   BCC PINIT    ;ERSTAUFRUF
0C64          68     PLA
0C65          C90D   CMP #$0D     ;LF FEHLT !
0C67          D014   BNE PRINT
0C69          A90A   LDA #$0A
0C6B          F010   BEQ PRINT
0C6D          ;INITIALISIERUNG DER SCHNITTSTELLE
0C6D PINIT    A938   LDA #$38
0C6F          8D21A0 STA $A021
0C72          A97F   LDA #$7F
0C74          8D20A0 STA $A020
0C77          A93C   LDA #$3C
0C79          8D21A0 STA $A021
0C7C          60     RTS
0C7D
0C7D          ;EIGENTLICHE ROUTINE
0C7D PRINT    AA     TAX          ;CHAR IN ACCU
0C7E          297F   AND #$7F
0C80          48     PHA
0C81          8A     TXA
0C82 L1       2C20A0 BIT $A020    ;DELAY WEIL
0C85          30FB   BMI L1       ;BUSY
0C87          A234   LDX #$34
0C89          2980   AND #$80
0C8B          D012   BNE BILD     ;GRAFIK?
0C8D          68     PLA          ;NEIN
0C8E          8D20A0 STA $A020
0C91          8E21A0 STX $A021
0C94          A260   LDX #$60     ;)STROBE(
0C96 L2       CA     DEX
0C97          D0FD   BNE L2
0C99          A23C   LDX #$3C
0C9B          8E21A0 STX $A021
0C9E          60     RTS
0C9F BILD     68     PLA          ;GRAFIK !
0CA0          8D24A0 STA $A024
0CA3          8E25A0 STX $A025
0CA6          A260   LDX #$60
0CA8 L3       CA     DEX
0CA9          D0FD   BNE L3
0CAB          A23C   LDX #$3C
0CAD          8E25A0 STX $A025
0CB0          60     RTS
0CB1          ;DRUCKROUTINE FUR BASIC U.AE.
0CB1 PLIST    AD0002 LDA TOP      ;START
0CB4          8DC10C STA TL+1     ;IN TOP
0CB7          AD0102 LDA TOP+1
0CBA          8DC20C STA TL+2
0CBD          206D0C JSR PINIT
0CC0 TL       ADFFFF LDA $FFFF    ;DUMMY
0CC3          C900   CMP #0       ;(WIRD VOM
0CC5          F00D   BEQ OFF      ;PROGRAMM
0CC7          207D0C JSR PRINT    ;GEAENDERT)
0CCA          EEC10C INC TL+1
0CCD          D0F1   BNE TL
0CCF          EEC20C INC TL+2
0CD2          D0EC   BNE TL
0CD4 OFF      60     RTS
0CD5                 .END
```

Bild 3. Das Assemblerprogramm zur Druckersteuerung. Der Assembler hat hier allerdings bei den Strings ab 0C00 jeweils nur die beiden ersten Hex-Bytes als Objektcode ausgedruckt