Anmerkung: Da die Unterprogramme mit Übergabevariablen aus dem Hauptprogramm (zum Beispiel wt$ oder tx$) versorgt werden und die Arbeitsvariablen der Unterprogramme, wie zum Beispiel tl (Textlänge), vk (Vorkommastellen), nk (Nachkommastellen) oder u$ (Vor- und Nachkommastellen bei Dezimalpunkt- und Nullenausgabe), im Hauptprogramm (siehe zum Beispiel Zeile 1770) beliebig definiert werden können, lassen sich diese als Bausteine für andere Programme nutzen.

Änderungen bei Verwendung einer Typenradschreibmaschine

Brother CE-60/CE-70,
Privileg P 6000,
Schweiz: Migro-Office,
auch: CE-50 m. IF-An.,
ebenso: Schönschreibdrucker Brother EM-80/100/200

Bei Verwendung eines Schönschreibdruckers beziehungsweise einer Typenradmaschine von Brother zusammen mit dem WDT-Interface (Wicher Digital Technik, Flörsbachtal, baugleich mit Quelle und Neckermann) sind folgende Änderungen im Listing notwendig:

```
1310 input" Anzahl der Kopien : ";ak:print
1311 print:print" Bitte warten Sie !"
1312 bl=30:ifa=2anda$="j"thenbl=18
1313 ifa=2anda$="n"thenbl=20
1314 ifa=2thenax=2:goto1319
1315 ifza=1thenht=3
1316 ifza=2thenht=2
1317 ifza=3thenht=1
1318 ax=0:ifa$="j"thenax=3:goto1319
1319 z=bl-ht-ax-(v)
1320 open1,4,7
1330 s=0:print#1,f$:print#1,fo$:print#1,fb$
1480 print#1,chr$(30);:print#1,chr$(6);
1500 print#1:print#1:print#1:print#1,ff$
1510 print#1,chr$(25);:print#1,chr$(7);
1530 print#1:print#1:print#1,ar$
ready.
1710 print#1:print#1:print#1,x1$:print#1
1720 print#1,x2$;nr$:print#1,x3$;d$
ready.
2350 forq=1toz:print#1:nextq
ready.
1480 print#1,chr$(27);chr$(31);chr$(9);
:print#1,chr$(27);chr$(69);
ready.
1510 print#1,chr$(27);chr$(31);chr$(13);
:print#1,chr$(27);chr$(82);
ready.
```

Automatischer Seitenvorschub, komprimierte Schrift und automatische Unterstreichung

```
45000 forl=1tolen(wt$)
45010 vu$=mid$(wt$,1,1)
45020 ifvu$=chr$(219)thenvu$=chr$(190)
45030 ifvu$=chr$(64)thenvu$=chr$(189)
45040 ifvu$=chr$(186)thenvu$=chr$(221)
45050 ifvu$=chr$(91)thenvu$=chr$(188)
45060 ifvu$=chr$(60)thenvu$=chr$(220)
45070 ifvu$=chr$(93)thenvu$=chr$(187)
45080 ifvu$=chr$(62)thenvu$=chr$(219)
45090 ww$=ww$+vu$
45100 nextl:wt$=ww$:ww$="":return
(45110 entfällt)
```

Zusatzänderung bei Verwendung des Original-Interface Brother IF-50

Kopplung über den User-Port

In der letzten Ausgabe brachten wir zwei VC 20 dazu, gegeneinander zu spielen. Nun lassen wir zwei C 64, oder einen VC 20 und einen C 64, Daten ohne jegliches Interface über den User-Port austauschen.

Seit einigen Monaten betreibe ich zwei gekoppelte C 64, die sich zu meiner vollen Zufriedenheit die Befehle eines fünfstimmigen Synthesizer-Programms teilen. Während es für meine speziellen Anforderungen bereits genügt, ein einzelnes Byte rasch zu transportieren, können die vorhandenen Routinen ohne viel Mühe so erweitert werden, daß man
a) Basicprogramme
b) Variablen,
c) Arrays und
d) Maschinenprogramme überträgt und gegebenenfalls ausführt.

In diesem Artikel erläutere ich Ihnen die Grundroutinen anhand der Basicprogrammübertragung.

Neben der eingebauten seriellen Schnittstelle hat der C 64 wie alle Computer von Commodore einen programmierbaren User-Port, der eine Verbindung nach »draußen« darstellt und entsprechend genutzt werden kann. Die Adresse des User-Ports liegt beim C 64 auf 56577 ($dd01). Eine Abfrage mit PEEK (56577) zeigt den Wert 255. Ein POKE-Befehl ändert daran nichts, weil das Betriebssystem beim Initialisieren den User-Port als Eingang definiert und sich unbeschaltete Eingänge als logisch »1« verhalten. Bevor man den User-Port als Ausgang definiert, müssen einige Sicherheitsvorkehrungen getroffen werden, denn der User-Port des einen Computers wird Draht für Draht mit dem des anderen verbunden. Dabei darf immer nur ein Computer den User-Port als Ausgang betreiben, sonst kommt es zur Zerstörung der Port-Bausteine. Die Adresse des Richtungsregisters ist 56579 ($dd03). In diesem Register definiert ein gesetztes Bit das entsprechende User-Bit als Ausgang. Dadurch wird ein gemischter Betrieb möglich, der in meinem Programm jedoch nicht vorkommt.

Als Voraussetzung muß eine physikalische Verbindung in Form von Drähten und zwei User-Port-Steckern geschaffen werden.

Es sind insgesamt 11 Drähte, welche die beiden Computer verbinden. Ich habe die Verbindung mit einer 4 m langen, sorgfältig verlöteten Flachleitung realisiert und bisher keine Störung bemerkt.

Auf Seite 143 des C 64 Handbuchs ist die Kontaktbelegung des User-Port-Steckers aufgeführt.

Die Daten-Bit-Kontakte PB0 bis PB7 liegen genau in der Mitte des Steckers und werden »Bit für Bit« verbunden, also »C« an »C«, »D« an »D« und so weiter.

Auch die Masse-Leitungen »A« werden miteinander verbunden.

Die beiden übrigen Kontakte »B« und »M« werden gekreuzt, so daß »B« mit »M« verbunden ist und umgekehrt.

Das Programm ist ausführlich erklärt. Eine Besonderheit ist die Speicherbereichswahl. Jeder Autor glaubt, der $c-Block sei ganz frei und warte auf genau sein Programm.

Um hier eine bequeme Übernahmemöglichkeit in eigene Routinen zu bieten, lädt sich der Maschinenteil des Programms an das obere verfügbare RAM-Ende, schützt das Segment vor den Strings und löscht sich letztlich selbst. Die SYS-Adressen werden auf dem Bildschirm angezeigt. Dann können im Direkt-Modus Basicprogramme hin- und hergeschoben werden, oder aus Maschinenroutinen einzelne Bytes transportiert werden. Die Basicprogramm-Übertragung hat die angenehme Eigenschaft, das transportierte Programm an das bereits im Speicher befindliche Programm anzuhängen, es handelt sich also um eine Art Merge-Funktion. Der »NEW«-Befehl schafft wieder leere Verhältnisse, wenn's mal zu voll geworden ist.

Programmerklärung

Die Punkte »..« im Basicprogramm sind Platzhalter für die $-Seite, in die das Maschinenprogramm geladen wurde. Im folgenden gehe ich von Seite $ 7f aus. $7f00 = sys 32512 überträgt ein Basicprogramm vom Sender zum Empfänger, wo es an das im Speicher befindliche Programm angehängt wird. Zunächst wird der User-Port des Senders als Ausgang definiert. Dann wird ein Zeiger in $58 eingerichtet, der von Basic-Start ($2b) bis Basic-Ende ($2d) alle Bytes der Byte-Sende-Routine mit »jsr 7f5e« übergibt. Nach Übertragung des letzten Bytes wird der User-Port wieder als Eingang geschaltet (= Sicherheitsfall).

Mit sys 32551 ($7f27) wird ein Basicprogramm empfangen. Durch Subtraktion von 2 vom Basic-Zeiger wird die Koppeladresse gewonnen, ab der die empfangenen Bytes gespeichert werden sollen. Die Byte-Empfangs-Routine steht ab $7f7c. Der Empfänger erkennt das Basicprogramm-Ende an drei folgenden Nullen ($#00), wonach er den Empfang abbricht, durch einen CLR-Befehl ($a660) die Basic-Zeiger neu setzt, durch $a533 die Programmzeilen neu bindet und schließlich in die Basicwarteschleife nach $e385 springt.

$7f5e ist die Byte-Sende-Routine. Der Akkumulator des Prozessors wird in das Datenregister des User-Ports geschrieben, anschließend wird das dritte Bit in Register $dd00 auf »0« gesetzt. Nun bleibt das fünfte Bit in Register $dd0d des anderen Computers solange »1«, bis es gelesen wird. Durch den Lesevorgang wird es gleichzeitig gelöscht. Diesen Mechanismus kann man einfach zur Übertragungskontrolle nutzen. Das dritte Bit von $dd00 ist die VALID-Flagge. Wenn sie gesetzt ist, sind die Daten gültig. Damit der Sender weiß, wann er das Byte übertragen ansehen kann, sendet der Empfänger ein QUITTUNGs-Signal. Dieses kommt im fünften Bit des Registers $dd0d beim Sender an. Auch hier wird durch das Lesen gleichzeitig gelöscht. Braucht der Empfänger zu lange, um das Byte anzunehmen, zum Beispiel dann, wenn er gar nicht eingeschaltet ist, bricht der Sender nach zirka 1,8 Millisekunden Wartezeit den Sendevorgang ab und schaltet den USER-Port wieder auf Eingangs-Modus. Dadurch soll verhindert werden, daß die beiden Computer gleichzeitig senden, was der Hardware schaden würde. Nach erfolgter Übertragung wird das dritte Bit in $dd00 wieder auf »1« gesetzt, dabei ändert sich in $dd0d des anderen Computers nichts, es ist halt notwendig, um es wieder auf »0« setzen zu können.

$7f7c ist die Byte-Empfangs-Routine. Hier wartet der Empfänger solange in der VALID-Schleife, bis die Daten gültig sind. Wird nichts gesendet, so wartet der Empfänger fortwährend und merkt nichts. Zum Abbruch muß man dann die RUN/STOP- und die RESTORE-Taste gleichzeitig drücken. Sind die Daten gültig, so werden sie in den Akkumulator des Prozessors geladen. Der Empfänger quittiert durch einmaliges Nullen des dritten Bits von $dd00.

$7f8f definiert den User-Port als Eingang. $7f92 definiert den User-Port als Ausgang. Der Trick mit »BIT mmnn« fand auch hier Anwendung. Man spart immerhin ein Byte.

$7f9d initialisiert den User-Port. Zwar setzt das Betriebssystem den User-Port als Eingang, nimmt aber nicht die Flag in $dd0d weg, wodurch es direkt nach dem Einschalten zu einer VALID-Meldung kommt, die eine richtige Übertragung verhindert. Die VALID-Schleife in $7f9d liest so lange VALID-Flags, bis keine mehr kommen, und springt dann zur Eingangs-Richtungs-Routine.

Praktische Ausführung

— Verbindungsleitung löten (1:1).
— Programm erfassen, SAVEn und RUNnen.

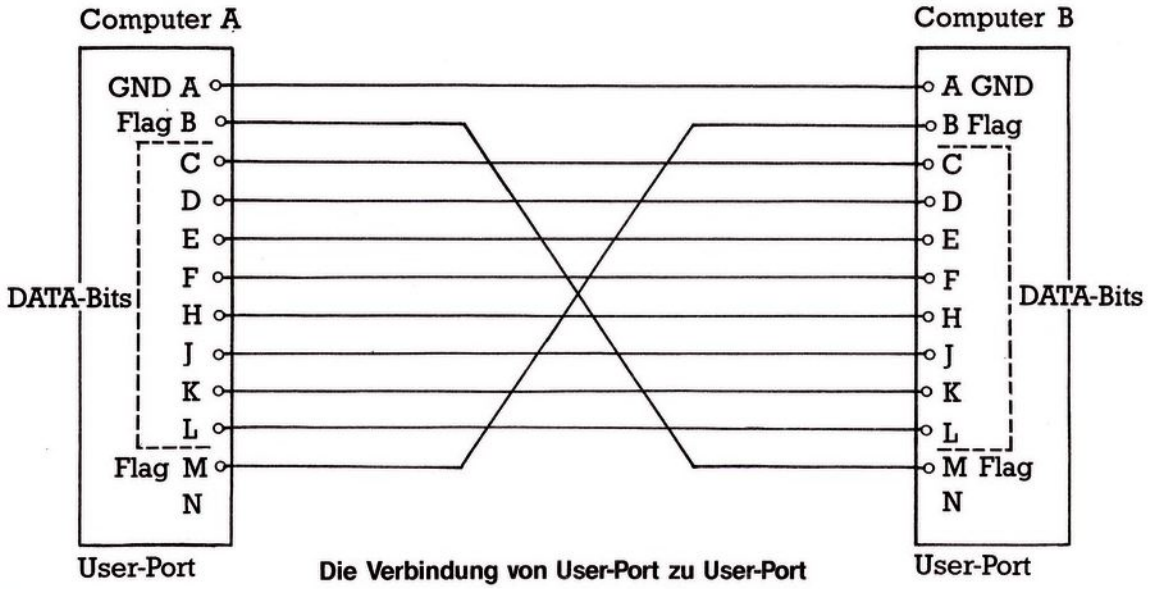
Soll zum Beispiel von Computer A ein Programm zu Computer B übertragen werden, so muß zuerst Computer B mit »sys 32551« vorbereitet werden, dann wird Computer A mit »sys 32512« zum Senden veranlaßt. Im umgekehrten Fall beschwert sich Computer A mit der Meldung »device not present error«, welche aus dem Basicinterpreter für diesen Zweck entliehen wurde. Die Übertragungsrate liegt bei etwa 11 000 Bytes pro Sekunde.

(Johannes Mockenhaupt/rg)

```
10 rem " Johannes Mockenhaupt
20 rem " Hochstadenstr. 28
30 rem "
40 rem " D-5000 Köln 1
                        ,den 29.März 1984
50 :
60 rem " C 64 / VC 20 / Koppelung
70 rem " über den programmierbaren USER
Port.
80 :
100 bytes=165:                  rem " Anz
ahl Bytes Maschinen-PRG
110 hs=peek(56)-1:              rem " hoe
chste aktuelle RAM-Seite minus 1.
120 if peek(55)>0 then hs=hs-1:rem " kei
n 'glattes' RAM-Ende: Sicherheits-Seite
130 :
140 print "■sys"hs*256":BASIC PRG senden
.
```

Listing zur Übertragung von Basicprogrammen zwischen zwei C 64.

```
150 print "■sys"hs*256+39":BASIC PRG emp
fangen.
160 print "■sys"hs*256+94":1 BYTE senden
170 print "■sys"hs*256+124":1 BYTE empfa
ngen
180 print "■sys"hs*256+143":USER-Port au
f Eingang
190 print "■sys"hs*256+146":USER-Port au
f Ausgang
200 print "■sys"hs*256+157":QUICK-Port I
nitialisierung
290 :
300  for i=hs*256 to i+bytes-1
310   read a
320   b=a
330    if a<0 then a=a+hs+1
340   poke i,a
350   s=s+b
360  next i
370 s=s-20510
380 if s<>0 then print "■■Data-Fehler■:"
s"betraegt die Differenz.":end
390 :
400 print "■■QuickPort■ ist jetzt initia
lisiert.
410 poke 56,hs:     rem " neue RAM-top Ad
resse
420 sys hs*256+157:rem " Initialisierung
430 new:            rem " Setzen der BASI
C-Zeiger auf neuen Bereich.
440 :
6000 rem " Transport-Routinen      BASIC-
PRG  Senden:     $..00
6010 :
6100 data  32,146, -1:rem " ..00  20 92
..  jsr ..92     USER-Port auf Ausgang
6110 data 165, 43:     rem " ..03  a5 2b
    lda 2b       L-Byte Basic-Start
6120 data 164, 44:     rem " ..05  a4 2c
    ldy 2c       H-Byte Basic-Start
6130 data 133, 88:     rem " ..07  85 58
    sta 58        merken
6140 data 132, 89:     rem " ..09  84 59
    sty 59        merken
6200 data 160,  0:     rem " ..0b  a0 00
    ldy#00       Y-Register vorbereiten.
6210 data 177, 88:     rem " ..0d  b1 58
    lda (58).y   1 Byte aus PRG holen
6220 data  32, 94, -1:rem " ..0f  20 5e
..  jsr .. 5e    und senden.
6230 data 230, 88:     rem " ..12  e6 58
    inc 58       Zeiger erhoehen L-Byte
6240 data 208,  2:     rem " ..14  d0 02
    bne ..18     ohne Uebertrag
6250 data 230, 89:     rem " ..16  e6 59
    inc 59       Zeiger erhoehen H-Byte
6300 data 165, 89:     rem " ..18  a5 59
    lda 59       Zeiger      H-Byte
6310 data 197, 46:     rem " ..1a  c5 2e
    cmp 2e       BASIC-Ende H-Byte
6320 data 144,237:     rem " ..1c  90 ed
    bcc ..0b     noch nicht erreicht.
6330 data 165, 88:     rem " ..1e  a5 58
    lda 58       Zeiger      L-Byte
6340 data 197, 45:     rem " ..20  c5 2d
    cmp 2d       BASIC-Ende L-Byte
6350 data 144,231:     rem " ..22  90 e7
    bcc ..0b     noch nicht erreicht.
6360 data  76,143, -1:rem " ..24  4c 8f
..  jmp ..8f     USER-Port auf Eingang
6370 :
7000 rem " Transport-Routinen      BASIC-
PRG  Empfangen: $..27
7010 :
7100 data  56:         rem " ..27  38
    sec          Start-Adresse berechnen
7110 data 165, 45:     rem " ..28  a5 2d
    lda 2d        L-Byte Basic-Ende
7120 data 233,  2:     rem " ..2a  e9 02
    sbc#02        $02 substrahieren
```



Die Verbindung von User-Port zu User-Port

Listing zur Übertragung von Basicprogrammen zwischen zwei C 64 (Schluß)

```
7130 data 133, 45:     rem " ..2c  85 2d
    sta 2d          und speichern.
7140 data 165, 46:     rem " ..2e  a5 2e
    lda 2e          H-Byte Basic-Ende
7150 data 233,  0:     rem " ..30  e9 00
    sbc#00          Carry-Flag subtrahieren
7160 data 133, 46:     rem " ..32  85 2e
    sta 2e          und speichern.
7200 data 160,  0:     rem " ..34  a0 00
    ldy#00          Y-Register vorbereiten.
7210 data  32,124, -1:rem " ..36  20 7c
..  jsr ..7c        1 Byte empfangen
7220 data 145, 45:     rem " ..39  91 2d
    sta (2d).y      und speichern.
7230 data 230, 45:     rem " ..3b  e6 2d
    inc 2d          BASIC-PRG Ende erhoehen
7240 data 208,  2:     rem " ..3d  d0 02
    bne ..41        ohne Uebertrag
7250 data 230, 46:     rem " ..3f  e6 2e
    inc 2e          H-Byte erhoehen
7300 data 168:         rem " ..41  a8
    tay             Ende-Bedingung pruefen
7310 data 208,  8:     rem " ..42  d0 08
    bne ..4c        kein Ende
7320 data 165,252:     rem " ..44  a5 fc
    lda fc          2.Byte=0 ?
7330 data 208,  4:     rem " ..46  d0 04
    bne ..4c        nein, kein Ende
7340 data 165,253:     rem " ..48  a5 fd
    lda fd          3.Byte=0 ?
7350 data 240,  9:     rem " ..4a  f0 09
    beq ..55        ja, dann Ende
7360 data 165,252:     rem " ..4c  a5 fc
    lda fc          vorletztes Byte kellern
7370 data 133,253:     rem " ..4e  85 fd
    sta fd
7380 data 132,252:     rem " ..50  84 fc
    sty fc          letztes    Byte kellern
7390 data  24:         rem " ..52  18
    clc             Sprungbedingung
7400 data 144,223:     rem " ..53  90 d9
    bcc ..34        weiter machen.
7500 data  32, 96,166:rem " ..55  20 60
a6  jsr a660        CLR-Befehl
7510 data  32, 51,165:rem " ..58  20 33
a5  jsr a533        BASIC-Zeilen binden
7520 data  76,133,227:rem " ..5b  4c 85
e3  jmp e385        in READY-Modus springen
7530 :
8000 rem " Transport-Routinen: 1 BYTE Se
nden:     $..5e
8010 :
8100 data 141,  1,221:rem " ..5e  8d 01
dd  sta dd01        AKKU in USER-Port
8110 data 162,147:     rem " ..61  a2 93
    ldx#93          VALID-Signal senden
8115 data 142,  0,221:rem " ..63  8e 00
dd  stx dd00
8120 data 173, 13,221:rem " ..66  ad 0d
dd  lda dd0d        QUITTUNG laden
8125 data 208, 11:     rem " ..69  d0 0b
    bne ..76        QUITTUNG empfangen
8130 data 202:         rem " ..6b  ca
    dex             abzaehlen der Zeit
8135 data 208,248:     rem " ..6c  d0 f8
    bne ..66        Ueberschreitung?
8140 data  32,143, -1:rem " ..6e  20 8f
..  jsr ..8f        USER-Port als Eingang
8150 data 162,  5:     rem " ..71  a2 05
    ldx#05          Fehler-Nummer Bet.System
8155 data  76, 58,164:rem " ..73  4c 3a
a4  jmp a43a        Fehler-Routine Rechner
8160 data 162,151:     rem " ..76  a2 97
    ldx#97          VALID-Flag aus
8165 data 142,  0,221:rem " ..78  8e 00
dd  stx dd00        VALID zuruecknehmen
8170 data  96:         rem " ..7b  60
    rts             RETURN
8180 :
8200 rem " Transport-Routinen: 1 BYTE Em
pfangen: $..7c
8202 :
8204 data 173, 13,221:rem " ..7c  ad 0d
dd  lda dd0d        VALID holen
8210 data 240,251:     rem " ..7f  f0 fb
    beq ..7c        ''  warten
8220 data 173,  1,221:rem " ..81  ad 01
dd  lda dd01        BYTE  holen
8230 data 162,147:     rem " ..84  a2 93
    ldx#93          QUITTUNG-Signal senden
8235 data 142,  0,221:rem " ..86  8e 00
dd  stx dd00
8240 data 162,151:     rem " ..89  a2 97
    ldx#97          QUITTUNG-Signal loeschen
8245 data 142,  0,221:rem " ..8b  8e 00
dd  stx dd00
8250 data  96:         rem " ..8e  60
    rts             RETURN
8260 :
9000 rem " Richtungs-Register auf Eingan
g:  $..8f
9010 rem "                          Ausgan
g:  $..92
9020 :
9100 data 162,  0:     rem " ..8f  a2 00
    ldx#00          USER-Port als Eingang
9110 data  44,162,255:rem " ..92  a2 ff
    ldx#ff          USER-Port als Ausgang
9120 data 160,151:     rem " ..94  a0 97
    ldy#97
9130 data 140,  0,221:rem " ..96  8c 00
dd  sty dd00        VALID-Flag zuruecksetzen
9140 data 142,  3,221:rem " ..99  8e 03
dd  stx dd03        USER-Richtung setzen
9150 data  96:         rem " ..9c  60
    rts             RETURN
9160 :
9200 rem " Initialisierung des Ports:
    $..9d
9210 :
9220 data 173, 13,221:rem " ..9d  ad 0d
dd  lda dd0d        VALID laden
9230 data 208,251:     rem " ..a0  d0 fb
    bne ..9d        Flag gesetzt:noch einmal
9240 data  76,143, -1:rem " ..a2  4c 8f
..  jmp ..8f        USER-Port als Eingang

ready.
```