ter somit auch um einiges schneller arbeiten kann. Leider ist sie aber (effektiv) wiederum nur wenig mehr als halb so hoch wie bei anderen Z80-Computern, was bedeutet, daß Turbo-Pascal auf anderen CP/M-Systemen noch schneller ist.

Trotzdem muß man sagen, daß Turbo-Pascal der schnellste derzeit für den C 128 verfügbare Pascal-Compiler ist.

Ein Benchmark-Programm, in dem eine Integer-Variable von 0 bis 2000 hochgezählt wurde, ergab bei der Ausführung einen Zeitbedarf von nur 0,4 Sekunden (zum Vergleich: Profi-Pascal 1,0 Sekunden; Oxford-Pascal 2,3 Sekunden).

Betrachtet man diese Zeiten, so wird es klar, daß Turbo-Pascal eindeutig der schnellste Compiler ist. Die Arbeitsgeschwindigkeit von Turbo-Pascal zeigt, was ein Z80-Prozessor alles vermag. Ein kleiner Anstoß vielleicht für manchen Interessierten, den Z80 näher kennenlernen zu wollen. Ganz sicher aber ein gutes Argument für die Qualität von Turbo-Pascal. (M.Thomas/ev)

Info: Turbo-Pascal ist ein eingetragenes Warenzeichen von Borland International. Vertrieb in Deutschland: Heimsoeth Software, Fraunhoferstr. 13, 8000 München 5, Preis 225 Mark.

Was uns gefiel

— Sicherheitskopie kann vom Besitzer angefertigt werden
— Editor mit den Qualitäten eines Textverarbeitungssystems
— Editor-Kommandos WordStar-kompatibel
— Extrem schneller Compilations-Vorgang
— Hohe Geschwindigkeit des übersetzten Programms
— Sehr komfortable Fehlerkorrektur
— Einbindung von Maschinensprache möglich
— Automatische Overlay-Technik
— Hoher Bedienungskomfort und hohe Bediensicherheit
— Wirth-Standard wird unterstützt
— Viele sinnvolle, zusätzliche Standardprozeduren und Funktionen
— Standardtyp String mit entsprechenden Funktionen implementiert
— Direkter Zugriff auf CP/M-System-Ebene und CPU-Register
— Hohe Genauigkeit bei reellen Zahlen (11 Stellen)
— Hoher Programmierkomfort durch viele Compiler-Optionen
— Automatisches Finden von Laufzeit-Fehlern
— Systemmeldungen editierbar
— Ausführliches, gut verständliches Handbuch
— Sinnvolles, größeres Beispielprogramm im Lieferumfang
— Unterstützung der Turbo-Pascal-Käufer durch den Hersteller
— Gemessen an der Leistung, sehr preiswert

Was uns nicht gefiel

— Handbuch im Taschenbuchformat wird intensive Benutzung nicht überstehen
— Kein Compilerprotokoll auf Drucker möglich

Die Testergebnisse:

		Ausführung	
	Compilation	m. Ausg.	o. Ausg.
Oxford-Pascal	3,7 sec	34,7 sec	2,3 sec
Profi-Pascal	24,8 sec (Disk)	25,1 sec	1,0 sec
Turbo-Pascal	0,5 sec	30,6 sec	0,4 sec

Doppelte Grafikauflösung für C 128

Ein kleines Maschinenprogramm macht dem 80-Zeichen-Videocontroller des C 128 Beine und bringt eine Grafikauflösung von 640 x 200 Punkten.

Wie Sie als stolzer C 128-Besitzer vielleicht wissen, besitzt Ihr Computer zwei Arten der Zeichendarstellung. Die üblichen 40 Zeichen pro Zeile und den 80-Zeichen-Bildschirm.

Schalten Sie auf 80 Zeichen um, so übernimmt ein anderer Video-Chip die Arbeit des uns vom C 64 her bekannten VIC II und zaubert ein 80-Zeichen-Feature auf den Bildschirm. Sehen können Sie dabei allerdings nur etwas, wenn Sie auch einen Monitor an der RGB-Buchse des C 128 angeschlossen haben, denn nur dort sind 80 Zeichen pro Zeile möglich. Der 8563-Videocontroller sorgt in diesem Modus für ein anständiges Bild.

Doch er kann noch mehr. Neben Buchstaben und Zahlen ausgeben ist er fähig, Punktgrafik zu erzeugen, und das in doppelter Auflösung, also statt mit den bekannten 320 mal 200 Punkten nun 640 mal 200 Punkte. Sie haben richtig gelesen. Das sind insgesamt 128000 Bildpunkte, die einzeln ansprechbar sind. Prima, werden Sie sagen, der C 128 hat ja die vielen tollen Grafikbefehle ... Doch halt ! Die Freude ist ein wenig verfrüht. Die doppelte Auflösung des 8563 wird nämlich unverständlicherweise von diesen Basic-Befehlen **nicht** ausgenutzt. Die ganzen fantastischen Grafikbefehle des C 128 sprechen nur die vom C 64 bekannte 320 x 200 Punkte-Grafik an. Warum das so ist, das weiß nur Commodore allein.

Auch der Versuch, die Punkte einfach in den Grafikspeicher des 8563 zu POKEn wird fehlschlagen, denn dieser Grafikspeicher ist vom Prozessor aus nicht ansprechbar. Der Videocontroller 8563 steht nämlich im Genuß eines eigenen Zeichenspeichers von 16 KByte, der nicht im normalen Adreßbereich liegt und nur ihm selbst zugänglich ist.

Doch ganz so eigenständig ist der VDC 8563 nun auch nicht. Es muß selbstverständlich ein Informationsaustausch zwischen Videoprozessor und dem übrigen Computer stattfinden können. Den gibt es natürlich auch.

Die Verbindung besteht allerdings nur aus 2 Byte im Input/Output-Bereich mit den Adressen $D600 und $D601. Durch sie hindurch drängt sich der gesamte Informationsverkehr von VDC 8563 und Computer.

Denn der VDC 8563 muß viel wissen, wenn er ein ordentliches Bild erzeugen will. Da er nur seinen »Privatspeicher« von 16 KByte kennt, kann er auf den Zeichengenerator nicht direkt zugreifen. Damit er trotzdem die Zeichen erzeugen kann, wird er beim Anschalten des Computers mit den nötigen Bytes aus dem Zeichengenerator über den Engpaß $D600/$D601 gefüttert.

Dies geschieht übrigens auch, wenn Sie mit der ASCII/DIN-Taste auf den anderen Zeichensatz umschalten. Die erhaltenen Zeichen legt er in seinem RAM ab, damit er nun darauf zugreifen kann.

Die Aufteilung seines Speichers sieht dann folgendermaßen aus:

2 KByte Zeichen
2 KByte Zeichenattribute
4 KByte Zeichendefinitionen
8 KByte liegen brach

Diese Konfiguration gilt für den Textmodus des 8563. Doch wehe, man setzt im Register 25 das Bit für den Grafikmodus, dann läßt der VDC 8563 Zeichen Zeichen sein und bearbeitet seine 16 KByte im »Bitmapping«-Modus. Das heißt für jedes gesetzte Bit läßt er einen Punkt auf dem Monitor leuchten, für jedes ungesetzte Bit eben nicht. Sein gesamter Speicher ist nun Grafikspeicher.

16 KByte RAM mal 8 Bit ergibt nach sorgfältigem Rechnen genau 128000 Bit, was in unserem Falle 128000 ansteuerbare Bildpunkte bedeutet. Doch wie, werden Sie fragen, kann man durch nur 2 Byte ($D600/$D601) die Register des VDC 8563 oder gar seinen Speicher manipulieren? Die Antwort ist ganz einfach: Sie lautet indirekte Adressierung.

In das erste Verbindungsbyte ($D600) schreibt man die Nummer des Registers, das man ansprechen will (der VDC 8563 hat deren 31). Danach liest man den Wert des angesteuerten Registers über das zweite Byte ($D601), oder man schreibt den gewünschten Wert hinein. Eine einfache Sache.

Wie aber kann nun der 16-KByte-Speicher des Videocontrollers manipuliert werden?

Der 8563 besitzt mehrere Register, von denen einige Informationen über die Speicheraufteilung der 16 KByte geben (Bild 1).

So gibt es Register, die beispielsweise die Startadresse des 80-Zeichen-Speichers enthalten. In den Registern 18 und 19 ist nun eine aktuelle Adresse des Videospeichers abgelegt, dessen Wert gerade bearbeitet werden soll. Und Register 31 hält den Inhalt dieser Adresse bereit.

Man muß also nach obengenanntem Schema die Register 18 und 19 (HI/LO) mit der Adresse des Videospeichers belegen, die man ansprechen will, und kann dann den Inhalt dieser Adresse über das Register 31 auslesen oder sie mit dem gewünschten Wert beschreiben.

REG									
0	HT7	HT6	HT5	HT4	HT3	HT2	HT1	HT0	Horizontal Total
1	HD7	HD6	HD5	HD4	HD3	HD2	HD1	HD0	Horizontal Displayed
2	HP7	HP6	HP5	HP4	HP3	HP2	HP1	HP0	Horizontal Sync Position
3	VW3	VW2	VW1	VW0	HW3	HW2	HW1	HW0	Vert/Horz Sync Width
4	VT7	VT6	VT5	VT4	VT3	VT2	VT1	VT0	Vertical Total
5	--	--	--	VA4	VA3	VA2	VA1	VA0	Vertical Total Adjust
6	VD7	VD6	VD5	VD4	VD3	VD2	VD1	VD0	Vertical Displayed
7	VP7	VP6	VP5	VP4	VP3	VP2	VP1	VP0	Vertical Sync Position
8	--	--	--	--	--	--	IM1	IM0	Interlace Mode
9	--	--	--	CTV4	CTV3	CTV2	CTV1	CTV0	Character Total Vertical
10	--	CM1	CM0	CS4	CS3	CS2	CS1	CS0	Cursor Mode, Start Scan
11	--	--	--	CE4	CE3	CE2	CE1	CE0	Cursor End Scan Line
12	DS15	DS14	DS13	DS12	DS11	DS10	DS9	DS8	Display Start Address hi
13	DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0	Display Start Address lo
14	CP15	CP14	CP13	CP12	CP11	CP10	CP9	CP8	Cursor Position hi
15	CP7	CP6	CP5	CP4	CP3	CP2	CP1	CP0	Cursor Position lo
16	LPV7	LPV6	LPV5	LPV4	LPV3	LPV2	LPV1	LPV0	Light Pen Vertical
17	LPH7	LPH6	LPH5	LPH4	LPH3	LPH2	LPH1	LPH0	Light Pen Horizontal
18	UA15	UA14	UA13	UA12	UA11	UA10	UA9	UA8	Update Address hi
19	UA7	UA6	UA5	UA4	UA3	UA2	UA1	UA0	Update Address lo
20	AA15	AA14	AA13	AA12	AA11	AA10	AA9	AA8	Attribute Start Adr hi
21	AA7	AA6	AA5	AA4	AA3	AA2	AA1	AA0	Attribute Start Adr lo
22	CTH3	CTH2	CTH1	CTH0	CDH3	CDH2	CDH1	CDH0	Character Tot(h), Dsp(v)
23	--	--	--	CDV4	CDV3	CDV2	CDV1	CDV0	Character Dsp(v)
24	COPY	RVS	CBRATE	VSS4	VSS3	VSS2	VSS1	VSS0	Vertical smooth scroll
25	TEXT	ATR	SEMI	DBL	HSS3	HSS2	HSS1	HSS0	Horizontal smooth scroll
26	FG3	FG2	FG1	FG0	BG3	BG2	BG1	BG0	Foregnd/Bgnd Color
27	AI7	AI6	AI5	AI4	AI3	AI2	AI1	AI0	Address Increment / Row
28	CB15	CB14	CB13	RAM	--	--	--	--	Character Base Address
29	--	--	--	UL4	UL3	UL2	UL1	UL0	Underline scan line
30	WC7	WC6	WC5	WC4	WC3	WC2	WC1	WC0	Word Count
31	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0	Data
32	BA15	BA14	BA13	BA12	BA11	BA10	BA9	BA8	Block Start Address hi
33	BA7	BA6	BA5	BA4	BA3	BA2	BA1	BA0	Block Start Address lo
34	DEB7	DEB6	DEB5	DEB4	DEB3	DEB2	DEB1	DEB0	Display Enable Begin
35	DEE7	DEE6	DEE5	DEE4	DEE3	DEE2	DEE1	DEE0	Display Enable End
36	--	--	--	--	DRR3	DRR2	DRR1	DRR0	DRAM Refresh rate

Beschreibung der MAPPED-Register:

$D600: Lesezugriff ergibt STATUS
Schreibzugriff setzt Registerinhalt

$D601: Daten: Ein-/Ausgabe

Zeichen-Speicher: $0000...$07CF
Attribut-Speicher: $0800...$0FCF
Zeichengenerator: $2000...$3FFF

--	--	R5	R4	R3	R2	R1	R0
STATUS	LP	VBLANK	--	--	--	--	--
D7	D6	D5	D4	D3	D3	D2	D0

ALT	RVS	UL	FLASH	R	G	B	I

Bild 1. Die Register des VDC 8563

```
100 BANK 15
110 SYS DEC("1400") : REM *** GRAFIK EINSCHALTEN
120 SYS DEC("1406") : REM *** BILDSCHIRM LÖSCHEN
130 FOR X = 0 TO 639
140 Y=SIN(X/100)     : REM *** SINUSWERT BERECHNEN
150 Y=99*Y+100       : REM *** WERT AN BILDSCHIRMAUSGABE ANPASSEN
160 SYS DEC("140F"), INT(X/256), X AND 255, Y
170 NEXT X
180 SYS DEC("1403") : REM *** TEXT EINSCHALTEN
190 SYS DEC("1409") : REM *** ZEICHENSATZ NEU LADEN, EDITOR INITIALISIEREN
```

Listing 2. Eine Demonstration der hochauflösenden 80-Zeichen-Grafik

Um Ihnen den Aufwand zu ersparen, ein eigenes Programm schreiben zu müssen, das die Grafik des 8563 ausnutzt, haben wir ein Assemblerlisting (Listing 1) dazu abgedruckt. Es stammt aus dem Commodore 128-Handbuch von Peter Rosenbeck (Markt & Technik-Verlag). Sie können dieses Programm direkt mit dem in den C 128 integrierten Maschinensprache-Monitor eingeben.

Es übernimmt die Arbeit der gerade erwähnten Prozedur der indirekten Adressierung des VDC 8563 und bietet auch eine Routine zum Punkte setzen und löschen. Ein unentbehrliches Werkzeug für die Arbeit mit der 640 mal 200 Punkte-Grafik auf dem 80-Zeichenbildschirm.

Die einzelnen Routinen können über den SYS-Befehl angesprochen werden, wie es das kleine Basic-Beispielprogramm (Listing 2) zeigt. Die X- und Y-Koordinaten werden einfach mit dem SYS-Befehl übergeben (Zeile 160). Das Beispielprogramm erzeugt eine Sinuskurve auf dem Monitor.

Wer Spaß daran hat, kann sich weitere Routinen (zum Beispiel zum Linien ziehen) dazuschreiben und somit die Grafikfähigkeiten des VDC 8563 voll ausnutzen.

(M.Thomas/ev)

```
1400 4c 1e 14 jmp $141e ;Grafikmodus anschalten
1403 4c 26 14 jmp $1426 ;Grafikmodus einschalten
1406 4c 87 14 jmp $1487 ;Grafikbildschirm löschen
1409 4c 62 ff jmp $ff62 ;Zeichensatz neu laden
140c 4c 81 ff jmp $ff81 ;Editor initialisieren
140f 4c a3 14 jmp $14a3 ;Punkt setzen
1412 4c 9d 14 jmp $149d ;Punkt löschen
1415 4c ?? ?? jmp $???? ;frei für Erweiterungen
1418 4c ?? ?? jmp $???? ;frei für Erweiterungen
141b 4c ?? ?? jmp $???? ;frei für Erweiterungen

;Grafikmodus anschalten

141e a9 80    lda #$80 ;Bit 7 setzen
1420 a2 19    ldx #$19 ;Register 25
1422 20 cc cd jsr $cdcc ;Register 25 mit
                        $80 besetzen
1425 60       rts

;Grafikmodus ausschalten

1426 a9 40    lda #$40 ;Bit 6 setzen
1428 a2 19    ldx #$19 ;Register 25
142a 20 cc cd jsr $cdcc ;Register 25 mit
                        $40 besetzen
142d 60       rts

;aktuelle Adresse setzen (in X und Y)

142e a9 12    lda #$12 ;Register 18
1430 8d 00 d6 sta $d600 ;ansteuern
1433 8e 01 d6 stx $d601 ;HI von Adresse nach Reg. 18
1436 20 45 14 jsr $1445 ;Warten auf Statusbit
1439 a9 13    lda #$13 ;Register 19
143b 8d 00 d6 sta $d600 ;ansteuern
143e 8c 01 d6 sty $d601 ;LO von Adresse
                        nach Reg. 19
1441 20 45 14 jsr $1445 ;Warten auf Statusbit
1444 60       rts

;Warten bis Statusbit gesetzt

1445 2c 00 d6 bit $d600 ;Bit 7 (Status) gesetzt
1448 10 fb    bpl $1445 ;nein, dann warte
144a 60       rts

;Warten bis Statusbit gelöscht

144b 2c 00 d6 bit $d600 ;Bit 7 (Status) gelöscht
144e 30       bmi $144b ;nein, dann warten
1450 60       rts

;Wortzähler Null setzen

1451 a9 1e    lda #$1e ;Register 30
1453 8d 00 d6 sta $d600 ;ansteuern
1456 20 45 14 jsr $1445 ;Warten auf Statusbit
1459 a9 00    lda #$00 ;Register 30
145b 8d 01 d6 sta $d601 ;Null setzen
145e 60       rts

;Datenbyte Null setzen

145f a9 1f    lda #$1f ;Register 31
1461 8d 00 d6 sta $d600 ;ansteuern
1464 20 45 14 jsr $1445 ;Warten auf Statusbit
1467 a9 00    lda #0   ;Register 31
1469 8d 01 d6 sta $d601 ;Null setzen
146c 60       rts

;Datenbyte nach A holen

146d a9 1f    lda #$1f ;Register 31
146f 8d 00 d6 sta $d600 ;ansteuern
1472 20 45 14 jsr $1445 ;Warten auf Statusbit
1475 ad 01 d6 lda $d601 ;Byte nach A holen
1478 60       rts

;Datenbyte (in A) in aktuelle Adresse
;ablegen

1479 48       pha       ;A retten
147a a9 1f    lda #$1f ;Register 31
147c 8d 00 d6 sta $d600 ;ansteuern
147f 20 45 14 jsr $1445 ;testen auf Statusbit
1482 68       pla       ;A wieder holen
1483 8d 01 d6 sta $d601 ;in aktuelle Adr. speichern
1486 60       rts


;Bildschirm löschen

1487 a2 00    ldx #$00 ;HI Adresse von Bildschirm in X
1489 a0 00    ldy #$00 ;LO Adresse in Y
148b 20 2e 14 jsr $142e ;aktuelle Adresse setzen
148e 20 51 14 jsr $1451 ;Wortzähler Null setzen
1491 20 5f 14 jsr $145f ;Datenbyte Null setzen
1494 c8       iny       ;nächste Adresse
1495 d0 f4    bne $148b
1497 e8       inx
1498 e0 40    cpx #$40 ;letzte Adresse von Bildschirm?
149a d0 ef    bne $148b ;nein, dann nächste Adresse
149c 60       rts

;Punkt löschen

149d 48       pha       ;A retten
149e a9 00    lda #$00 ;Flag für Löschen
14a0 4c a6 14 jmp $14a6 ;Punkt löschen

;Punkt setzen

14a3 48       pha       ;A retten
14a4 a9 ff    lda #$ff Flag für Setzen
14a6 85 c3    sta $c3   ;zwischenspeichern
14a8 68       pla       ;A wieder holen
14a9 20 db 14 jsr $14db ;Adressberechnung
                        (X-Koord. in A,X
                        ; Y-Koord. in Y)
14ac b0 ee    bcs $149c ;Angaben außerhalb
                        des Bereichs
14ae 85 c4    sta $c4   ;Bitmaske (in A)
                        zwischenspeichern
14b0 a4 c1    ldy $c1   ;LO Adresse nach Y
14b2 a6 c2    ldx $c2   ;HI Adresse nach X
14b4 20 2e 14 jsr $142e ;aktuelle Adresse
                        (in X/Y) setzen
14b7 20 6d 14 jsr $146d ;Datenbyte aus aktueller
                        Adr. nach A holen
14ba 48       pha       ;retten
14bb a5 c3    lda $c3   ;Flag für setzen/löschen
                        nach Adr. f0 06
14bd f0 06    beq $14c5 ;Punkt löschen, dann
                        zu Löschen
14bf 68       pla       ;Datenbyte wieder holen
14c0 05 c4    ora $c4   ;mit Bitmaske verknüpfen
                        (Bit setzen)
14c2 4c ce 14 jmp $14ce ;zum Abspeichern
14c5 68       pla       ;Datenbyte wieder holen
14c6 85 c3    sta $c3   ;zwischenspeichern
14c8 a5 c4    lda $c4   ;Bitmaske nach A
14ca 49 ff    eor #$ff ;alle Bits umdrehen
14cc 25 c3    and $c3   ;mit Datenbyte verknüpfen
                        (löschen)
14ce 48       pha       ;neues Datenbyte retten
14cf a4 c1    ldy $c1   ;LO Adresse nach Y
14d1 a6 c2    ldx $c2   ;HI Adressenach X
14d3 20 2e 14 jsr $142e ;aktuelle Adresse setzen
14d6 68       pla       ;neues Datenbyte wieder holen
14d7 20 79 14 jsr $1479 ;Datenbyte in aktueller
                        Adr. ablegen

14da 60       rts

;Adresse berechnen (X-Koord. in A/X; Y-Koord. in Y)
;Adresse nach $C1/$C2; Bitmaske nach A

14db c9 03    cmp #$03 ;A größer gleich 3?
14dd b0 55    bcs $1534 ;wenn ja, dann ord. zu groß
14df c9 02    cmp #$02 ;A kleiner 2?
14e1 d0 04    bne $14e7 ;wenn ja, dann X-Koord. ok
14e3 e0 80    cpx #$80 ;LO von X-Koord. zu groß?
14e5 10 4d    bpl $1534 ;wenn ja, keine
                        Adreßberechnung
14e7 c0 c8    cpy #$c8 ;Y-Koord. zu groß?
14e9 b0 49    bcs $1534 ;wenn ja, dann keine
                        Adreßberechnung
14eb 48       pha       ;HI X-Koord. retten
14ec 8a       txa       ;LO X-Koord.
14ed 48       pha       ;retten
14ee 98       tya
14ef 29 0f    and #$0f
14f1 aa       tax
14f2 bd 40 15 lda $1540,x;LO-Wert aus Tabelle 1
14f5 85 c1    sta $c1   ;ablegen
14f7 bd 50 15 lda $1550,x;HI-Wert aus Tabelle 2
14fa 85 c2    sta $c2   ;ablegen
14fc 98       tya
14fd 29 f0    and #$f0
14ff 4a       lsr
1500 4a       lsr
1501 4a       lsr
1502 4a       lsr
1503 aa       tax
1504 bd 60 15 lda $1560,x;HI-Wert aus Tabelle 3
1507 18       clc
1508 65 c2    adc $c2   ;dazuzählen
150a 85 c2    sta $c2
150c 68       pla
150d aa       tax
150e 68       pla
150f a8       tay
1510 b9 70 15 lda $1570,y;LO-Wert aus Tabelle 4
1513 18       clc
1514 65 c1    adc $c1   ;dazuzählen
1516 85 c1    sta $c1
1518 90 02    bcc $151c
151a e6 c2    inc $c2
151c 8a       txa
151d 29 f8    and #$f8
151f 4a       lsr
1520 4a       lsr
1521 4a       lsr
1522 18       clc
1523 65 c1    adc $c1
1525 85 c1    sta $c1
1527 90 02    bcc $152b
1529 e6 c2    inc $c2
152b 8a       txa
152c 29 07    and #$07
152e aa       tax
152f bd 73 15 lda $1573,x;Bitmaske aus Tabelle 5
1532 18       clc       ;holen
1533 60       rts
1534 38       sec
1535 60       rts

;Tabellen für Adreßberechnung

Tabelle 1

.: 1540 00 50 a0 f0 40 90 e0 30
.: 1548 80 d0 20 70 c0 10 60 b0

Tabelle 2

.: 1550 00 00 00 00 01 01 01 02
.: 1558 02 02 03 03 03 04 04 04

Tabelle 3

.: 1560 00 05 0a 0f 14 19 1e 23
.: 1568 28 2d 32 37 3c 41 46 00

Tabelle 4

.: 1570 00 20 40

Tabelle 5 (Bitmasken)

.: 1573 80 40 20 10 08 04 02 01
```

Listing 1. Assemblerprogramm zur Nutzung der verborgenen Grafikfähigkeiten des C 128