GripS

Version 1.15

Grafik Interface Prozessor

Umbausatz zu GRIP

Umbau von GRIP zu GripS

GripsS ist ein Umbausatz für den Grafik-Interface-Prozessor GRIP der Firma Conitec. GripS besteht aus einer Aufsteckplatine und einem EPROM. Durch GripS wird GRIP sehr viel schneller und vollkommen bildstörungsfrei. Dies wird erreicht durch eine Änderung des Timings der Bildschirmspeicherzugriffe, so daß jederzeit, evtl. um einen Wait-Zyklus verzögert, auf den Bildschirmspeicher zugegriffen werden kann, also nicht nur während der Strahlrücklaufphasen. Außerdem sind jetzt Baudraten bis 19.200 Baud möglich (meistens läuft dabei der Buffer noch nicht einmal voll!).

Voraussetzungen für die Installation

Voraussetzung für die Anwendbarkeit von GripS ist, daß mindestens die IC's Z18, Z34, Z35, Z36 und Z37 von GRIP gesockelt sind. GripS-1 unterstützt die Farberweiterungskarte nicht, diese darf auch nicht angeschlossen werden.

Installation von GripS

Die IC's Z18, Z34, Z35 und Z37 werden aus ihren Sockeln entfernt. Die Jumper J4 und J8 werden unterbrochen (Leiterbahn auf der Bauteilseite (J4) bzw. Lötseite (J8) entfernen). Der Mittenkontakt des Jumpers J8 wird mit GND verbunden, für J4 werden Pfosten eingelötet. Die Leiterbahn, die zu Pin 24 von Z1 führt, ist zu unterbrechen, ein Draht von diesem Pin zu Pin 9 von Z35 zu legen. Pin 6 von Z36 wird aus dem Sockel gezogen, abgebogen und mit einem Stück Draht mit Pin 35 von Z30 verbunden.
Beim Unterbrechen von Leiterbahnen ist größte Vorsicht angebracht, um keine weiteren Leiterbahnen zu beschädigen. Das Unterbrechen geschieht am einfachsten mit einem Skalpell oder scharfen Messer, wobei ein kurzes Stück herausgeschnitten und mit dem heißen Lötkolben entfernt wird.

Der Quarz X1 auf der GRIP-Platine sollte eine Frequenz von 16 MHz haben, andernfalls kann das Bild verzerrt sein oder der Monitor nicht synchronisieren.

Die GripS-Platine wird dann vorsichtig auf die freigewordenen Sockel gesteckt (achten Sie darauf, daß die Kontakte in die richtigen Sockelpins gelangen) und das angelötete Kabel auf die Pfosten von J4 (auf richtige Polarität achten, siehe Plan im Anhang).

Ersetzen Sie nun das Original-Eprom durch das mitgelieferte.

GRIP & GripS sind jetzt einsatzbereit, nach Einschalten sollten ein kurzer Ton zu hören sein und die Statuszeile am oberen Bildschirmrand erscheinen.

Fehlersuche

An Pin 5 von Z9 sollte eine Rechteckschwingung mit einer Frequenz von 2,4576 MHz messbar sein, an Pin 6 von Z1 eine Frequenz von 4 MHz.

Ist dies der Fall, die Karte funktioniert aber trotzdem nicht, d.h. es erscheint kein Bild, so ziehen Sie bitte die IC's Z3, Z30, Z32 aus ihren Fassungen. Wenn Z30 nicht bestückt ist, springt das Programm (so es korrekt ausgeführt wird) in eine Selbsttest-Routine.

Nach dem Einschalten sollte ein Ton von 880 Hz zu hören sein. Ist dies nicht der Fall, prüfen Sie, ob bei den IC's Z6 (Pin 6), Z7 (Pin 11), Z8 (Pin 14), Z9 (Pin 30), Z14 (Pin 1), Z33 (Pin 1) kurze LOW-Impulse und bei Z30 (Pin 23) kurze HIGH-Impulse meßbar sind. Das Programm wird dann ordnungsgemäß ausgeführt.

Schalten Sie dann die Karte aus und stecken Z3 in die Fassung. Nach dem Einschalten sollte ein Ton von 440 Hz zu hören sein. Z3 ist dann in Ordnung.

Wieder ausschalten, Z32 in die Fassung stecken, einschalten. Nun sollten Sie einen Ton von 220 Hz hören. Der dynamische Speicher funktioniert dann. Wenn Sie soweit sind, die Karte aber trotzdem nicht funktioniert, müssen Sie den Fehler anderweitig lokalisieren.

Sollten sporadisch einzelne Streifen oder Punkte auf dem Bildschirm sichtbar werden, kann ein zu langsamer RAM 4164 die Ursache sein, Sie sollten RAMs mit höchstens 150 ns Zugriffszeit verwenden.

Anschluß

Wenn Sie die Karte auf den ECB-Bus stecken, achten Sie darauf, daß die Anschlüsse für +12V und -12V auf Pin 13a bzw. 14a liegen!

Wenn Sie die Karte an der V24-Schnittstelle betreiben, sollten Sie die CTS- und RTS-Leitungen für das Handshake benutzen. Andernfalls können Zeichen verloren gehen. Es kann auch das XON/XOFF-Protokoll verwendet werden, es muß durch eine Doppelescape-Sequenz eingestellt werden. Stellen Sie dir richtige Baudrate ein und vergewissern Sie sich, daß sie die richtige Anzahl an Bits/char und Stopbits und die richtige Parität verwenden (siehe Doppelescape-Sequenzen). Diese Parameter können Sie auch interaktiv mit der Tastatur verändern (siehe Setup-Zeile).

Bei der seriellen Tastatur wird die Polarität automatisch vom Programm erkannt, bei der parallelen Tastatur muß die Polarität nichtinvertiert sein, Strobe negativ (LOW-Impuls).

Jumper

Hier wird nur der Jumper J3 erklärt, die Funktion der übrigen Jumper ist dem GRIP–Handbuch zu entnehmen.

Legen Sie die Karte so, daß der **96–polige Stecker nach links** zeigt.

Jumper J3, linke Hälfte (Festlegung der Quelle)

Die Karte wird am ECB–Bus betrieben.


Die Karte wird seriell mit 19200 Baud, 8 Bit/char betrieben.


Die Karte wird seriell mit 9600 Baud, 8 Bit/char betrieben.


Die Karte wird seriell mit 2400 Baud, 8 Bit/char betrieben.


Die Karte wird zu Testzwecken lokal betrieben. Die Tastatur ist direkt mit dem Terminal verbunden.

Jumper J3, rechte Hälfte (Festlegung der Tastatur)

Es wird der parallele Tastatureingang benutzt (8 Bit).


Es wird der serielle Tastatureingang mit 1200 Baud, 8 Bit/char benutzt.


Es wird der serielle Tastatureingang mit 1200 Baud, 7 Bit/char benutzt.


Es wird der serielle Tastatureingang mit 600 Baud, 8 Bit/char benutzt.


Es wird der serielle Tastatureingang mit 600 Baud, 7 Bit/char benutzt.

Doppelescape–Sequenzen

spezifiziere V24–Empfängerbaudrate ESC ESC 0*a*
spezifiziere V24–Senderbaudrate ESC ESC 1*a*
spezifiziere serielle Tastaturbaudrate ESC ESC 3*a*

wobei *a* folgende Bedeutung hat:

a	= 0	Baudrate wie J3 (def)
	1	5 0 Baud
	2	7 5 Baud
	3	1 1 0 Baud
	4	1 5 0 Baud
	5	3 0 0 Baud
	6	6 0 0 Baud
	7	1 2 0 0 Baud
	8	2 4 0 0 Baud
	9	4 8 0 0 Baud
	:	9 6 0 0 Baud
	;	1 9 2 0 0 Baud

Beispiel: ESC ESC 1 7 (1Bh 1Bh 30h 37h) setzt die Senderbaudrate auf 1200 Baud.

setze V24–Betriebsart ESC ESC 2*x*

wobei *x* ein Byte ist, dessen Bits folgende Bedeutung haben:

x = 0w1w0.s1.s0.p1.p0.q

w1w0	=	0 0	Wortlänge 8 Bit
		0 1	Wortlänge 7 Bit (def)
		1 0	Wortlänge 6 Bit
		1 1	Wortlänge 5 Bit
s1s0	=	0 1	1 Stoppbit (def)
		1 0	1,5 Stoppbits
		1 1	2 Stoppbits
p1p0	=	0 1	keine Parität (def)
		1 0	ungerade Parität
		1 1	gerade Parität
q	=	0	XON/XOFF–Protokoll aus (def)
		1	XON/XOFF–Protokoll ein

Beim XON/XOFF–Protokoll sendet die Schnittstelle ein XOFF–Zeichen ↑S, wenn während des Datenempfangs der Empfangsbuffer überzulaufen droht. Daraufhin muß die angeschlossene Einheit das Senden unterbrechen. Ist der Buffer wieder leer, wird mit einem XON–Zeichen ↑Q zum Weitersenden aufgefordert.

Dieses XON/XOFF-Protokoll ist nur erforderlich, wenn nicht mit RTS/CTS-Handshake gearbeitet werden kann.

Beispiel: ESC ESC 2 ↑ T (1Bh 1Bh 32h 14h) setzt 8 Bit Wortlänge, 1,5 Stoppbits, ungerade Parität und kein XON/XOFF-Protokoll.

definiere Tastatur-Kanal und -Parameter ESC ESC 4 *x*

wobei *x* ein Byte ist, dessen Bits folgende Bedeutung haben:

x = 0w.i.c.0.0.k1.k0

w = 0 Wortlänge 8 Bit
1 Wortlänge 7 Bit

i = 0 Tastatur nichtinvertiert (def)
1 Tastatur invertiert

c = 0 paralleler Tastatureingang
1 serieller Tastatureingang

k1k0 = 0 0 kein Tastenklick
0 1 leiser Tastenklick
1 0 normaler Tastenklick (def)
1 1 lauter Tastenklick

Beispiel: ESC ESC 4 0 (1Bh 1Bh 34h 30h) setzt 8 Bit Wortlänge, invertierte serielle Tastatur ohne Tastenklick.

Spooler einschalten ESC ESC 5 0
Spooler ausschalten (def) ESC ESC 5 3

Der Spooler faßt 32 KByte Daten und dient der Zwischenspeicherung der Daten zum Drucker. Wenn der Spooler eingeschaltet ist, erscheint in der Statuszeile ein Druckersymbol. Der nicht-leere Spoolerspeicher kann an dem dunklen "Papier" in diesem Druckersymbol erkannt werden.

Zeichensatz auswählen

ASCII (def)	ESC ESC 7 0
Deutsch	ESC ESC 7 1
Skandinavisch	ESC ESC 7 2
Dänisch	ESC ESC 7 3
Britisch	ESC ESC 7 4
Französisch	ESC ESC 7 5
International	ESC ESC 7 6
Griechisch / Sonderzeichen	ESC ESC 7 7
selbstdefiniert	ESC ESC 7 8

Die ersten 7 Zeichensätze unterscheiden sich nur in wenigen Zeichen. Der achte

enthält griechische Buchstaben, mathematische Zeichen und Sonderzeichen. Der neunte Zeichensatz kann selbst definiert werden. Alle Zeichen lassen sich gleichzeitig auf dem Bildschirm darstellen, eine Umschaltung zwischen den Zeichensätzen ist jederzeit, auch innerhalb der Zeile, möglich. Die Zeichensätze sind im Anhang aufgelistet.

Textformat wählen ESC ESC 8 *x s*

Dabei ist *x* das ASCII–Zeichen (Zeilenzahl+31), *s* das ASCII–Zeichen (Spaltenzahl+31). Default und maximales Format ist 34 Zeilen und 96 Spalten.

Beispiel: ESC ESC 8 7 o (1Bh 1Bh 38h 37h 6fh) schaltet auf 24 x 80 (TeleVideo–Standard), ESC ESC 8 A DEL (1Bh 1Bh 38h 41h 7Fh) schaltet auf 34 x 96 (maximales Format).

Videosignal an verwendeten Monitor anpassen ESC ESC 9 *x*

wobei *x* ein Byte ist, dessen Bits folgende Bedeutung haben:

x = 0.i.w1.w0.h1.h0.v1.v0

i = 0 kein Zeilensprung (def)
1 Pseudo–Zeilensprung

w1w0 = 0 0 kein Horizontal–Syncimpuls
0 1 Horizontal–Syncimpuls kurz
1 0 Horizontal–Syncimpuls mittel (def)
1 1 Horizontal–Syncimpuls lang

h1h0 = 0 0 Bild nach links
0 1 horiz. Bildlage normal (def)
1 0 Bild nach rechts
1 1 Bild stark nach rechts

v1v0 = 0 0 Bild nach oben
0 1 vert. Bildlage normal (def)
1 0 Bild nach unten
1 1 Bild stark nach unten

Beispiel: ESC ESC 9 e (1Bh 1Bh 39h 65h) schaltet bei normaler Bildlage den Pseudo–Zeilensprung–Modus ein.

Emulationen

Turbo

TeleVideo-Modus einschalten (24 x 80) mit größerem Zeilenabstand ESC ESC A
Die Zeichen werden in einem 8 x 11 Raster dargestellt.

Televideo-Modus einschalten (34 x 96) ESC ESC a

Tektronix-Modus einschalten (mit Statuszeile) ESC ESC T

Tektronix-Modus einschalten (ohne Statuszeile) ESC ESC t

Setup-Zeile

Setup-Taste definieren ESC ESC ! *aa*

Dabei bedeutet *aa* den Hex-Code der Setup-Taste (default ist 80h).

Einige der obigen Konfigurationen können interaktiv mittels der Setup-Taste geändert werden. Nach Drücken dieser Taste erscheint am unteren Bildschirmrand die Set-Up-Zeile. Mit den Tasten TAB (↑ I) und BS (↑ H) kann der Cursor von einem Feld zum nächsten bzw. zum vorhergehenden bewegt werden, mit der Taste **T** (oder **t** oder ↑ T) kann der Wert in diesem Feld zyklisch verändert werden. Mit den Zifferntasten kann auf eine andere Set-Up-Zeile mit neuen Feldern umgeschaltet werden, mit der Setup-Taste wird die Set-Up-Zeile wieder ausgeschaltet. Solange die Set-Up-Zeile sichtbar ist, bewirkt die Taste **H** (oder **h**), daß eine Hardcopy des Bildschirminhalts auf dem Drucker erzeugt wird. Mit der Setup-Zeile kann ein **Monitor-Modus** eingeschaltet werden, so daß Steuerzeichen nicht mehr interpretiert sondern als Kürzel auf den Bildschrim geschrieben werden. Dies ist für die Kontrolle der Ausgabe von Programmen wichtig.

interaktive Tasten-Umdefinition

Tasten-Umdefinitions-Taste definieren ESC ESC " *a*

Dabei bedeutet *aa* den Hex-Code der Umdefinitions-Taste (default ist 81h).

Die Tastatur-Belegung kann interaktiv mittels der Tasten-Umdefinitions-Taste geändert werden. Das Umdefinieren geht folgendermaßen vor sich: Drücken Sie die Umdefinitions-Taste, dann die umzudefinierende Taste, dann geben Sie den String ein, der von dieser Taste erzeugt werden soll, dann drücken Sie wieder die Umdefinitions-Taste. Diese Umdefinition wird in die Tasten-Umcodetabelle eingefügt. Sie geht natürlich mit dem Ausschalten verloren, ebenso, wenn die Tasten-Umcodetabelle mittels Datentransfer (s. d.) neu geschrieben wird. Ausnahmen von dieser Prozedur sind die Setup-Taste und die Umdefinitions-Taste selbst. Diese können auch interaktiv umdefiniert werden, jedoch nach einem anderen Schema:

Umdefinitions-Taste drücken, Set-Up-Taste drücken, neue Set-Up-Taste drücken bzw.
Umdefinitions-Taste zweimal drücken, neue Umdefinitionstaste drücken

Datentransfer

Der Datentransfer läuft immer von dem eingestellten Quellenkanal zum Rechner bzw. von diesem zum eingestellten Zielkanal.

Bei den Datentransferkommandos ist vom Hexformat und von binär die Rede. Hexformat meint dabei 2 bzw. 4 hexadezimale Ziffern (0..9,A..F), wobei statt (A..F) auch (:..?) zugelassen ist. GripS antwortet jedoch immer mit (0..9,A..F). Binär meint 1 oder 2 ASCII-Zeichen im Bereich 00h bis FFh. Bei 16-Bit-Binärwerten steht immer das niederwertige Byte zuerst.

Folgende Kanäle sind implementiert:

Centronics

Die Übertragung kann über den Spooler laufen (wenn dieser eingeschaltet wird), und wird dadurch beschleunigt.

ECB-Bus

Der ECB-Bus ist in Empfangsrichtung mit 250 Bytes gepuffert.

V24-Schnittstelle

Die V24-Schnittstelle ist in Empfangsrichtung mit 250 Bytes gepuffert.

Video-RAM

Es kann auf das dynamische RAM (Video-RAM) direkt zugegriffen werden. Die untere Page wird für den Bildschirm verwendet, die Home-Position wandert jedoch beim Scrollen. Die obere Page wird vom Spooler belegt, wenn er eingeschaltet ist, ansonsten ist sie frei.

programmierbarer Zeichensatz

Es können die Zeichen 20h (Blank) bis 7fh (DEL) programmiert werden. Der programmierbare Zeichensatz kann zuvor mit einem der Standardzeichensätze vorbelegt werden, wenn nur einzelne Zeichen umdefiniert werden sollen. Jedes Zeichen wird durch 8 Bytes definiert, die einen 8 x 8 Punkte-Block bilden. Die Bytes sind die Zeilen des Blocks, der oben links mit Bit 0 von Byte 1 beginnt, d. h. die Bytes werden gegenüber der üblichen Schreibweise gespiegelt dargestellt.

Userprogramm

Es ist Platz für 200 Bytes ab Adresse 4300h. Das Programm kann an dieser Adresse mit einem vorgegebenen Wert für das A-Register gestartet werden und muß mit einem RETURN (C9h) enden. Außer den Index-Registern dürfen alle Register verändert werden, das IX-Register darf auch dann nicht verändert werden, wenn es später

wiederhergestellt wird! Der Stack läuft von Adresse 4400h ab herunter und kann durch das Laden des Userprogramms nicht überschrieben werden!

Tasten-Umcodetabelle

Für jede gedrückte Taste kann statt des Original-Tastencodes ein beliebiger (max. 254 Zeichen langer) String zum Rechner gesendet werden. Die Umcodierung erfolgt über eine Tabelle, die bis zu 256 Bytes enthalten darf. Das Format ist: Tastencode (1 Byte), Stringlänge (1 Byte), String, nächster Tastencode usw. Die Stringlänge 0 beendet die Tabelle.

Beispiel: Die Tasten Y und Z sollen vertauscht werden, eine Funktionstaste, die den Code 84h sendet, soll D I R[CR] erzeugen. In die Umcodetabelle muß dann eingetragen werden:

59h	01h	5ah	5ah	01h	59h	(Y mit Z vertauschen)
79h	01h	7ah	7ah	01h	79h	(y mit z vertauschen)
19h	01h	1ah	1ah	01h	19h	(↑Y mit ↑Z vertauschen)
84h	04h	44h	49h	52h	0dh	(84h sendet D I R[CR])
00h	00h					(beendet die Tabelle)

Melodiegenerator

Für jeden Ton sind zwei Bytes erforderlich, das erste Byte *b1* bestimmt die Tonhöhe (Frequenz) f nach folgender Formel:

f = 76.8 kHz / *b1* für gerades *b1*
f = 6.144 kHz / *b1* für ungerades *b1*

Es lassen sich also Töne zwischen 38.4 kHz (*b1* = 2) und 24.1 Hz (*b1* = 255) erzeugen. Das zweite Byte *b2* bestimmt die Lautstärke und Länge des Tons. Es ist folgendermaßen aufgebaut:

b2 = 0.1.1.1.0.d4.d3.d2.d1.d0

1.1.1.0 = 0 0 Pause
0 1 leiser Ton
1 0 mittellauter Ton
1 1 lauter Ton

d4..d0 = Dauer in Einheiten von 20 ms

Die Tonerzeugung beginnt jeweils nach Übertragung des 2. Bytes. Eine kontinuierliche Tonerzeugung ist gewährleistet, wenn die Generierung der Bytes schneller erfolgt als das Abspielen. Der Host-Puffer gibt dabei einen gewissen Spielraum.

Bei Dauer = 0 wird ein Dauerton erzeugt, der abgeschaltet wird durch Senden eines Bytes 00h zum Melodiegenerator (also kein zweites Byte).

Zielkanal einstellen

Centronics (def)	ESC ESC C 0
ECB–Bus	ESC ESC C 1
V24–Schnittstelle	ESC ESC C 2
Video–RAM direkt	ESC ESC C 3
programmierbarer Zeichensatz	ESC ESC C 4
Userprogramm	ESC ESC C 5
Tasten–Umcodetabelle	ESC ESC C 6
Melodiegenerator	ESC ESC C 7

Quellenkanal einstellen

ECB–Bus	ESC ESC c 0
V24–Schnittstelle	ESC ESC c 1
serieller Tastatureingang	ESC ESC c 2
paralleler Tastatureingang	ESC ESC c 3
Video–RAM direkt	ESC ESC c 4

Default–Quellenkanal ist Kanal 1, falls GripS vom ECB–Bus aus betrieben wird, sonst Kanal 0.

Zielkanal initialisieren ESC ESC P

Je nach eingestelltem Zielkanal wird folgendes getan:

- Centronix: der Spooler wird gelöscht
- Video–Ram: Adresse 0000h einstellen
- Progr. Zeichensatz: Zeichen 20h einstellen
- User–Programm: Adresse 4300h einstellen
- Tasten–Umcodetabelle: Löschen und auf Anfang einstellen

Setze Video–RAM–Adresse im Hex–Format	ESC ESC I *aaaa*
Setze Video–RAM–Adresse binär	ESC ESC i *xx*

Dabei bedeutet *aaaa* bzw. *xx* eine 16–Bit–Adresse im Hex– bzw. Binärformat. (Im Binärformat erst LOW–Byte, dann HIGH–Byte.) Das höchstwertige Bit gibt die Page an. Diese Adresse wird beim Einschreiben oder Auslesen um eins hochgezählt.

Beispiel: ESC ESC I 4 0 0 0 oder ESC ESC i 0 0 h 4 0 h setzen die Adresse auf 4000h in Page 0.

Setze zu definierendes Zeichen im programmierbaren Zeichensatz ESC ESC J *a*

Dabei bedeutet *a* das ASCII–Zeichen, das programmiert werden soll. Nach jeweils 8 übertragenen Bytes wird das nächste Zeichen programmiert.

Sende ein Byte im Hex–Format zum Zielkanal	ESC ESC D*aa*
Sende ein Byte binär zum Zielkanal	ESC ESC d *x*

Dabei bedeutet *aa* bzw. *x* ein Byte, das zum Zielkanal übertragen wird.

Hole ein Byte im Hex–Format vom Quellenkanal ESC ESC E
Hole ein Byte binär vom Quellenkanal ESC ESC e

GripS antwortet mit dem Byte im Hex– bzw. Binärformat.

Sind vom Quellenkanal keine Daten mehr verfügbar, so wird 00h gesendet

Sende Byte–Folge im Hex–Format zum Zielkanal ESC ESC F *aa* .. CR
Sende Byte–Folge binär zum Zielkanal ESC ESC f *xx y* ..

Dabei bedeutet *aa* .. bzw. *y* .. eine Bytefolge im Hex– bzw. Binärformat.
xx ist ein 16–Bit–Wert, der die Anzahl der folgenden Binärbytes angibt (LOW–Byte, dann HIGH–Byte).

Hole Bytefolge im Hex–Format vom Quellenkanal ESC ESC G *aaaa*
Hole Bytefolge binär vom Quellenkanal ESC ESC g *xx*

Dabei bedeutet *aaaa* bzw. *xx* die Zahl der zu holenden Bytes (binär: LOW–Byte vor HIGH–Byte). GripS antwortet mit der entsprechenden Zahl von Bytes im Hex– bzw. Binärformat.

Direktmodus

Bytes mit gesetztem Bit 7 werden direkt zum Zielkanal gesendet. Das Bit 7 wird dabei gelöscht.

sonstige Kommandos

Bildschirminhalt zum Zielkanal senden (Hardcopy) ESC ESC H

Der Bildschirminhalt (ohne Statuszeile) wird zum Zielkanal gesendet.
In dieser Zeit kann der Bildschirminhalt nicht verändert werden. Ist der Zielkanal der Drucker (Centronics) und ist der Spooler eingeschaltet, so ist das Terminal wesentlich weniger lange blockiert. Die Daten werden in einem Format gesendet, daß der gewählte Drucker sie verarbeiten kann.

Drucker für Hardcopy auswählen

Epson RX–80 oder FX–80 (x/y–Seitenverhältnis 1.2:1) (default)	ESC ESC h 0
Gemini–10X	ESC ESC h 1
Siemens PT–88	ESC ESC h 2
Epson FX–80 (x/y–Seitenverhältnis 1:1)	ESC ESC h 3
Epson LQ–1500	ESC ESC h 4

Bei Epson und Gemini wird die Druckkopfbewegung optimiert, der Druckkopf bewegt sich nur zum letzten zu druckenden Punkt.

Ergänzungen ab Version 1.11

Folgendes **Problem** existiert bei GripS (wie auch bei GRIP, TeleVideo u.a.):

Es gibt Escape-Sequenzen (z.B. ESC ?), die bewirken, daß das Terminal ein oder mehrere ASCII-Zeichen an den Rechner schickt (hier: die Cursor-Position). Wird gleichzeitig jedoch eine Taste gedrückt, so kann der Rechner nicht unterscheiden, ob das Zeichen, das er gerade vom Terminal bekommen hat, von der Tastatur stammt oder die Antwort auf die Escape-Sequenz ist.

Betroffen von diesem Problem sind bei GripS alle Funktionen, die eine Antwort der GRIP-Karte an den Rechner bewirken, dieses sind:
ESC ESC E, ESC ESC e, ESC ESC G, ESC ESC g, ESC ESC L, ESC ESC M,
ESC ESC N, ESC ESC n, ESC ESC u, ESC ESC v,
im TeleVideo-Modus: ESC ?, ESC M
im Tektronix-Modus: ESC P.

Dieses Problem wirkt sich besonders dann sehr störend aus, wenn der Rechner in einer Schleife z.B. dauernd hintereinander ein Zeichen vom Quellkanal liest und die Tastatur abfragt, um z.B. einen Abbruch zu ermöglichen. Da GripS (wie jedes andere Terminal) von der Tastatur kommende Zeichen immer ungefragt (d.h. ohne Anforderung des Rechners) an den Rechner weitergibt, erscheinen sie dem Rechner u. U. als die Antwort-Zeichen auf seine Escape-Sequenz und die wirklichen Antwort-Zeichen als Tastendrücke.

Abhilfe: Um diesem Problem abzuhelfen, darf das Terminal die von der Tastatur kommenden Zeichen nicht einfach an den Rechner senden, sondern muß sie solange speichern, bis der Rechner sie anfordert. Dazu muß der Rechner natürlich abfragen können, ob ein solches Zeichen gespeichert ist. Ab Version 1.11 gibt es jetzt 4 neue Doppel-Escape-Sequenzen:

ESC ESC #	Status abfragen. GripS antwortet mit einem Byte, wobei Bit 0 = 1 bedeutet: Zeichen im V24-Empfangsbuffer, Bit 1 = 1 bedeutet: V24-Sender ist zum Senden bereit, Bit 2 = 1 bedeutet: Zeichen im Tastaturbuffer.
ESC ESC '	Tastatur lesen. GripS antwortet mit dem nächsten Zeichen aus dem Tastaturbuffer.
ESC ESC & 1	Tastatur sperren, Zeichen von der Tastatur können nur noch mit ESC ESC ' gelesen werden.
ESC ESC & 0	Tastatur freigeben, Zeichen von der Tastatur werden direkt an den Rechner gesendet (Normalzustand).

Programme, die dieses Verfahren benutzen sollen, müssen also am Anfang die Tastatur sperren (ESC ESC & 1), die Tastatur mit ESC ESC ' auslesen und vor der Rückkehr in das Betriebssystem die Tastatur wieder freigeben (ESC ESC & Ø). In Turbo-Pascal könnte das etwa folgendermaßen aussehen:

1. Es werden neue I/O-Funktionen geschrieben, die die Doppelescape-Sequenzen benutzen:

```
Function NewConSt: Boolean;
Begin
  Write (#27#27,'#');              (* GripS antwortet mit Status  *)
  NewConSt := Bios (2) And 4 = 4 (* Bit 2 abfragen               *)
End;

Function NewConIn: Char;
Begin
  Repeat Until NewConSt;          (* Warten, bis Taste gedrueckt   *)
  Write (#27#27,'''');            (* GripS antwortet mit Taste     *)
  NewConIn := Chr (Bios (2))      (* über Bios einlesen            *)
End;
```

2. Das Hauptprogramm wird folgendermaßen geändert:

```
Begin
  Write (#27#27,'&1');            (* Tastatur sperren              *)
  ConStPtr := Addr (NewConSt); (* neue Statusroutine NewConSt    *)
  ConInPtr := Addr (NewConIn); (* neue Tastaturroutine NewConIn *)
  ...
  ...
  Write (#27#27,'&Ø');            (* Tastatur freigeben            *)
End.
```

Die Funktionen bzw. Prozeduren KeyPressed und Read benutzen jetzt die neuen I/O-Treiber mit den Doppelescape-Sequenzen.

3. Alle Funktionen, die Doppelescape-Sequenzen benutzen, bei denen GripS ein oder mehrere Zeichen zurückschickt, dürfen zum Einlesen dieser Zeichen nicht Read benutzen, sondern müssen direkt das Bios aufrufen, da Read ja nur Zeichen von der Tastatur lesen kann.

zum Beispiel WhereX:

```
Function WhereX: Integer;       (* X-Koordinate des Cursors lesen *)
Var X, Y: Char;
Begin
  Write (#27,'?');              (* GripS antwortet mit Y X CR     *)
  Y := Chr (Bios (2));          (* Y-Koordinate lesen             *)
  X := Chr (Bios (2));          (* X-Koordinate lesen             *)
  Y := Chr (Bios (2));          (* CR lesen                       *)
  WhereX := Ord (X) - 31
End;
```

Sollten Sie Fragen oder Anregungen haben oder Fehler in der GripS-Software feststellen, schreiben Sie mir bitte oder rufen mich an:

Stefan Klöcker
Pontstr. 72
5100 Aachen
Tel.: Ø241/49748

```
;=========================================================
; Beispiel fuer ein selbstdefiniertes Hardcopyprogramm
;=========================================================
; Es wird ein Hardcopy-Programm fuer Epson-FX80 als User-
; Programm geladen und bei jedem Hardcopy-Aufruf aus-
; gefuehrt.
; Der FX80 dient hier nur als Beispiel, dieses Programm ist
; in GripS schon vorhanden. Fuer andere Drucker kann es
; aber als Anregung dienen.
;=========================================================
; Dieses Programm wird von CP/M aus aufgerufen.
; Das Hardcopy-Programm bleibt in GripS aktiv bis ein
; neues User-Programm geladen wird oder bis zum Reset.

Bdos    equ     5

        ; Laden als User-Programm vorbereiten

        ld      de,load
        ld      c,9
        call    Bdos

        ; Hardcopy-Programm hexadezimal senden

        ld      hl,usrprog
        ld      bc,usrprog_len
send_loop:
        ld      a,(hl)  ; Byte holen
        srl     a
        srl     a
        srl     a
        srl     a
        add     a,'0'
        call    send    ; oberes Nibble
        ld      a,(hl)
        and     0fh
        add     a,'0'
        call    send    ; unteres Nibble
        inc     hl
        dec     bc
        ld      a,b
        or      c
        jr      nz,send_loop

        ; Userprogramm starten

        ld      de,start
        ld      c,9
        call    Bdos
        jp      0       ; back in the U.S.S.R.

send:   push    hl
        push    bc
        ld      c,2
        ld      e,a
        call    Bdos
        pop     bc
        pop     hl
        ret

load:   defb    esc, esc, 'C5'  ; Zielkanal Userprogramm
```

```
        defb    esc, esc, 'P'   ; Zielkanal initialisieren
        defb    esc, esc, 'F'   ; Daten hex. zum Zielkanal
        defb    '$'

start:  defb    cr
        defb    'Hardcopyprogramm fuer FX80 geladen'
        defb    esc, esc, 'm '  ; Userprogramm starten
        defb    esc, esc, 'C0'  ; Zielkanal Centronics
        defb    '$'

;============================================================
; Speicherstellen im GripS-Ram

home    equ     4000h   ; Adr. linke obere Bildschirmecke } gelten ab
hardcopy equ    4017h   ; Sprung zur Hardcopy-Routine     } Version 1.12
zielsend equ    401Dh   ; Zeichen in A zum Zielkanal

; Ascii-Werte

cr      equ     0Dh             ; Carriage Return
ff      equ     0Ch             ; Form Feed
esc     equ     1Bh             ; Escape


usrprog:.phase 4300h            ; laeuft unter GripS auf 4300h
        ld      hl,hcprog       ; Hardcopy-Programm
        ld      (hardcopy+1),hl ; in GripS-Software einbinden
        ret                     ; Ready for take off.

;------------------------------------------------------------
; ESC ESC H oder Set-Up-'H' ruft diese Routine auf:
; (druckt das Bild um 90 Grad verdreht)
;------------------------------------------------------------

hcprog: push    hl              ; alle Register muessen
        push    de              ; gerettet werden
        push    bc
        push    af

        ld      hl,(home)
        ld      bc,8*96-8
        add     hl,bc           ; rechte obere Ecke

        ld      b,96            ; 96 Zeilen auf dem Drucker
next_Line:
        push    bc
        push    hl

        ld      a,esc           ; Grafikmodus Plottergrafik
        call    zielsend
        ld      a,'*'
        call    zielsend
        ld      a,5
        call    zielsend
        ld      a,low (35*8*2)
        call    zielsend
        ld      a,high (35*8*2)
        call    zielsend

        ld      c,35            ; 35 Bildschrimzeilen
        ld      de,8*96-8       ; Offset zur naechsten Zeile
```

```
next_Block:
        set     7,h             ; Video liegt oben
        ld      b,8             ; 8 Byte pro Block
next_Byte:
        ld      a,(hl)          ; Byte holen und als
        call    zielsend        ; Druckerspalte zum Zielkanal
        call    zielsend        ; nochmal fuer quadratisches Bild
        inc     hl
        djnz    next_Byte

        add     hl,de           ; naechste Zeile
        dec     c
        jr      nz,next_Block

        ld      a,cr            ; neue Zeile
        call    zielsend
        ld      a,esc           ; Vorschub 24/216": nahtlos
        call    zielsend
        ld      a,'J'
        call    zielsend
        ld      a,24
        call    zielsend

        pop     hl
        ld      bc,-8           ; naechste Spalte
        add     hl,bc
        pop     bc
        djnz    next_Line

        ld      a,ff            ; neue Seite
        call    zielsend

        pop     af
        pop     bc
        pop     de
        pop     hl

        ret

        .dephase

usrprog_len equ $-usrprog

        end
```

Auf Wunsch und gegen Berechnung programmiere ich auch ein Hardcopy-Programm für jeden anderen Drucker.

```
(*=================================================================*)
(*== GRAPH.P                                    (Klöcker, 85/11/05) ==*)
(*== Routinen zur Nachbildung der IBM-Routinen in Turbo-Pascal   ==*)
(*== (ab Version GripS 1.11)                                     ==*)
(*==                                                             ==*)
(*== Erweiterungen:                                             ==*)
(*== zusätzliche Farbe 2 = invertieren bei Plot, Draw und Circle ==*)
(*=================================================================*)

Const Esc        = #27;
      BW40       = 0;
      BW80       = 2;

Procedure TextMode (Mode: Integer);
Begin
  If Mode = BW40 Then Write (Esc,'Gp');
  If Mode = BW80 Then Write (Esc,'G0')
End;

Function WhereX: Integer;
Var X,Y: Char;
Begin
  Write (Esc,'?');
  Read (Kbd, Y, X, Y);
  WhereX := Ord (X) - 31
End;

Function WhereY: Integer;
Var X,Y: Char;
Begin
  Write (Esc,'?');
  Read (Kbd, Y, X, X);
  WhereY := Ord (Y) - 31
End;

Var HiResOff: Boolean;

Procedure GraphMode; Begin ClrScr; HiResOff := True End;

Procedure HiRes; Begin ClrScr; HiResOff := False End;

Const ColorCode: Array [0..2] Of Byte = (17, 19, 18);
      Tektr     = #20;
      PointPlot = #28;
      Vector    = #29;

Procedure Plot (X, Y, Color: Integer);
Begin
  If Y < 280 Then
    Begin
      If HiResOff Then X := X Shl 1;
      Y := (279 - Y) Shl 1;
      Write (Tektr, Chr (ColorCode [Color mod 3]),
             Tektr, PointPlot,
             Tektr, Chr (Y Shr 5 + 32), Chr (Y And 31 + 96),
                    Chr (X Shr 5 + 32), Chr (X And 31 + 64))
    End
End;
```

```
Procedure Draw (X1, Y1, X2, Y2, Color: Integer);
Begin
  If (Y1 < 280) And (Y2 < 280) Then
    Begin
      If HiResOff Then Begin X1 := X1 Shl 1; X2 := X2 Shl 1 End;
      Y1 := (279 - Y1) Shl 1; Y2 := (279 - Y2) Shl 1;
      Write (Tektr, Chr (ColorCode [Color mod 3]),
             Tektr, Vector,
             Tektr, Chr (Y1 Shr 5 + 32), Chr (Y1 And 31 + 96),
                    Chr (X1 Shr 5 + 32), Chr (X1 And 31 + 64),
             Tektr, Chr (Y2 Shr 5 + 32), Chr (Y2 And 31 + 96),
                    Chr (X2 Shr 5 + 32), Chr (X2 And 31 + 64))
    End
End;

Procedure Circle (X, Y, Radius, Color: Integer);
Begin
  If Y < 280 Then
    Begin
      If HiResOff Then Begin X := X Shl 1; Radius := Radius Shl 1 End;
      Y := (279 - Y) Shl 1;
      Write (Tektr, Chr (ColorCode [Color mod 3]),
             Tektr, Vector,
             Tektr, Chr (Y Shr 5 + 32), Chr (Y And 31 + 96),
                    Chr (X Shr 5 + 32), Chr (X And 31 + 64),
             Tektr, Esc, 'K',
             Tektr, Chr (Radius Shr 5 + 32), Chr (Radius And 31 + 64))
    End
End;

Function GetDotColor (X, Y: Integer): Integer;
Var P: Char;
Begin
  GetDotColor := 0;
  If Y < 280 Then
    Begin
      If HiResOff Then X := X Shl 1;
      Y := (279 - Y) Shl 1;
      Write (Tektr, Vector,
             Tektr, Chr (Y Shr 5 + 32), Chr (Y And 31 + 96),
                    Chr (X Shr 5 + 32), Chr (X And 31 + 64),
             Tektr, Esc, 'P');
      Read (Kbd,  P);  If P = '.' Then GetDotColor := 1
    End
End;

Procedure Sound (f: Integer);
Var fi: Integer;
Begin
  If f > 24 Then
    Begin
      If f > 301 Then fi := Round (38400.0 / f) Shl 1
                 Else fi := Round (3072.0  / f) Shl 1 + 1;
      Write (Esc,Esc,'C7', Esc,Esc,'d', Chr (fi), Esc,Esc,'d', Chr(96))
    End
End;

Procedure NoSound;
Begin
  Write (Esc,Esc,'C7', Esc,Esc,'d', Chr(0))
End;
```

```
Program Sierpinski;

{ Zeichne die Sierpinski-Kurven der Ordnung 1 bis N
  Algorithmus aus:
  N. Wirth, Algorithmen und Datenstrukturen, Teubner. S. 189 ff. }

{$A- rekursive Umgebung }
{$I GRAPH.P }

Const N = 4; H0 = 256;

Var I,H,X,Y,X0,Y0: Integer;

Procedure B (I:Integer); Forward;
Procedure C (I:Integer); Forward;
Procedure D (I:Integer); Forward;

Procedure A (I:Integer);
Begin If I > 0 Then
      Begin A (I-1); X := X+H; Y := Y-H; Draw (X-H,Y+H,X,Y,1);
            B (I-1); X := X+2*H; Draw (X-2*H,Y,X,Y,1);
            D (I-1); X := X+H; Y := Y+H; Draw (X-H,Y-H,X,Y,1);
            A(I-1)
      End
End;
Procedure B;
Begin If I > 0 Then
      Begin B (I-1); X := X-H; Y := Y-H; Draw (X+H,Y+H,X,Y,1);
            C (I-1); Y := Y-2*H; Draw (X,Y+2*H,X,Y,1);
            A (I-1); X := X+H; Y:=Y-H; Draw (X-H,Y+H,X,Y,1);
            B (I-1)
      End
End;
Procedure C;
Begin If I > 0 Then
      Begin C (I-1); X := X-H; Y := Y+H; Draw (X+H,Y-H,X,Y,1);
            D (I-1); X := X-2*H; Draw (X+2*H,Y,X,Y,1);
            B (I-1); X := X-H; Y := Y-H; Draw (X+H,Y+H,X,Y,1);
            C (I-1)
      End
End;
Procedure D;
Begin If I > 0 Then
      Begin D (I-1); X := X+H; Y := Y+H; Draw (X-H,Y-H,X,Y,1);
            A (I-1); Y := Y+2*H; Draw (X,Y-2*H,X,Y,1);
            C (I-1); X := X-H; Y := Y+H; Draw (X+H,Y-H,X,Y,1);
            D (I-1)
      End
End;

Begin GraphMode;
  I := 0; H := H0 Div 4; X0 := 2*H; Y0 := 3*H;
  Repeat I := I+1; X0 := X0-H;
     H := H Div 2; Y0 := Y0+H;
     X := X0; Y := Y0;
     A (I); X := X+H; Y := Y-H; Draw (X-H,Y+H,X,Y,1);
     B (I); X := X-H; Y := Y-H; Draw (X+H,Y+H,X,Y,1);
     C (I); X := X-H; Y := Y+H; Draw (X+H,Y-H,X,Y,1);
     D (I); X := X+H; Y := Y+H; Draw (X-H,Y-H,X,Y,1);
  Until I =N
End.
```

Standard-Zeichensatz in programmierbaren Zeichensatz einkopieren

ASCII (def)	ESC ESC K 0
Deutsch	ESC ESC K 1
Skandinavisch	ESC ESC K 2
Dänisch	ESC ESC K 3
Britisch	ESC ESC K 4
Französisch	ESC ESC K 5
International	ESC ESC K 6
Griechisch / Sonderzeichen	ESC ESC K 7

Wenn nur einzelne Zeichen eines bestimmten Zeichensatzes geändert werden sollen, kann der programmierbare Zeichensatz dadurch vorbelegt werden.

Lichtgriffel-Position abfragen ESC ESC L

GripS antwortet mit 3 Bytes *x*, *z* und *s*.

Dabei ist *x* ein Byte, dessen Bits folgende Bedeutung haben:

x = 0.0.n/p.err.sen/a2/a1/a0

n/p	= Lichtgriffel nicht ausgelöst	
err	= ERROR-Eingang	
sen	= SENSE-Eingang	0 1
/a2../a0	= Lichtgriffel-Adressbits innerhalb eines 8x8-Pixelblocks	2 3
	Jeder Block ist dazu in 8 Unterblöcke aufgeteilt, dessen	4 5
	Nummer */a2../a0* angeben.	6 7

z und *s* sind ASCII-Zeichen, die die Zeile+32 und die Spalte+32 angeben.

Die Position ist nur gültig, wenn der Lichtgriffel ausgelöst hat und der Bildschirm nicht gescrollt wurde. Je nach Ansprechgeschwindigkeit des Lichtgriffels kann die Position in horizontaler Richtung etwas verschoben sein. Dies muß durch das Anwendungsprogramm korrigiert werden. Der SENSE-Eingang wird nur korrekt wiedergegeben, wenn der Lichtgriffel ausgelöst hat, der ERROR-Eingang ist davon unabhängig. GripS ist sofort wieder frei für ein neues Auslösen.

Userprogramm starten (Hexformat) ESC ESC M*aa*
Userprogramm starten (binär) ESC ESC m*x*

Der Wert *aa* bzw. *x* wird in das A-Register geladen, dann das Userprogramm gestartet. Beim Kommando im Hexformat wird der Wert des A-Registers nach Rückkehr des Userprogramms als zweistelliger Hexwert gesendet.

Direktes Portauslesen (Hexformat) ESC ESC N*pp*
Direktes Portauslesen (binär) ESC ESC n*p*

Das Port *pp* bzw. *p* wird ausgelesen, GripS antwortet mit dem gelesenen Wert *bb* bzw. *b*.

Direkte Portausgabe (Hexformat) ESC ESC O *pp bb*
Direkte Portausgabe (binär) ESC ESC o *p b*

Das Byte *bb* bzw. *b* wird auf das Port *pp* bzw. *p* ausgegeben.

Monitor-Testbild einschalten ESC ESC Q

Das Testbild beinhaltet Einstellmarken und die Zeichensätze.

Bildschirm und alle Puffer löschen ESC ESC R

Alle Funktionen auf die Default-Werte zurücksetzen, einzig die Uhr läuft weiter.

Statuszeile einschalten ESC ESC S
Statuszeile ausschalten ESC ESC s

Uhr stellen und starten ESC ESC U *h m s*
Uhr abfragen ESC ESC u

Bei "Uhr abfragen" antwortet GripS mit *h m s*.

Dabei sind *h*, *m*, *s* ASCII-Zeichen mit folgender Bedeutung:

h = Stunde (BCD-Format) + 32
m = Minute (BCD-Format) + 32
s = Sekunde (BCD-Format) + 32

Beispiel: ESC ESC U C 7 P stellt 23:17:30 Uhr ein. Die Uhr beginnt zu laufen, sobald das letzte Zeichen übertragen wurde. Das BCD-Format ist das gleiche, wie es von CP/M 3.0 für die BDOS-TIME-Funktion verwendet wird.

Datum stellen ESC ESC V *dddd*
Datum abfragen ESC ESC v

Dabei gibt *dddd* die Anzahl der Tage seit dem 1. Januar 1978 im Hexformat an bzw. GripS antwortet mit diesem Wert.

Beispiel: ESC ESC V 0 0 0 1 stellt den 1.1.1978 ein, ESC ESC V 0 A B 6 stellt den 4.7.1985 ein. Auch CP/M 3.0 verwendet dieses Datum-Format für die BDOS-TIME-Funktion.

Wecker stellen ESC ESC W *h m*

Die Weckzeit (nur Stunden und Minuten) wird eingestellt, *h* und *m* wie bei "Uhr stellen". Bei Erreichen der eingestellten Weckzeit spielt GripS eine kleine Melodie (Greensleeves).

Beispiel: ESC ESC W 7 8 stellt die Weckzeit auf 17:18.

TeleVideo–Modus

Cursorbewegung

Cursor ab ↑ J
↑ V

↑ J bewirkt auf der untersten Zeile ein Scrollen, ↑ V nicht.

Cursor auf ↑ K
ESC j

ESC j bewirkt auf der obersten Zeile ein Scrollen, ↑ K nicht.

Cursor links ↑ H

Am Anfang einer Zeile wird der Cursor auf das Ende der vorherigen Zeile gesetzt, am Anfang des Bildschirms bleibt der Cursor stehen.

Cursor rechts ↑ I

Am Ende der Zeile wird der Cursor auf den Anfang der nächsten Zeile gesetzt, am Ende des Bildschirms wird zusätzlich gescrollt.

Anmerkung: Der Code ↑ I bewirkt beim TeleVideo ein Tabulieren, bei GripS jedoch die Bewegung des Cursors um ein Zeichen nach rechts. Der entsprechende Code bei TeleVideo ist ↑ L, dieser wird hier jedoch zum Bildschirmlöschen verwendet.

Cursor an den linken Zeilenrand CR

Cursor auf eine neue Zeile ↑ _

Wirkt wie CR ↑ J.

Cursor in die Home–Position ↑ ^

Cursoradressierung ESC = *z s*

Bewegt den Cursor auf die Position (z,s), wobei *z* das ASCII–Zeichen (Zeilennummer+32) und *s* das ASCII–Zeichen (Spaltennummer+32) ist. Die Home–Position ist (0,0).

Cursorposition lesen ESC ?

GripS antwortet mit der Position des Cursors *z s* CR, wobei *z* und *s* wie bei der Cursoradressierung definiert sind.

Attribute

Zeichenattribute

ESC G *x*

Dabei ist *x* ein Byte, dessen Bits folgende Bedeutung haben:

x = 0.b.s1.s0.u.i.d.v

b = Breitschrift
u = Unterstreichen
i = Invertieren
d = Durchstreichen
v = unsichtbare Schrift
s1s0 = 00: Index tief
s1s0 = 01: Kleinschrift
s1s0 = 10: Index hoch
s1s0 = 11: normale Schrift

Für *b* = 0 und *s1* = *s0* = 1 gilt dann:

Normale Schrift	ESC G 0
Unsichtbare Schrift	ESC G 1
Durchgestrichene normale Schrift	ESC G 2
Invertierte Schrift	ESC G 4
Invertierte durchgestrichene Schrift	ESC G 6
Unterstrichene Schrift	ESC G 8
Durchgestrichene unterstrichene Schrift	ESC G :
Invertierte unterstrichene Schrift	ESC G <
Invertierte durchgestrichene unterstrichene Schrift	ESC G >

Anmerkung: Bei TeleVideo wird das Bit *d* zum Blinken verwendet.
Dies ist bei GripS von der Hardware her nicht zu leisten.

Invertieren ein

ESC)

Invertieren aus

ESC (

Anmerkung: Diese Codes werden bei TeleVideo für halbe Helligkeit verwendet.
Dies ist bei GripS von der Hardware her nicht zu leisten.

Cursorattribute

Unsichtbarer Cursor	ESC . 0
Blinkender Block	ESC . 1
Stehender Block	ESC . 2
Blinkende Linie	ESC . 3
Stehende Linie	ESC . 4

Editierkommandos

Zeile einfügen ESC E

Die Zeile, auf der der Cursor steht, und alle folgenden Zeilen werden eine Zeile nach unten geschoben, die letzte Zeile verschwindet. Es entsteht eine leere Zeile wo der Cursor stand, dieser steht am Anfang dieser neuen Zeile.

Zeile löschen ESC R

Die Zeile, auf der der Cursor steht, wird gelöscht, alle folgenden Zeilen werden nach oben nachgezogen. Der Cursor steht am Anfang der Zeile, die letzte Zeile ist leer.

Zeichen einfügen ESC Q

Schiebt eine Leerstelle an der Cursorposition ein, das letzte Zeichen der Zeile geht verloren.

Zeichen löschen ESC W

Löscht das Zeichen unter dem Cursor, alle weiteren Zeichen der Zeile rücken nach, das letzte Zeichen der Zeile ist leer.

Löschkommandos

Bildschirm löschen ↑ L
↑ Z
ESC *
ESC :
ESC ,
ESC ;

Der Bildschirm wird vollständig gelöscht, der Cursor steht auf Home–Position. ESC +

Anmerkung: Der Code ↑ L bewirkt beim TeleVideo eine Cursorbewegung nach rechts.

Bildschirm ab Cursorposition löschen ESC Y
ESC y

Der Bildschirm wird von der Cursorposition bis zum Ende gelöscht, der Cursor verändert seine Position nicht.

Zeile ab Cursorposition löschen ESC T
ESC t

Die Zeile wird von der Cursorposition bis zum Ende gelöscht, der Cursor verändert seine Position nicht.

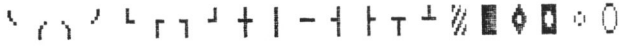
sonstige Kommandos

Glockensignal (piepsen) ↑ G

Grafik–Modus einschalten ESC $
Grafik–Modus ausschalten ESC %

Ab dem Einschalten des Grafikmodus werden statt der Buchstaben spezielle Strichgrafikzeichen dargestellt, nach dem Ausschalten wieder Buchstaben. Bereits angezeigte Zeichen verändern ihre Darstellung nicht! Folgende Zeichen können dargestellt werden:

A B C D E F G H I J K L M N O P Q R S T U



Anmerkung: Die Zeichen ab 'P' sind im TeleVideo nicht vorhanden. Sie können jedoch z. B. für Platinenlayoutprogramme verwendet werden.

Meldungszeile beschreiben ESC f
Meldungszeile anzeigen ESC g
Meldungszeile ausschalten ESC h

Nach ESC f wird in die Meldungszeile geschrieben bis zu einem CR.
Die Meldungszeile ist standardmäßig ausgeschaltet.

temporär auf Tektronix–Modus umschalten ↑ T

Es wird für ein Tektronix–Kommando (ein Kontrollzeichen oder eine vollständige ESC–Sequenz) bzw. eine Koordinatensequenz, die mit */x* endet, in den Tektronix–Modus umgeschaltet, danach ist GripS wieder im TeleVideo–Modus.

Anmerkung: Dieser Code hat beim TeleVideo eine andere Bedeutung.

Terminalidentifikation senden ESC M

Es wird der Text G r i p S 1 CR gesendet. Ein Programm kann daran den Terminaltyp erkennen.

Tektronix-Modus

GripS emuliert ein Tektronix-Grafikterminal T4010 (teilweise auch T4014). Bedingt durch die geringe Zeilenzahl von 280 auf einem normalen Monitor ohne Zeilensprung-verfahren ist der darstellbare Bereich allerdings auf 768 x 560 Punkte beschränkt. Dabei ist jede ungerade Zeile unsichtbar (ausgenommen ist der Vectormode, da werden je zwei Zeilen auf eine Zeile abgebildet). GripS verhält sich aber so, als sei der Bildschrim 1024 x 1024 Punkte groß, es ist eben nur ein Ausschnitt zu sehen.
Alle Koordinaten beziehen sich auf den Ursprung, der unten links liegt.

Das Tektronix-Terminal kennt verschiedene Modes (Alphamode, Vectormode, Pointplot-Mode, Incremental-Mode und Gin-Mode). Letzterer ist nicht implementiert. Die Kommandos beziehen sich auf die interne Grafikcursor-Position (der Cursor ist nur im Alphamode sichtbar), die durch ASCII-Zeichen eingestellt werden kann, solange sich das Terminal im Vector- oder Pointplot-Mode befindet. Im Alphamode werden die Zeichen übereinander geschrieben (wie bei einem Drucker), das darunterliegende Zeichen bleibt also sichtbar.

Es kann jederzeit in den TVI-Modus zurückgeschaltet und von dort aus wieder der Tektronix-Modus aufgerufen werden, ohne daß sich der Bildschirminhalt oder eingestellte Parameter ändern. Nur die Meldungszeile wird abgeschaltet, ihr Inhalt geht aber nicht verloren.

Alphamode einschalten ↑ _

Es erscheint ein Grafikcursor, der bei jedem Zeichen um 8 Punkte nach rechts rückt.

Alphamode einschalten, Cursor an Zeilenanfang ↑ M

Wie oben, jedoch steht der Cursor auf x-Koordinate 0.

Bildschirm löschen, Alphamode einschalten ESC ↑ L

Der Bildschirm wird gelöscht, der Alphamode eingeschaltet und der Cursor steht oben links.

Vectormode einschalten ↑]

Die erste folgende Koordinate ist Anfangskoordinate, d. h., es wird dorthin noch kein Vektor gezeichnet. Dann wird von jeder Koordinate zur nächsten ein Vektor gezeichnet.

Vektor doch zeichnen ↑ G

Wird dieses Kommando im Vektormode vor der ersten Koordinate gegeben, dann wird doch ein Vektor zur ersten Koordinate gezeichnet. Das ist sinnvoll, wenn der Vectormode verlassen und mit "Vectormode einschalten" wieder betreten wurde.
Es kann dann direkt weitergezeichnet werden.

Pointplot–Mode einschalten ↑ \

Es werden auf den folgenden Koordinaten Punkte gezeichnet.

Incremental–Mode einschalten ↑ ^

Es kann wie mit einem mechanischen Plotter herumgefahren werden.

Zeichne in der Hintergrundfarbe (löschen) ↑ Q
Zeichne durch Invertieren ↑ R
Zeichne in der Vordergrundfarbe (default) ↑ S

Zeichne durchgehende Linien ESC
Zeichne gepunktete Linien ESC a
Zeichne Strichpunkt–Linien ESC b
Zeichne kurz gestrichelte Linien ESC c
Zeichne lang gestrichelte Linien ESC d

Diese Kommandos beziehen sich auf das Aussehen der Vektoren.

Fläche füllen ESC F

Die Fläche um die aktuelle Cursorposition, (die vollständig umrandet sein muß, der Bildschrimrand gilt auch als Rand), wird ausgefüllt.

Zeichne einen Kreis ESC K*hr lr*

Zeichne um die augenblickliche Cursorposition herum einen Kreis mit Radius r. Dabei ist *hr* ein ASCII–Zeichen mit der Nummer (Radius div 32 + 32) und *lr* ein ASCII–Zeichen mit der Nummer (Radius mod 32 + 64). Zeichnen durch Invertieren funktioniert leider nicht besonders gut.

Pixel abfragen ESC P

Wenn der Punkt auf der aktuellen Cursorposition hell ist, antwortet GripS mit . , sonst mit ␣.

Schreibrichtung wählen (Alphamodus, die Zeichen erscheinen gedreht)

von links nach rechts ESC R 0
von unten nach oben ESC R 1
von rechts nach links ESC R 2
von oben nach unten ESC R 3

Cursor links ↑ H
Cursor rechts ↑ I
Cursor ab ↑ J
Cursor auf ↑ K

Diese Kommandos versetzen im Alphamode den Cursor um eine Zeichenbreite bzw. –höhe.

Koordinaten

Eine vollständige Koordinate besteht aus je 10 Bit für X- und Y-Richtung. Sie wird durch bis zu 4 lesbare ASCII-Zeichen übertragen, die jeweils 5 Bit zur Koordinate beitragen. Dabei bedeuten:

hy	=	␣ . . ?	(20h..3Fh): obere 5 Bit der Y-Koordinate = Y div 32 + 32
/y	=	` . . DEL	(60h..7Fh): untere 5 Bit der Y-Koordinate = Y mod 32 + 96
hx	=	␣ . . ?	(20h..3Fh): obere 5 Bit der X-Koordinate = Y div 32 + 32
/x	=	@ . . _	(40h..5Fh): untere 5 Bit der X-Koordinate = Y div 32 + 64

Es müssen nicht alle Koordinatenteile übertragen werden, wenn sich nur einige ändern. Sobald */x* übertragen wurde, wird die Koordinate als vollständig aufgefaßt. Folgende Formate sind erlaubt:

hy /y hx /x	Koordinate völlig neu
hy /x	ly und hx beibehalten
/y hx /x	hy beibehalten
/y /x	hy und hx beibehalten
/x	Y und hx beibehalten

Incremental-Mode

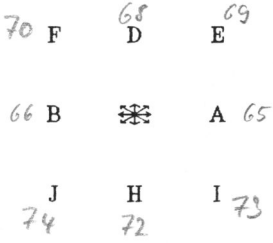
Es wird von der letzten Grafikkoordinate an in 1-Punkt-Schritten weitergezeichnet. Der "Stift" kann gehoben und gesenkt werden (auch Invertieren und Löschen wurde implementiert). In Y-Richtung ist jede ungerade Zeile unsichtbar.

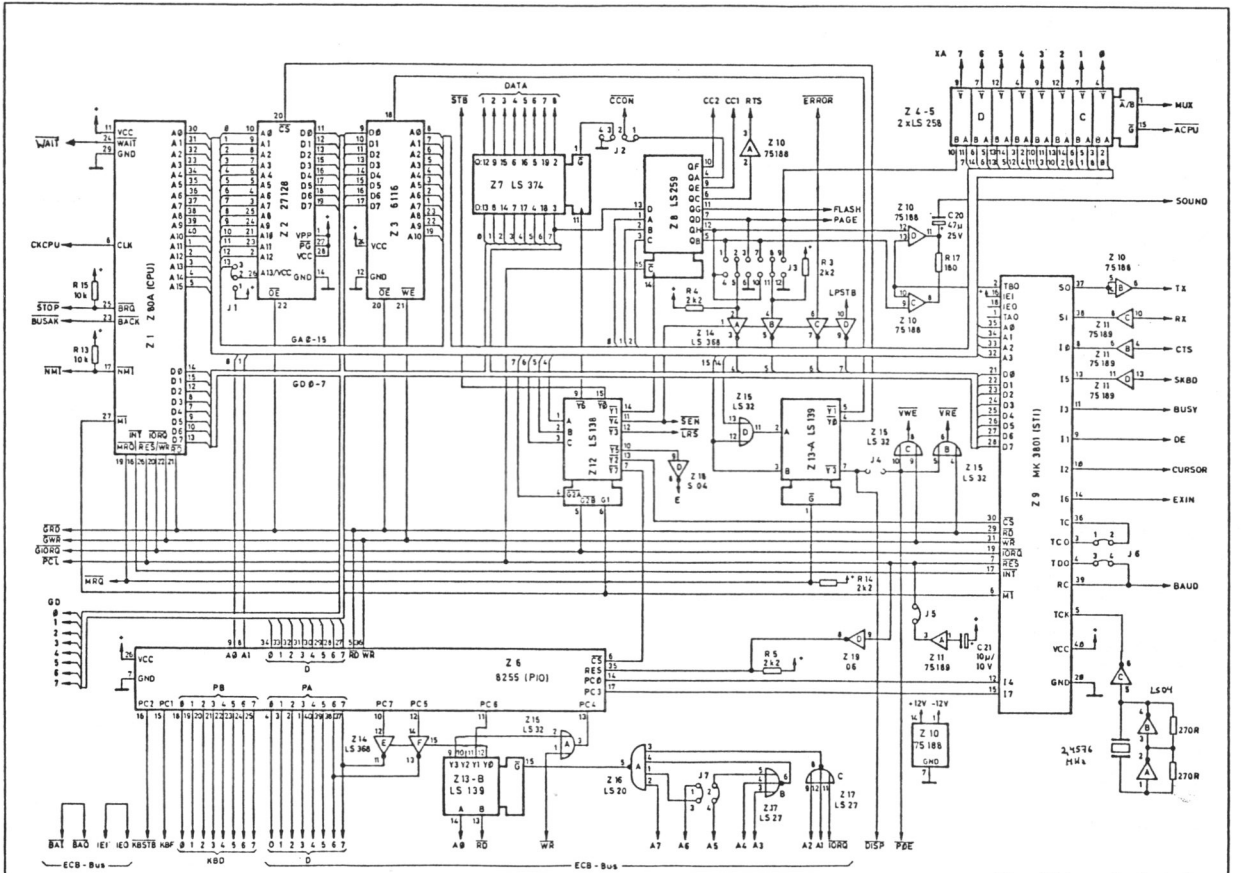
Zunächst muß die "Stiftinformation" gegeben werden, diese kann auch jederzeit geändert werden:

nicht zeichnen ("Stift oben")	]
zeichnen ("Stift abgesenkt")	P
löschen ("radieren")	Q
invertieren	R

Die beiden letzten Kommandos gibt es beim Tektronix nicht.

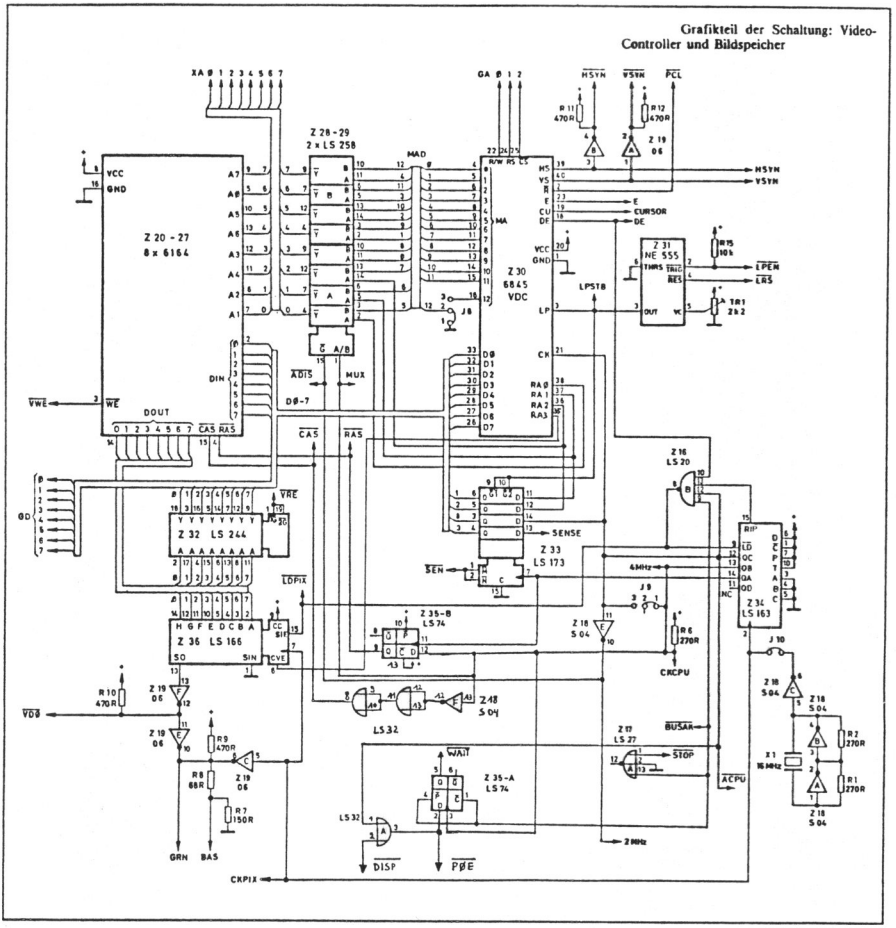
Dann kann der "Stift" mit folgenden Zeichen bewegt werden:

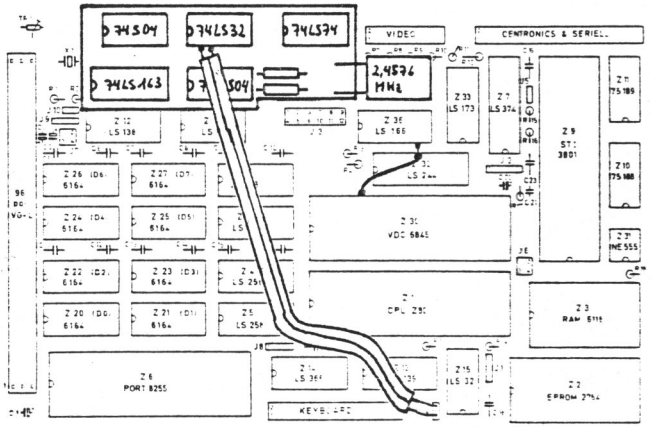




Dieser Schaltungsteil zeigt den Prozessorbaustein und die Schnittstellen

Grafikteil der Schaltung: Video-Controller und Bildspeicher





Bestückungsplan

Zeichensatz ASCII

```
 !"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{¦}~▒
```

Zeichensatz Deutsch

```
 !"#$%&'()*+,-./0123456789:;<=>?
§ABCDEFGHIJKLMNOPQRSTUVWXYZÄÖÜ^_
`abcdefghijklmnopqrstuvwxyzäöüß▒
```

Zeichensatz Skandinavisch

```
 !"§¤%&'()*+,-./0123456789:;<=>?
ÉABCDEFGHIJKLMNOPQRSTUVWXYZÄÖÅ^_
éabcdefghijklmnopqrstuvwxyzäöåü▒
```

Zeichensatz Dänisch

```
 !"#¤%&'()*+,-./0123456789:;<=>?
ÉABCDEFGHIJKLMNOPQRSTUVWXYZÆØÅÜ_
éabcdefghijklmnopqrstuvwxyzæøåü▒
```

Zeichensatz Britisch

```
 !"£#%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{¦}~▒
```

Zeichensatz Französisch

```
 !"£#%&'()*+,-./0123456789:;<=>?
àABCDEFGHIJKLMNOPQRSTUVWXYZ°ç§^_
`abcdefghijklmnopqrstuvwxyzéùè¨▒
```

Zeichensatz International

```
 !"#¤%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{¦}~▒
```

Zeichensatz Griechisch / mathematisch

```
|{}[[]]||(||]|√ ┌-∮∫-⊂⊃½¼∞∝≈≃·×±≤≠≥÷
∃⊗⊖Ψ‡ЄNΛ*†ΠΠΩìòöℵΓΘΣÖΞΕΔςΤ♀♂♦♥♣♠
∀αβψφεⁿλητπκωμνορ℘∂στξεδχυζ↑↓←→▒
```