SINIX-Schnittstellen
Benutzerhandbuch

Ausgabe April 1987 (SINIX V1.2, V2.0)

Bestell-Nr. U2300-J-Z95-2
Printed in the Federal Republic of Germany
10000 AG 4875.(12500)

Siemens Aktiengesellschaft

Vorwort

Dieses Schnittstellenhandbuch richtet sich an geübte SINIX-Anwender und C-Programmierer.
Wir möchten Ihnen mit diesem Buch Hinweise geben, wie Sie die Möglichkeiten Ihres PC-X/PC-X10 oder PC-MX2 besser nutzen können.

Soweit keine anderen Aussagen gemacht werden, beziehen sich die Angaben und Beschreibungen auf:

- SINIX Version 1.2A und PC-X/PC-X10
- SINIX Version 2.0, PC-MX2 und die Bedieneinheit 97801

Auf der mitgelieferten Diskette 'SSHB' finden Sie Quellprogramme, Include-Dateien und Objektmodule, die Ihnen bei der Lösung kniffliger Aufgaben helfen.

Das Inhaltsverzeichnis der Diskette und die Quellprogramme sind in Anhang ausgedruckt.

Achtung

Benutzen Sie unter den Kennungen root und admin keine Kommandos, deren Wirkungsweise Sie nicht kennen. Die Konsistenz Ihres SINIX-Systems kann dabei zerstört werden.

Wichtige Hinweise

- Siemens PC entsprechen bei bestimmungsgemäßem Gebrauch der allgemeinen Genehmigung nach Postverfügung Nr. 1115.
  Alle an die PC angeschlossenen Geräte müssen ebenfalls die oben genannten Bedingungen erfüllen.
- Da die in diesem Benutzerhandbuch beschriebenen Funktionen über den für das Produkt vereinbarten Funktionsumfang hinausgehen, kann keine Haftung für die gegebenen Informationen gewährt werden.

Eine Bitte an Sie

Keine erklärende Dokumentation kann perfekt sein. Eine Dokumentation lebt. Sie lebt auch von Ihren Anregungen, Ideen oder Verbesserungsvorschlägen.
Helfen Sie uns, indem Sie uns Ihre Stolpersteine mitteilen, damit wir sie aus dem Weg räumen können.

Manualredaktion K D ST QM2
Otto-Hahn-Ring 6, 8 München 83

Inhalt

1 Bedieneinheit 97801-302/-303

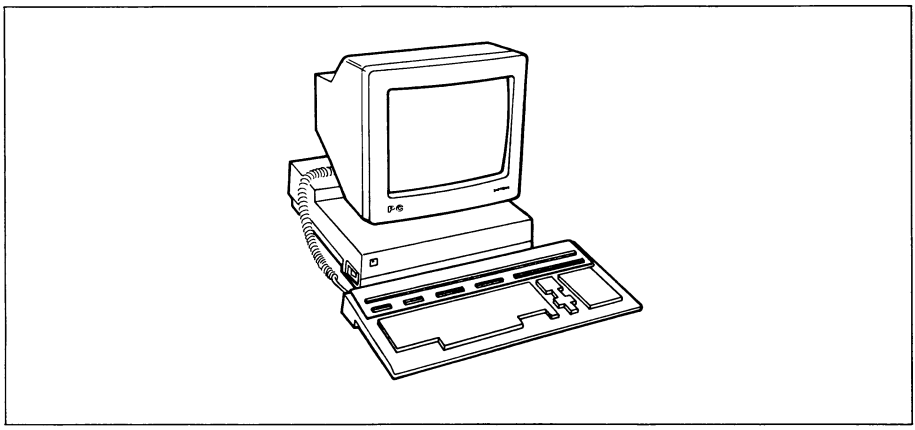


Bild 1-1 Bedieneinheit 97801

Die Bedieneinheit für SINIX-Mehrplatzsysteme besteht aus Tastatur, Bildschirm und Steuereinheit.

Der Bildschirm kann 25 x 80 Zeichen darstellen. Für jedes einzelne Zeichen kann die Darstellungsart (normal, invers, halbhell, blinken, unterstrichen) abgespeichert werden. Mit einem Regler an der Unterkante des Bildschirms kann die Bildhelligkeit eingestellt werden.
Die Helligkeit der Halbhelldarstellung wird an der Rückseite der Bedieneinheit eingestellt.
Die Bedieneinheit arbeitet im 'Reflected Copy Mode', d.h. alle eingegebenen Daten werden nicht sofort am Bildschirm angezeigt, sondern zum PC gesendet und wenn die Software es für erforderlich hält, zum Bildschirm zurückübertragen. So wird z.B. ein eingegebenes Kennwort nicht sichtbar, da es nicht zum Bildschirm zurückübertragen wird.

Die Bedieneinheit 97801 besteht aus:

- Bildschirm und Steuereinheit 97801-302/-303
- Tastatur

Hinweis

Die Bedieneinheit kann am **PC-MX** nur mit den Tastaturbelegungen **deutsch** und **international** betrieben werden, da der PC-MX **kein** Tastaturladeprogramm besitzt (siehe Kapitel 8 Tastaturbelegung).

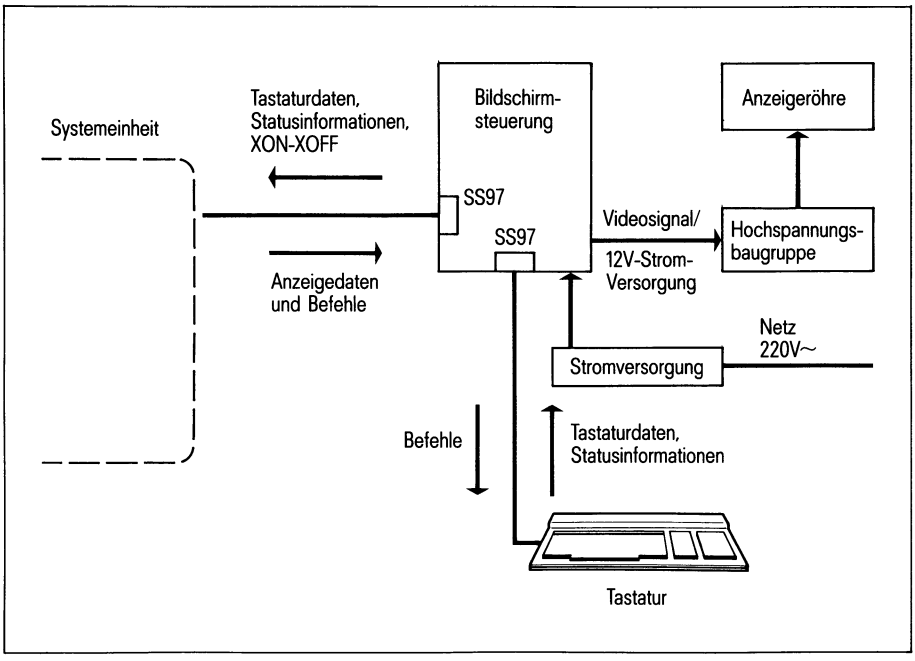


Bild 1-2 Funktionsübersicht Bedieneinheit

1.1 Steuereinheit

Die Steuereinheit enthält neben der Bildschirmsteuerung die Stromversorgung der Bedieneinheit. Zusätzlich ist in der Steuereinheit 97801-303 eine AFP-Datenübertragungseinrichtung eingebaut.

1.1.1 Stromversorgung

Die Stromversorgung liefert alle zum Betreiben der Bedieneinheit erforderlichen Spannungen. Die Ablenkspannung (17000 Volt) für die Bildröhre wird im Monitor erzeugt.

Achtung

Überlassen Sie Reparaturen grundsätzlich dem Service!

1.1.2 Bildschirmsteuerung

Die Bildschirmsteuerung ermöglicht

- das stellenrichtige Einschreiben der vom PC übertragenen Daten in den Bildwiederholspeicher,
- das Ausführen von Befehlen
  (z.B. Cursor positionieren, das Löschen usw.),
- das Übertragen der von der Tastatur eingegebenen Daten zum PC.

Die im Bildwiederholspeicher eingeschriebenen Daten werden mit einer Wiederholrate von 66 Hz am Bildschirm angezeigt.

1.2 Tastatur

Jedes Betätigen einer Taste erzeugt einen Platzcode, der zur Bildschirmsteuerung übertragen wird. Hier wird der Platzcodes entsprechend der geladenen Tastaturbelegungstabelle in ASCII-Zeichen umgewandelt.

Hinweis

Die Firmware (Programm und Zeichenspeicher) in Steuereinheit und Tastatur ist **immer** gleich, unabhängig von den länderspezifischen Tastaturbelegungen.

Tastaturvarianten (die Nummer finden sie an der Tastaturunterseite), wahlweise mit folgenden Belegungen:

- international 97801-131
- international und deutsch 97801-132
- Alle anderen Tastaturbelegungen werden durch Laden der Tastaturbelegungstabelle und Austausch der Tastenkappen der internationalen Tastatur festgelegt.

Das Laden des Zeichensatzes im Bildschirm entsprechend der verwendeten Tastatur erfolgt immer bei Ausgabe des Begrüßungsbildschirmes. Welcher Zeichensatz geladen wird, ist in der Datei /etc/termcap für jede Bedieneinheit hinterlegt (siehe 8.2.1: Beispiel 1).

1.2.1 Tastaturvarianten

Die Tastatur der Bedieneinheit 97801 hat wahlweise folgende Belegungen:

- international und deutsch 97801-132
- international 87801-131
- Die internationale Tastatur kann mit folgenden Tastenkappensätzen auf nationale/internationale Tastaturbelegung umgerüstet werden.
  - international und schwedisch 97801-144
  - international und dänisch 97801-145
  - international und französisch 97801-146
  - international und belgisch AZERTY 97801-147
  - international und spanisch 97801-149
  - international und italienisch 97801-150
  - international und britisch 97801-153
  - international und norwegisch 97801-154

2 Schnittstellen des PC-X10

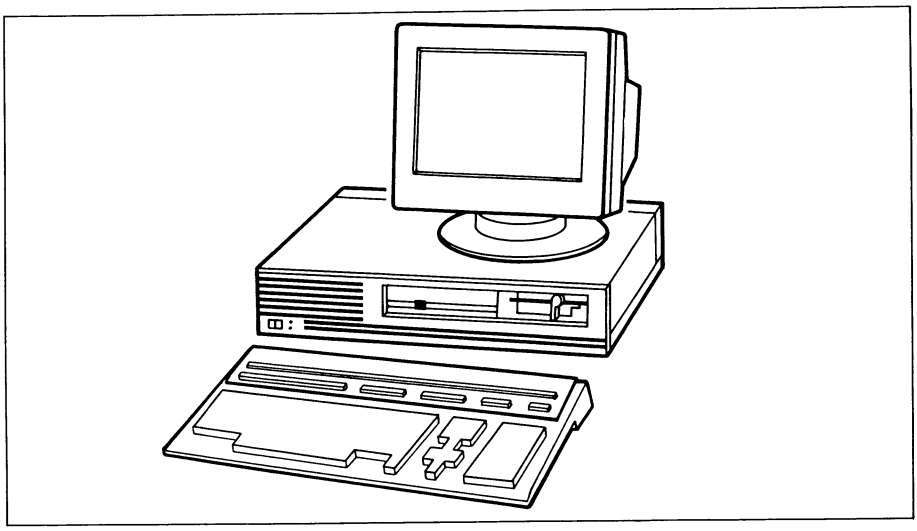


Bild 2-1 Personal Computer PC-X10 (9781)

Aufbau und Arbeitsweise

Das Kernstück des PC-X10 ist die Systemeinheit. Sie besteht aus:

- dem Systemboard,
- der Bildschirmsteuerung für Text oder Grafik,
- der Stromversorgung,
- einem Diskettenlaufwerk,
- einer Festplatte mit Controller,
- dem Frontplatte,
- den Einbauplätzen für Erweiterungsflachbaugruppen.

Die Bildschirm-Steuerung ist auf den Erweiterungsbus des Systemflachbaugruppe aufgesteckt.

Die Textversion der Bildschirm-Steuerung bildet mit Tastastur und Bildschirm die Bedieneinheit des PC-X10.
Sie steuert und überträgt über den internen Erweiterungsbus die Ein- und Ausgaben zwischen Betriebssystem und Tastatur bzw. Bildschirm.
Dabei wird eine PC-Bedieneinheit vom Typ 97801 nachgebildet.

Die Grafikversion der Bildschirm-Steuerung bildet zusammen mit Tastastur, Bildschirm und Maus die Bedieneinheit des PC-X10.
Soweit möglich wird dabei eine PC-Bedieneinheit vom Typ 97801 nachgebildet.

Tastatur, Bildschirm und bei der Grafikversion die Maus werden über je einen 9-poligen Stecker vom Typ AMPLIMITE HD20 oder ähnlich am Anschlußfeld der Systemeinheit angeschlossen.

Die Nachbildung der Bedieneinheit arbeitet im 'Reflected Copy Mode', d.h. alle eingegebenen Daten werden nicht sofort am Bildschirm angezeigt, sondern zum Betriebssystem gesendet. Je nach Programm-Modus wird dieses Zeichen zum Bildschirm zurückübertragen. So wird z.B. ein eingegebenes Kennwort nicht sichtbar, da es nicht zum Bildschirm zurückübertragen wird.

Funktionsübersicht

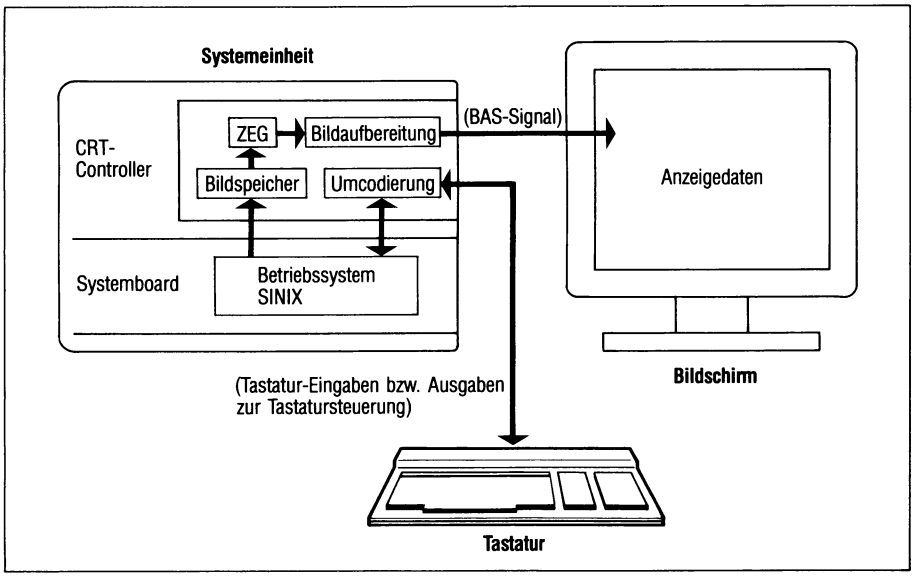


Bild 2-2 Funktionsübersicht

2.1 Schnittstellen im Grundausbau des PC-X10

Der PC-X10 verfügt über eine Reihe von parallelen und seriellen HW-Schnittstellen. Sie unterscheiden sich in

- vom Benutzer beeinflußbare (externe) Schnittstellen und
- für SINIX reservierte und vom Benutzer nicht veränderbare Schnittstellen.

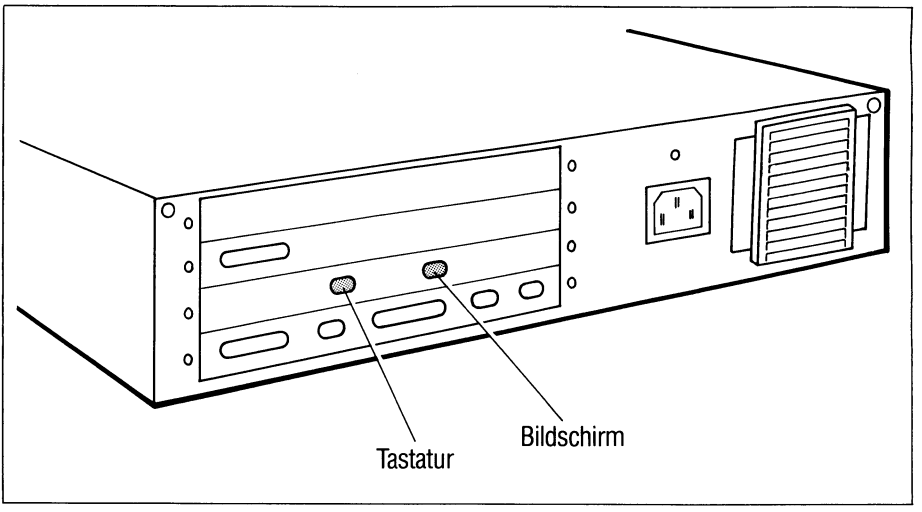


Bild 2-3 Anschlußfeld der Systemeinheit (Textversion)

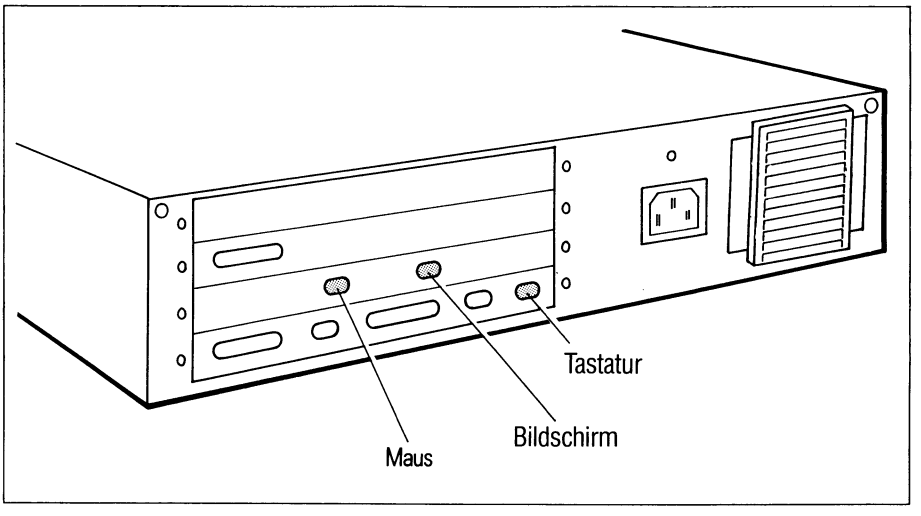


Bild 2-4 Anschlußfeld der Systemeinheit (Grafikversion)

Am Systemboard sind folgende Schnittstellen vorhanden:

Schnittstelle	Typ	Beeinflußbar	Anschluß
Schnittstelle für Diskettenlaufwerk	intern	parametrisierbar über SINIX	Berg-Stecker intern
SCSI-Schnittstelle für Festplatte (Controller)	intern/ parallel	nein	Berg-Stecker intern
Schnittstelle für Erweiterungs-flachbaugruppen	intern/ parallel parallel	nein	96-poliger Er-weiterungsstecker weiterungsstecker
Schnittstelle für Drucker-Anschluß	extern/ seriell	ja	Anschlußfeld St1 Ebene A / RS232
		ja	Anschlußfeld St2 Ebene A / V.11
Reserve-Schnittstelle	extern/ seriell	ja	Anschlußfeld St3 Ebene A / RS232
		ja	Anschlußfeld St4 Ebene A / V.11
Reserviert für Tastatur (Grafik)	extern/ seriell	nein	Anschlußfeld St5 Ebene A / V.11-Ta

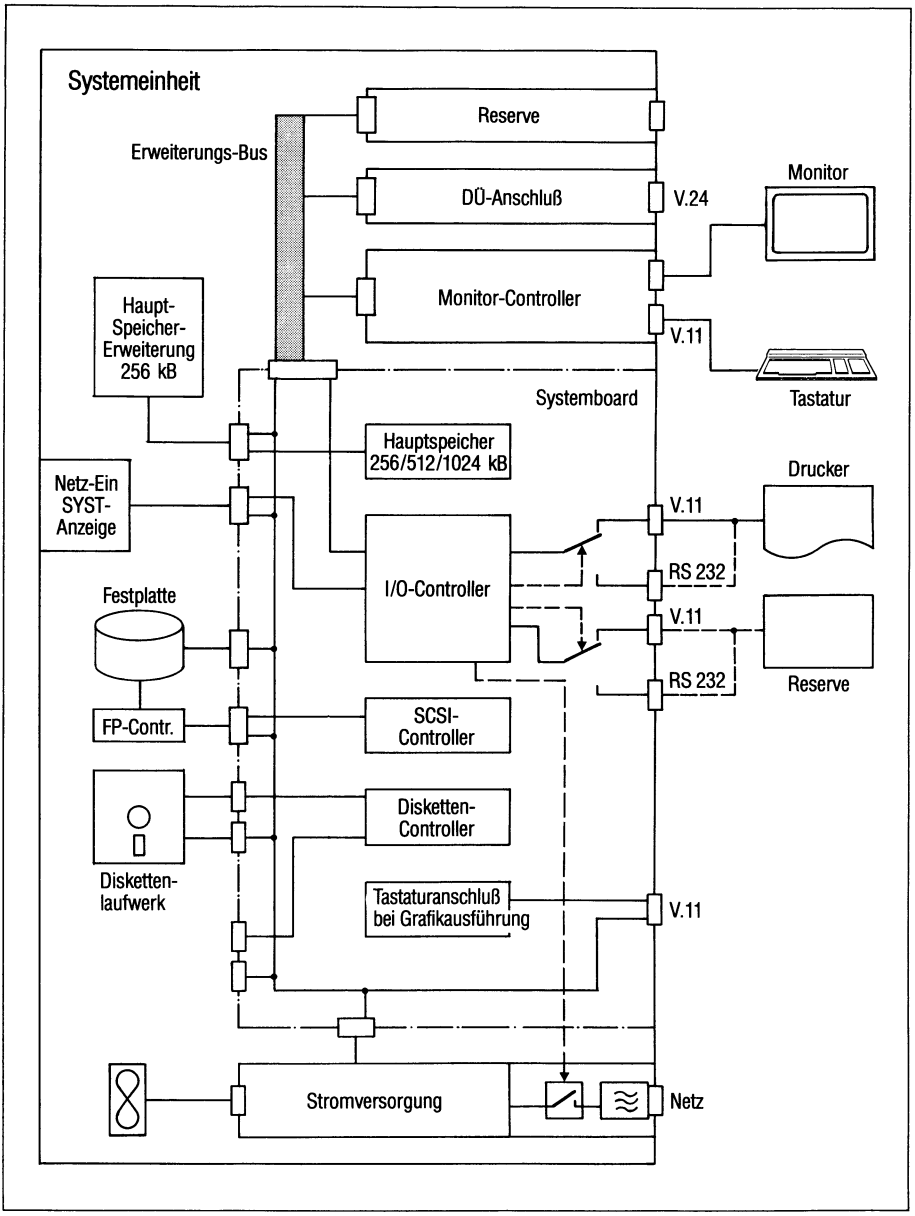


Bild 2-5 Blockschaltbild der Systemeinheit

2.2 Bildschirmsteuerung

Der Bildschirm steht auf der Systemeinheit. Er ist dreh- und neigbar und entspricht den ergonomischen Anforderungen für Bildschirm-Arbeitsplätze.

Der Bildschirm ist über ein Kabel an die in der Systemeinheit eingebaute Bildschirm-Steuerung angeschlossen. Die Ansteuerung des Bildschirms erfolgt über eine BAS-Schnittstelle (Bild-Austast-Synchronisiersignal). Die Versorgungsspannung von +12 V erhält der Bildschirm über das gleiche Anschlußkabel von der Stromversorgung der Systemeinheit.

Die Bildschirmsteuerung ermöglicht:

- das stellenrichtige Einschreiben der vom Betriebssystem übergebenen Daten in den Bildwiederholspeicher,
- das Ausführen von Befehlen, wie z.B. Cursor positionieren, Löschen und Verschieben von Zeichen/Zeilen,
- das Übertragen der von der Tastatur eingegebenen Daten zum Betriebssystem.

Die Bildhelligkeit läßt sich durch einen Regler am Bildschirm den Lichtverhältnissen am Arbeitsplatz anpassen.

Zeichen-Bildschirm

Der Bildschirm kann 25 x 80 Zeichen darstellen. Für jedes einzelne Zeichen können die Darstellungsarten (normal, invers, halbhell, blinken, unterstrichen) in beliebiger sinnvoller Kombination gewählt werden.

Der Pegel von Halbhelldarstellungen ist auf der Rückseite der Systemeinheit an der Bildschirm-Steuerung einstellbar.

Die Bildwiederholfrequenz beträgt 66 Hz. Die Darstellungsart ist positiv, d.h schwarze Schrift auf weißem Hintergrund.

Die Bildschirm-Steuerung bildet mit Tastastur und Bildschirm die Bedieneinheit des PC-X10. Sie steuert und überträgt über den internen Erweiterungsbus die Ein- und Ausgaben zwischen Betriebssystem und Tastatur bzw. Bildschirm.

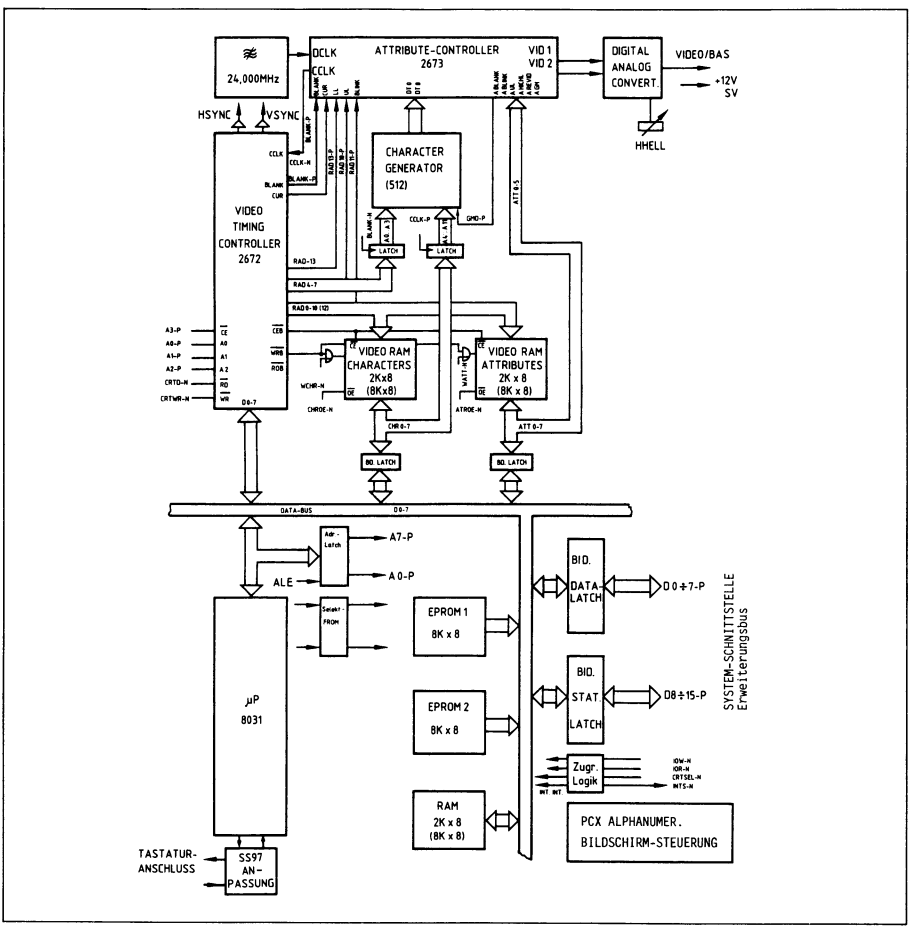


Bild 2-6 Blockschaltbild der Text-Bildschirm-Steuerung

Grafik-Bildschirm

Der Bildschirm kann 25 Zeilen zu 80 Zeichen Text und 640 x 400 Bildpunkte Grafik darstellen.

Für jedes einzelne Zeichen können die Darstellungsarten (normal, invers, blinken, unterstrichen) in beliebiger sinnvoller Kombination gewählt werden.
Die Darstellungsart **blinken** belastet den Prozessor des PC-X10. Dadurch wird die Leistung des PC-X10 stark vermindert.
Die Darstellungsart **blinken** sollte daher sparsam verwendet werden.

Halbhelldarstellung ist mit der Grafik-Bildschirm-Steuerung nicht möglich, da diese Bildschirm-Steuerung nur eine Video-Speicher-Ebene besitzt. Sie wird durch unterlegte Darstellung simuliert.

Die Bildwiederholfrequenz beträgt 70 Hz. Die Darstellungsart ist positiv, d.h schwarze Schrift auf weißem Hintergrund.

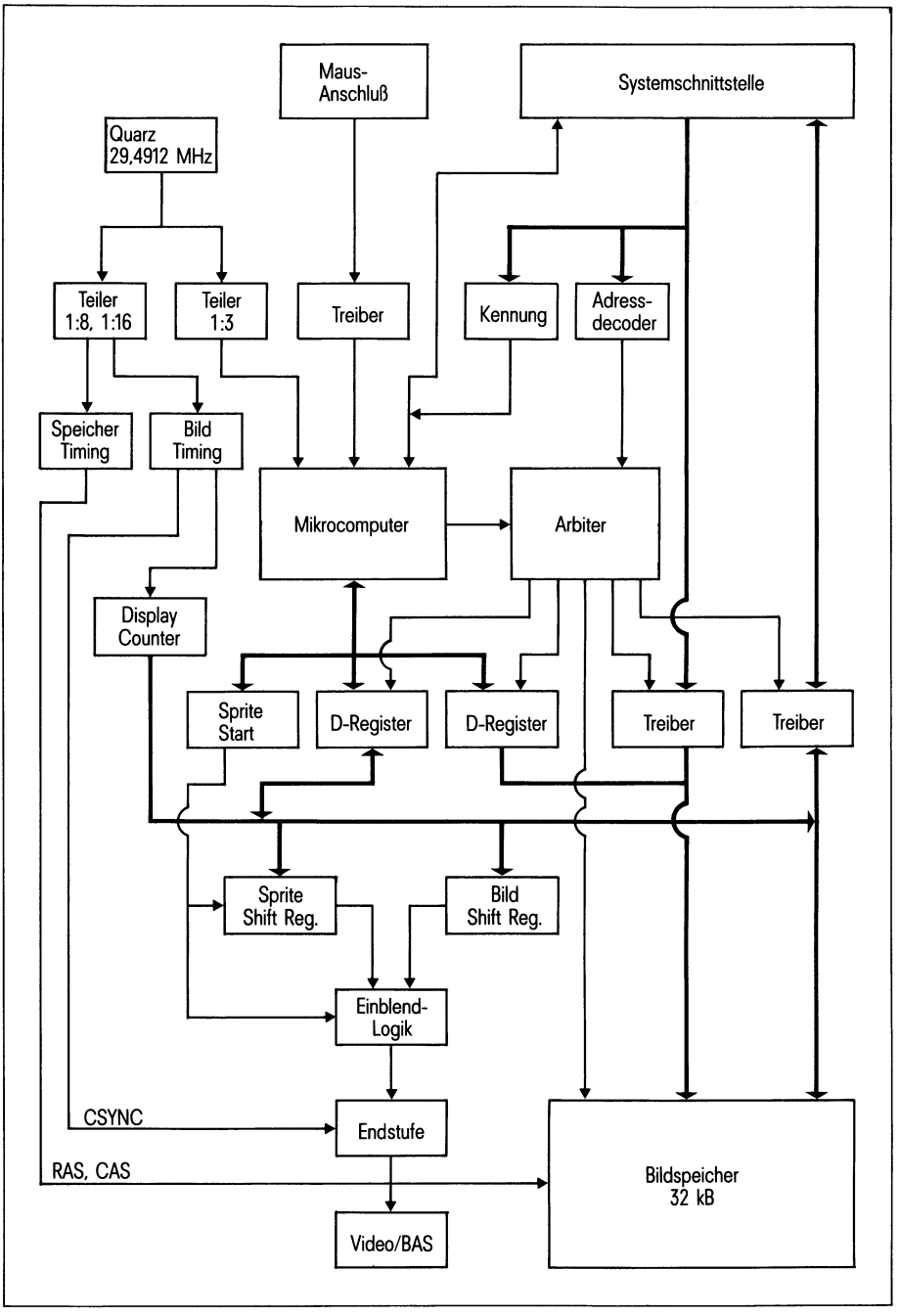


Bild 2-7 Blockschaltbild der Grafik-Bildschirm-Steuerung

2.3 Tastatur

Jedes Betätigen einer Taste erzeugt einen Platzcode, der seriell zur Bildschirm-Steuerung übertragen wird. Hier erfolgt eine Umcodierung des Platzcodes entsprechend dem geladenen Zeichensatz in ASCII-Zeichen. Diese werden dem Betriebssystem übergeben.

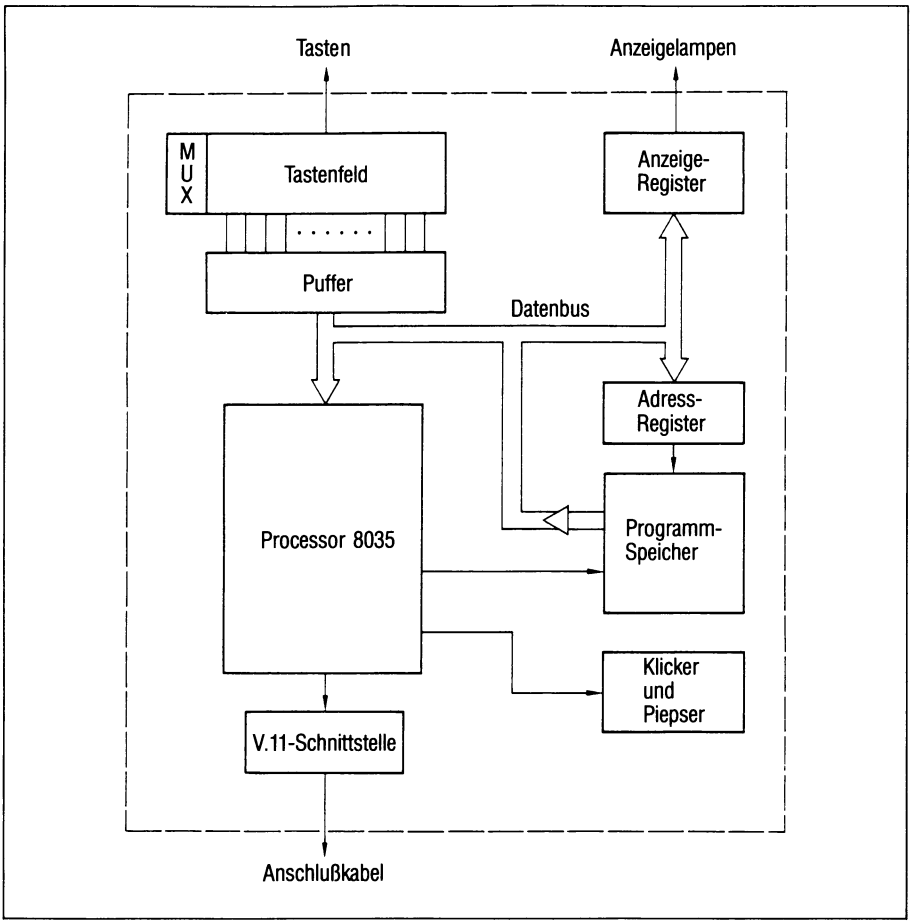


Bild 2-8 Blockschaltbild der Tastatur

Hinweise

- Die Firmware (Programm und Zeichenspeicher) in der Bildschirm-Steuerung ist **immer** gleich, unabhängig von den länderspezifischen Tastaturen.
- Die Firmware (Programm und Platzcodespeicher) in der Tastatur ist länderspezifisch.
- Das Laden des Zeichensatzes (entsprechend der verwendeten Tastatur) im Bildschirm erfolgt immer bei Ausgabe des Begrüßungsbildschirmes. Welcher Zeichensatz geladen wird, ist in der Datei /etc/termcap terminalspezifisch hinterlegt (siehe 1.7, Beispiel 1).
- Tastaturvarianten des PC-X10 (die Nummer finden sie an der Tastaturunterseite), wahlweise mit folgenden Belegungen:
  - international 97811-131
  - international und deutsch 97811-132
  - Alle anderen Tastaturbelegungen werden durch Laden der Tastaturbelegungstabelle und Austausch der Tastenkappen festgelegt.
    Tastenkappensätze zum Umrüsten der internationalen Tastatur auf nationale/internationale Tastaturbelegung:

– international und schwedisch	97801-144
– international und dänisch	97801-145
– international und französisch	97801-146
– international und spanisch	97801-149
– international und italienisch	97801-150
– international und britisch	97801-153
– international und norwegisch	97801-154

2.4 Beeinflußbare externe Schnittstellen (Stecker 1 - 4)

Von den an der Systemeinheit befindlichen 5 Steckeranschlüssen an der unteren Anschlußleiste werden die Stecker 1 bis 4 von SINIX standardmäßig wie folgt verwendet:

- Stecker 1 (RS232) frei
- Stecker 2 (V.11/SS97) Drucker
- Stecker 3 (RS232) frei
- Stecker 4 (V.11/SS97) frei

Stecker 5 ist für den Tastaturanschluß im Zusammenhang mit dem Einsatz des Grafik-Bildschirm-Controller reserviert.

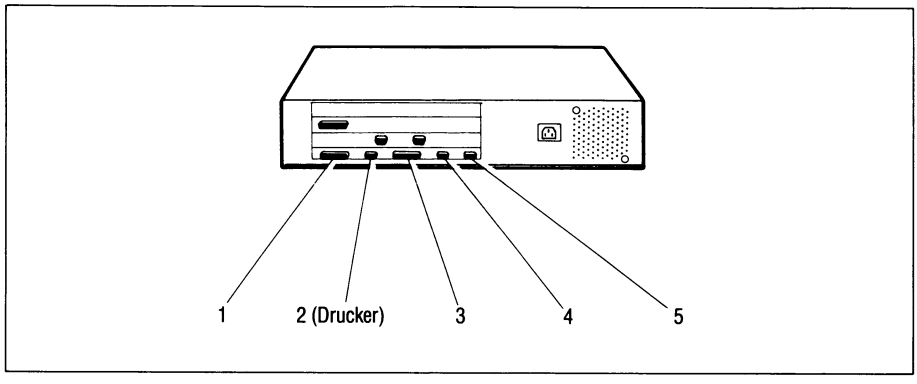


Bild 2-9 Externe Schnittstellen

2.4.1 Umschalten der Schnittstelle SS97/RS232

Stecker 1/2 und Stecker 3/4 werden von je einem seriellen Ein-/Ausgabe-Controller gesteuert.
Jeder der beiden Controller kann softwaremäßig einer RS232-Schnittstelle (V.28-Pegel) oder einer SS97-Schnittstelle (V.11-Pegel) zugeordnet werden. Die Sendedaten eines Controllers werden an beiden Schnittstellen (SS97/RS232) angeboten. Die Empfangsdaten werden je nach Zuordnung nur von einer Schnittstelle an den Controller weitergereicht.

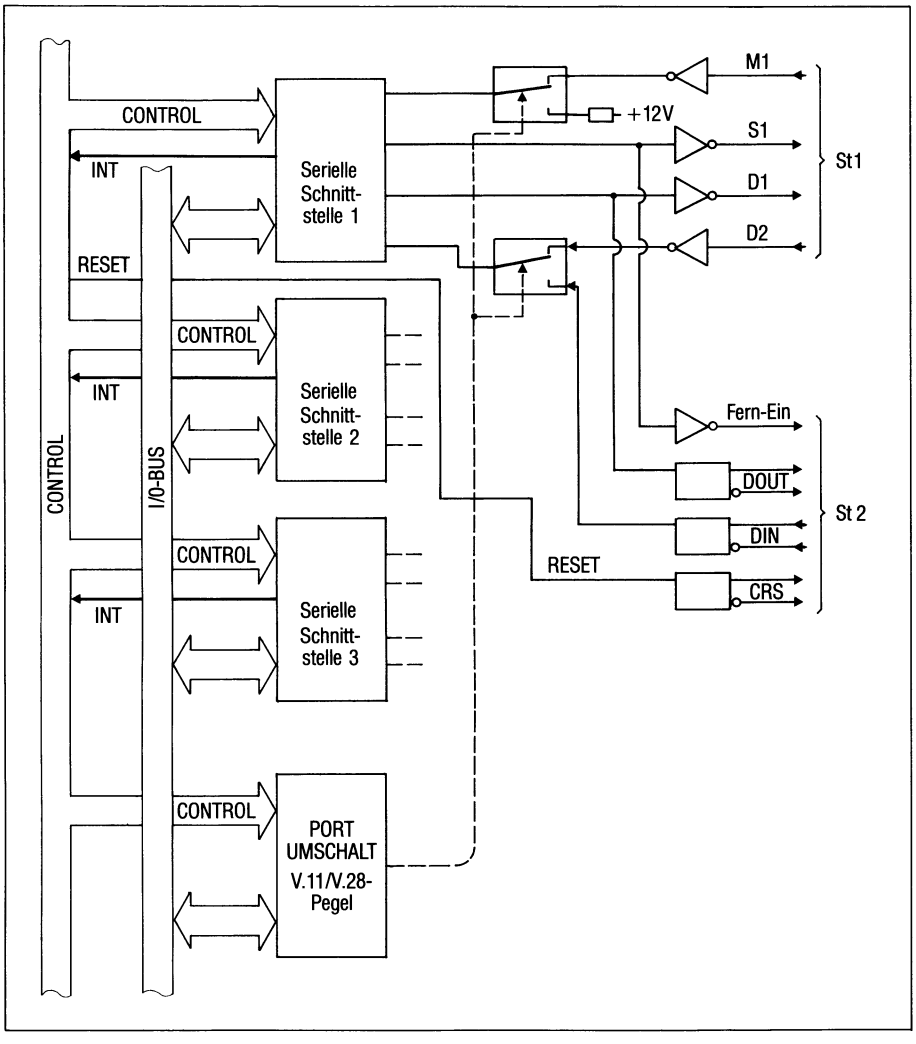


Bild 2-10 Umschaltung SS97/RS232

Mit dieser Umschaltung wird nur die Hardware-Schnittstelle bezüglich Pegel und Belegung verändert. Das logische Protokoll, mit dem das angeschlossene Gerät betrieben wird, bleibt unverändert.

Es können nur Geräte angeschlossen werden, die die Datenflußsteuerung mit XON-XOFF (DC1-DC3) durchführen.

Die Adressierung der jeweiligen Schnittstelle unter SINIX ist wie folgt vorzunehmen:

- Adresse der beiden seriellen Kanäle am Systembord mit Majornummer 3
- Adresse des jeweiligen Anschlusses:

```
Stecker 1   → Minornummer 131    (RS232)
Stecker 2   → Minornummer   1    (SS97)
Stecker 3   → Minornummer 130    (RS232)
Stecker 4   → Minornummer   0    (SS97)
```

2.4.2 Beispiel: Anschluß eines Druckers an die Schnittstelle RS232

a) Einrichten der Gerätedateien

Für jede physikalische Hardwareschnittstelle muß in dem Dateiverzeichnis /dev ein Eintrag vorhanden sein.
Wird bei der Installation des SINIX-Systems ein Drucker eingerichtet, wird die dazu notwendige Gerätedatei angelegt,

z.B. Eintrag für einen Drucker 9001:

```
crw-rw-rw- 1 root          3,   1 Mar 12 17:49 lp9001-1-D1
```

die Majornummer 3 bezeichnet die seriellen SS auf dem Systemboard, die Minornummer 1 schaltet auf den SS97-Anschluß (St2) um.

SS97: Coderahmen 7 bit + Parity (ungerade) Übertragungsgeschwindigkeit 9600 bit/s

b) Einrichten einer Gerätedatei für Drucker mit RS232-Anschluß

```
cd /dev
/etc/mknod lpfremd c 3 131
```

c) Ausgeben einer Datei auf den Drucker

```
cat DATEINAME >/dev/lpfremd
```

Werden keine Parameter der Schnittstelle verändert, so werden

- die Daten mit 9600 bit/s und
- jede Zeile, die mit 0x0a(Line Feed) abgeschlossen ist, mit 0x0d,0x0d (Carriage return und Line Feed)

ausgegeben.

Ist eine andere Ausgabegeschwindigkeit gefordert, so ist die Einstellung der gewünschten Übertragungsgeschwindigkeit mit dem Kommando stty möglich.

3 Schnittstellen des PC-MX2

3.1 Schnittstellen im Grundausbau (SS97/RS232)

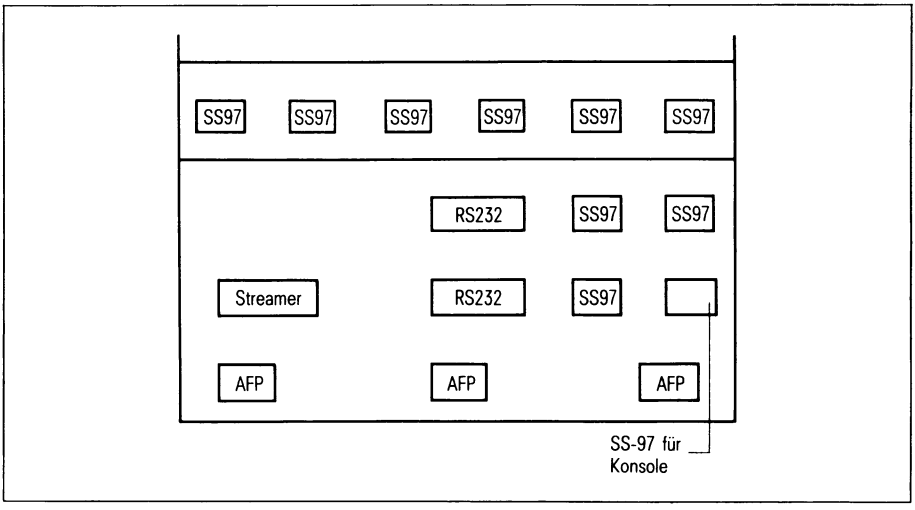


Bild 3-1 Anschlußfeld PC-MX2

Im Grundausbau enthält der PC-MX2 einen E/A-Prozessor SERAD mit 4 Schnittstellen SS97 und 2 Schnittstellen RS232 .
Der PC-MX2 kann mit einem E/A-Prozessor SERAG mit 6 Schnittstellen SS97 oder einem zweiten E/A-Prozessor SERAD erweitert werden.

Geräteadressen des PC-MX2

Die angegebenen Minor-Nummern gelten für Drucker oder Fremdgeräte. Bei Standardterminals gelten nur die letzten beiden Stellen.

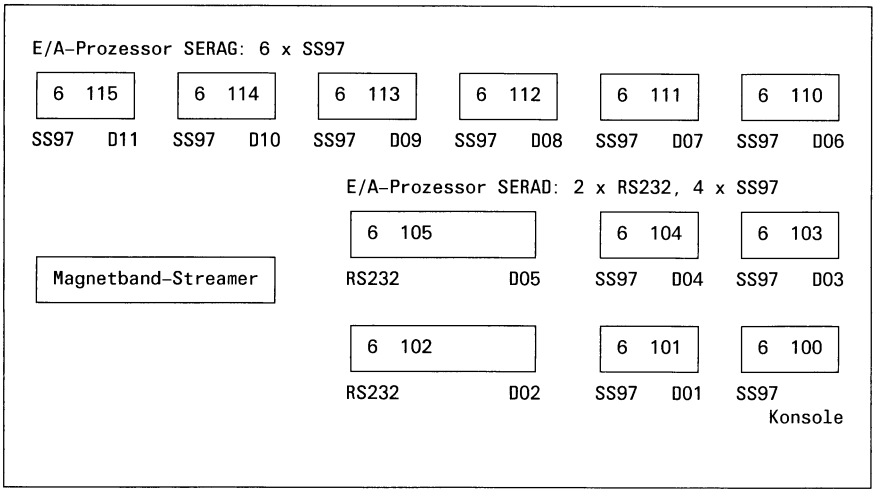


Bild 3-2 PC-MX2 Geräteadressen

3.1.1 Beispiel

Anschluß eines Druckers an die Schnittstelle RS232

a) Einrichten der Gerätedateien

Für jede physikalische Hardwareschnittstelle muß in dem Dateiverzeichnis /dev ein Eintrag vorhanden sein. Wird bei der Installation des SINIX-Systems ein Drucker eingerichtet, wird diese notwendige Gerätedatei angelegt, z.B. Eintrag für einen Drucker 9001:

```
crw-rw-rw- 1 root          6,101 Mar 12 17:49 lp9001-D01
```

Die Majornummer 6 bezeichnet den Treiber für die E/A-Baugruppen. Die Minornummer 101 legt den Ausgang, den E/A-Pozessor und den Coderahmen fest.

```
101 bedeutet:
|||
|||___ 2. Anschluß
||
||____ 1. E/A-Prozessor
|
|_____ Standardeinstellung ist 9600 Baud und Odd-Parity,
       die Schnittstelle ist jedoch beliebig einstellbar.
```

```
Einrichten einer Gerätedatei

cd /dev
/etc/mknod lpfremd c 6 101
           |       | | |
           |       | | |___ Minornummer für 1. RS232C-Schnittstelle
           |       | |
           |       | |____ Majornummer
           |       |
           |       |_____ Character Device
           |
           |_____________ frei gewählter Name
```

b) Ausgeben einer Datei auf den Drucker

```
cat DATEINAME > /dev/lpfremd
```

Werden keine Parameter der Schnittstelle verändert, so werden

– die Daten mit 9600 bit/s und Odd-Parity

ausgegeben.

Ist eine andere Übertraungsgeschwindigkeit gefordert, so kann sie mit dem tty#-Kommando wie folgt eingestellt werden:

```
stty    1200 >/dev/lpfremd        Geschwindigkeit setzen
```

Dieses Kommando kann man in die Datei /etc/rc einfügen. Damit wird bei jedem Systemstart der Kanal automatisch richtig eingestellt.

3.2 Ein-/Ausgabeprozessor

3.2.1 SERAG: 6 x SS97

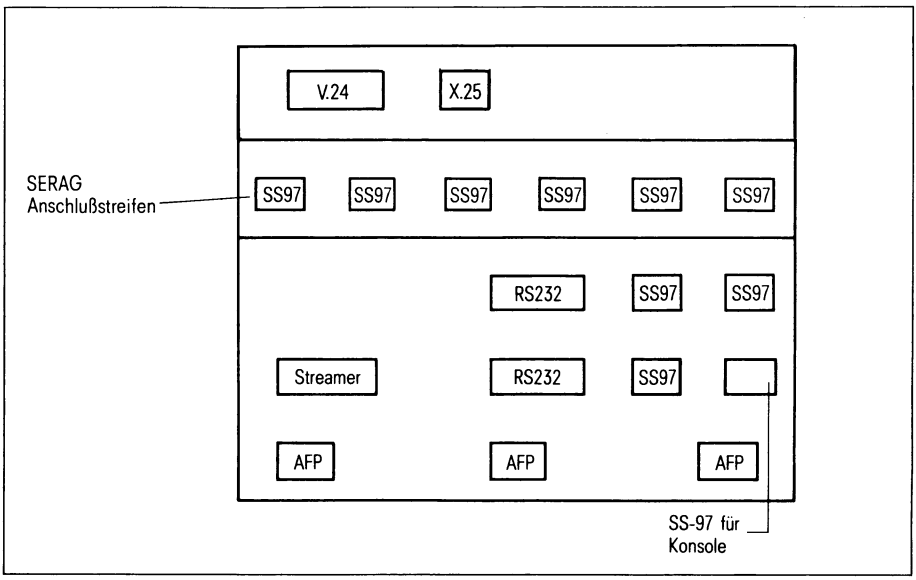


Bild 3-3 Anschlußfeld der Systemeinheit E/A-Prozessor SERAG

An diese Schnittstellen werden die Bildschirme für das Mehrplatzsystem und die Drucker angeschlossen.
Ein Umschalten der Pegel ist nicht möglich.

Die Leitung FE/PO (Fern-Ein) dieser Schnittstellen kann benutzt werden, um Peripheriegeräte vom PC aus einzuschalten.
Mit OPEN auf einen Kanal wird die entsprechende Leitung FE/PO auf 0 Volt geschaltet.

Die Leitung UH (Hilfsspannung) dieser Schnittstellen kann benutzt werden, um Peripheriegeräte mit Hilfsspannung +12 V zum Schalten der Netz-Ein-Logik zu versorgen. Der Strom darf 30 mA nicht übersteigen.

3.2.2 SERAD: 4 x SS97 und 2 x RS232

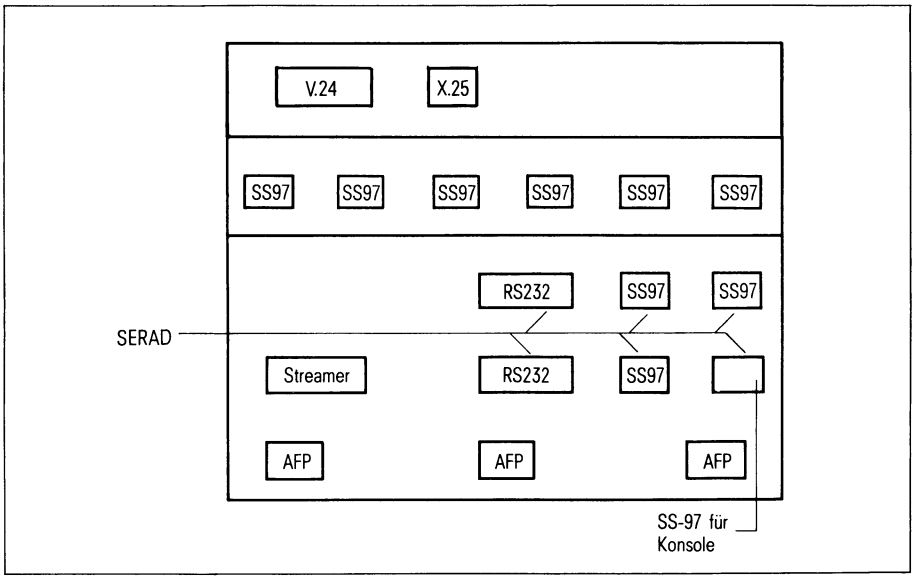


Bild 3-4 Anschlußfeld der Systemeinheit E/A-Prozessor SERAD

An diese Schnittstellen werden die Bildschirme für das Mehrplatzsystem und die Drucker angeschlossen. Es gibt 2 Kanäle mit RS232-Pegel und 4 Kanäle mit V.11-Pegel (SS97).

SS97: Die Leitung FE/PO (Fern-Ein) dieser Schnittstelle kann benutzt werden, um Peripheriegeräte vom PC aus einzuschalten.
Mit OPEN auf einen Kanal wird die entsprechende Leitung FE/PO auf 0 Volt geschaltet.
Die Leitung UH (Hilfsspannung) dieser Schnittstellen kann benutzt werden, um Peripheriegeräte mit Hilfsspannung +12 V zum Schalten der Netz-Ein-Logik zu versorgen. Der Strom darf 30 mA nicht übersteigen.

RS232: Unter Umständen kann das Signal S1 (Pin 20) zum Einschalten der Peripheriegeräte verwendet werden. S1 wird vom 1.'OPEN' auf den Kanal bis zum 'CLOSE' aktiv gehalten (> +3 V).

3.2.3 Mehrere Ein-/Ausgabeprozessoren

Eine Umrüstung kann nur durch den Siemens-Servie erfolgen.

Hinweis

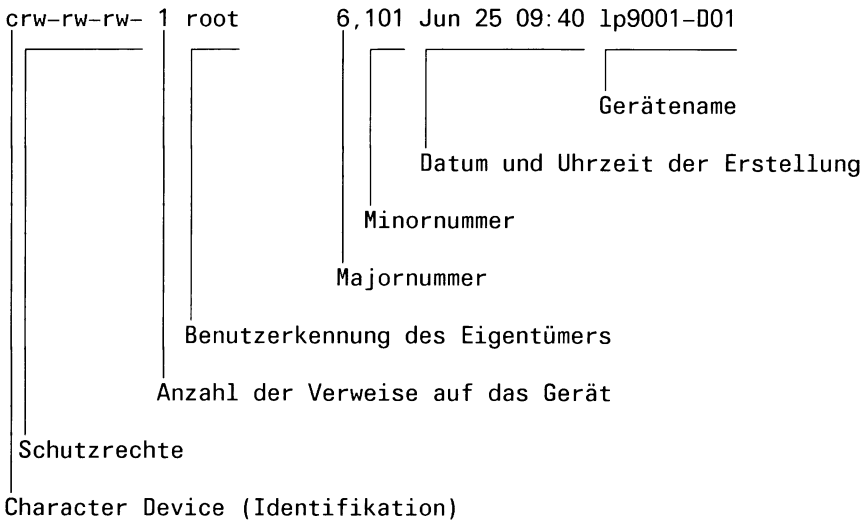
Zwei E/A-Prozessoren müssen in aufsteigender Reihenfolge gesteckt sein. Ist z.B. der 1.E/A-Prozessor auf Einbauplatz 7, dann muß der 2.E/A-Prozessor auf Einbauplatz 8 stecken.
Am 2.E/A-Prozessor muß die I/O-Adresse 1100 eingestellt werden.
Die E/A-Prozessoren arbeiten ohne Interupt im Polling-Verfahren.

3.2.4 Allgemeines über Gerätedateien

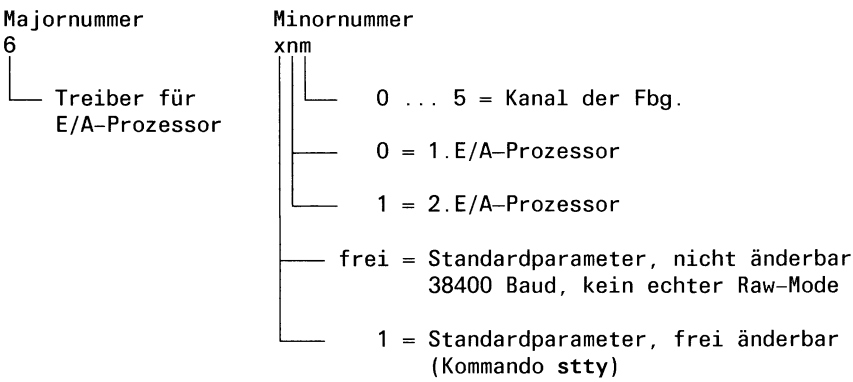
Über die Gerätedateien werden die angeschlossenen Geräte direkt angesprochen. Vom System ist die Möglichkeit geschaffen, über Gerätedateien die Leitungsparameter

- gezielt zu verändern oder
- eine Veränderung der Leitungsparameter zu verhindern.

Aufbau einer Gerätedatei in /dev



Festlegung der Major- und Minornummern



Standardparameter

- Bei den 'Hunderter-Kanälen':

  ```
  Übertragungsgeschwindigkeit 9600 bit/s
  erase = '^H'; kill = '^X'
  odd -n1 -tabs cbreak
  ```

- Bei Kanälen ohne 'Hunderter-Nummer':

  ```
  Übertragungsgeschwindigkeit 38400 bit/s
  erase = '^H'; kill = '^X'
  odd -n1 -tabs cbreak
  ```

  Zeichenrahmen und Geschwindigkeit lassen sich nicht beeinflussen!

Gegenüber diesen Vorgaben besitzen die Einträge in /etc/ttys (ob Login-Gerät) - und davon abhängig, die in /etc/ttytype vorgegebenen Werte - Vorrang!

Beispiel

```
crw-rw-rw- 1 root          6,105 Jun 24 14:55 /dev/lpfremd
```

An Kanal 5 des ersten E/A-Prozessors ist ein Drucker angeschlossen, der mit 9600 bit/s betrieben wird.

4 Physikalische Schnittstellen

4.1 Schnittstelle SS97

Die serielle Schnitstelle SS97 ist vollduplexfähig. Die Datenübertragung erfolgt erdfrei über verdrillte Aderpaare. Der Datenfluß wird durch ein XON/XOFF-Protokoll gesteuert.

4.1.1 Steckerbelegung

Es sind 9-polige Stecker der Serie HDP 20, z.B. von AMP, zu verwenden. Am Peripheriegerät sind immer Stifte einzubauen.

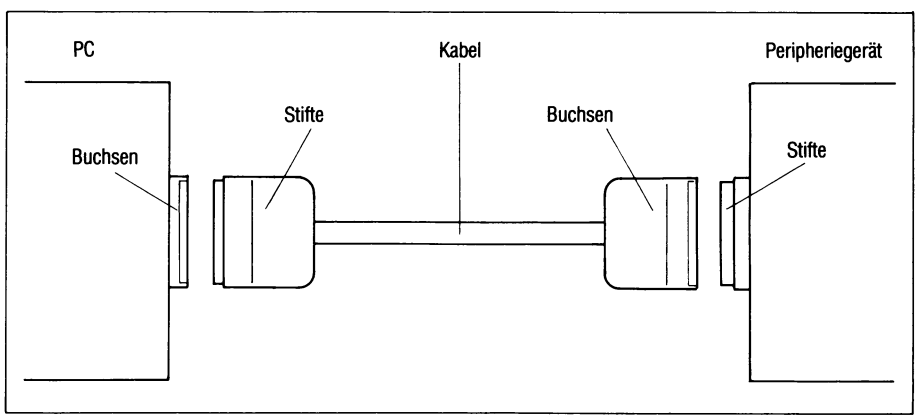


Bild 4-1 Steckerbelegung (9-poliger Stecker)

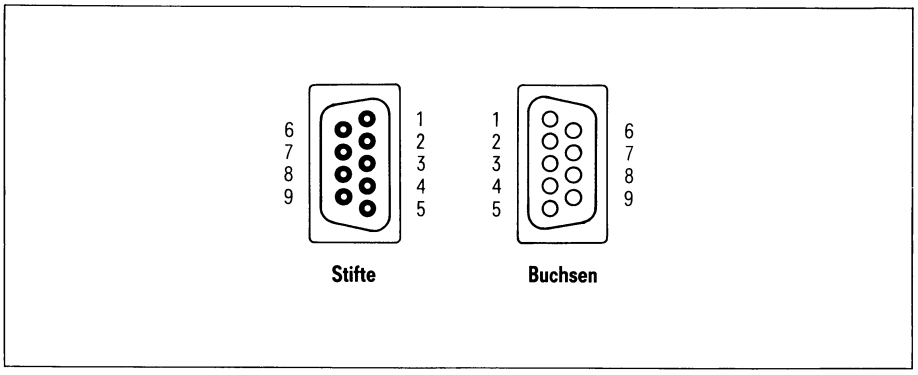


Bild 4-2 Numerierung von Buchsen und Stiften der 9-poligen Steckverbindung

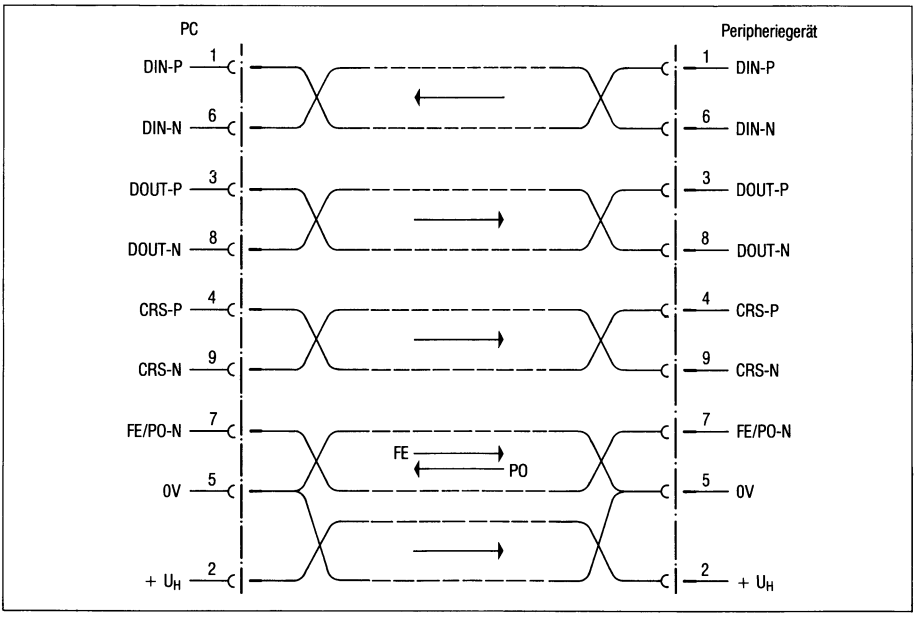


Bild 4-3 Zuordnung der Signale zum Stecker und logische Signalrichtung

Stift	Bezeichnung	Erklärung
1	DIN-P	Empfangsdaten
6	DIN-N	
3	DOUT-P	Sendedaten
8	DOUT-N	
4	CRS-P	Rücksetzsignal vom Pc bei Netz-Ein
9	CRS-N	
7	FE/PO-L	Fern-Ein (Einschaltsignal vom PC zum Peripheriegerät)/ Power On (Einschaltsignal von der Konsole zum PC)
5	0 V	Masseleitung
2	UH	Hilfsspannung +12 V (max. 30 mA)

4.1.2 Leitungen

Die Leitungen

- Sendedaten (DIN),
- Empfangsdaten (DOUT) und
- Rücksetzen (CRS)

sind symmetrisch mit AMD-Bausteinen 26 LS 31 bzw. 26 LS 32 (oder kompatiblen) auszuführen, um einen großen Störabstand zu gewährleisten. Diese Bausteine bieten außerdem den Vorteil, daß sie nur eine Versorgungsspannung (+5 V) benötigen.
Die Versorgungsspannung des Senders sowie die Versorgung des Empfängers, die ja von getrennten Stromversorgungen gespeist werden, darf nur einen max. Spannungsunterschied von 8 V annehmen, da sonst eine Zerstörung der Bausteine eintreten kann.
Für eine störungsfreie Datenübertragung ist ein **wesentlich** geringerer Potentialunterschied unbedingt notwendig.

Wird komponentenspezifisch die eine oder andere Signalleitung nicht benötigt, so muß sichergestellt sein, daß der Empfänger der Gegenstelle einen definierten Zustand (Stoppolarität) einnimmt. Dazu muß jeder Empfängerbaustein mit Abschlußwiderständen, wie in Bild 4-4 vorgeschlagen, beschaltet werden. Der entsprechende Sender kann entfallen.

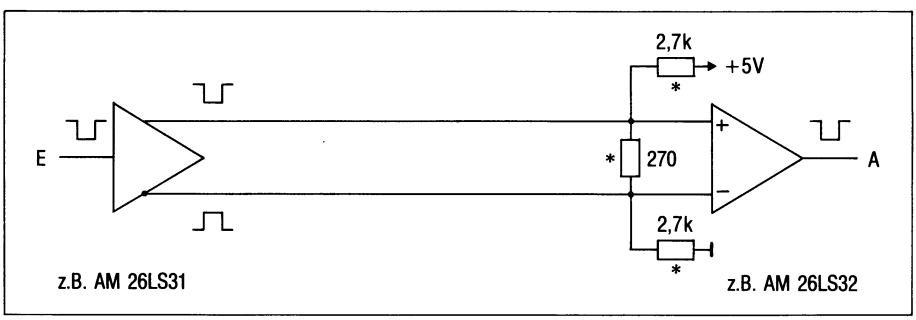


Bild 4-4 Beschaltung nicht benutzter Signalleitungen

* Diese Widerstände bewirken einen definierten Zustand (z.B. Stoppolarität) des Empfängers bei abgezogener bzw. getrennter Leitung.

Kabellängen

Die max. Kabellänge kann 500 m betragen.
Es gibt konfektionierte Kabel mit Längen von 5, 10, 20, und 30 m. Die Kabel können auch gekoppelt werden (z.B. 2 x 30 m).
Für größere Entfernungen müssen Kabel - entsprechend der o.g. Einschränkungen - selbst gefertigt werden (siehe Aufbaurichtlinien).

Die Güte der Datenübertragung ist stark von der Potentialgleichheit der Peripheriegeräte abhängig. Die Datenübertragung wird durch Ausgleichsströme gestört, die über den Schirm des Kabels fließen!
Also: PC und Peripheriegeräte an einem Netz mit gemeinsamem Bezugspunkt (z.B. Stockwerkverteiler) anschließen!

4.1.3 Signalzustände

Definition

a) Treibereingänge/Empfängerausgänge

H ≙ log. 1 in TTL-Definition
L ≙ log. 0 in TTL-Definition

b) Treiberausgänge/Empfängereingänge

H ≥ +3 V
L ≤ +2 V

DIN-Leitung

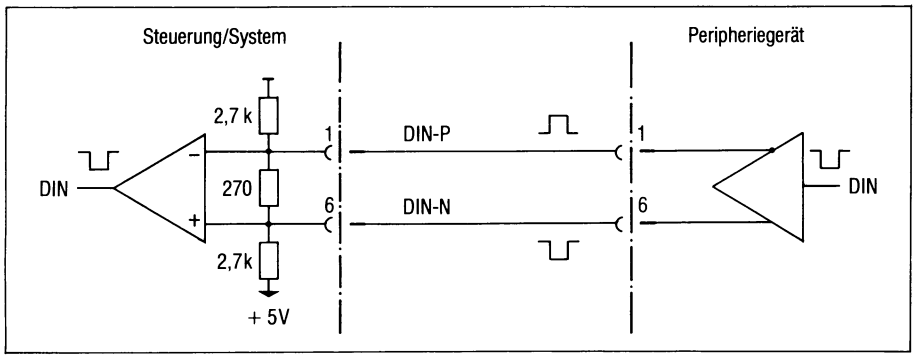


Bild 4-5 DIN-Leitung

DIN	DIN-P	DIN-N
H	L	H = log. 1 ≙ Stoppolarität
L	H	L = log. 0 ≙ Startpolarität

Wenn keine Daten gesendet werden, muß Stoppolarität anliegen.

DOUT-Leitung

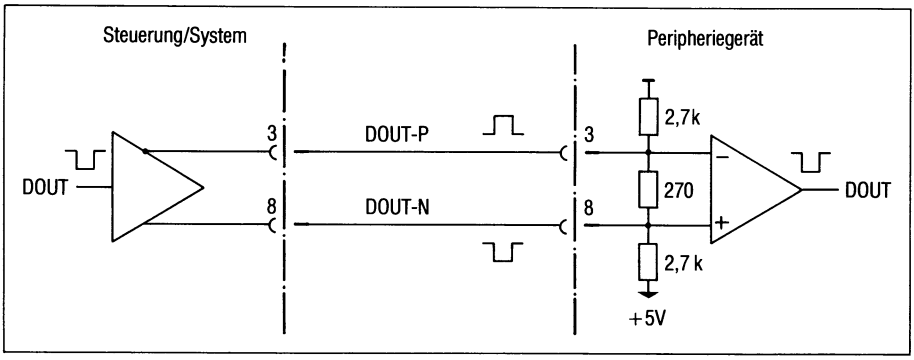


Bild 4-6 DOUT-Leitung

DOUT	DOUT-P	DOUT-N
H	L	H = log. 1 ≙ Stoppolarität
L	H	L = log. 0 ≙ Startpolarität

Wenn keine Daten empfangen werden, liegt Stoppolarität an.

CRS-Leitung

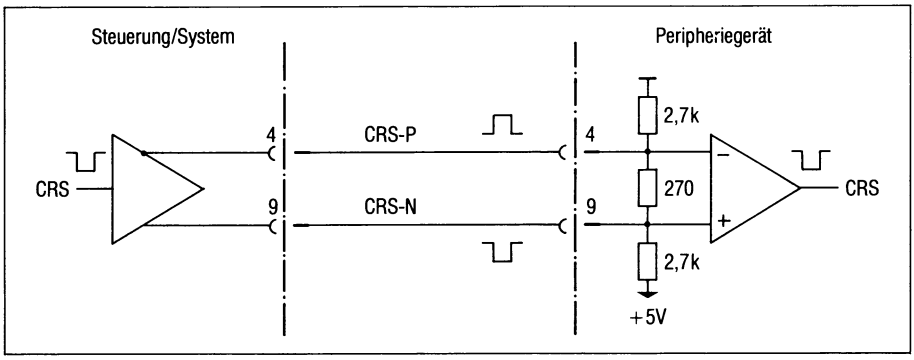


Bild 4-7 CRS-Leitung

CRS	CRS-P	CRS-N
H	L	H ≙ Ruhezustand
L	H	L ≙ Arbeitszustand (t CRS > 70 ms), das Peripheriegerät muß sich rücksetzen

4.1.4 Beschaltung der Leitung FE/PO und UH

Kann ein Peripheriegerät nur manuell über den eigenen Netzschalter eingeschaltet werden, so ist diese Steuerleitung bedeutungslos, darf jedoch nicht beschaltet werden.

Für die Übertragung der Signale FE/PO und UH ist wahlweise eine der folgenden Schaltungen vorgesehen:

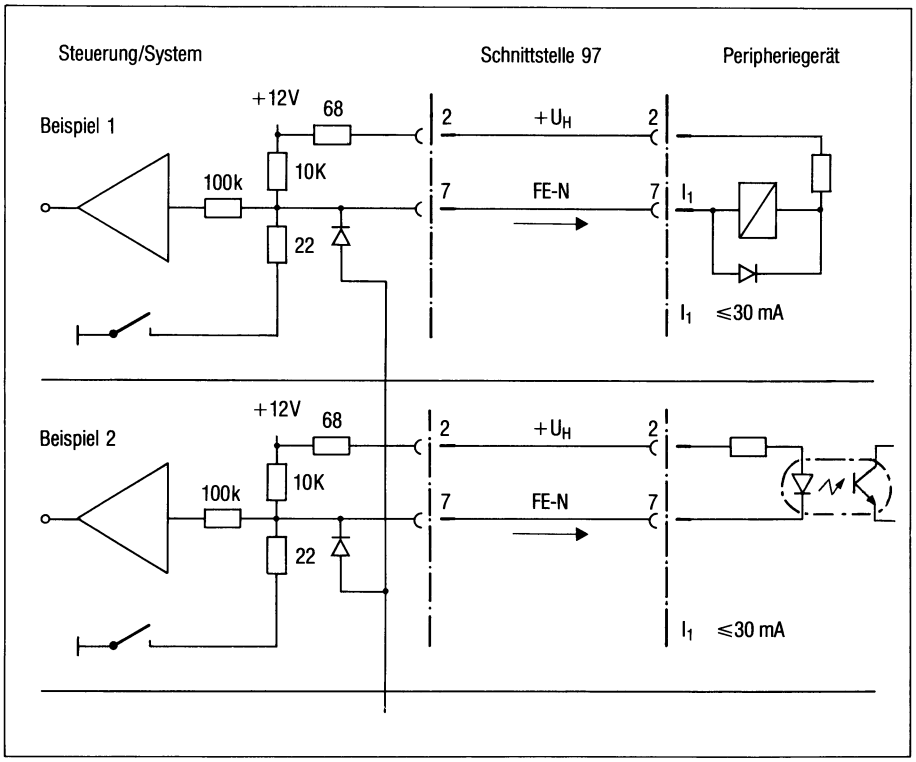


Bild 4-8 Schaltungsvorschläge für Fern-Ein

4.1.5 Schnittstelle zur Bedieneinheit und zum Drucker

E/A-Prozessor SERAG: 6 x SS97 (97802-201) und
E/A-Prozessor SERAD: 4 x SS97 und 2 x RS232 (97802-202)

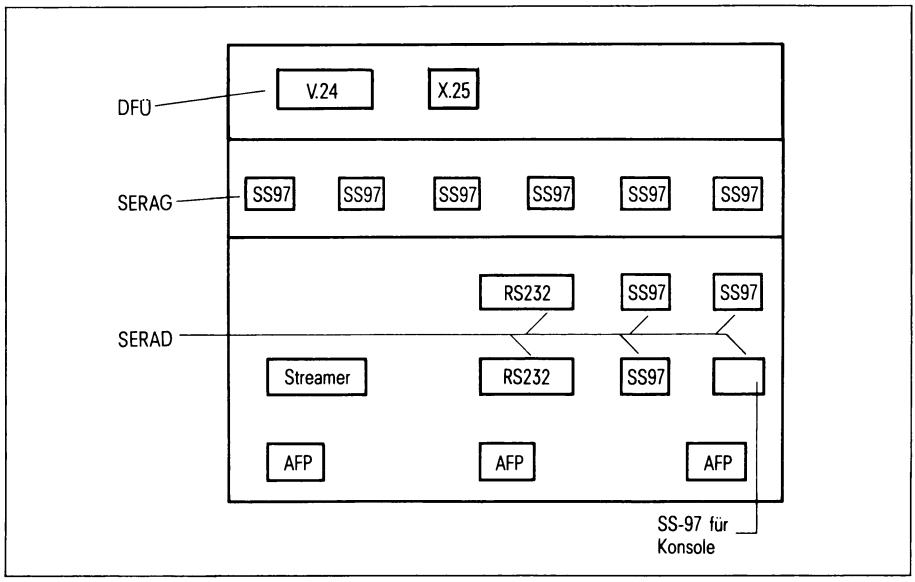


Bild 4-9 Anschlußfeld der Systemeinheit PC-MX mit E/A-Prozessor SERAD

An diese Schnittstellen werden die Bildschirme für das Mehrplatzsystem und die Drucker angeschlossen.

Bedieneinheit

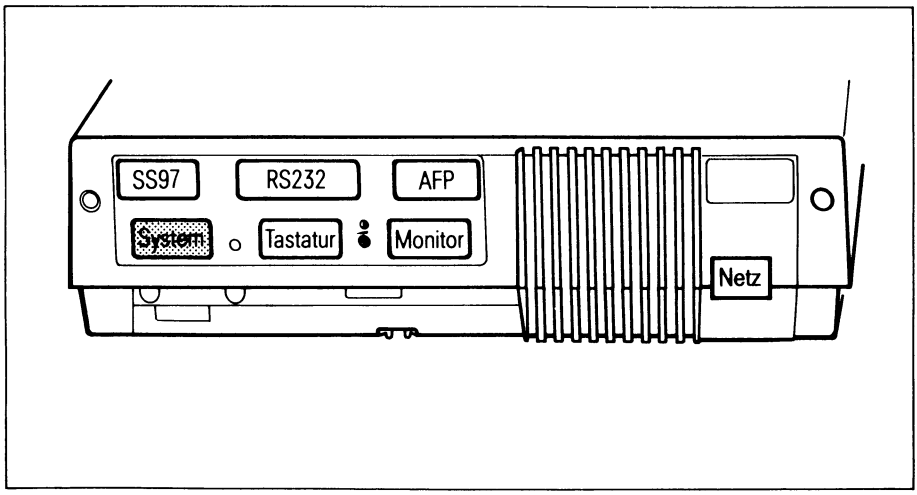


Bild 4-10 SS97 (Stifte) an der Bedieneinheit

Kennwerte der Übertragung

Übertragungsverfahren	:	asynchron
Übertragungsgeschwindigkeit		
– Bedieneinheit	:	38400 bit/s
– Drucker	:	9600 bit/s
Zeichenlänge	:	7 bit + Parity
Parität	:	ungerade
Stopbit	:	einfach
Startbit	:	einfach

4.1.6 Schnittstelle zur Tastatur

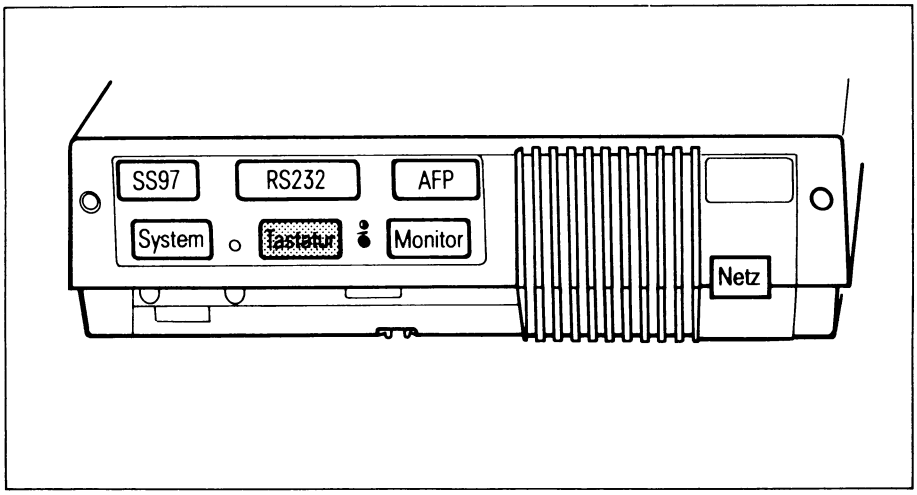


Bild 4-11 Anschlüsse an der Bedieneinheit

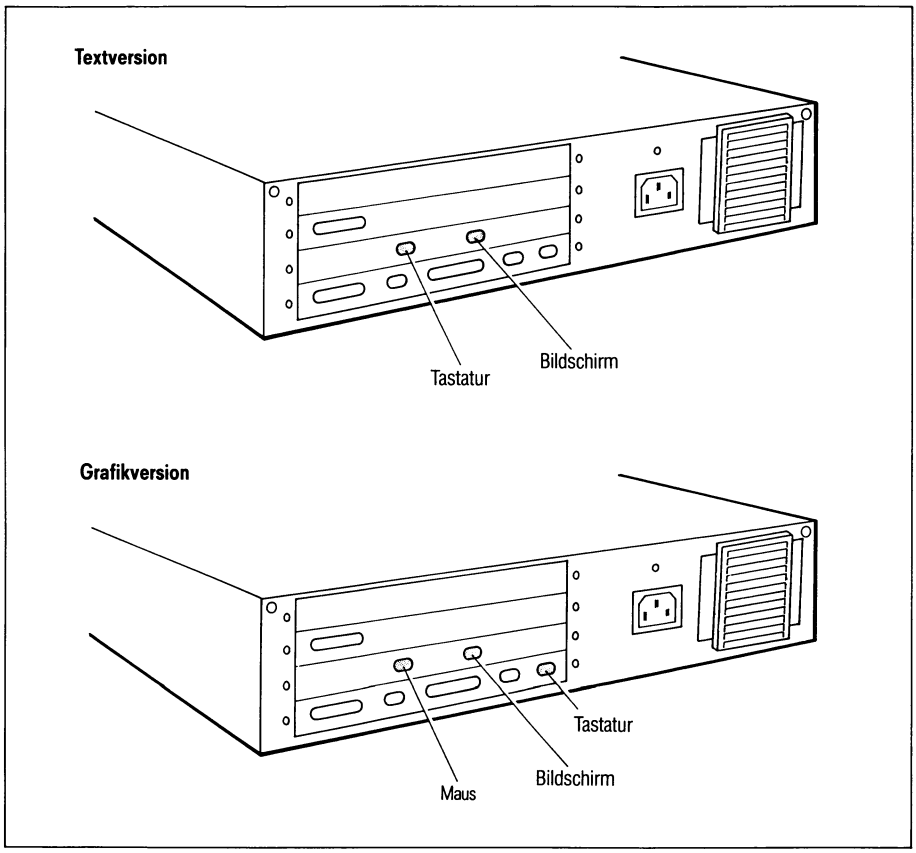


Bild 4-12 Tastatur-Anschluß an der Systemeinheit des PC-X/PC-X10

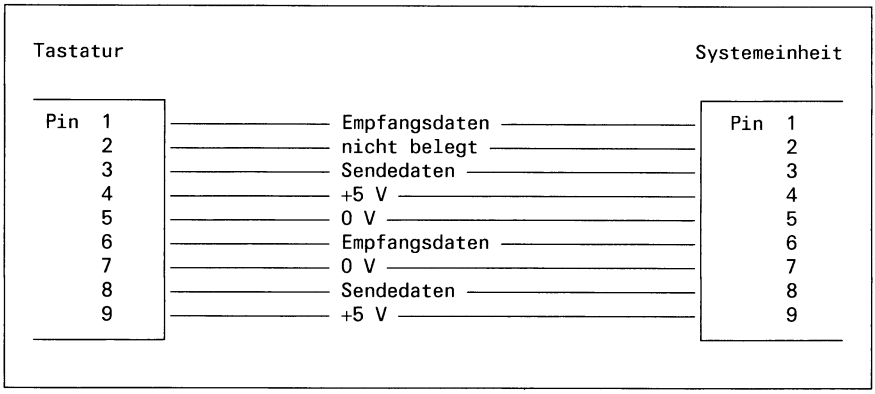


Bild 4-13 Steckerbelegung

Buchse	Bezeichnung	Erklärung
3	DOUT-P	Daten vom Bildschirm zur Tastatur
8	DOUT-N	
1	DIN-P	Daten von der Tastatur zum Bildschirm
6	DIN-N	
4	+5 V	Tastatur-Versorgungsspannung
9	+5 V	
7	0 V	Masseleitung
5	0 V	Masseleitung
2		nicht belegt

Kennwerte der Übertragung

Übertragungsverfahren	:	asynchron
Übertragungsgeschwindigkeit	:	600 bit/s
Zeichenlänge	:	8 bit
Parität	:	ohne
Stopbit	:	einfach
Startbit	:	einfach

Hinweis

Über diese Schnittstelle werden die Platzcodes der Tasten, die Schlüsselschalterinformationen und die Ausweisleserinformationen übertragen.

4.2 Bildschirm-Anschluß

4.2.1 Hardware-Schnittstelle

Der Anschluß des Bildschirms erfolgt über einen 9-poligen Anschlußstekker. Der Stecker wird an der Rückseite

- der Systemeinheit des PC-X oder
- der Bedieneinheit

am Anschlußfeld der Bildschirmsteuerung angeschlossen.

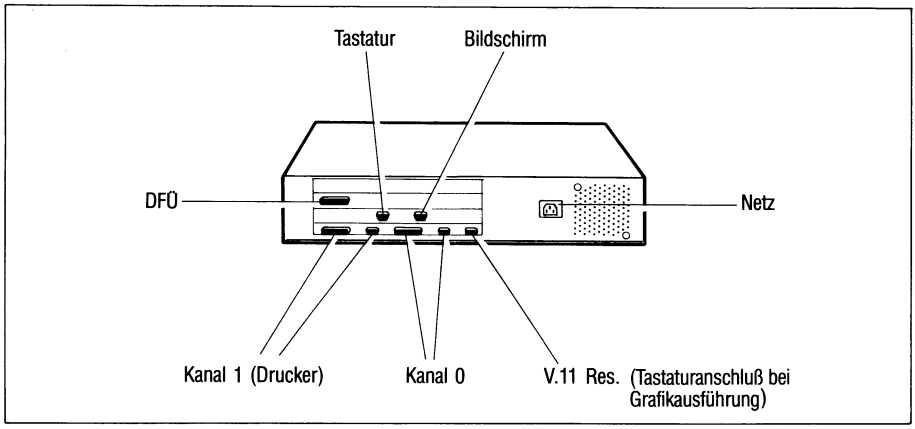


Bild 4-14 Anschlußfeld der Systemeinheit des PC-X

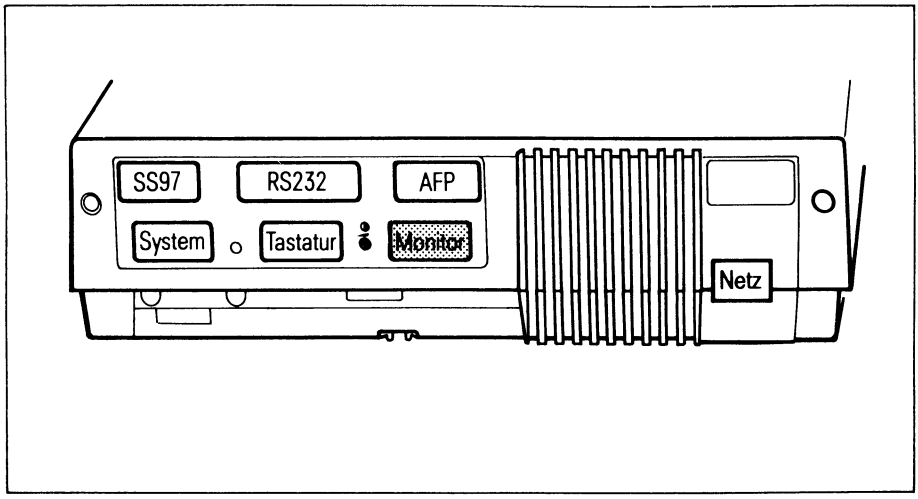


Bild 4-15 Anschlußfeld der Steuereinheit der Bedieneinheit

4.2.2 PIN-Belegung des Bildschirm-Anschlusses

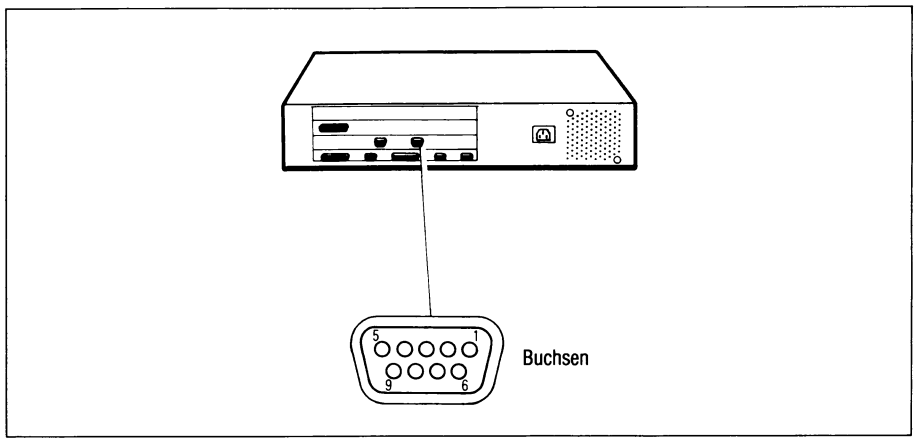


Bild 4-16 Bildschirm-Anschluß an der Systemeinheit

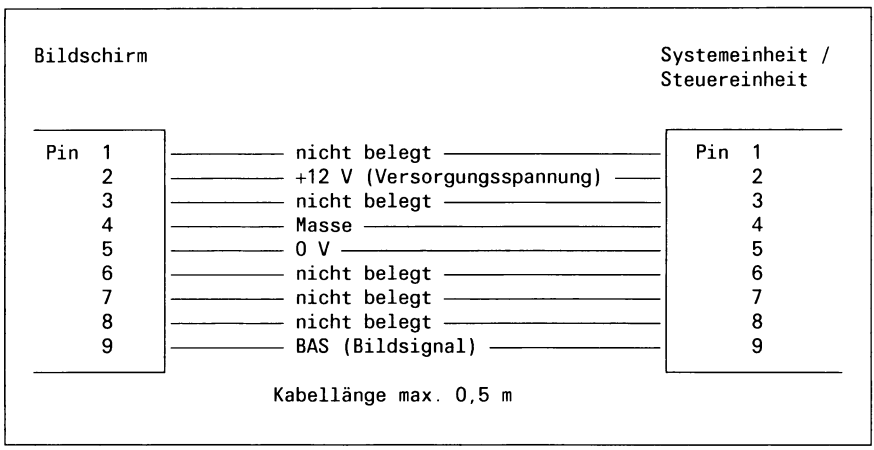


Bild 4-17 Pin-Belegung

4.4 Schnittstelle RS232

4.4.1 Pin-Belegung

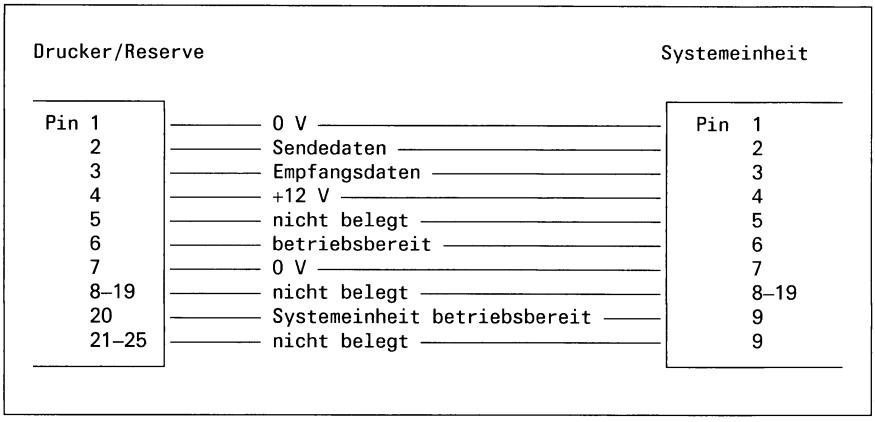


Bild 4-18 Pin-Belegung der Schnittstelle RS232

Stift	Bezeichnung nach EIA	DIN	CCITT	Erklärung	
1	PG	E1	101	Schutzerde	nicht angeschlossen
2	TD	D1	103	Sendedaten	
3	RD	D2	104	Empfangsdaten	
4	RTS	S2	105	Sendeteil einschalten	
5	CTS	M2	106	Sendebereitschaft	
6	DSR	M1	107	Betriebsbereitschaft	nicht angeschlossen
7	SG	E2	102	Signalerde	
8	DCD	M5	109	Empfangssignalpegel	nicht angeschlossen
20	DTR	S1.2	108.2	DEE betriebsbereit	
24	-	T1	113	Sendeschrittakt	nicht versorgt

4.4.2 Elektrische Kennwerte

Schnittstellen-Ersatzschaltbild

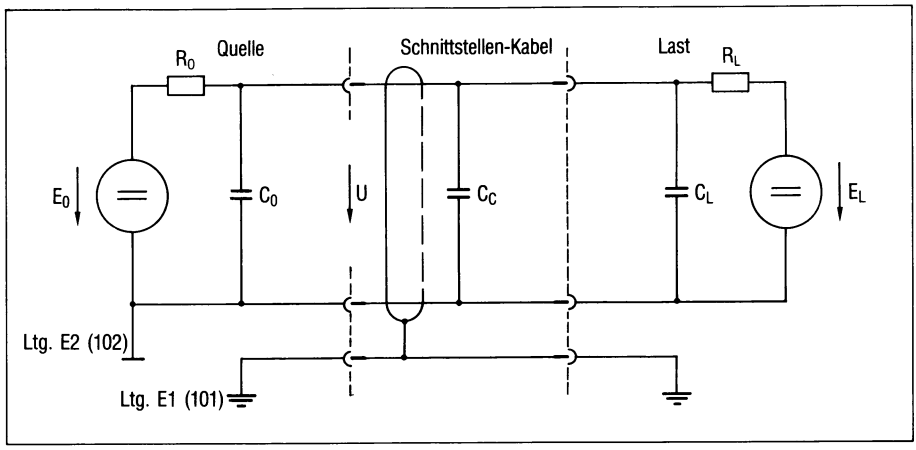


Bild 4-19 Schnittstellen-Ersatzschaltbild

U ≙ Betriebssp. an der Steckverbindung
> +3 V bzw. < -3 V bei R = 3000 Ohm

E_0 ≙ Leerlaufspannung der Quelle
+25 V ≥ E_0 ≤ -25 V

R_0 ≙ Quellenwiderstand
Bei Ausfall der Stromversorgung R_0 ≥ 300 Ohm.

C_0 ≙ Quellenkapazität, nicht definiert.

C_C ≙ Kapazität des Schnittstellenkabels
≤ 2000 pF

C_L ≙ Lastkapazität
≤ 2500 pF

R_L = Gleichstromwiderstand
3000 Ohm ≤ R_L ≤ 7000 Ohm

E_L ≙ Leerlaufspannung der Last
+24 V ≥ E_L ≤ -24 V

Logische Zustände der Schnittstellenleitungen

Datenleitungen (TD und RD)

negativ (-3 V bis -25 V) ≙ log. 1 ≙ Stoppolarität ≙ A-Polarität
positiv (+3 V bis +25 V) ≙ log. 0 ≙ Startpolarität ≙ Z-Polarität

Taktleitungen (nicht realisiert)

negativ (-3 V bis -25 V) ≙ AUS-Zustand
positiv (+3 V bis +25 V) ≙ EIN-Zustand

Steuer- und Meldeleitungen (DTR, RTS und CTS)

negativ (-3 V bis -25 V) ≙ AUS-Zustand
positiv (+3 V bis +25 V) ≙ EIN-Zustand

Undefinierter Zustand

Der Bereich zwischen +3 V und -3 V wird als Übergangsbereich gekennzeichnet. Der Zustand des Signals oder der Leitung kann nicht eindeutig definiert werden, wenn die Spannung sich in diesem Übergangsbereich befindet.

Eine Ausnahme bildet eine ausgefallene oder abgeschaltete Stromversorgung oder eine Unterbrechung der Schnittstellenleitungen.
Folgende Leitungen sind dann als im AUS-Zustand zu bewerten:
S2 (105), M1 (107), S1 (108).

Signalverzerrung

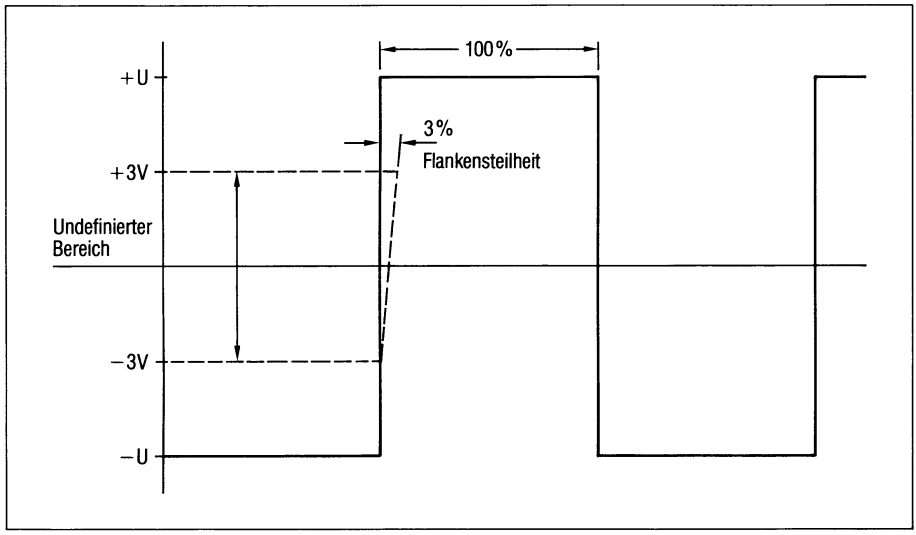


Bild 4-20 Impulsdiagramm

Die Abweichung der Flankensteilheit im undefinierten Bereich darf 3% der Schrittdauer betragen.

Beispiel:

Übertragungsgeschwindigkeit = 4800 bit/s

Schrittdauer (100%) = 208,3 ms

Abweichung (3%) = 6,249 ms

4.4.3 Bemerkungen

Signalabhängigkeiten

Wenn der PC senden will, gibt er DTR (S1) und RTS (S2) aus. Daten werden

- erst gesendet, wenn CTS (M2) aktiv ist und
- solange gesendet, wie das Peripheriegerät CTS (M2) aktiv hält.

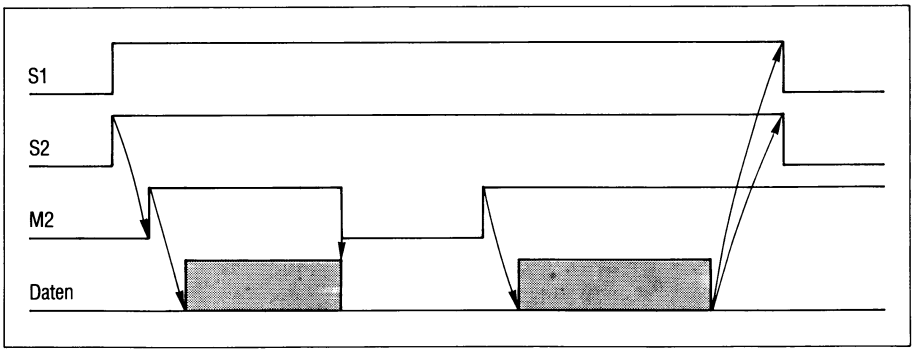


Bild 4-21 Signalabhängigkeiten

Normen

Die Norm RS232 nach EIA (Electronic Industries Associates) beschreibt eine serielle, unsymmetrische Doppelstrom-Schnittstelle zwischen Datenendeinrichtungen (DEE) und bedingt zwischen Datenendeinrichtung (DEE) und Datenübertragungseinrichtung (DÜE).
Das internationale Pendant nach CCITT ist die Schnittstellendefinition nach V.24 und V.28.
Dabei beschreibt die V.24-Schnittstellendefinition die funktionellen Eigenschaften (das Protokoll) zwischen DEE und DÜE und die V.28-Schnittstellendefinition die elektrischen Eigenschaften (Physical Layer) der Schnittstelle. Die V.24-Vereinbarungen sind jedoch viel umfangreicher als die der RS232, da sie zusätzlich auch die Wahlinformation, Hilfskanäle, Prüfschleifen usw. behandelt.

Realisierung

Die einzelnen Leitungen sind mit den Bausteinen 75188 als Sender und 75189 als Empfänger (oder kompatible Bausteine) ausgeführt mit Versorgungsspannungen von +12 V und -12 V.

Anschlußbedingungen

Die max. Kabellänge kann 30 m betragen. Die über diese geschirmte Schnittstelle miteinander verbundenen Peripheriegeräte müssen am selben Netzverteiler (220 V) angeschlossen sein, um Ausgleichsströme zu verhindern.

5 Software-Schnittstelle termcap

Was ist termcap?

termcap ist eine Programmierhilfe für C-Programmierer, mit der terminalunabhängige Anwendungen erstellt werden können. Programme, die mit Hilfe von termcap geschrieben wurden, sind leichter zu portieren und zu warten. Alle bildschirmorientierten SINIX-Standard-Anwendungen verwenden termcap. Jeder Siemens SINIX-PC enthält daher eine termcap-Datei mit dem Namen /etc/termcap, in der Steuerzeichenfolgen definiert sind. Die zum termcap-Paket gehörenden Bibliotheksroutinen werden mit dem C-Entwicklungs-System (CES) ausgeliefert.

Woher kommt termcap?

termcap wurde Ende der siebziger Jahre von Bill Joy an der Universität von Berkley (Kalifornien) entwickelt, um den von ihm entwickelten bildschirmorientierten Editor vi mit Terminals unterschiedlichen Typs ablauffähig zu machen.

Warum termcap?

Bildschirmorientierte Anwendungen wie Editoren, Emulationen oder benutzerfreundliche Bedienoberflächen nutzen in der Regel eine Fülle von Funktionen, die ihnen moderne Terminals bieten. Unglücklicherweise finden sich kaum zwei Terminals unterschiedlichen Typs, die über den gleichen Funktionsumfang oder die gleichen Steuerzeichensequenzen zu deren Auslösung verfügen.

Um dieser Vielfalt zu begegnen, kann man die Steuersequenzen vieler verschiedener Terminals in die Programme aufnehmen oder beim Betreiben von unterschiedlichen Terminals am selben Rechner mehrere Versionen des selben Programms auf der Platte halten. Beim Einsatz eines neuen Typs einer Bedieneinheit müßte jedoch in jedem Fall eine neue Version des Programms erstellt und auf der Platte abgelegt werden.

Das Problem kann behoben werden, indem man die Steuersequenzen in einer Datei ablegt. Diese wird beim Laden des Programms jedesmal eingelesen und kann bei Bedarf, falls sich z.B. ein Parameter ändert, leicht verändert und erweitert werden. Um zu verhindern, daß für viele Anwen-

dungen eigene Dateien mit eigenen Formaten und eigene Auswertungsroutinen erstellt werden, sollte deshalb bei der Erstellung neuer Programme termcap verwendet werden.

Neben der Hinterlegung von Steuerzeichen sind andere Anwendungen von termcap denkbar, z.B. die Definition von Eingabesequenzen, um beim Einsatz eines Programms die Möglichkeit zu erlauben, bestimmte Funktionen durch Funktionstasten auf der Tastatur auszulösen.

Wie funktioniert termcap?

termcap besteht aus einer Bibliothek mit C-Routinen und einer Datei, in der Beschreibungen von Terminals abgelegt sind. Diese Beschreibungen haben eine spezielle Form, die von den Bibliotheksroutinen verstanden wird.

Programme, die mit termcap arbeiten, haben Routinen aus dieser Bibliothek miteingebunden. Mit Hilfe der Routinen wird die termcap Datei teilweise eingelesen und die eingelesenen Daten verarbeitet, d.h. die in der termcap-Datei enthaltenen Informationen werden in programmeigene Datenbereiche und Strukturen kopiert. Da in den Angaben neben einfachen Steuersequenzen auch komplexe Formate zur dynamischen Generierung von Sequenzen nach Angaben von Parametern, z.B. Cursor-Positionierungsanweisungen, enthalten sind, werden Funktionen angeboten, die diese Formate auswerten (siehe auch CES-Manual).

5.1 Die termcap-Datei

Eine termcap-Datei ist eine Datei, die Beschreibungen der Funktionen von Terminals im termcap-Format enthält. termcap-Dateien können mehrfach auf einem Rechner vorhanden sein. Es existiert in /etc/termcap jedoch eine zentrale termcap-Datei, auf die von den termcap-Routinen zugegriffen wird, sofern vom Benutzer nicht explizit eine andere termcap-Datei bestimmt wurde. Sie enthält nur Informationen in Form von abdruckbaren Zeichen, d.h. eventuell in einer Steuersequenz vorhandene nicht abdruckbare Zeichen werden symbolisch definiert.

5.1.1 Aufbau und Inhalt einer termcap-Datei in SINIX

Die termcap-Datei enthält einen oder mehrere Terminaleinträge. Ein Terminaleintrag besteht aus einem oder mehreren Feldern und darf eine Länge von 1024 Zeichen nicht überschreiten.

Als Beispiel für einen Terminaleintrag hier der sogenannte Standard-Eintrag aus der termcap-Datei der Siemens SINIX-PC's:

```
standard|97801:\
        :co#80:li#24:am:bs:bt=\E[Z:cm=\E[%i%d;%dH:nd=\E[C:\
        :up=\E[A:ce=\E[0K:cd=\E[0J:cl=\E[H\E[2J:dl=\E[M:al=\E[L:\
        :sr=\E[T:sf=\E[S:ae=\E[2m:a\=EE[m:so=\E[7m:se=\E[m:ti=\E)w:\
        :ic=\E[@:dc=\E[P:us=\E[4m:ue=\E[m:ta=^I:cs=\E[%i%d;%dr:\
        :ku=\E[A:kd=\E[B:kr=\E[C:kl=\E[D:kh=\E[H:\
        :k0=\E[@:k1=\E[P:k2=\Eo:k3=\Ep:k4=\E[L:k5=\E[M:\
        :k6=\E\072:k7=\E9:k8=\E[T:k9=\E[S:l0=\E>:l1=\Em:l2=^D:\
        :F1=\E\:F2=\E;:F3=\E":F4=\E#:F5=\E$:\
        :F6=\E%:F7=\E&:F8=\E':F9=\E<:Fa=\E=:\
        :P1=\E@:P2=\EA:P3=\EB:P4=\EC:P5=\ED:\
        :P6=\EF:P7=\EG:P8=\EH:P9=\EI:Pa=\EJ:\
        :Pb=\EK:Pc=\EL:Pd=\EM:Pe=\EN:Pf=\EO:\
        :Pg=\EP:Ph=\EQ:Pi=\E\:Pj=\Ed:Pk=\ET:\
        :y0=^NB^O:y1=^NC^O:y2=^ND^O:y3=^NE^O:\
        :y4=^NA^O:y5=^N@^O:y6=\E[2;7m:\
        :y7=\E[m:ya=\012:yb=\177:yc=\015:yd=^R:\
        :ye=^X:yf=^H:Pl=\Eg:GS=^N:GE=\O:GV=«:GH=A:
```

Ein Terminaleintrag beginnt mit einem Feld, das den gesamten Eintrag kennzeichnet. Das Feld enthält mehrere symbolische Bezeichnungen, getrennt durch senkrechte Striche. In SINIX bezeichnet der erste Name den gesamten Eintrag mit Hinblick auf seinen hauptsächlichen Verwendungszweck. Der zweite Name ist eine Kombination aus der Typenbezeichnung des Terminals und einem aus dem ersten Namen abgeleiteten Präfix. Es können noch weitere Namen angegeben werden.

In den nachfolgenden Feldern werden die einzelnen Funktionen und Dimensionen des Terminals beschrieben. Alle Feldeinträge beginnen mit einem zwei Zeichen langen Code, der das Feld symbolisch bezeichnet. Es gibt drei Arten von Feldern:

Typ 1 Boolesche Felder zeigen das Vorhandensein bestimmter Funktionen an:
z.B. :bs: für ein Terminal, das in der Lage ist, mit dem Backspace-Zeichen (in C durch '\b' dargestellt) um ein Zeichen nach links zu gehen.

Typ 2 Numerische Felder geben die Grösse eines bestimmten Bereichs an: so wird z.B. die Anzahl von 24 verfügbaren Zeilen auf dem Bildschirm durch :li#24: angezeigt.

Typ 3 Zuweisungsfelder, in denen Steuerzeichensequenzen definiert werden. Das Feld für die Funktion 'Löschen Zeile' z.B. ist :dl=\E[M: im Eintrag für die Bedieneinheit vom Typ 97801.

5.1.2 Definition von Steuerzeichenfolgen

Nicht abdruckbare Zeichen

Im Beispiel zu Typ 3 finden wir ein nicht abdruckbares Sonderzeichen ('\E') in einer Sequenz eingetragen. Dies geschieht durch sog. Escape-Sequenzen, die an die der C-Sprache angelehnt sind. So gibt es die Sequenzen '\n', '\r', '\t', '\b' und '\f' für die Steuerzeichen New-Line, Carriage-Return, Tab, Backspace und Form-Feed respektive. Die Sequenz '\E' wird in das Steuerzeichen Escape umgewandelt.
Weiterhin ist es möglich, beliebige Bit-Kombinationen durch ein '\', gefolgt von drei Oktalziffern, darzustellen. So muß z.B. das Zeichen ':' mit '\072' eingetragen werden, da es sonst als Feldtrennzeichen interpretiert würde.

Möchte man eine binäre Null definieren, muß sie als '\200' kodiert sein. Andernfalls würde bei der Verarbeitung der Steuerzeichenfolge an dieser Stelle abgebrochen werden, da termcap wie alle C-Programme nur mit Zeichenketten arbeitet, die durch eine binäre Null abgeschlossen sind.
Das höchstwertige Bit wird später von den termcap-Ausgaberoutinen zurückgesetzt und tatsächlich '\000' an das Terminal gesendet.

Steuerzeichen können auf der Tastatur durch gleichzeitiges Drücken der Taste CTRL und einer weiteren Taste erzeugt werden. Falls gewünscht, kann ein Steuerzeichen auch in Anlehnung an diese Möglichkeit durch die Notation '^x' dargestellt werden, z.B. '^D' für die Taste END. Die Fluchtsymbole '^' und '\' werden selbst durch '\^' und '\ \' festgelegt.

5.1.3 Wartezeiten nach Terminalfunktionen

Bei der Vereinbarung von Steuerzeichensequenzen kann nach dem Zeichen '=' optional eine Wartezeit in Millisekunden angegeben werden, die durch Senden von Füllzeichen nach dem Aussenden der Sequenz erreicht wird. Die Angaben können als einfacher Integerwert wie z.B. '30' oder in der Form '4.5*' erfolgen. Letztere Form gibt die Anzahl von Millisekunden pro von der Operation betroffenen Einheiten an. Moderne Terminals, wie die der Siemens PC's, benötigen - außer nach te (RIS = ESC c, ca. 1,5 sec.) - keine Wartezeiten nach komplexen Funktionen.

In diesem Zusammenhang ist die Bibliotheksroutine tputs zu beachten, die diese Angaben decodiert.

5.1.4 Cursor-Bewegungen

Einige Steuerzeichenfolgen, wie z.B. solche zur absoluten Cursor-Bewegung, müssen in der Regel mit aktuellen Parametern versorgt an das Terminal geschickt werden. Bei der Beschreibung solcher Funktionen werden anstelle der Parameter Platzhalter eingetragen, die später von der Bibliotheksroutine tgoto durch die richtigen Werte ersetzt werden. Das Format dieser Platzhalter ist ähnlich wie in der C-Bibliotheksroutine printf. Für die unterschiedlichen Terminaltypen gibt es eine Vielzahl von speziellen Formaten, wie folgt:

Format	Beschreibung
%%	steht für %
%d	wie in printf
%2	wie %2d in printf
%3	wie %3d in printf
%.	wie %c in printf, d.h. 1 Byte wie übergeben
%+x	addiert x zum übergebenen Wert, dann wie %.

Die folgenden Formate werden zusätzlich zu den o.g. eingetragen, um das allgemeine Format der Umwandlung zu steuern, führen jedoch nicht zur Ausgabe weiterer Zeichen:

Format	Beschreibung
%>xy	falls übergebener Wert größer x addiere y
%r	vertausche Reihenfolge von Zeilen- und Spaltenangabe
%i	erhöhe Zeilen- und Spaltenwert um 1
%n	führe 'exklusiv oder' mit oktal 0140 durch
%B	BCD-Ausgabe (Binary Coded Decimal) (16*(x/10)) + (x%10)
%D	BCD - Umgekehrte Kodierung (x-2*(x%16))

Das Feld 'cm' für die Terminals der SINIX-PC's mit der Folge '\E[%i%d;%dH' bedeutet daher: Ausgabe der Zeilen- und Spaltenadresse als Dezimalzahl in ASCII-Zeichen beginnend mit 1, z.B. für Zeile 4, Spalte 36 die Folge '\E[4;36H'.

Ein anderes Beispiel (für das Siemens-Terminal 6265):
'^P%.%.', d.h. nach CTRL P folgt die Adresse binär in je einem Byte.

5.1.5 Funktionsfeldkennungen

Grundsätzlich ist die Bezeichnung eines Feldes frei wählbar, jedoch hat Bill Joy durch seinen vi-Editor Konventionen dafür geschaffen, wie bestimmte Terminalfunktionen zu bezeichnen sind.

Hier ein Auszug aus den in der termcap-Datei der SINIX-PC's vergebenen Feldbezeichnungen:

Feldbezeichnung	Typ	Beschreibung
ae	3	Ausschalten des alternativen Zeichensatzes
al	3	Einfügen Zeile
am	1	Cursor springt beim Erreichen des Zeilenendes auf nächste Zeile
as	3	Einschalten des alternativen Zeichensatzes
bc	3	Backspace-Folge, wenn bs nicht gesetzt
bs	1	Terminal kennt Backspace-Zeichen '\b'
bt	3	Rückwärts-Tabulator-Zeichen
cd	3	Löschen ab Cursor-Position bis Bildschirmende
ce	3	Löschen ab Cursor-Position bis Zeilenende
cl	3	Löschen Bildschirminhalt
cm	3	Cursor bewegen
co	2	Anzahl Spalten pro Zeile
cs	3	Einstellen Scroll-Bereich
dc	3	Löschen Zeichen
dl	3	Löschen Zeile
do	3	Cursor eine Zeile nach unten
ho	3	Cursor an Bildschirmanfang
ic	3	Einfügen Zeichen
is	3	Terminal-Initialisierungs-Sequenz
kd	3	Von der Taste [↓] gesendete Zeichenfolge
kh	3	Von der Taste [↖] gesendete Zeichenfolge
kl	3	Von der Taste [←] gesendete Zeichenfolge
kr	3	Von der Taste [→] gesendete Zeichenfolge
ku	3	Von der Taste [↑] gesendete Zeichenfolge
li	2	Anzahl Zeilen am Bildschirm
nd	3	Cursor nach rechts
se	3	Ausschalten Invers-Modus
sf	3	Scrollen nach oben
so	3	Einschalten Invers-Modus
sr	3	Scrollen nach unten
ta	3	Tabulator-Zeichen
ti	3	Terminalinitialisierungssequenz vor der ersten Cursor-Bewegung
te	3	Terminal rücksetzen (mit Verzögerung)
ue	3	Ausschalten Unterstreichen-Modus
up	3	Cursor nach oben
us	3	Einschalten Unterstreichen-Modus

5.2 Einträge in der /etc/termcap-Datei

Von vielen Programmen wird der termcap-Eintrag auch dazu verwendet, die Zeichenfolgen zu definieren, die von Funktionstasten gesendet werden. Bis auf wenige Ausnahmen, z.B. 'kh', 'kl' usw., gibt es für die Benennung dieser Felder keine allgemein verbreitete Konvention, so daß heute nahezu jede Anwendung hier ihre eigenen Feldbezeichnungen kennt. Da andererseits die Länge eines Terminaleintrags in einer termcap-Datei auf 1024 Zeichen beschränkt ist und z.T. sehr viele Definitionen pro Anwendung existieren, mußten in SINIX spezielle Einträge für einige Standard-Anwendungen in der Datei /etc/termcap gemacht werden, die leider viele gleiche Felder beinhalten.

Zu den einzelnen Einträgen können folgende Anmerkungen gemacht werden:

Der Eintrag für das COBOL-Laufzeitsystem (LZS) darf eine Länge von 512 Zeichen nicht überschreiten, da im LZS nicht mehr Platz für ihn reserviert ist. Er enthält das Nötigste, um einen Bildschirm zu bedienen, sowie eine Definition des Cursor-Blocks und der ersten 10 Funktionstasten in 'k0' bis 'k9'.

Der 'menu'-Eintrag ist für das Menü-System auf den SINIX-PC's eingerichtet. Das Menü-System verwendet einige der Felder auf recht spezielle Weise, so daß die allgemeine Nutzung dieses Eintrags nicht empfehlenswert ist. Es werden eine Reihe von Semi-Grafik-Zeichen definiert.

Der Multiplan-Eintrag enthält die Definitionen der Funktionstasten für Multiplan (Microsoft).

Der 'em9750'-Eintrag erlaubt die Veränderung der Funktionstastenbelegung und die Einstellung verschiedener anderer Parameter in der 9750-Nachbildung des TRANSIN-Pakets. Für den normalen SINIX-Betrieb ist er nicht geeignet.

Der Eintrag 'standard' wird von anderen Anwendungen benützt, so z.B. vom CED-Editor. Neben der üblichen und vollständigen Beschreibung des 97801-Terminals enthält er die Definition des gesamten Cursor- und Funktionstastenfeldes. Hier eine Auflistung der Zuordnung zwischen Tastenbeschriftung und termcap-Feldbezeichnung:

Taste	Feldbezeichnung
F1 - F9	P1 - P9
F10 - F20	Pa - Pk
HELP	10
START	11
END	12
INS CHAR	k0
DEL CHAR	k1
INS WORD	k2
DEL WORD	k3
INS LINE	k4
DEL LINE	k5
Springen Wort links	k6
Springen Wort rechts	k7
dicker Pfeil unten	k8
dicker Pfeil oben	k9

Neben den o.g. Einträgen befindet sich in /etc/termcap ein weiterer Eintrag pro angeschlossenem Terminal. Die Bedeutung dieser Einträge wird im Abschnitt Terminalinitialisierung näher beschrieben.

5.3 Umgebungsvariablen

Jedes SINIX-Programm erhält beim Laden vom aufrufenden Programm, in der Regel der Shell, eine Reihe von Schlüsselwortparametern übergeben, die seine sog. 'Umgebung' darstellen. Unter diesen Umgebungsvariablen befinden sich auch Parameter zur Steuerung der termcap-Routinen.

Ein SINIX-Benutzer kann sich die aktuellen Umgebungsvariablen mit dem Kommando 'printenv' ausgeben lassen. Der C-Programmierer verwendet die Funktion 'getenv( )' zum Suchen nach einer bestimmten Umgebungsvariablen.

5.3.1 Variable TERM

Mit der Umgebungsvariablen TERM wird einem Programm der Name des Eintrags übergeben, der mit termcap verwendet werden soll, d.h. mit welchem Terminaltyp gearbeitet wird. Für den SINIX-Benutzer wird diese Variable bereits beim 'Einloggen' automatisch gesetzt.

5.3.2 Variable TERMCAP

Die Umgebungsvariable TERMCAP bezeichnet die Quelle, aus der die Einträge zu holen sind. Ist der Inhalt der Variablen ein absoluter Dateiname (Pfadname mit '/' am Anfang) wird diese Datei als termcap-Datei eröffnet.
Anderenfalls wird angenommen, daß die TERMCAP-Variable einen termcap-Eintrag direkt als Wert enthält, und es wird diese Zeichenfolge verwendet (Voraussetzung: Der Inhalt der TERM-Variablen stimmt überein mit dem Namen im termcap-Eintrag).
Auf diese Weise können neue termcap-Einträge getestet oder von nicht privilegierten SINIX-Benutzern eigene Einträge verwendet werden.

Sollte keine TERMCAP-Variable vergeben oder der Inhalt ungültig sein, wird standardmäßig aus der Datei /etc/termcap gelesen.

Beispiel 1:

Eigene termcap Datei:

```
TERM=own
TERMCAP=/usr/ben/termcap
export TERM TERMCAP
programm
```

Beispiel 2:

termcap Eintrag in der TERMCAP-Variablen:

```
TERM=own
TERMCAP='own|:co#80:li#24:bs:cm=\E[%i%d;%dH:  ...'
export TERM TERMCAP
programm
```

5.4 Routinen in der termcap-Bibliothek

Die termcap-Bibliothek enthält eine Reihe von Routinen, mit denen die Einträge aus der termcap-Datei gelesen und einzelne Felder extrahiert werden können. Außerdem gibt es Routinen zur Auswertung von 'cm'-Feldern und zur Ausgabe von Steuerzeichenfolgen zusammen mit Füllzeichen.

Die Routinen in der termcap-Bibliothek werden beim Binden eines C-Programms durch die Angabe '-ltermcap' eingebunden.

Beispiel:

```
cc programm.c -o programm -ltermcap
```

5.4.1 Funktion tgetent

Die Funktion tgetent liest einen Eintrag aus der termcap-Datei:

```
tgetent(bp, name)
char *bp, *name;
```

bp zeigt auf einen Speicherbereich von mindestens 1024 Bytes Länge, in den der termcap-Eintrag gelesen wird. Dieser Speicherbereich kann später, wenn die Steuerzeichenfolgen in programmeigene Strukturen kopiert wurden, wieder freigegeben werden. Die Adresse des Speicherbereichs wird von tgetent abgespeichert, so daß bei Aufrufen an die anderen termcap-Routinen diese nicht angegeben werden muß. Symbolisch definierte, nicht abdruckbare Zeichen werden erst durch die nachfolgend beschriebenen Routinen umgesetzt.

name ist ein Zeiger auf eine null-terminierte Zeichenkette, die den einzulesenden Eintrag spezifiziert.

tgetent hat den Rückgabewert 1 wenn alles in Ordnung war und der Eintrag nach bp gelesen werden konnte. Der Wert -1 wird zurückgegeben, wenn die termcap-Datei nicht eröffnet werden konnte.
Der Wert 0 zeigt an, daß in der angegebenen termcap-Datei kein Eintrag für name existiert.

tgetent sucht in der 'Umgebung' des Programms nach der Variablen TERMCAP und wertet ihren Inhalt in der oben beschriebenen Weise aus.

Programmbeispiel

```
int ret;
char tcbuffer[1024],*getenv();

if ( ( ret = tgetent(tcbuffer, getenv("TERM")) ) <= 0 ) {
        printf("termcap: Fehler beim Einlesen (%d)\n", ret);
        exit(1);
}
```

5.4.2 Funktion tgetnum

tgetnum sucht ein Feld vom Typ 2 im eingelesenen Eintrag:

```
tgetnum(id)
char *id;
```

id ist ein Zeiger auf eine null-terminierte Zeichenkette, die die Bezeichnung eines Feldes enthält. Das angegebene Feld wird im Puffer gesucht und die Wertangabe als Integer zurückgeliefert. Falls das Feld nicht existiert, wird -1 zurückgegeben.

Programmbeispiel

```
int lines;

if ( ( lines = tgetnum("li") ) == -1 ) {
        printf("termcap: Zeilenanzahl nicht angegeben\n");
        exit(1);
}
```

5.4.3 Funktion tgetstr

tgetstr sucht nach einem Feld vom Typ 3:

```
char *
tgetstr(id, area)
char *id, **area;
```

Das durch den Zeiger id spezifizierte Feld wird im Puffer gesucht und an der Adresse hinterlegt, die über den Zeiger area eingetragen wurde.

Zu beachten ist, daß area kein Zeiger auf eine Zeichenkette ist, sondern als Zeiger auf einen Zeiger definiert ist. Wie im folgenden Beispiel zu sehen, muß daher die Adresse eines Zeigers auf einen Speicherbereich übergeben werden und nicht der Zeiger selbst. Der Grund für diesen komplizierten Mechanismus besteht darin, daß tgetstr neben dem Suchen und Abspeichern der Steuerzeichenfolge gleichzeitig den Zeiger auf den privaten Speicherbereich um die Länge der gefundenen Steuerzeichenfolge erhöht, so daß dieser anschließend auf den nächsten freien Speicherplatz zeigt.

tgetstr liefert als Rückgabewert die Adresse des Speicherplatzes, an dem die Steuerzeichenfolge abgelegt wurde.

Programmbeispiel

```
char strbuf[1024];
char *bp, *cm, *tgetstr();

bp = strbuf;
if ( ( cm = tgetstr("cm", &bp) ) == (char *)0 ) {
        printf("termcap: Cursorpositionierung unmoeglich\n");
        exit(1);
}
```

5.4.4 Funktion tgetflag

tgetflag sucht nach einem Feld vom Typ 1:

```
tgetflag(id)
char *id;
```

tgetflag liefert den Wert 1, wenn das Feld im eingelesenen Eintrag vorhanden ist, 0 falls nicht.

id ist ein Zeiger auf eine Zeichenkette.

Programmbeispiel

```
int canbs = 0;
char *bc, *tgetstr();

if ( tgetflag("bs") )
        canbs++;
else {
        if ( ( bc = tgetstr("bc", &bp) ) == (char *)0 ) {
                fprintf(stderr, "termcap: Kein bs und kein bc\n");
                exit(1);
        }
}
```

5.4.5 Funktion tputs

tputs dient zur Ausgabe der Steuerzeichenfolgen:

```
extern char PC;
extern short ospeed;

tputs(cp, affcnt, outc)
char *cp;
int affcnt;
int (*outc)();
```

Tputs wertet die Wartezeitangaben am Beginn von Steuerzeichenfolgen aus und übergibt die nötige Anzahl von Füllzeichen im Anschluß an die Steuerzeichenfolge. Die Ausgabe von tputs wird zeichenweise an die Funktion übergeben, deren Adresse im Zeiger outc steht. cp ist ein Zeiger auf die Steuerzeichenfolge, affcnt enthält die Anzahl der von der Operation betroffenen Einheiten.

Zur Berechnung der Anzahl von Füllzeichen, die nötig sind, um die angegebene Wartezeit zu erreichen, muß in der Variable ospeed die Übertragungsgeschwindigkeit zum angeschlossenen Terminal eingetragen sein. Dieser Wert kann durch einen Aufruf an den SINIX-Terminal-Treiber erhalten werden. Für Terminals mit Übertragungsgeschwindigkeiten größer als 9600 bit/s wird keine Wartezeitbearbeitung durchgeführt.

Falls ein anderes Füllzeichen als '\0' verwendet werden soll, muß es in die Variable PC eingetragen werden.

Programmbeispiel

```
#include <sgtty.h>

extern char PC;
extern short ospeed;

int outc();
struct sgttyb ttymodes;
char *cp;
gtty(1, &ttymodes);
ospeed = ttymodes.sg-ospeed;
         .
         .
         .
tputs(cp, 10, outc);
         .
         .
         .
```

Die Funktion outc:

```
outc(c)
char c;
{
         putchar(c);
}
```

5.4.6 Funktion tgoto

Die Routine tgoto wird zum Dekodieren der 'cm'- und 'cs'-Felder verwendet:

```
extern char *UP;
extern char *BC;

char *
tgoto(cm, destcol, destline)
char *cm;
int destcol, destline;
```

Die Routine versucht die Ausgabe von '\n', '^D' und '\000' zu verhindern, da diese u.U. von Terminaltreibern besonders behandelt werden. Um dennoch die gewünschte Adresse zu erreichen, wird an eine Koordinate in der unmittelbaren Umgebung gesprungen und durch die in UP und BC hinterlegten Folgen auf die richtige Stelle positioniert. UP und BC sind in der termcap-Bibliothek definierte Zeiger auf Zeichenketten, die vom aufrufenden Programm versorgt werden müssen.
Solche Programme sollten in der Regel auch den Modus 'Tabulatorzeichen expandieren' ausschalten, da falls möglich ein Tabulatorzeichen zur Positionierung verwendet wird.

Falls ein angegebenes %-Format von tgoto nicht verstanden wird, gibt die Funktion die Zeichenfolge 'OOPS' zurück.

Programmbeispiel

```
int outc();
char *tgoto();
char *cm;
extern char *UP, *BC;

UP = tgetstr("up", &bp);
if ( !canbs )
        BC = bc;
tputs(tgoto(cm, 35, 3), 3, outc);
```

5.5 Terminal initialisieren

5.5.1 /etc/getty

Das Systemprogramm /etc/getty (s. a. SINIX, Buch 1) hat neben der Entgegennahme der Benutzerkennung die Aufgabe, den SINIX-Terminal-Treiber und das SINIX-PC-Terminal einzustellen. So muß z.B. der Typ der an das Terminal angeschlossenen Tastatur in das Terminal geladen werden. Die einzustellenden Parameter und Steuerzeichenfolgen befinden sich in den Dateien /etc/ttytype und /etc/termcap.

Nach der Einstellung der Leitungsgeschwindigkeit wird in /etc/termcap ein Eintrag mit dem Namen des Terminalports, z.B. tty00, gesucht und das darin enthaltene 'is'-Feld an das Terminal gesendet. Dadurch kann eine port-spezifische Initialisierung durchgeführt werden. Als Port wird ein Terminalanschluß am SINIX-PC bezeichnet.

Zusätzlich enthält die Portbeschreibung in der termcap-Datei die Länderkennung der Tastaturbelegung im Feld LK. /etc/keyload wird von /etc/getty angestoßen und lädt die entsprechende Tastaturbelegungstabelle (siehe 8).

Nachdem der Typ des Terminals in /etc/ttytype festgestellt wurde, wird zusätzlich das 'ti'-Feld des für diesen Typ vorhandenen Eintrags, auf SINIX-Anlagen der Eintrag '97801', ausgesendet, da anschließend eine Cursor-Positionierung durchgeführt wird.

Am Ende der termcap-Datei befinden sich die Sequenzen für die Initialisierung der Bedieneinheiten.

```
console|:is=\E[21u\08 \08\E(K\E[7u:LK=deut
tty01|:is=\E[21u\08 \08\E(K\E[6u:LK=inter
tty03|:is=\E[21u\08 \08\E(K\E[6u:LK=inter
tty04|:is=\E[21u\08 \08\E(K\E[6u:LK=inter
```

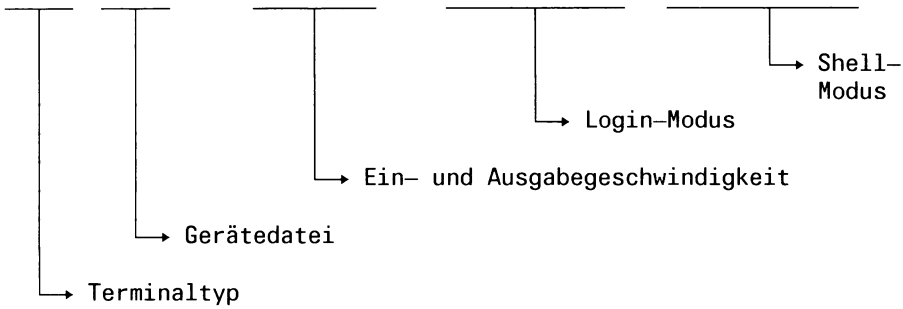
5.5.2 /bin/login

/bin/login stellt verschiedene Umgebungsvariablen ein, u.a. die Variable TERM. Der dem Port zugeordnete Terminaltyp steht in /etc/ttytype.

5.5.3 /etc/ttytype

Der angeschlossene Terminaltyp steht in Feld 1, die Gerätedatei in Feld 2.

```
97801  console  38400  38400  P0;RW;NL1;CR1  P0;EC;CM;F0;T0
97801  tty00    38400  38400  P0;RW;NL1;CR1  P0;EC;CM;F0;T0
97801  tty01    38400  38400  P0;RW;NL1;CR1  P0;EC;CM;F0;T0
```



5.6 Eingabesequenzen verarbeiten

Die termcap-Routinen enthalten keine Unterstützung von Eingabesequenzen, wie z.B. Cursor-Steuertasten oder Funktionstasten. Da diese Funktionen jedoch sehr häufig benötigt werden, soll hier ein Beispiel gezeigt werden, wie die Eingabefolgenerkennung durchgeführt werden kann. Die Routinen können in die termcap-Bibliothek aufgenommen werden und stehen dann allen C-Programmen zur Verfügung.

5.6.1 Funktion tbuild

Zu Beginn müssen alle Eingabesequenzen in einen binären Baum eingetragen werden, über den später die Folgen dekodiert werden. Jeder Folge wird ein Funktionsschlüssel mitgegeben, der nach erfolgreicher Decodierung zurückgeliefert wird. Der Speicherplatz für den binären Baum wird dynamisch vom System angefordert.

```
tbuild(path, key)
char *path;
int key;
```

path ist eine Zeichenkette, wie sie z.B. von tgetstr zurückgeliefert wird. Diese Zeichenfolge muß eingegeben werden, damit die mit key bezeichnete Funktion erkannt wird. key sollte außerhalb des Bereichs der ASCII-Zeichen liegen, damit normale Zeichen ungehindert durchgeschleust werden.

tbuild gibt -1 zurück, falls kein weiterer Ast wegen Speicherplatzmangel angelegt werden kann. Es wird nicht überprüft, ob ein Pfad mehrmals definiert wird.

5.6.2 Funktion txtract

txtract liest solange Zeichen für Zeichen, bis EOF oder eine Sequenz erkannt ist.

```
txtract(nextc)
int (*nextc)();
```

nextc ist ein Zeiger auf eine Funktion, die bei jedem Aufruf ein Zeichen oder -1 bei EOF liefert.

Rückgabewert ist der gesuchte Funktionsschlüssel, ein eingelesenes Zeichen oder -1 falls EOF erreicht wurde.

5.6.3 Anwendungsbeispiele

Erstellen des Baums:

```
#include <stdio.h>

#define TASTE_1 200

encode()
{
        char buffer[10];
        char *bp;
        bp = buffer;
        if ( tbuild(tgetstr("P1", &bp), TASTE_1) == -1 )
                fprintf(stderr, "tbuild: kein Speicherplatz\n");
}
```

Rückgewinnung von Eingabesequenzen:

```
#include <stdio.h>

decode()

{
        register int c;
        int nextc();

        for ( ; ; ) {
                switch ( c = txtract(nextc) ) {
                        case TASTE_1:
                                todo();
                                break;
                        case -1:
                                exit(0);
                                break;
                        default:
                                input(c);
                                break;
                }
        }
}

/*
 *      Da getchar ein Macro ist,
 *      muß es in eine Funktion verpackt werden.
 */

nextc()
{
        return getchar();
}
```

5.6.4 tbuild und txtract

Das folgende Quellprogramm finden Sie auf der Diskette 'SSHB' unter dem Dateinamen tbuild.c.
Ein kleines Demonstrationsprogramm zur Anwendung der Funktionen tbuild und txtract hat den Namen txample.c. Siehe außerdem auch c-get.c und c-test.c

```
static char SCCSID[] = "@(#)tbuild.c 1.2 85/05/07";
/* Mods:
 *      m01     jgr     Endekriterium in Sequenzpuffer: '\200' statt ' '.
 */

/*
 *      Funktionen tbuild und txtract
 */

/*
 *      Ast des binaeren Baums
 */

typedef struct tnode {          /* Knoten */
        struct tnode *tnext;    /* Zeiger naechstes Element */
        struct tnode *talt;     /* Zeiger alternatives Element */
        int          tkey;      /* Funktionskode */
} *node;

/*
 *      Die Umsetzschiene und der Sequenzpuffer
 */

static node tbase[128];
static char tseqbuf[20], *tbp;

/*
 *      Anfordern vom Speicherplatz fuer einen Ast
 */

static node
getnode()
{
        register node tp;
        node malloc();

        if ( tp = malloc(sizeof (struct tnode)) ) {
                tp->tnext = (node)0;
                tp->talt  = (node)0;
                tp->tkey  = 0;
        }
        return tp;
}

/*
 *      Erstellen eines binaeren Baums
 */

tbuild(path, fkey)
register char *path;
int fkey;
{
        register node tp;

        if ( path == (char *)0 || *path == ' ' )
                return 0;
        if ( ( tp = tbase[*path & 0177] ) == (node)0 ) {
            if ( ( tp = getnode() ) == (node)0 )
                return -1;
        }
        tbase[*path & 0177] = tp;
        while ( *++path ) {
            if ( tp->tnext ) {
                if ( tp->tkey == *path )
                    tp = tp->tnext;
                else {
```

```
                while ( tp->tkey != *path ) {
                    if ( tp->talt )
                        tp = tp->talt;
                    else {
                        if ((tp->talt=getnode()) == (node)0)
                            return -1;
                        tp = tp->talt;
                        tp->tkey = *path;
                        if ((tp->tnext=getnode()) == (node)0)
                            return -1;
                        break;
                    }
                }
                if ( tp->tnext )
                    tp = tp->tnext;
            }
        } else {
            if ( ( tp->tnext = getnode() ) == (node)0 )
                return -1;

            tp->tkey = *path;
            tp = tp->tnext;
        }
        tp->tkey = fkey;
        return 0;
}

/*
 *      Dekodieren einer Eingabesequenz
 */

txtract(tnextchar)

int (*tnextchar)();
{
        register node tp;
        register int c;

        if ( tbp ) {
                if ( *tbp != '\200' )                           /* m01 */
                        return *tbp++;
        }
        tbp = tseqbuf;
        c = (*tnextchar)();
        if ( c == -1 || ( tp = tbase[c & 0177] ) == (node)0 ) {
                tbp = (char *)0;
                return c;
        }
        *tbp++ = c;
        while ( tp->tnext ) {
                if ( ( c = (*tnextchar)() ) == -1 ) {
                        *tbp = '\200';                          /* m01 */
                        tbp = tseqbuf;
                        return *tbp++;
                }
                *tbp++ = c;
                if ( c == tp->tkey )
                        tp = tp->tnext;
                else {
                        while ( c != tp->tkey ) {
                                if ( tp->talt )
                                        tp = tp->talt;
                                else {
                                        *tbp = '\200';          /* m01 */
                                        tbp = tseqbuf;
                                        return *tbp++;
                                }
                        }
                        tp = tp->tnext;
                }
        }
        tbp = (char *)0;
        return tp->tkey;
}
```

6 Bildschirmfunktionen und Datenformate

6.1 Kommandoübersicht und Kurzbeschreibung der Normelemente

In den Tabellen bedeutet S = Standard nach Netz-Ein und RESET der Bedieneinheit.

6.1.1 Kommandos zum Modifizieren der Zeichensätze

Kommando	S	Erklärung
ESC (@	X	International → G0 (ladbare Hälfte)
ESC) @	X	International → G1
ESC (B		International A → G0 (ladbare Hälfte)
ESC) B		International A → G1
ESC (K		Deutscher Zeichensatz → G0 (ladbare Hälfte)
ESC) K		Deutscher Zeichensatz → G1
ESC (w		Klammern-Zeichensatz → G0 (ladbare Hälfte)
ESC) w		Klammern-Zeichensatz → G1
ESC (c		FACET-Zeichensatz → G0 (ladbare Hälfte)
ESC) c		FACET-Zeichensatz → G1
ESC (v		IBM-Zeichensatz → G0 (ladbare Hälfte)
ESC) v		IBM-Zeichensatz → G1
ESC (u		EURO-Symbole → G0 (ladbare Hälfte)
ESC) u		EURO-Symbole → G1
ESC (t		Mathematische Symbole → G0 (ladbare Hälfte)
ESC) t		Mathematische Symbole → G1
ESC (y		Blanks → G0 (ladbare Hälfte)
ESC) y		Blanks → G1
ESC (x		Bereitstellbereich G2 → G0 (ladbare Hälfte)
ESC) x		Bereitstellbereich G2 → G1
ESC * F		Laden des Bereitstellbereichs G2 mit einem Zeichensatz F = @ B K w ...
ESC R B ... ESC ⌈		Laden einer oder mehrerer Zeichengenerator-Adressen mit einem neuen Symbol (... ≙ 3 Byte Adresse und 28 Byte Muster im Hex-Format)
SI	X	Umschalten auf G0 (entspricht CTRL O von Tastatur)
SO		Umschalten auf G1 (entspricht CTRL N von Tastatur)
ESC [5 v		Umschalten innerhalb G0 (national/international)
ESC [1 0 v		Sperren der Taste CH CODE
ESC [1 1 v	X	Freigeben der Taste CH CODE
ESC [1 3 v		Aktuelle Codiertabelle abfragen

6.1.2 Kommandos zum Editieren, zur Cursor-Steuerung und zum Löschen

Kommando	S	Erklärung
ESC [Pn @		Pn (1 bis 80) Leerzeichen ab Cursor rechts einfügen
ESC [Pn L		Pn (1 bis 24 od 25) Leerzeilen ab Cursor einfügen
ESC [7 p	X	Cursor hell
ESC [6 p		Cursor dunkel
ESC [Pn A		Cursor um Pn Zeilen nach oben
ESC [Pn B		Cursor um Pn Zeilen nach unten
ESC [Pn C		Cursor um Pn Zeichen nach rechts
ESC [Pn D		Cursor um Pn Zeichen nach links
ESC [Pl ; Pc H		Cursor auf Pl-te Zeile und Pc-te Stelle positionieren
ESC [6 n		Aktuelle Cursor-Position abfragen
IS4 pl pc		Kurz-Cursor-Positionierung (entsprechend 810-Protokoll, keine ANSI-Norm)
ESC [5 p		Kurz-Cursor-Positions-Abfrage (Werte entsprechen Pl und Pc, siehe vorher)
BS		Cursor 1 Stelle nach links
		(entspricht ⌫ von der Tastatur)
HT		Cursor 1 Tabulatorstelle nach rechts
		(entspricht ⇥ von der Tastatur)
ESC [Pn Z		Cursor um Pn Tabulatorstellen nach links
LF		Zeilenvorschub (entspricht MENU von der Tastatur)
CR		Cursor auf 1.Stelle der Zeile
		(entspricht ↵ von der Tastatur)
ESC E		CR mit LF
ESC [s		Aktuelle Cursor-Position speichern
ESC [u		Cursor wird auf letzte gespeicherte Cursor-Position positioniert
ESC [1 0 u	X	Roll-Modus
ESC [1 1 u		Scroll-Modus
ESC [Pn S		Bildverschiebung um Pn Zeilen aufwärts
ESC [Pn T		Bildverschiebung um Pn Zeilen abwärts
ESC [Pn P		Es werden Pn Zeichen einschließlich Cursor-Position ausgefügt
ESC [Pn M		Es werden Pn Zeilen einschließlich Cursor-Position ausgefügt
ESC [Pn K		Löschen aller oder einiger Zeichen einer Zeile je nach Parameter Pn
ESC [Pn J		Löschen aller oder einiger Zeichen auf dem Bildschirm je nach Parameter Pn

6.1.3 Kommandos zur Initialisierung des Bildschirms und Zeichendarstellung

Kommando	S	Erklärung
ESC [Pt ; Pb r		Bildverschiebebereich festlegen, Cursor muß sich im Bereich befinden
ESC [1 u	X	24 Zeilen Modus einschalten (Sonderform des Bild-verschiebebereichs)
ESC [0 u		25 Zeilen Modus einschalten (Sonderform des Bild-verschiebebereichs)
ESC [Pn p		Blinken ein/aus für 24- oder 25-Zeilen-Mode bzw. der 25.Zeile
ESC [9 u	X	Auto-roll-Modus einschalten
ESC [8 u		Page Modus einschalten
ESC [P1 ;...Pn m		Attribut(e) für nachfolgende Zeichen einstellen
ESC [3 u	X	Löschmuster Blank
ESC [2 u		Löschmuster Nil
ESC [8 p		Bildschirm dunkel steuern
ESC [9 p	X	Bildschirm hell steuern
ESC [2 0 u	X	Hintergrund dunkel (weiß auf schwarz) } nicht für 15"-
ESC [2 1 u		Hintergrund hell (schwarz auf weiß) } Bildschirm
ESC [1 0 p		Hervorheben der Cursor-Position (nur für PC-X sinnvoll)
ESC [5 u	X	Video-Timeout einschalten
ESC [4 u		Video-Timeout ausschalten
ESC c		Rücksetzen des Bildschirms, startet jedoch nicht den Selbsttest

6.1.4 Tastaturkommandos

Kommando	S	Erklärung
ESC «		Tastatur sperren
ESC b	X	Tastatur freigeben
ESC [0 s		Tastenwiederholung aus
ESC [1 s	X	Tastenwiederholung ein
BEL		Akustisches Signal (entspricht CTRL G von Tastatur)
ESC [2 s	X	Clicker aus
ESC [3 s		Clicker ein
ESC [0 w		Anfordern des Schlüsselschalterstatus
ESC [7 u	X	Umcodierung der Tastencodes entsprechend der deutschen Tastaturbelegung
ESC [6 u		Umcodierung der Tastencodes für alle anderen Tastaturbelegungen
ESC R A ... ESC \		Laden weiterer nationaler Tastaturbelegungen (... ≙ Tastaturbelegungstabelle, 1024 Byte für Mehrplatzsysteme, 1280 Byte für PC-X/PC-X10)
ESC E 11 y		Lesen der aktuellen Tastaturbelegungstabelle (siehe Kapitel 8 Tastaturbelegung)

6.1.5 Servicekommandos

Kommando	S	Erklärung
ESC [3 v		Alle Steuerzeichen außer ESC werden am Bildschirm angezeigt und nicht ausgeführt
ESC [2 v	X	Zurückschalten in Normalbetrieb
ESC [4 v		Alle Steuerzeichen werden angezeigt, dadurch jedoch keine Rückkehr in Normalbetrieb möglich, die Bedieneinheit muß hierzu ausgeschaltet werden
ESC [3 y		Systemtest auslösen und Ergebnis abfragen
ESC [4 y		Firmware-Version der Bildschirm-Steuerung abfragen
ESC [5 y		Tastatur-Firmware-Version abfragen
ESC [8 v		Bildverschiebebereich wird mit nachfolgenden Zeichen gefüllt
ESC [9 v		Gesamten Zeichengenerator ausgeben
ESC [7 v		Vorige Kommandos (ESC [8 v und ESC [9 v) zurücksetzen

6.1.6 Kurzbeschreibung der Normelemente (DIN 66254/ISO 6492.2)

a) Direkte Steuerzeichen aus dem Steuerzeichensatz C0 0x01 bis 0x1f z.B. BS, CR, usw.
b) Direkte erweiterte Steuerzeichen aus dem Steuerzeichensatz C1: ESC und Zeichen der Spalte 4 und 5: 0x1b,0x40 bis 0x1b,0x5f
c) Steuerzeichenfolgen: Einleitungsfolgen, Parameter, Schlußzeichen

Einleitungsfolgen

Beispiel:

CSI	Control Sequence Introducer	ESC [	0x1b,0x5b
DCS	Device Control String	ESC P	0x1b,0x50
PU1	Private Use 1	ESC Q	0x1b,0x51
PU2	Private Use 2	ESC R	0x1b,0x52
ST	Stringterminator	ESC \	0x1b,0x5c

Parameter

Pn — Dezimalparameter (ASCII-Zahlenfolge (Spalte 3), die aus mehreren Stellen bestehen kann)

Pn1; Pn2; Pnn — Dezimalparameterfolge ; Trennzeichen für mehrere Dezimalparameter

(Pn) Hexadezimale Darstellung von Pn

z.B.	0x38	bedeutet die Zahl	8
	0x36,0x32	"	62

Home Position — 1.Zeile, 1.Stelle des Bildschirms

Standardwert-Annahme: Unterbleibt bei einer Steuerzeichenfolge die Übergabe eines Parameters, wird automatisch der Standardwert angenommen.

Es gibt jedoch auch Steuerzeichenfolgen, in denen gar keine Parameterangabe vorgesehen ist.

Schlußzeichen

ST String Terminator: ESC\ 0x1b,0x5c

F Final Code: ASCII-Zeichen aus den Spalten 4, 5, 6 oder 7 0x40 bis 0x7e, das die Bedeutung der Steuerzeichenfolge festlegt.

6.2 Funktionen und Befehle für die Bildschirmsteuerung

6.2.1 Codierung und Zeichendarstellung

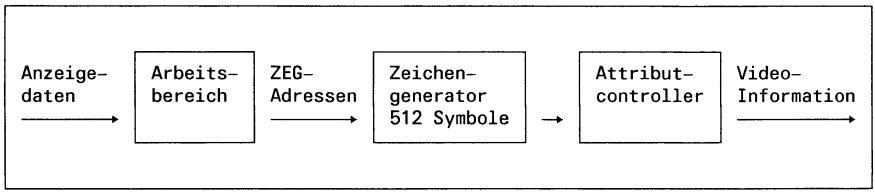


Bild 6-1 Schema der Zeichendarstellung

Die der Bildschirmsteuerung übergebenen 7-Bit-Codes ermöglichen das Arbeiten mit 128 verschiedenen Bitkombinationen. Davon beziehen sich 32 auf den Steuerzeichensatz C0 und 96 auf den in G0 bereitgestellten nationalen od. internationalen Schriftzeichensatz.

Durch das Steuerzeichen SO (0x0e, CTRL N) wird der Bereitstellungsbereich G1 aufgerufen, was bedeutet, daß die Schriftzeichen-Bitkombinationen jetzt entsprechend Zeichensatz G1 am Bildschirm angezeigt werden. Durch das Steuerzeichen SI (0x0f, CTRL O) wird wieder G0 zur Darstellung von Anzeigedaten am Bildschirm verwendet.

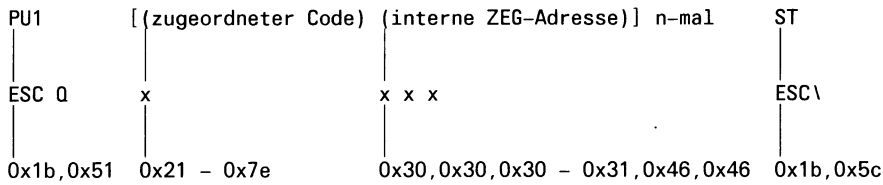
Die Bereitstellungsbereiche G0 und G1 können mit Zeichensätzen wie z.B. International, International A, Deutsch oder Graphik oder andere geladen werden. Das geschieht durch Übergabe der ESC-Folge 'ESC (F' bzw. 'ESC) F', wobei F ein für jeden Zeichensatz festgelegter Final Code ist. In einem Pseudo-Bereitstellungsbereich G2 mit dem Final Code 'x' kann ein beliebiger Zeichensatz aus den 512 Möglichkeiten des Zeichengenerators zusammengestellt werden.
Wenn ein optionaler Zeichengenerator-RAM-Speicher eingesetzt wird, können die 512-Standard-Symbole des Zeichengenerators mit anderen, frei definierbaren Symbolen überschrieben werden. Die frei definierten Symbole bleiben bis zur Steuersequenz R/S oder zum Ausschalten der Bedieneinheit erhalten (siehe Abschnitt 7.4 Ladbarer Zeichengenerator).
Der Bereitstellungsbereich G0 ist logisch in zwei Hälften geteilt. Die eine Hälfte beinhaltet fest den Zeichensatz International A, die andere Hälfte ist über die Steuerfolge 'ESC (F' ladbar. Dieser Bereich wird vorzugsweise mit dem jeweils aktuellen nationalen Zeichensatz geladen.

Standard-zeichensatz	Steuerfolgen zur Code-Übertragung in die Bereitstellungsbereiche	
	G0 nationale Hälfte	G1
International	ESC (@ 0x1b,0x28,0x40	ESC) @ 0x1b,0x29,0x40
International A	ESC (B 0x1b,0x28,0x42	ESC) B 0x1b,0x29,0x42
Deutsch	ESC (K 0x1b,0x28,0x4b	ESC) K 0x1b,0x29,0x4b
Klammern	ESC (w 0x1b,0x28,0x77	ESC) w 0x1b,0x29,0x77
Facet	ESC (c 0x1b,0x28,0x63	ESC) c 0x1b,0x29,0x63
IBM	ESC (v 0x1b,0x28,0x76	ESC) v 0x1b,0x29,0x76
Euro Symbole	ESC (u 0x1b,0x28,0x75	ESC) u 0x1b,0x29,0x75
Mathematische Symbole	ESC (t 0x1b,0x28,0x74	ESC) t 0x1b,0x29,0x74
Blanks	ESC (y 0x1b,0x28,0x79	ESC) y 0x1b,0x29,0x79
Pseudo Bereitstellungs-bereich G2	ESC (x 0x1b,0x28,0x78	ESC) x 0x1b,0x29,0x78

Um eine frei ladbare Codetabelle zu realisieren, wird ein Pseudo-Bereitstellungsbereich G2 verwendet.

In diesen Bereich kann mit 'ESC * F', also 0x1B,0x2a F, jeder gewünschte Vorratsbereich geladen werden.
Mit dem Controllstring 'PU1, Ladeinformation, ST' können dann einzelne GL-Adressen mit anderen Adressen direkt überschrieben werden.
Eignet sich kein vorhandener Vorratsbereich, so kann mit 'ESC * y' der G2 mit 'Blanks' hinterlegt werden.

Dieser Controlstring ist wie folgt aufgebaut:



Der so entstandene G2-Bereich kann mit 'ESC (x' bzw. 'ESC) x' nach G0 oder G1 geladen werden.

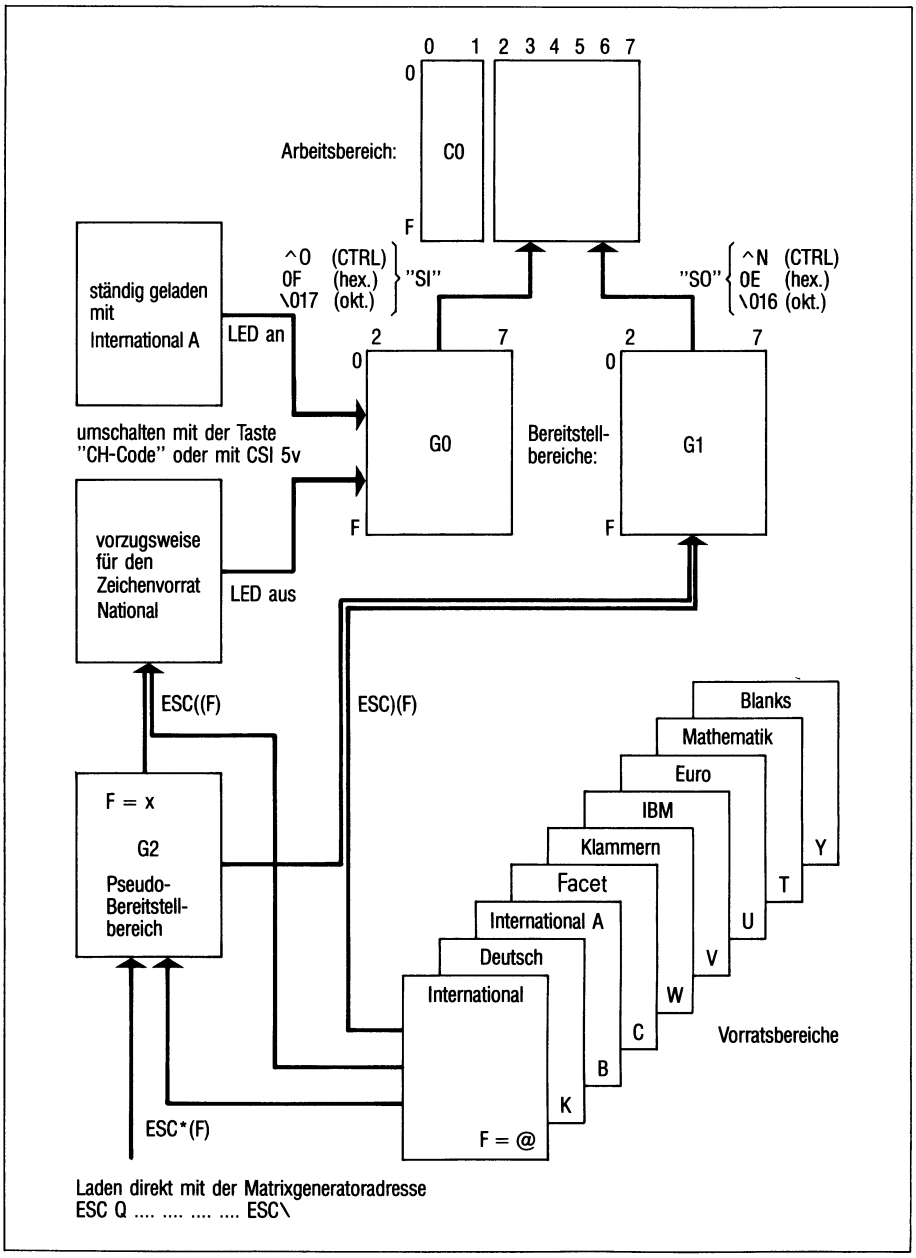


Bild 6-2 Übersicht der möglichen Codiertabellen

6.2.2 Befehle zur Codiertabellen-Umschaltung

Umschalten auf G0

```
SI  0x0f
```

Es wir die in G0 aktuelle Umsetztabelle verwendet.

Umschalten auf G1

```
SO  0x0e
```

Es wird entsprechend G1 umgesetzt.

Umschalten innerhalb G0

```
CSI 5 v  0x1b,0x5b,0x35,0x76
```

Es wird zwischen der fest mit 'International A' belegten Umsetztabelle und der anderen (vorzugsweise nationalen) in G0 geladenen Umsetztabelle gewechselt. Ist die 'International A'-Hälfte aktuell, so leuchtet die INT-LED der Tastatur.
Diese 'International-A'-Hälfte ist auch nach Netz-Ein und RESET aktiv.
Die Umschaltung kann auch mit der Taste [CH CODE] der Tastatur erfolgen.

Sperren der Taste [CH CODE]

```
CSI 10 v  0x1b,0x5b,0x31,0x30,0x76
```

Damit wird das Umschalten innerhalb von G0 für den Bediener ausgeschaltet.

Freigeben der Taste CH CODE

```
CSI 11 v  0x1b,0x5b,0x31,0x31,0x76
```

Der Bediener kann frei wählen zwischen dem Zeichensatz 'International A' und dem anderen (vorzugsweise nationalen) Zeichensatz in G0 (Standard nach Netz-Ein).

Codiertabellen Abfrage

```
CSI 13 v  0x1b,0x5b,0x31,0x33,0x76
```

Damit kann vom PC abgefragt werden, welche Codiertabelle momentan für den Bildschirm aktuell ist.

Das folgende Ergebnis wird zum PC zurückgeschickt:

```
DCS 13 v Pn ST  0x1b,0x50,0x31,0x33,0x76,(Pn),0x1b,0x5C
```

Pn:	0	(0x30)	G0	International A
	1	(0x31)	G0	National
	2	(0x32)	G1	International A in G0
	3	(0x33)	G1	National in G0

6.2.3 Anzeigedaten

Alle empfangenen Zeichen, die keine Steuerfunktionen darstellen, werden am Bildschirm an die aktuelle Cursor-Position geschrieben. Bei jedem Zeichen wird der Cursor automatisch eine Stelle weiter nach rechts positioniert. Wurde das Zeichen auf die letzte Stelle einer Zeile geschrieben, wird der Cursor in die nächste Zeile, 1.Stelle, positioniert.

Ist der Bildschirm im 24-Zeilen-Anzeige-Mode und wurde ein Zeichen an die letzte Stelle der Zeile 24 geschrieben, erfolgt eine Bildverschiebung nach oben (Scroll Up) und der Cursor wird an die 1.Stelle der 24.Zeile positioniert.
Im 'Page-Mode' wird der Cursor auf die 1.Stelle der 1.Zeile des Bildverschiebebereichs gesetzt.
Im 24-Zeile-Mode ist die 25.Zeile Systemzeile. Falls sie beschrieben werden soll, muß der Cursor durch einen Positionierbefehl in diese gebracht werden.
Wurde ein Zeichen auf die letzte Stelle der Systemzeile geschrieben, bleibt der Cursor auf dieser Position, bis ein erneuter Positionierbefehl erfolgt.
Wurde im 25-Zeilen-Mode ein Zeichen auf die letzte Stelle der 25.Zeile geschrieben, erfolgt eine Bildverschiebung nach oben und der Cursor wird in die 1.Stelle der 25.Zeile gebracht.
Im 'Page-Mode' wird der Cursor auf die 1.Stelle der 1.Zeile des Bildverschiebebereichs gesetzt.

Die beiden folgenden Text-Editier-Befehle erleichtern das Einschreiben von Anzeigedaten:

Zeichen einfügen **ICH (Insert character)**

```
CSI Pn @  0x1b,0x5b,(Pn),0x40
```

Von der Cursor-Position ab werden die rechts vom Cursor stehenden Zeichen Pnmal nach rechts geschoben; die über das Zeilenende 'hinausgeschobenen' Zeichen gehen verloren. Attribute werden innerhalb der Zeile ebenfalls verschoben. Ab der Cursor-Position werden Pn Stellen gelöscht, die alten Attribute werden gelöscht.

(Standardwert-Annahme: 1, Pnmax = 80 - Pn Cursor)

Zeile einfügen IL (Insert line)

```
CSI Pn L  0x1b,0x5b,(Pn),0x4c
```

Von der Cursor-Position ab werden die Zeile, in welcher der Cursor steht, und die folgenden Zeilen Pn-mal nach unten geschoben. Die über die 24. Zeile (im 24-Zeilen-Mode)/ 25. Zeile (im 25-Zeilen-Mode)/ bzw. unterste Zeile des Bildverschiebebereichs hinausgeschobenen Zeilen gehen verloren. Attribute werden mitverschoben.
Die Cursor-Zeile und Pn-1 folgenden Zeilen werden gelöscht, alte Attribute werden gelöscht.
Cursor zum Anfang der Zeile.

(Standardwert-Annahme: 1, Pnmax = 24/25/* - Pn cursor)

* bzw. Anzahl der Zeilen des Bildverschiebebereichs

6.2.4 Cursor-Befehle

Beim Einschalten der PC-MX2 Bedieneinheit bzw. beim Einschalten des PC-X wird der Bildschirm mit Blank gelöscht und der Cursor auf die 1.Stelle der 1.Zeile (Home Position) gebracht. Der Cursor blinkt mit einer Frequenz von ca. 3 Hz.

Cursor hell CON (Cursor on)

```
CSI 7 p  0x1b,0x5b,0x37,0x70
```

Standard nach RIS bzw. Netz-Ein.

Cursor dunkel COF (Cursor off)

```
CSI 6 p  0x1b,0x5b,0x36,0x70
```

Cursor nach oben **CUU (Cursor up)**

```
CSI Pn A  0x1b,0x5b,(Pn),0x41
```

Relative Verschiebung des Cursors auf die gleiche Zeichenposition der Pn-ten vorhergehenden Zeile bis maximal zum oberen Rand des Bildverschiebebereichs.

(Standardwert-Annahme: 1)

Cursor nach unten **CUD (Cursor down)**

```
CSI Pn B  0x1b,0x5b,(Pn),0x42
```

Relative Verschiebung des Cursors auf die gleiche Zeichenposition der Pn-ten nachfolgenden Zeile bis maximal zur 24. (24-Zeilen-Mode) /25.Zeile (25-Zeilen-Mode) /bzw. zum unteren Rand des Bildverschiebebereichs.

(Standardwert-Annahme: 1)

Cursor nach rechts **CUF (Cursor forward)**

```
CSI Pn C  0x1b,0x5b,(Pn),0x43
```

Relative Verschiebung des Cursors auf die Pn-te nachfolgende Zeichenstelle bis maximal zum rechten Bildrand.

(Standardwert-Annahme: 1)

Cusor nach links **CUB (Cursor backward)**

```
CSI Pn D  0x1b,0x5b,(Pn),0x44
```

Relative Verschiebung des Cursors auf die Pn-te vorhergehende Zeichenposition bis maximal zum linken Bildrand.

(Standardwert-Annahme: 1)

Absolute Cursor-Positionierung **CUP (Cursor position)**

```
CSI Pl;Pc H  0x1b,0x5b,(Pl),0x3b,(Pc),0x48
```

Absolute Cursor-Positionierung auf Pl-te Zeile, Pc-te Stelle. Pl bzw. Pc können auch 2-stellige Parameter sein.

(Standardwert-Annahme: Pc = Pl = 1 = Home Position)

Aktuelle Cursor-Position abfragen (PC r BE)

```
CSI 6 n  0x1b,0x5b,0x36,0x6E
```

Diese Kontroll-Folge fordert von der Bedieneinheit die aktuelle Cursor-Position an.

Absolute Cursor-Position senden CPR (Curser position report)

```
CSI Pl;Pc R  0x1b,0x5b,(Pl),0x3b,(Pc),0x52
```

Diese Folge meldet die absolute Position des Cursors mit 2 numerischen Parametern (Pl = Nummer der Zeile, Pc = Nummer der Spalte).

Hinweis

Zusätzlich gibt es auch verkürzte Cursor-Befehle (Cursor-Positionierung bzw. Cursor-Positionsabfrage.)
Diese entsprechen jedoch nicht den ANSI-Normen.

Kurz-Cursor-Positionierung

```
wie CUP:  IS4 Pl Pc  0x1c,(line + 20),(column + 20)
```

- Der Zeilenadressierung 0 bis 24 entsprechen die ASCII-Werte 0x20 bis 0x38 (Blank bis 8).
- Der Spaltenadressierung 0 bis 79 entsprechen die ASCII-Werte 0x20 bis 0x6f (Blank bis o).

Kurz-Cursor-Positions-Abfrage

```
wie CPR:  CSI 5 p  0x1b,0x5b,0x35,0x70
```

Die Reaktion erfolgt in der Form: IS4 Pl Pc (siehe oben).

Achtung

Die Folge IS4 0x1c entspricht dem Standard-Quit-Signal des TTY-Treibers und verursacht daher einen Core Dump, wenn der TTY-Treiber nicht entsprechend umparametrisiert wird. Dies kann durch Ändern des Quit-Characters oder durch Ändern des Modus (z.B. RAW) erfolgen.

Rückwärtsschritt **BS (Backspace)**

```
0x08
```

Der Cursor wird um 1 Stelle nach links bis maximal zum linken Bildrand geschoben.

Horizontal-Tabulator nach rechts **HT (Horizontal tabulation)**

```
0x09
```

Der Cursor wird auf die nächste Tabulator-Stelle oder, wenn in der betroffenen Zeile keine mehr besteht, bis zum rechten Bildrand geschoben.

Horizontal-Tabulator nach links CBT (Cursor backward tabulation)

```
CSI Pn Z  0x1b,0x5b,(Pn),0x5a
```

Der Cursor wird auf die Pn-te Tabulator-Stelle nach links bis maximal zum linken Bildrand verschoben.

(Standardwert-Annahme: 1)

Zeilenvorschub **LF (Line feed)**

```
0x0A
```

Der Cursor wird auf die entsprechende Zeichen-Stelle der nächsten Zeile positioniert. Ist der Cursor bereits in der 24./25.Zeile (24/25-Zeilen-Mode) bzw. in der untersten Zeile des Bildverschiebebereichs, erfolgt eine Bildverschiebung nach oben und der Cursor wird an die 1.Stelle der freiwerdenden Zeile gesetzt. Arbeitet man im 'Page-Mode', wird der Cursor an die 1.Stelle der letzten Zeile gebracht, es wird keine Bildverschiebung durchgeführt.

Cursor-Rücklauf CR (Carriage return)

```
0x0D
```

Der Cursor wird an die 1.Zeichenstelle der Zeile gebracht.

Nächste Zeile NEL (Next line)

```
ESC E  0x1b,0x45
```

Ausführung wie LF + CR.
Der Cursor wird an die erste Stelle der nächsten Zeile gesetzt. Ist der Cursor bereits in der 24./25.Zeile (24-/25-Zeilen-Mode), erfolgt automatisch eine Roll-Up-Funktion und ein Positionieren des Cursors auf die 1.Stelle der 24. bzw. 25.Zeile.

Speichern der Cursor-Position CS (Cursor save)

```
CSI s  0x1b,0x5b,0x7
```

Die aktuelle Cursor-Position wird gespeichert, bis ein erneutes CS erfolgt oder rückgesetzt wird.

Rücksprung CRST (Cursor restore)

```
CSI u  0x1b,0x5b,0x75
```

Der Cursor wird auf die letzte mit CS abgespeicherte Stelle positioniert. Wurde kein CS-Kommando gegeben, so wird die Home Position angesteuert.

6.2.5 Bildverschiebe-Befehle

Verschiebemodus

Grundsätzlich kann zwischen Roll- und Scroll-Bewegungen gewählt werden. Während im Roll-Modus der gesamte Bildverschiebebereich verschoben wird, verschiebt sich im Scroll-Modus nur der sich oberhalb oder unterhalb der aktuellen Cursor-Zeile befindliche Teil des Bildverschiebebereichs.

Diese Auswahl bestimmen auch die von den Tasten 'Bild nach oben' und 'Bild nach unten' ausgelösten Funktionen.

Roll-Modus

```
CSI 10 u  0x1b,0x5b,0x31,0x30,0x75
```

Der gesamte Bildverschiebebereich (Zeilen und Attribute) wird verschoben.
Dadurch entstehen an der Bildverschiebebereichs-Ober- oder -Untergrenze eine oder mehrere Leerzeilen.
Alle Attribute dieser Leerzeile(n) werden gelöscht.
Die Cursor-Position wird nicht verändert.
Dieser Mode ist Standardvorgabe nach RIS bzw. Netz-Ein.

Scroll-Modus

```
CSI 11 u  0x1b,0x5b,0x31,0x31,0x75
```

Der obere oder untere Bildverschiebebereich (Zeilen und Attribute) werden – von der aktuellen Cursor-Zeile an - um eine oder mehrere Zeilen nach oben oder unten verschoben.
Die Attribute der dadurch entstehenden Leerzeile(n) werden gelöscht. Die Cursor-Position wird nicht verändert.

Bildverschiebung aufwärts **RU/SU (Roll-/Scroll-up)**

```
CSI Pn S  0x1b,0x5b,(Pn),0x53
```

Mit Pn wird die Anzahl der Zeilen angegeben.

(Standardwert-Annahme: 1)

Bildverschiebung abwärts **RD/SD (Roll-/Scroll-down)**

```
CSI Pn T  0x1b,0x5b,(Pn),0x54
```

Mit Pn wird die Anzahl der Zeilen angegeben.

(Standardwert-Annahme: 1)

6.2.6 Löschbefehle

Zeichen löschen **DCH (Delete Character)**

```
CSI Pn P  0x1b,0x5b,(Pn),0x50
```

Das Zeichen auf Cursor-Position und die Pn-1 folgenden Zeichen werden gelöscht, die in dieser Zeile rechts vom Cursor übrigbleibenden Zeichen (mit Attributen) werden nachgeschoben. Pn Stellen am rechten Zeilenende werden ebenso wie zugehörige Attribute gelöscht.

(Standardwert-Annahme: 1, Pnmax = 80)

Zeile löschen **DL (Delete line)**

```
CSI Pn M  0x1b,0x5b,(Pn),0x4d
```

Die Zeile, in welcher der Cursor steht, und die Pn-1 folgenden Zeilen werden gelöscht. Die folgenden, nicht entfernten Zeilen und zugehörige Attribute werden nach oben nachgeschoben. Am unteren Bildende werden Pn Zeilen und die zugehörigen Attribute gelöscht. Der Cursor wird an den

Anfang der Zeile positioniert. Dieser Befehl bezieht sich auf den Bildverschiebebereich. Ist der 24-Zeilen-Anzeigemode eingestellt, wird die 25.Zeile (System-Zeile) durch den Befehl DL nicht beeinflußt.

(Standardwert-Annahme: 1, Pnmax = Bildverschiebeuntergrenze - Pl)

Zeile löschen EL (Erase in line)

```
CSI Pn K  0x1b,0x5b,(Pn),0x4b
```

Einige oder alle Zeichen einer Zeile werden abhängig vom übergebenen numerischen Parameter Pn gelöscht.

	Löschen ab Cursor-Position bis Zeilenende	Löschen von Zeilenanfang bis einschließlich Cursor-Positon	Löschen aller Zeichen der Zeile
Löschen mit Blank, Attribute werden gelöscht	0 (0x30)	1 (0x31)	2 (0x32)
Löschen mit Nil, Attribute werden gelöscht	3 (0x33)	4 (0x34)	5 (0x35)
Löschen mit Blank, mit zuletzt gültigem Attribut vorbelegen (sofern Attribute nicht gespeichert sind)	6 (0x36)	7 (0x37)	8 (0x38)
Löschen mit Nil, mit zuletzt gültigem Attribut vorbelegen (sofern Attribute nicht gespeichert sind)	9 (0x39)	10 (0x31,0x30)	11 (0x31,0x31)
Löschen mit Blank, ohne Verändern der Attribute	12 (0x31,0x32)	13 (0x31,0x33)	14 (0x31,0x34)
Löschen mit Nil, ohne Verändern der Attribute	15 (0x31,0x35)	16 (0x31,0x36)	17 (0x31,0x37)

(Standardwert-Annahme: 0)

Nach Löschbefehlen mit PN = 0...5 wird für den Rest des Bildschirms automatisch das Kommando "Einfrieren Attribute" erzeugt.

Bildschirm löschen ED (Erase in display)

```
CSI Pn J  0x1b,0x5b,(Pn),0x4a
```

Der Befehl bezieht sich auf den Bildverschiebebereich.

Einige oder alle Zeichen eines Bildes werden abhängig vom übergebenen numerischen Parameter Pn gelöscht.

	Löschen ab Cursor-Position bis Bildende Verschiebebereich	Löschen ab Bildanfang bis einschließlich Cursor-Positon	Löschen aller Zeichen des Bildes, Cursor geht zum Bildanfang Verschiebebereich
Löschen mit Blank, Attribute werden gelöscht	0 (0x30)	1 (0x31)	2 (0x32)
Löschen mit Nil, Attribute werden gelöscht	3 (0x33)	4 (0x34)	5 (0x35)
Löschen mit Blank, mit zuletzt gültigem Attribut vorbelegen (sofern Attribute nicht gespeichert sind)	6 (0x36)	7 (0x37)	8 (0x38)
Löschen mit Nil, mit zuletzt gültigem Attribut vorbelegen (sofern Attribute nicht gespeichert sind)	9 (0x39)	10 (0x31,0x30)	11 (0x31,0x31)
Löschen mit Blank, ohne Verändern der Attribute	12 (0x31,0x32)	13 (0x31,0x33)	14 (0x31,0x34)
Löschen mit Nil, ohne Verändern der Attribute	15 (0x31,0x35)	16 (0x31,0x36)	17 (0x31,0x37)

(Standardwert-Annahme: 0)

6.2.7 Darstellungsarten (Attribute)

Durch die Befehlsfolge SGR (Select graphic rendition) wird die Darstellung der nach dieser Befehlsfolge zum Bildschirm übertragenen Anzeigedaten festgelegt.
Die Darstellungsart bleibt solange gültig, bis durch eine neue SGR-Befehlsfolge entweder eine neue Darstellungsart festgelegt oder die Attribute gespeichert werden.
Die Art der Darstellung kann beliebig oft geändert werden und schränkt die Anzahl der nutzbaren Zeichenpositionen auf dem Bildschirm nicht ein.

Hinweis für PC-X10

Die Emulation der Bedieneinheit auf dem PC-X10 kann die Darstellungsarten nicht wie die Bedieneinheit 97801 behandeln.

Darstellungsarten auswählen **SGR (Select graphic rendition)**

```
CSI P1;...Pn m  0x1b,0x5b,(P1),0x3b,...(Pn),0x6d
```

Folgende Darstellungsarten, die durch numerische Parameter gekennzeichnet werden, sind möglich:

Parameter	Darstellungsart
0 0x30	Normal (Standardwert-Zuweisung)
2 0x32	Halbhell
4 0x34	Unterstrichen
5 0x35	Blinken
7 0x37	Invers
8 0x38	Dunkel (unterdrückte Darstellung)
50 0x35,0x30	Attribute speichern *)

Hinweis für PC-X10

- Der PC-X10 kann die Attribute nicht speichern.
- Die Grafikversion des PC-X10 benutzt zwei verschiedene Zeichensätze für normale und hervorgehobene Darstellung.
  Ein Wechsel der Darstellungsart zwischen Normal und Hervorgehoben kann deshalb auch einen Wechsel der Zeichenstze bedeuten.

- Das Attribut Blinken senkt die Systemleistung des PC-X10.
- Das Attribut Halbhell ergibt die **normale** Darstellung des Textes.

*) Durch den Befehl 'Attribute speichern' werden alle in diesem Augenblick für den gesamten Bildschirm wirksamen Attribute festgehalten. Werden neue Anzeigedaten (ohne SGR-Befehlsfolgen) auf den Bildschirm ausgegeben, sind diese Attribute gültig.
Die gespeicherten Attribute können durch Befehle wie 'Bildschirm löschen mit Blank/Nil und Attribute löschen' oder 'Zeile löschen mit Blank/Nil und Atttribute löschen' wieder auf normale Darstellung umgewandelt werden.

Folgende Kombinationen sind erlaubt:

Kombination		Darstellungsart
2;4	0x32,0x3b,0x34	Halbhell, unterstrichen
2;4;7	0x32,0x3b,0x34,0x3b,0x37	Halbhell, invers, unterstrichen
2;5	0x32,0x3b,0x35	Blinken zwischen normal und halbhell
2;5;7	0x32,0x3b,0x35,0x3b,0x37	Blinken zwischen halbhell und halbhell invers
2;7	0x32,0x3b,0x37	Halbhell, invers
4;5	0x34,0x3b,0x35	Blinken zwischen normal und unterstrichen
4;5;7	0x34,0x3b,0x35,0x3b,0x37	Blinken invers, unterstrichen
4;7	0x34,0x3b,0x37	Unterstrichen, invers
5;7	0x35,0x3b,0x37	Blinken zwischen normal und invers
2;4;5	0x32,0x3b,0x34,0x3b,0x35	Blinken halbhell, unterstrichen

Blinken

```
CSI Pn p  0x1b,0x5b,(Pn),0x70
```

Pn: 0 (0x30) Blinken ein, Zeile 1 - 25 bzw. 1 - 24 (25/24-Zeilen-Mode)
1 (0x31) Blinken aus, Zeile 1 - 25 bzw. 1 - 24 (25/24-Zeilen-Mode)
2 (0x32) Blinken ein, Zeile 25
3 (0x33) Blinken aus, Zeile 25

Hinweis für PC-X10 Grafikversion

Das Attribut **Blinken** belastet den Hauptprozessor des PC-X10 zusätzlich und senkt dadurch die Leistung des PC-X10.

Bildschirm dunkel steuern

```
CSI 8 p  0x1b,0x5b,0x38,0x70
```

Damit wird das Anzeigen des Bildes verhindert ohne den Bildinhalt zu beeinflussen.

Bildschirm hell steuern

```
CSI 9 p  0x1b,0x5b,0x39,0x70
```

Der Bildinhalt wird wieder angezeigt.

(Standard)

Hervorheben

```
CSI 10 p  0x1b,0x5b,0x31,0x30,0x70
```

Damit wird an der aktuellen Cursor-Position das Attribut 'Invers' invertiert. Alle anderen Attribute bleiben unverändert.

6.2.8 Bildschirm-Format und weitere Einstellung

Bildverschiebebereich

```
CSI Pt;Pb r  0x1b,0x5b,(Pt),0x3b,(Pb),0x72
```

Durch obige Befehlsfolge läßt sich der Bildverschiebebereich zwischen die Zeile Pt und Pb legen.
Diese Einstellung bezieht sich auf die Funktionen Zeile einfügen, Cursor nach oben/unten, Bildverschiebung aufwärts/abwärts, Bildschirm löschen und Bildschirm füllen. Diese Funktionen werden nur ausgeführt, wenn der Cursor sich im definierten Bildverschiebebereich befindet.
Durch obige Befehlsfolge wird eine früher erfolgte Bildschirm-Format-Einstellung (24-/25-Zeilen-Mode) überschrieben.

Bildschirm-Mode (Sonderformen des Bildverschiebebereichs)

24-Zeilen-Mode (Standard)

```
CSI 1 u  0x1b,0x5b,0x31,0x75
```

Der Bildschirm besteht aus einem zusammenhängenden 24-Zeilen Feld; die 25.Zeile ist Systemzeile.
Dieser Mode wird auch nach dem Einschalten der Bedieneinheit oder nach übergabe des Befehls RIS (Reset to initial state) von der SE zur BE wirksam.

25-Zeilen-Mode

```
CSI 0 u  0x1b,0x5b,0x30,0x75
```

Der Bildschirm besteht aus einem zusammenhängenden 25-Zeilen Feld.

Löschmuster

```
CSI 3 u  0x1b,0x5b,0x33,0x75
```

Alle Funktionen bei welchen Zeichen gelöscht werden (Zeichen einfügen, Zeile einfügen, Scroll up, Scroll down, Zeichen entfernen, Zeile entfernen) benutzen das Zeichen **Blank** als Löschmuster.
Dieses Zeichen wird nach Einschalten der Bedieneinheit oder nach dem Befehl RIS verwendet (Standard).

```
CSI 2 u  0x1b,0x5b,0x32,0x75
```

Alle Funktionen, bei denen Zeichen gelöscht werden, benutzen das Zeichen **NIL** als Löschmuster.

Video-Timeout

Einschalten

```
CSI 5 u  0x1b,0x5b,0x35,0x75
```

Erfolgt 10 Minuten lang keine Ausgabe vom PC, wird der Bildschirm dunkelgesteuert. Ein danach gesendetes Zeichen bewirkt die Anzeige des alten Bildschirminhalts, wobei auch das neu gesendete Zeichen zur Anzeige kommt. Auch durch jede beliebige Tastatureingabe wird das Bild wieder zur Anzeige gebracht, wobei jedoch die erste Tastatureingabe nicht an den PC weitergegeben wird.
Der Video-Timeout ist nach Einschalten der Bedieneinheit oder nach dem Befehl RIS eingestellt (Standard).

Ausschalten

```
CSI 4 u  0x1b,0x5b,0x34,0x75
```

Ausschalten des 10-Minuten-Video-Timeouts.

Auto-Roll-Mode (Standard)

```
CSI 9 u  0x1b,0x5b,0x39,0x75
```

Steht der Cursor auf der letzten Position der letzten Zeile des Bildverschiebebereichs und wird ein Zeichen auf den BS geschrieben, erfolgt automatisch eine Bildverschiebung nach oben und der Cursor wird in die 1.Spalte der frei werdenden Zeile gebracht.
Das Kommando 'Line feed' in der letzten Zeile bewirkt ebenfalls eine Bildverschiebung nach oben und Positionieren des Cursors in die 1.Spalte der untersten Zeile des Bildverschiebebereichs.

Page-Mode

```
CSI 8 u  0x1b,0x5b,0x38,0x75
```

Steht der Cursor auf der letzten Position der letzten Zeile des Bildverschiebebereichs und wird ein Zeichen auf den BS geschrieben, erfolgt die Positionierung des Cursors auf die Home Position des definierten Bildverschiebebereichs.
Das Kommando 'Line Feed' in der letzten Zeile bewirkt lediglich die Positionierung des Cursors in die 1.Spalte der betreffenden Zeile.

6.2.9 Rücksetz-Befehl

Rücksetzen **RIS (Reset to initial state)**

```
ESC c  0x1b,0x63
```

Dieser Befehl bewirkt den Aufruf der BE-Initialisierungs-Routine. Das bedeutet, die Bedieneinheit wird in den Zustand gebracht, der auch nach Einschalten des Gerätes besteht. RIS bewirkt aber nicht das Durchlaufen der internen Systemtest-Routinen.
Nach dem RIS ist eine Pause $\geq$ 1,5 sec. einzuhalten, bis Folgezeichen gesendet werden.

6.3 Funktionen und Befehle für Tastatur

6.3.1 Tastaturbefehle

DMI (Disable manual input)

ESC « 0x1b,0x60

Von der Tastatur zur Bildschirm-Steuerung der BE übertragene Daten werden nicht angenommen.

EMI (Enable manual input)

ESC b 0x1b,0x62

Von der Tastatur zur Bildschirm-Steuerung der BE übertragene Daten werden angenommen, umcodiert und zum PC weiterübertragen (Standard).

LED1 - LED6 an

Diese Funktion ist abhängig von der Tastaturbestückung mit LED's.

CSI Pn q 0x1b,0x5b,(Pn),0x71

Pn = 1...6

Die Kombination mehrerer LED's in einer Befehlsfolge ist zulässig.

Beispiel: CSI 1;3;5 q

(Standardwert-Annahme: 1)

LED's aus

```
CSI 0 q  0x1b,0x5b,0x30,0x71
```

Die Kombination mit LED-an-Befehlen in einer Befehlsfolge ist zulässig.

Beispiel: CSI 0;1;3;5 q

Repeat aus/an

```
CSI Pn s  0x1b,0x5b,(Pn),0x73
```

Pn: 0 (30) Repeat aus
1 (31) Repeat ein

(Standardwert-Annahme: 1)

Wenn bei Tastendauerbetätigung der Befehl 'Repeat aus' zur Tastatur geschickt wird, beendet die Tastaturlogik das Senden des Tastencodes, bis die Taste losgelassen und erneut betätigt oder der Befehl 'Repeat an' zur Tastatur übermittelt wird.

Akustischen Alarm auslösen **BEL**

```
0x07
```

Clicker aus/an

```
CSI Pn s  0x1b,0x5b,(Pn),0x73
```

Pn: 2 (32) Clicker aus
3 (33) Clicker an

Wenn eine Taste betätigt wird, kann das Ansprechen durch ein kurzes 'Click' signalisiert werden.
Durch Senden der entsprechenden Steuerfolge kann der Clicker aus- oder eingestellt werden. Nach Netz-Ein oder einem Masterreset (RIS) ist der Clicker ausgestellt.

Hinweis

Die Tastaturen 97801-131 und 97801-132 erzeugen kein Clicker-Signal. Das Clicker-Signal wird von der Bedieneinheit erzeugt.

Magnetkartenleser

```
CSI Pn t  0x1b,0x5b,(Pn),0x74
```

Pn: 0 (30) aus
1 (31) ein
2 (32) Status senden

Nachdem ein Befehl 'Magnetkartenleser-Status-Senden' vom PC zur BE übergeben worden ist, wird der Status wie folgt zum PC gesendet:

```
DCS t XX ST  0x1b,0x50,0x74,(XX),0x1b,0x5c
```

XX: @ (0x40) Magnetkartenleser vorhanden
A (0x41) Magnetkartenleser nicht installiert

Wenn eine Karte in den Leser eingeführt wird, reagiert die Tastatur nicht mehr auf eine Tastenbetätigung bis der Ausweis gelesen ist.

Nach dem Lesen der Karte wird folgender Device Control String von der BE zum PC gesendet:

```
DCS t XX (Kartendaten) ST
                    0x1b,0x50,0x74,(XX),(Kartendaten),0x1b,0x5C
```

XX: C (0x43) Kartendaten im 5-Bit-ABA-Format wurden erfolgreich gelesen *)
D (0x44) Kartendaten im 7-Bit-SIPASS-Format wurden erfolgreich gelesen
E (0x45) Timeout-Fehler. Es wurde eine Karte im Ausweisleser aber nicht innerhalb 1 sec. gelesen
F (0x46) Lesegeschwindigkeit zu langsam
G (47) Lesegeschwindigkeit zu schnell
H (48) Falsches Startzeichen
I (49) Falsches Endzeichen

Kartendaten: max. 40 Zeichen

Wenn der Lesestatus XX aus einem Fehlercode (E...I) besteht, werden im Device Control String keine Kartendaten übertragen.
Wird als Lesestatus C oder D gesendet, werden max. 40 Kartendaten zur SE übertragen.

*) Da laut DIN 66254 ABA-Code nicht im Device-Control-String übertragen werden darf, wird von der BE eine Umcodierung der Kartendaten in den ASCII-Code durchgeführt. Wurde bei der Umcodierung ein Parity-Fehler erkannt,wird das betreffende Zeichen durch den Code (7F) (Schmierzeichen) ersetzt.

Entfernt man die Karte aus dem Ausweisleser, wird folgende Meldung zur SE abgesetzt:

```
DCS t B ST  0x1b,0x50,0x74,0x42,0x1b,0x5C
```

6.3.2 Tastenbelegung

Die Tastatur erzeugt 8-Bit-Platz-Codes von (00) bis (FF). Jede Taste (außer [SHIFT] [CTRL] [LOCK] [CAPS] [CH CODE] erhält 2 Tasten-Platz-Codes, die durch Betätigung der entsprechenden Taste ohne/mit gleichzeitiger Betätigung der Taste [SHIFT] erzeugt und zur Bildschirmsteuerung der Bedieneinheit gesendet werden.

Die 8-Bit-Codes der Tastatur werden von der Bildschirmsteuerung über Firmwaretabellen umcodiert und der so entstandene Leitungscode (7 Bit) gelangt zum PC.

So können verschiedene Tastenbelegungen realisiert werden, ohne den Platzcode der Tastatur verändern zu müssen.

Durch ESC-Folgen wird der Bildschirmsteuerung mitgeteilt, welche Tastaturvariante Verwendung findet:

Deutsche Tastenbelegung (Standard)

```
CSI 7 u  0x1b,0x5b,0x37,0x75
```

Alle anderen Tastaturen

```
CSI 6 u  0x1b,0x5b,0x36,0x75
```

6.3.3 Zusammenfassung von Tastenbelegung und Zeichensätzen

Tastatur International

Zur Unterstützung dieser Tastaturvariante sind, nach dem Umschalten mit CSI 6 u, keine weiteren Vorgaben vom System nötig.
Der Zeichensatz 'International A' ist nach RIS bzw. Netz-Ein immer geladen.

Tastatur Deutsch/International umschaltbar

Um neben dem ASCII-Schriftzeichensatz 'Deutsche-Referenzversion mit Umlauten' auch die 'Internationale-Referenzversion' von der selben Tastatur verarbeiten zu können, sind verschiedene Tasten mit 2 Symbolen je Shift-Ebene versehen.
Nach dem Einschalten des Bildschirms sind alle 3 Bereitstellungsbereiche (beide Hälften von G0 und G1) mit dem Zeichensatz 'International A' geladen, G0 ist aufgerufen, die INT-LED leuchtet. Ist das Arbeiten mit den nationalen Symbolen - entsprechend der konfigurierten Tastatur - erforderlich, müssen in der Initialisierung der BE folgende Aktivitäten erfolgen:

Übergabe der Steuerzeichenfolge ESC (K 0x1b,0x28,0x4B aus TERMCAP vom PC zur BE. Dadurch wird der Deutsche Zeichensatz in die für den nationalen Zeichensatz reservierte Hälfte des Bereitstellungsbereichs G0 geladen. Danach bewirkt die Taste [CH CODE] folgende Code-Umschaltungen: Solange G0 aktiv ist (mit 'SI' (0F) eingeschaltet bzw. default): Bei jeder Betätigung der Taste wird die jeweils andere Hälfte von G0 verbindlich und die - in den beiden Zeichensätzen unterschiedlichen - Zeichen am Bildschirm werden aktualisiert. Ist die nationale Hälfte von G0 gewählt, ist die INT-LED aus; ist die immer vorhandene internationale Hälfte gewählt, leuchtet die INT-LED.

Solange G1 aktiv ist (mit 'SO' 0x0e eingeschaltet), hat die Taste [CH CODE] keine Wirkung.

Die selbe Reaktion wie mit der Taste [CH CODE] erreicht man auch durch die Steuerfolge:

```
CSI 5 v  0x1b,0x5b,0x35,0x76
```

Andere national/international umschaltbare Tastaturen

Nach dem Umschalten der Tastatur mit CSI 6 u ist die Behandlung analog zu der deutsch/internationalen Tastatur.
Aus TERMCAP wird bei der Initialisierung der BE zuerst im Pseudo-Bereitstellungsbereich G2 der entsprechende nationale Zeichensatz zusammengestellt. Der Zeichensatz wird mit folgender Steuerzeichenfolge nach G0 geladen:

```
ESC ( x  0x1b,0x28,0x78
```

Mit der Taste [CH CODE] (bzw. CSI 5 v) wird wieder zwischen der nationalen und internationalen Hälfte von G0 umgeschaltet. Es erfolgt jedoch keine Aktualisierung des Bildschirms.

6.4 Diagnose-Funktionen

6.4.1 DIP-FIX-Schalter-Information abfragen (nur bei PC-MX2)

Die DIP-FIX-Schalter befinden sich auf der Flachbaugruppe TECAC innerhalb der Steuereinheit. Sie sind nicht von außen zugänglich.

```
CSI 0 y  0x1b,0x5b,0x30,0x79
```

```
DCS 0 y Pn ST  0x1b,0x50,0x30,0x79,(Pn),0x1b,0x5c
```

Pn ist eine 3-stellige Zahl $128 \leq Pn \leq 255$

Die Wertigkeit der Brückenschalter S1 bis S8 (2^0 - 2^7) ist N-aktiv, d.h.

- Schalter offen r 1
- Schalter geschlossen r 0

Der Schalter S8 (2^7) ist im Normalbetrieb immer offen zu halten. Der mögliche codierbare Bereich beginnt daher bei 128.

6.4.2 Testmodus

Um den Testmodus aufzurufen, muß der Schalter S8 (2^7) **vor** dem Einschalten des Bildschirmes geschlossen sein. Für normalen Arbeitsbetrieb muß dieser Schalter geöffnet sein.

DIP-FIX-Schalter-Wertigkeit:

Schalter	8	7	6	5	4	3	2	1	
Wertigkeit	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
	0	X	X	X	X	X	1	1	→ Bildschirm-Dauertest
	0	X	X	X	X	X	0	1	→ Bildschirm-Einstellmuster
	0	X	X	X	X	X	X	0	→ Local Back Loop

0 = Schalter geschlossen
1 = Schalter offen
X = beliebig

Bildschirm-Dauertest

Jedes Zeichen mit interner Matrixgeneratoradresse 0x000 bis 0x1FF wird der Reihe nach im gesamten Bildverschiebebereich dargestellt.

Bildschirm-Einstellmuster

Am Bildschirm wird 2000 mal das Zeichen B dargestellt.

Local Back Loop

Alle von der Tastatur oder von der Bildschirm-Steuer-Logik erzeugten Zeichen werden vom USART wieder zur Bildschirm-Steuerung zurückgeschickt und entweder am Bildschirm angezeigt bzw. als Kommando abgearbeitet. Damit können alle Funktionen der Bedieneinheit auch ohne Systemanschluß von der Tastatur ausgelöst werden.

6.4.3 Monitor-Funktion für Steuerzeichen

Mit CSI 3 v werden alle Steuerzeichen außer ESC-Folgen hexadezimal angezeigt und nicht ausgeführt. Mit CSI 2 v erfolgt Rückschalten in den Normalbetrieb.

Mit CSI 4 v - anschließend an CSI 3 v ausgegeben - werden alle Steuerzeichen und alle Steuerzeichenfolgen (auch ESC) angezeigt und nicht ausgeführt.
Ein Rückschalten ist nur durch Ausschalten der BE möglich.

Hinweis

Einplatzsysteme unter SINIX V1.2A ignorieren die Monitor-Funktion.

- Nach CSI 2 v ignoriert die Nachbildung der Bedieneinheit das Steuerzeichen ESC.
- Abdruckbare Zeichen der Bildschirmkommandos werden auf den Bildschirm ausgegeben.
- Steuerzeichen außer ESC werden ausgewertet.

6.4.4 Systemtest (Bildschirm und Tastatur)

PC-MX2:

Nach dem Einschalten der Bedieneinheit oder dem aktiven Signal CRS der SS97 erfolgt automatisch der Aufruf der Systemtest-Routinen.
Werden diese ohne Erkennen eines Fehlers durchlaufen, geht die rote Leuchtdiode an der Rückseite des Bildschirms aus.
Im Fehlerfall erfolgt ohne Aufforderung durch den PC-MX2 keine Fehlermeldung an das System. Am Bildschirm wird nach Möglichkeit der Fehlercode entsprechend dem nachfolgend beschriebenen DCS angezeigt. Vom PC können diese internen Routinen der Bedieneinheit durch folgende Steuerfolge aufgerufen werden:

```
CSI Pn y  0x1b,0x5b,(Pn),0x79
```

Pn:	1	(0x31)	Bedieneinheit-Systemtest auslösen
	2	(0x32)	Testergebnis abfragen *)
	3	(0x33)	Bedieneinheit-Systemtest auslösen und Ergebnis abfragen (Kombination von 1 + 2)
	4	(0x34)	Bildschirm Firmware-Version abfragen
	5	(0x35)	Tastatur Firmware-Version abfragen

Hinweis

Einplatzsysteme unter SINIX V1.2A ignorieren die Funktionen 1, 2 und 3.

*) Ein pauschales Testergebnis kann auch durch folgende Befehlsfolge angefordert werden:

```
CSI 5 n    0x1b,0x5b,0x35,0x6e
```

Folgende Reaktionen sind zu erwarten:

```
CSI Pn n    0x1b,0x5b,(Pn),0x6e
```

Pn: 0 (0x30) BE betriebsbereit
3 (0x33) BE nicht betriebsbereit

Der Bedieneinheit-Systemtest beinhaltet:

ROM-Checksummen-Test
RAM-Test
USART-Test
VIDEO-Test
Tastatur-Test

Testergebnis

Die Testergebnis-Auswertung darf frühestens 10 sec. nach Testauslösung erfolgen bzw. das Ergebnis wird automatisch als folgender Device Control String an den PC gesandt:

```
DCS 2 y Pn1;Pn2...ST
            0x1b,0x50,0x32,0x79,(Pn1),0x3b,(Pn2),...0x1b,0x5c
```

Mögliche Codierungen für

Pn1...Pnn:	0	(0x30)	kein Fehler
	1	(0x31)	ROM1-Checksummenfehler
	2	(0x32)	ROM2-Checksummenfehler
	3	(0x33)	RAM-Fehler
	4	(0x34)	VIDEO-Fehler
	5	(0x35)	Tastatur ROM-Checksummenfehler
	6	(0x36)	Tastatur RAM-Fehler
	7	(0x37)	Tastatur Timeout (keine Tastatur angeschlossen)
	8	(0x38)	USART1-Fehler

Am Bildschirm werden die Fehlercodierungen in der 25.Zeile ausgegeben z.B. ERROR: 7

Firmware-Version der Bildschirmsteuerung

Die Firmware-Version wird als DCS zur SE gesandt:

```
DCS 4 y Pn ST  0x1b,0x50,0x34,0x79,(Pn),0x1b,0x5c
```

Pn ist eine 6-stellige Zahl, z.B. 000017 oder 820113

Firmware-Version der Tastatur

Bei Anforderung der Firmware-Version erfolgt folgende Rückmeldung:

```
DCS 5 y Pn ST  0x1b,0x50,0x35,0x79,(Pn),0x1b,0x5c
```

Pn ist eine 6-stellige Zahl, z.B. 80xxyy
(xx = Tastatur-Variante, yy = Firmware-Version)

Bildschirm-Einstell-Funktion

```
CSI Pn v  0x1b,0x5b,(Pn),0x76
```

Pn = 8 (0x38) Der gesamte Bildverschiebebereich wird mit jedem AnzeigeZeichen, das vom PC gesendet wird, beschrieben. Ergänzung: Zu den horizontalen Verschiebegrenzen können auch vertikale Begrenzungen des Verschiebebereichs definiert werden. Befehlsfolge:

CSI Pn1;Pn2 z 0x1b,0x5b,(Pn1),0x3b,(Pn2),0x7a

Pn1 ist die linke, Pn2 die rechte Bildgrenzen-Spalte des so zu bildenden Fensters ($1 \leq Pn1 \leq Pn2 \leq 80$).

Pn = 7 (0x37) Jedes vom PC gesendete Anzeige-Zeichen wird wieder einfach dargestellt. Die linke/rechte Fenstergrenze wird automatisch wieder 1 bzw. 80.

Hinweis für PC-X10 und PC-X mit Grafik-Bildschirmsteuerung

Einplatzsysteme unter SINIX V1.2A ignorieren diese Funktionen.

Zeichenvorrat ausgeben und Dauertest

```
CSI 9 v  0x1b,0x5b,0x39,0x76
```

Auf dem Bildschirm wird der gesamte Zeichengenerator, also alle verschiedenen Zeichen, ausgegeben.

Diese beiden Folgen sind kombinierbar und mit folgender Steuerfolge aufzurufen:

```
CSI 8 v CSI 9 v
              0x1b,0x5b,0x38,0x76,0x1b,0x5b,0x39,0x76
```

Es werden Testbilder mit nacheinander allen 512 Zeichen des Matrixgenerators ausgegeben. Dieser Test entspricht dem Dauertest, der mit Schalter S8 des DIP-FIX-Diagnose-Schalters im Anschlußfeld des Bildschirms einschaltbar ist.

7 Zeichensätze und Adressen des Matrixgenerators

Der verwendete Matrixgenerator enthält 512 verschiedene Zeichen in einer 8x14-Punkt-Auflösung.
Bei der Darstellung auf dem 25KHz-12"-Bildschirm kommt eine 9x14-Punkt-Auflösung zur Anwendung.
Um dies mit der gegebenen Matrixgenerator-Information zu ermöglichen, wird durch eine Zusatzschaltung vor jedem Zeichen eine zusätzliche Spalte eingefügt.
Diese Spalte beinhaltet normalerweise Leerinformation. Nur bei Zeichen, die das volle Zeichenraster beanspruchen, wird, entsprechend der Information des vorherigen Zeichens, in der betreffenden Reihe Information eingefügt oder nicht eingefügt.
Dadurch ist eine durchgehende Darstellung möglich.

Hinweis

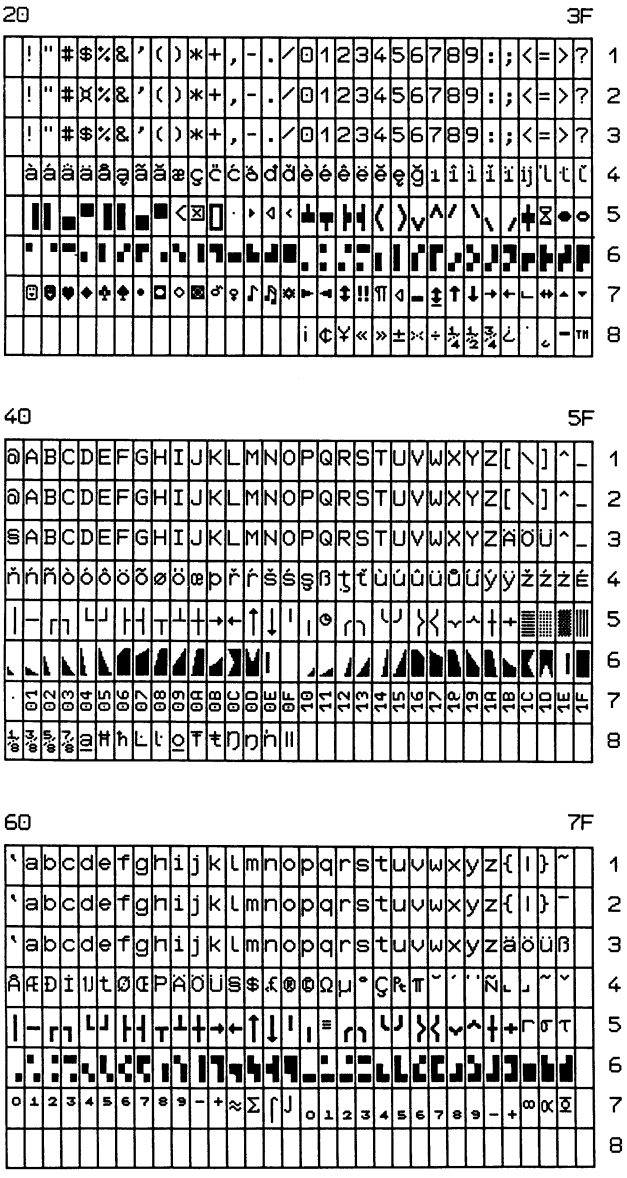
Bedieneinheiten mit Schwarz-Weiß-Bildschirm haben ein anderes Zeichengenerator-ROM als die "alten" Bedieneinheiten mit Schwarz-Grün-Bildschirm. Der Zeichensatz Mosaik wurde durch den Zeichensatz Facet ersetzt.
Bedieneinheiten mit Schwarz-Weiß-Bildschirm können mit dem "alten" Zeichengenerator-ROM ausgerüstet werden.
Das "alte" Zeichengenerator-ROM ist unter der Bestell-Nummer

PC-X/MX 12 ZOLL BS 97801/397P26361/D311/V12

erhältlich.

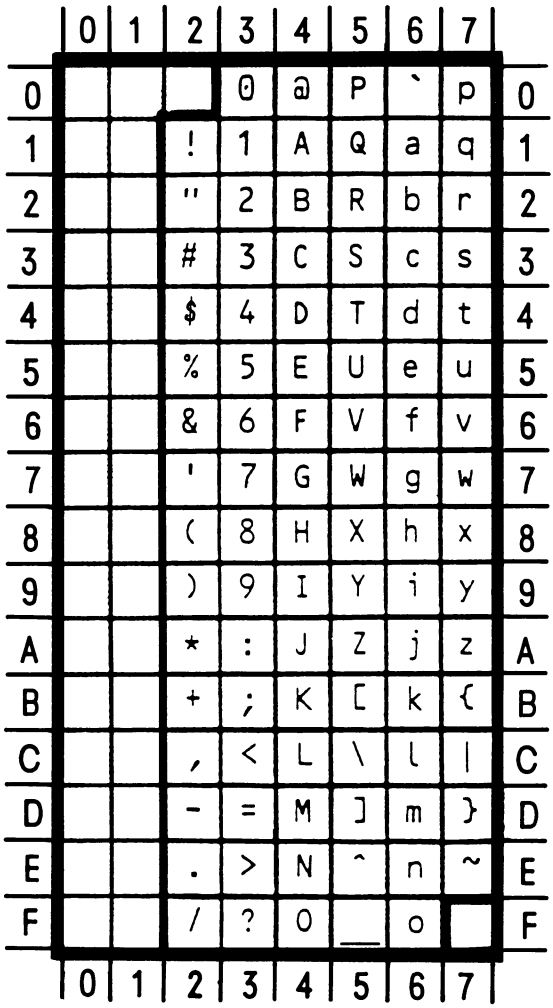
Das Zeichengenerator-ROM muß vom Service ausgetauscht werden.

7.1 Übersicht der zu Zeichensätzen zusammengefaßten Zeichen



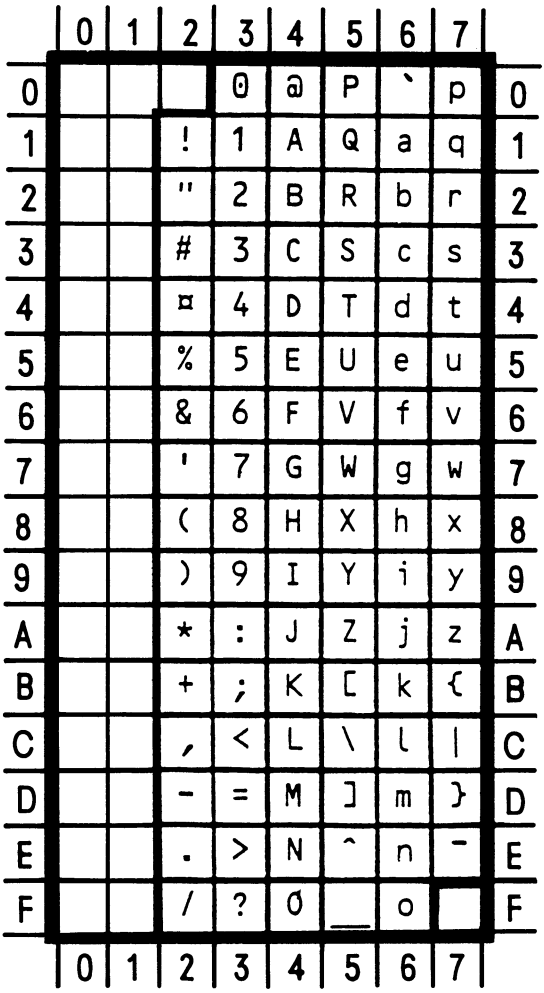
1 = Zeichensatz International A
2 = Zeichensatz International
3 = Zeichensatz Deutsch
4 = Zeichensatz Euro
5 = Zeichensatz Klammern
6 = Zeichensatz Mosaic
7 = Zeichensatz IBM
8 = Zeichensatz Mathematik

7.1.1 Zeichensatz: International A



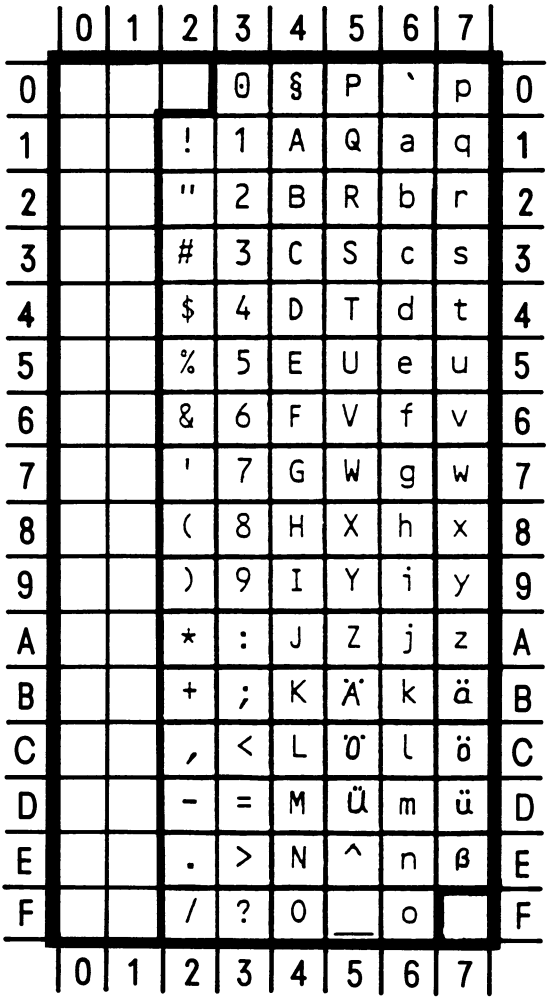
	0	1	2	3	4	5	6	7	
0				0	@	P	`	p	0
1			!	1	A	Q	a	q	1
2			"	2	B	R	b	r	2
3			#	3	C	S	c	s	3
4			$	4	D	T	d	t	4
5			%	5	E	U	e	u	5
6			&	6	F	V	f	v	6
7			'	7	G	W	g	w	7
8			(	8	H	X	h	x	8
9			)	9	I	Y	i	y	9
A			*	:	J	Z	j	z	A
B			+	;	K	[	k	{	B
C			,	<	L	\	l	|	C
D			-	=	M	]	m	}	D
E			.	>	N	^	n	~	E
F			/	?	O	_	o		F
	0	1	2	3	4	5	6	7	

7.1.2 Zeichensatz: International



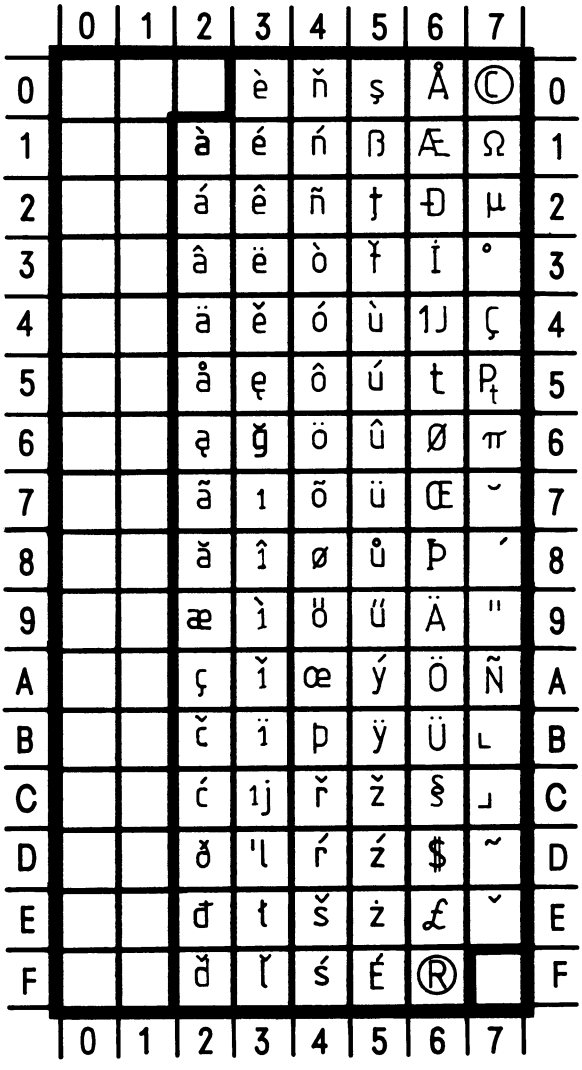
	0	1	2	3	4	5	6	7	
0				0	@	P	`	p	0
1			!	1	A	Q	a	q	1
2			"	2	B	R	b	r	2
3			#	3	C	S	c	s	3
4			¤	4	D	T	d	t	4
5			%	5	E	U	e	u	5
6			&	6	F	V	f	v	6
7			'	7	G	W	g	w	7
8			(	8	H	X	h	x	8
9			)	9	I	Y	i	y	9
A			*	:	J	Z	j	z	A
B			+	;	K	[	k	{	B
C			,	<	L	\	l	|	C
D			-	=	M	]	m	}	D
E			.	>	N	^	n	~	E
F			/	?	O	_	o		F
	0	1	2	3	4	5	6	7	

7.1.3 Zeichensatz: Deutsch

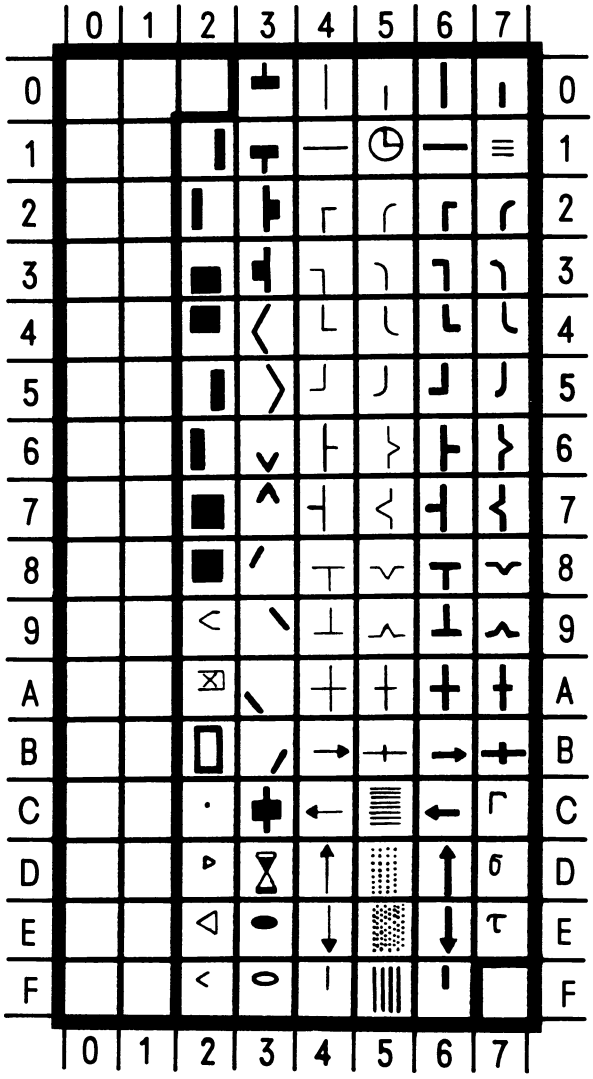


	0	1	2	3	4	5	6	7	
0				0	§	P	`	p	0
1			!	1	A	Q	a	q	1
2			"	2	B	R	b	r	2
3			#	3	C	S	c	s	3
4			$	4	D	T	d	t	4
5			%	5	E	U	e	u	5
6			&	6	F	V	f	v	6
7			'	7	G	W	g	w	7
8			(	8	H	X	h	x	8
9			)	9	I	Y	i	y	9
A			*	:	J	Z	j	z	A
B			+	;	K	Ä	k	ä	B
C			,	<	L	Ö	l	ö	C
D			-	=	M	Ü	m	ü	D
E			.	>	N	^	n	ß	E
F			/	?	O	_	o		F
	0	1	2	3	4	5	6	7	

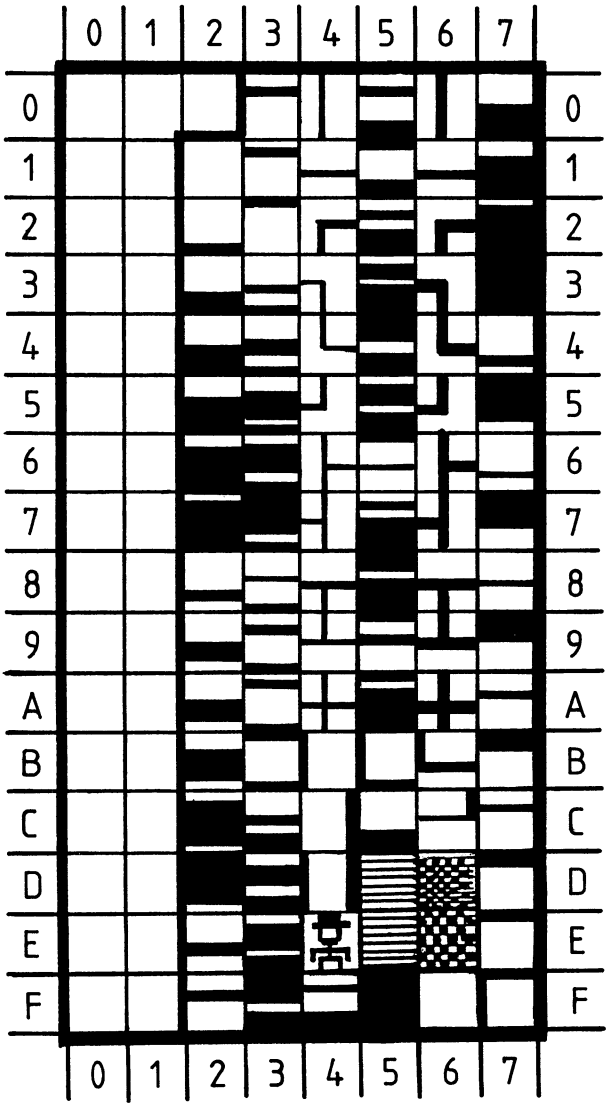
7.1.4 Zeichensatz: Euro



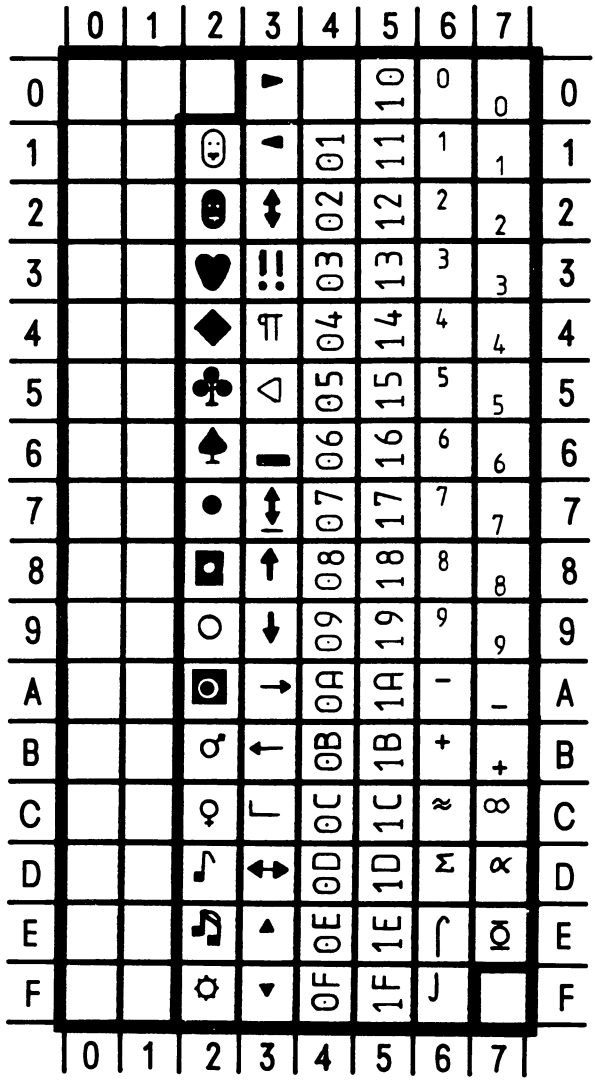
7.1.5 Zeichensatz: Klammern



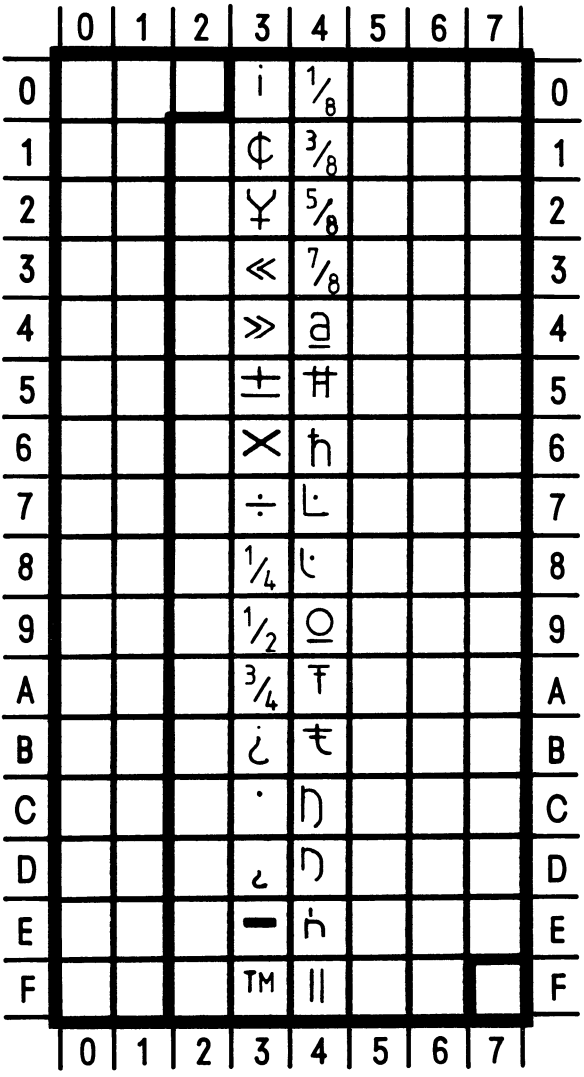
7.1.6 Zeichensatz: Facet



7.1.7 Zeichensatz: IBM



7.1.8 Zeichensatz: Mathematisch

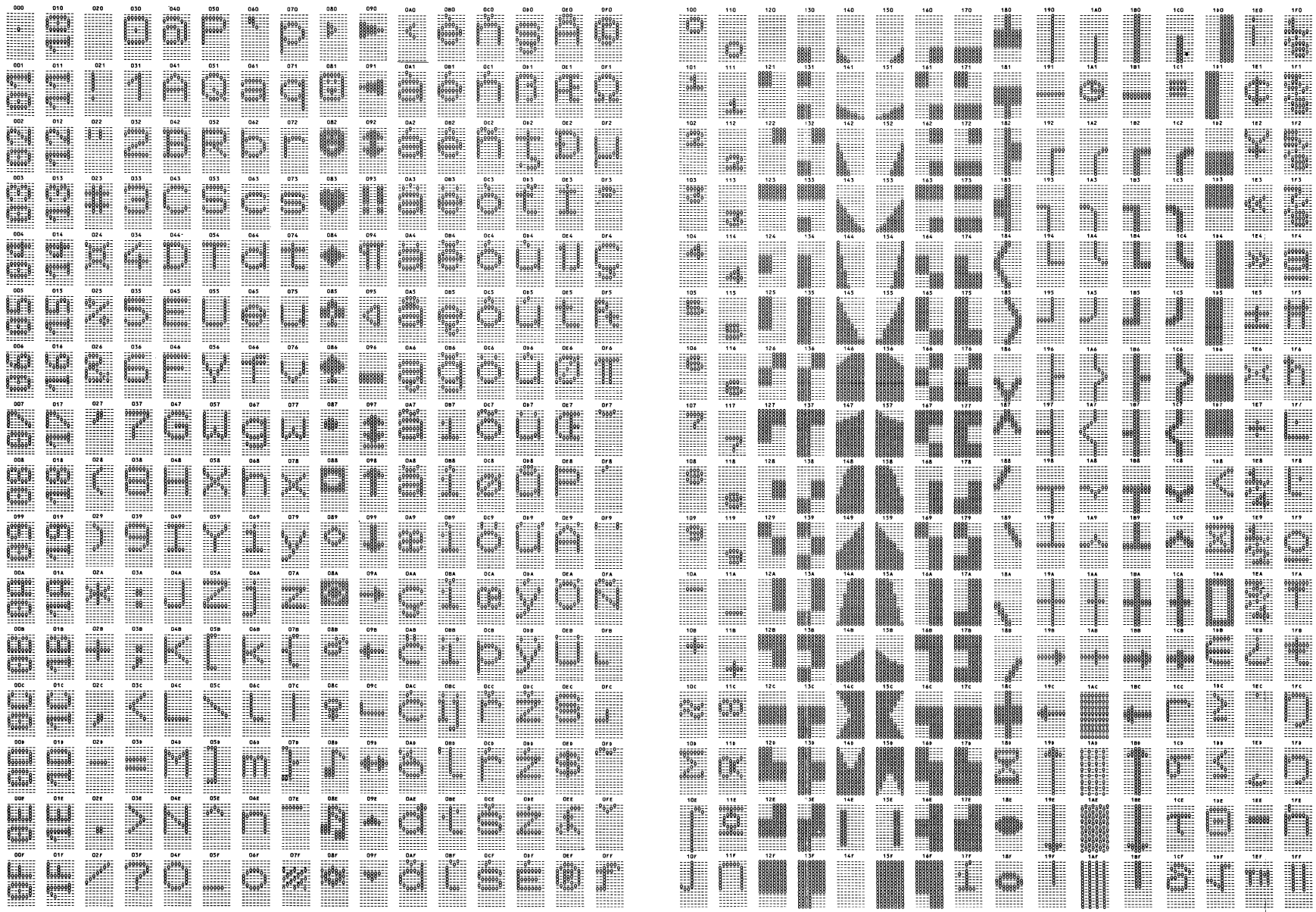


7.2 Unterschiedliche Zeichen bei nationalen Tastatur-Varianten

Gegenüberstellung mit interner Matrixgeneratoradresse bei PC-X und PC-MX2.

Hex. Code	International A	International	Deutsch	Norwegisch	Schwedisch	Dänisch	Französisch	Britisch	Italienisch	Spanisch
23	023	023	023	023	023	023	023	0EE	0EE	023
	#	#	#	#	#	#	#	£	#	#
24	0ED	024	0ED	0ED	024	0ED	0ED	0ED	0ED	0ED
	$	¤	$	$	¤	$	$	$	$	$
40	040	040	0EC	040	0DF	040	0A1	040	0EC	040
	@	@	§	@	E	@	à	@	§	@
5B	05B	05B	0E9	0E1	0E9	0E1	0F3	05B	0F3	1E0
	[	[	Ä	Æ	Ä	Æ	°	[	°	i
5C	05C	05C	0EA	0E6	0EA	0E6	0AA	05C	0AA	0FA
	\	\	Ö	ø	Ö	ø	ç	\	ç	Ñ
5D	05D	05D	0EB	0E0	0E0	0E0	0EC	05D	0B1	1EB
	]	]	Ü	Å	A	A	§	]	é	¿
5E	05E	05E	05E	0EB	0EB	0EB	05E	05E	05E	05E
	^	^	^	Ü	Ü	Ü	^	^	^	^
5F	05F	05F	05F	05F	05F	05F	05F	05F	05F	05F
	_	_	_	_	_	_	_	_	_	_
60	060	060	060	060	0B1	060	060	060	0D4	060
	`	`	`	`	é	`	`	`	ù	`
7B	07B	07B	0A4	0A9	0A4	0A9	0B1	07B	0A1	027
	{	{	ä	χ	ä	χ	é	{	à	´
7C	07C	07C	0C6	0C8	0C6	0C8	0D4	07C	0C3	0C2
	|	|	ö	ø	ö	ø	ù	|	ò	ñ
7D	07D	07D	0D7	0A5	0A5	0A5	0B0	07D	0B0	0AA
	}	}	ü	å	å	å	è	}	è	ç
7E	0FD	07E	0D1	0F9	0D7	0D7	0F9	07E	0B9	0F9
	~	–	ß	¨	ü	ü	¨	–	ì	¨

7.3 Zeichen des Matrixgenerators mit internen Adressen



7.4 Ladbarer Zeichengenerator

Während der Initialiserungsphase des Terminals wird der Ladbare-Zeichengenerator mit dem aus 512 Zeichen bestehenden Grundzeichenvorrat geladen.
Wenn erforderlich, können einzelne Zeichen beliebig neu gebildet werden.
Die Ladeinformation für die Neubildung eines Zeichens im Zeichengenerator ist wie folgt aufgebaut:

ESC R B	Adresse		ESC\
(1B) (52) (42)	3 Bytes	28 Bytes	(1B) (5C)
Header	ZG-Ladeinformation		Stringterminator

Als Zeichengenerator-Ladeinformation sind die Werte von 30 H bis 39 H und von 41 H bis 46 H zulässig.
Die ersten 3 Bytes der ZG-Ladeinformation ergeben die Adresse des Zeichens im Zeichengenerator welches neu aufgebaut wird.
Der Aufbau des Zeichens steckt in den unmittelbar nachfolgenden 14 Wertepaaren (= 28 Bytes).
Die gesamte Zeichengenerator-Ladeinformation wird zunächst zwischengespeichert. Erst nach Empfang des Stringterminators beginnt die Auswertung und der Eintrag in den Zeichengenerator.
Dazu werden die ersten 3 Bytes in ASCII-Zeichen umgewandelt und ergeben die Adresse des zu verändernden Zeichens im Zeichengenerator (000 H bis 1FF H).
Jeweils 2 ASCII-Zeichen werden aus dem Zwichenspeicher geholt, in die entsprechenden Binärwerte umgewandelt und zu einem Byte zusammengefügt. Dabei bildet der erste von beiden Binärwerten das "high nibble", der zweite das "low nibble" des neuen Scanline-Wertes, der sodann im Zeichengenerator abgespeichert wird.
Auf diese Weise wird das neue Zeichen von der Scanline 0 (oben) bis zur Scanline 14 (unten) gebildet.

Beispiel

Wenn mehrere Zeichen verändert werden sollen, so sind die einzelnen Zeichengenerator-Ladeinformationen durch die Angabe eines Separatorzeichens (; = 3B H) voneinander zu trennen.

Die zuletzt vom System zum Terminal übermittelte Zeichengenerator-Information kann vom System abgefragt werden.

Die Steuerfolge dafür lautet:

CSI	12	y"=	ESC	[	1	2	y	
				(1B)	(5B)	(31)	(32)	(79)

Bemerkung

Die Laderoutine hat ein Timeout von ca. 80 ms. Wenn innerhalb dieser Zeit kein neues Zeichen empfangen wird, wird die Laderoutine abgebrochen und die bisherige Zeichengenerator-Information bleibt gültig.

Um die neue Zeichengenerator-Information wirksam werden zu lassen

- darf der Timeout (ca. 80 ms) nicht überschritten werden
- muß die gesamte ZG-information genau 31 Bytes oder 31 und ein Vielfaches von 31 Bytes + Separator lang sein
- dürfen nur zulässige Zeichen empfangen worden sein (30 H...39 H und 41 H...46 H)
- muß am Ende der Stringterminator (1B) (5C) empfangen worden sein.

7 Zeichensätze und Adressen des Matrixgenerators

Der verwendete Matrixgenerator enthält 512 verschiedene Zeichen in einer 8x14-Punkt-Auflösung.
Bei der Darstellung auf dem 25KHz-12"-Bildschirm kommt eine 9x14-Punkt-Auflösung zur Anwendung.
Um dies mit der gegebenen Matrixgenerator-Information zu ermöglichen, wird durch eine Zusatzschaltung vor jedem Zeichen eine zusätzliche Spalte eingefügt.
Diese Spalte beinhaltet normalerweise Leerinformation. Nur bei Zeichen, die das volle Zeichenraster beanspruchen, wird, entsprechend der Information des vorherigen Zeichens, in der betreffenden Reihe Information eingefügt oder nicht eingefügt.
Dadurch ist eine durchgehende Darstellung möglich.

Hinweis

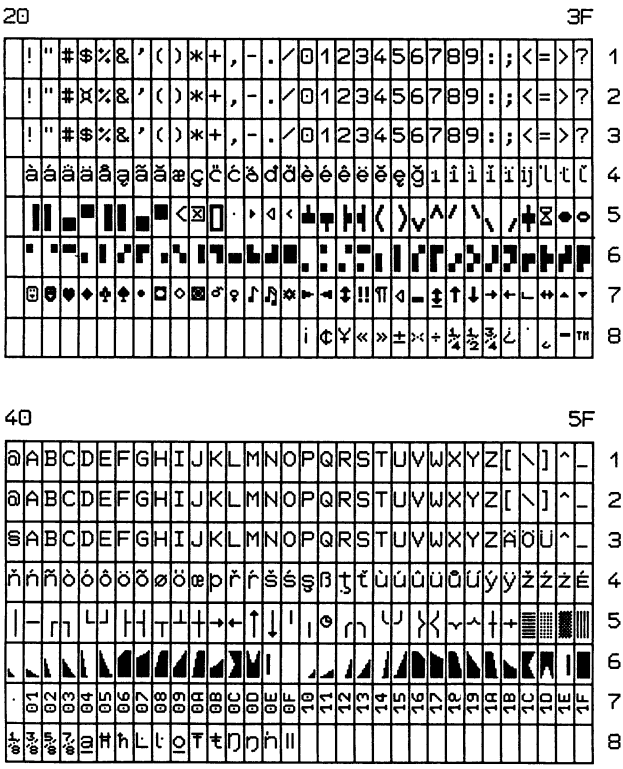
Bedieneinheiten mit Schwarz-Weiß-Bildschirm haben ein anderes Zeichengenerator-ROM als die "alten" Bedieneinheiten mit Schwarz-Grün-Bildschirm. Der Zeichensatz Mosaik wurde durch den Zeichensatz Facet ersetzt.
Bedieneinheiten mit Schwarz-Weiß-Bildschirm können mit dem "alten" Zeichengenerator-ROM ausgerüstet werden.
Das "alte" Zeichengenerator-ROM ist unter der Bestell-Nummer

PC-X/MX 12 ZOLL BS 97801/397P26361/D311/V12

erhältlich.

Das Zeichengenerator-ROM muß vom Service ausgetauscht werden.

7.1 Übersicht der zu Zeichensätzen zusammengefaßten Zeichen



1 = Zeichensatz International A
2 = Zeichensatz International
3 = Zeichensatz Deutsch
4 = Zeichensatz Euro
5 = Zeichensatz Klammern
6 = Zeichensatz Mosaic
7 = Zeichensatz IBM
8 = Zeichensatz Mathematik

8 Tastaturbelegungen (PC-MX2/PC-X10)

Die Tastaturen von PC-X10 und der Bedieneinheit des PC-MX2 unterscheiden sich weder in der Hardware noch im Design.
Einplatzsysteme unter SINIX V1.2A und Mehrplatzsysteme unter SINIX V2.0 initialisieren die Tastatur über unterschiedliche Wege.
Grund dafür ist die unterschiedliche Systemauslegung.

Grundsätzlich gilt:

- Beim Konfigurieren des PC wird für für jede Bedieneinheit ein Eintrag in der Datei /etc/termcap vorgenommen.
  Der Eintrag in /etc/termcap besteht aus 2 Feldern:

  is enthält die Steuerfolge, die den Bildschirm und den Ladbaren-Zeichengenerator initialisiert.

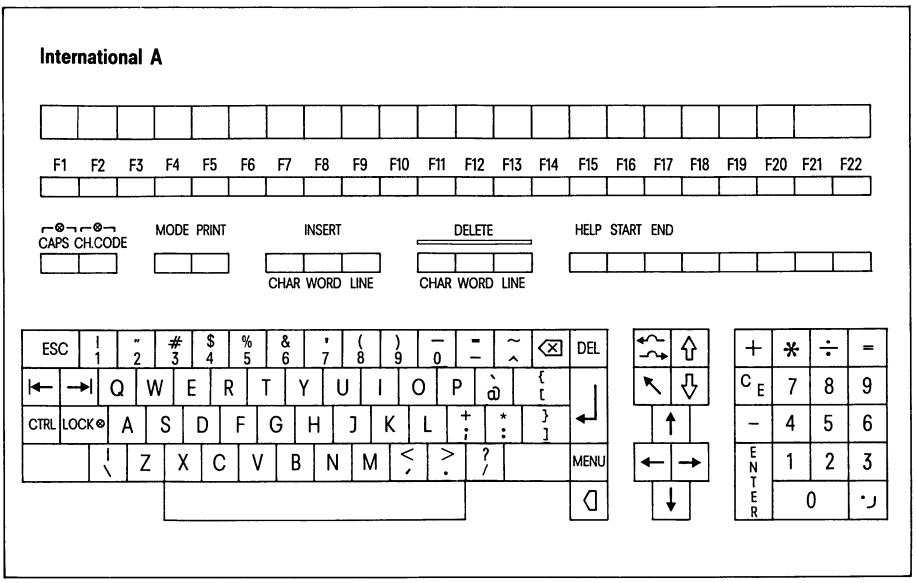
  LK entält die Kennung für die Tastaturbelegungstabelle.

- Beim Einschalten des PC und bei Betätigung der Taste END wird dieser Eintrag ausgewertet und die Zeichensätze entsprechend geladen bzw. die Tastaturvariante vorgegeben.
  /etc/getty liest beide Felder ein und übergibt den Inhalt des Feldes LK an /etc/keyload .
  /etc/keyload übergibt die entsprechende Tastaturbelegungstabelle an die Bedieneinheit.
  Danach initialisert /etc/getty Bildschirm und Zeichengenerator der Bedieneinheit mit dem Inhalt des Feldes is .

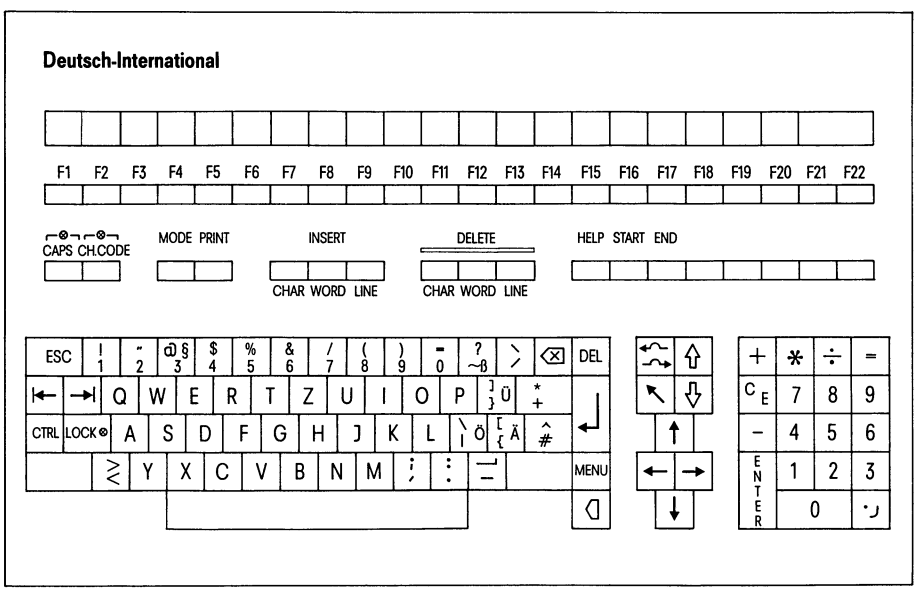
- Nationale Tastaturvarianten werden nur durch den Eintrag in /etc/termcap und durch den Austausch der Tastenkappen auf der internationalen Tastatur 97801-131/97811-131 erstellt.

8.1 Nationale und internationale Tastaturbelegungen

International (V1) 97801-131/97811-131



Deutsch/International (V2) 97801-132/97811-132



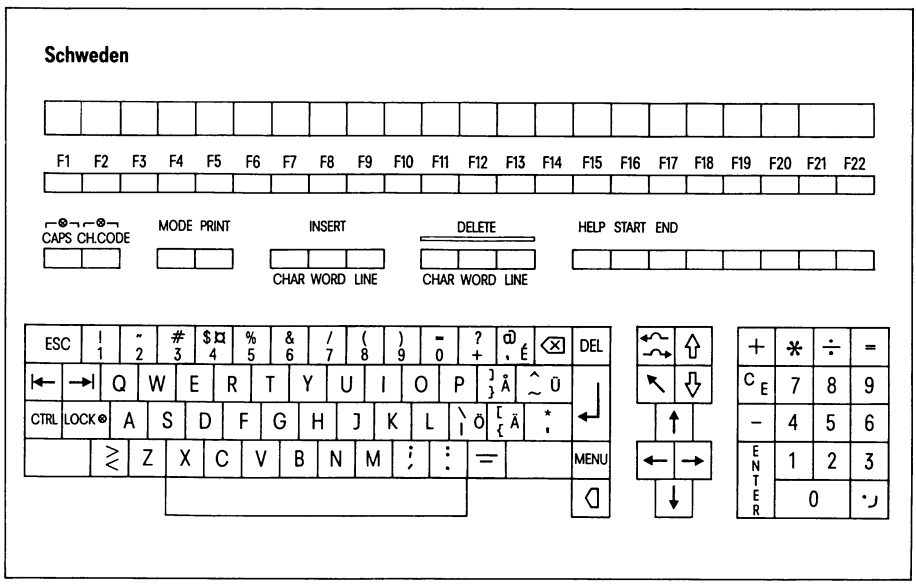
Belgisch-Flämisch

Für Belgisch-Flämisch wird International angeboten.

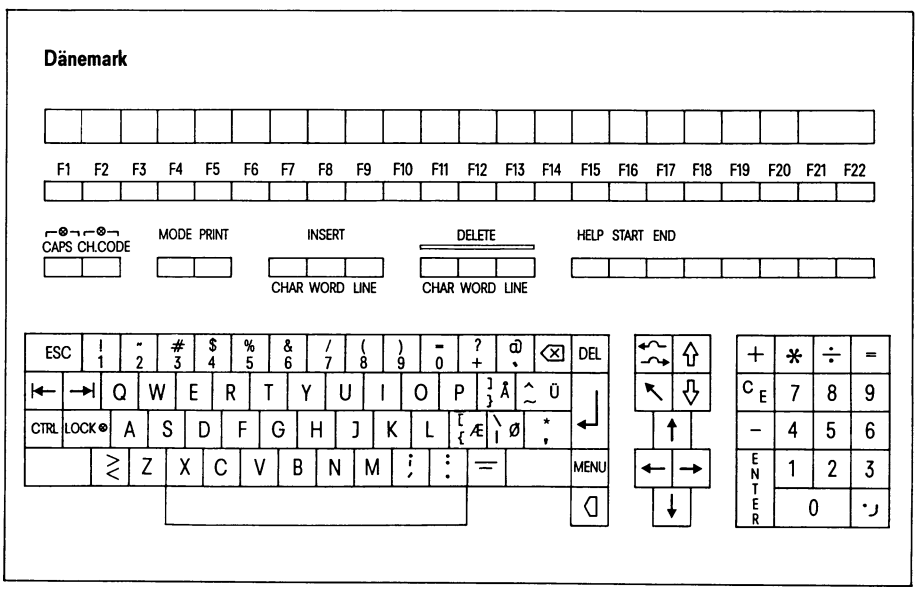
Belgisch-Französisch

Für Belgisch-Französisch wird Französisch angeboten.

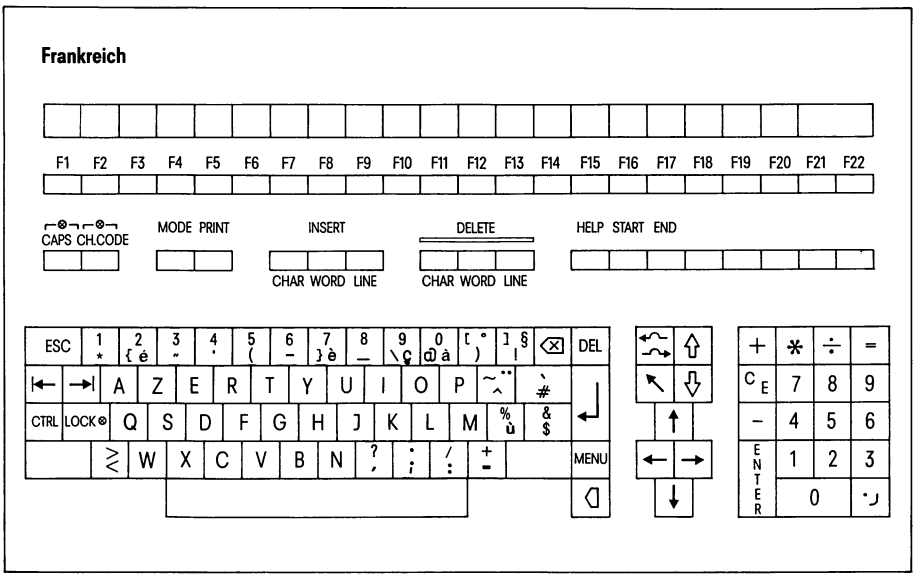
Schwedisch (V4) 97801-131/97811-131 und Tastenkappensatz 97801-144/97811-144



Dänisch (V5) 97801-131/97811-131 und Tastenkappensatz 97801-145/97811-145



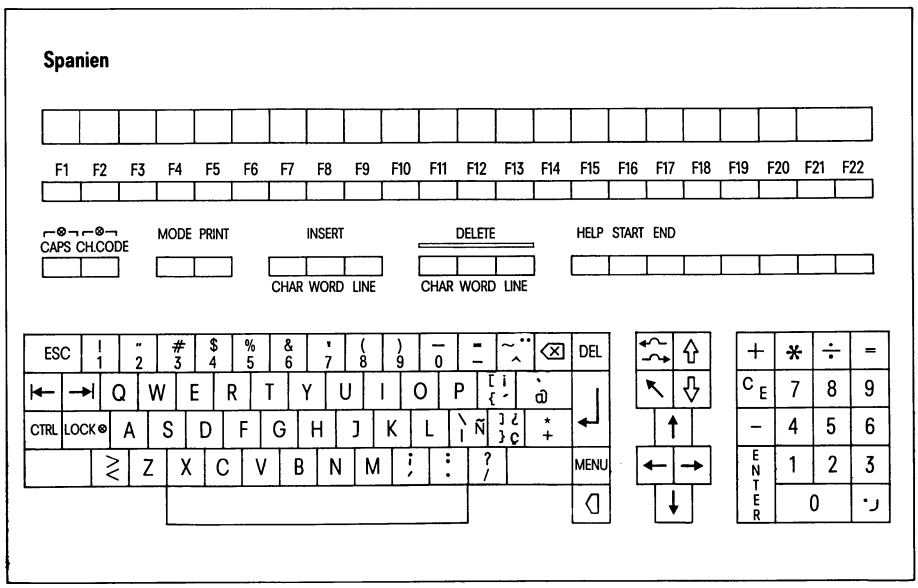
Französisch (V6) 97801-131/97811-131 und Tastenkappensatz 97801-146/97811-146



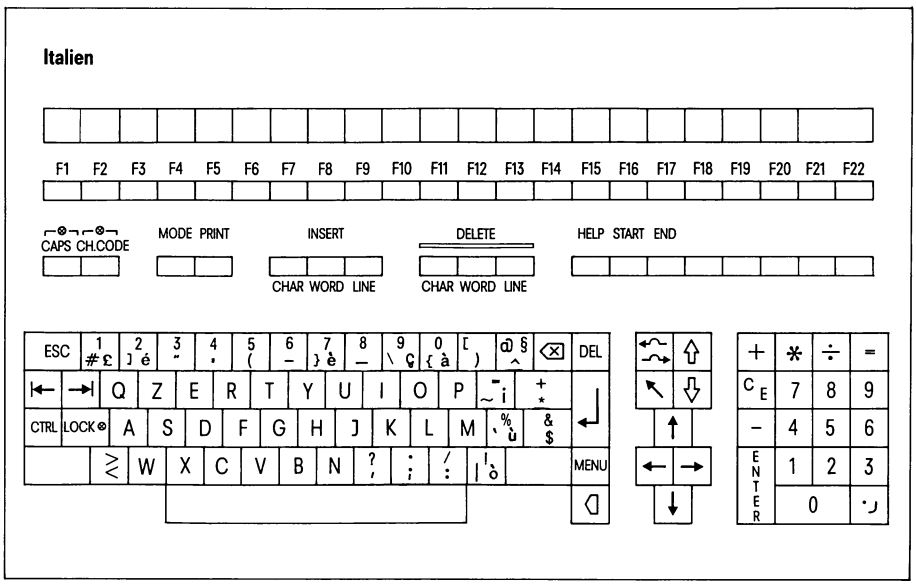
Schweizerisch

Für Schweizerisch wird Deutsch/International und Französisch angeboten.

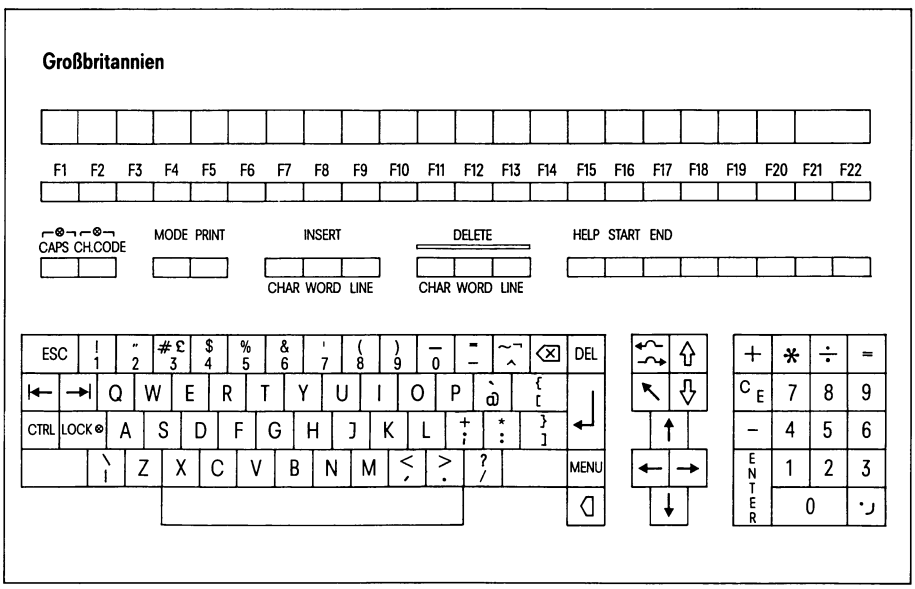
Spanisch (V9) 97801-131/97811-131 und Tastenkappensatz 97801-149/97811-149



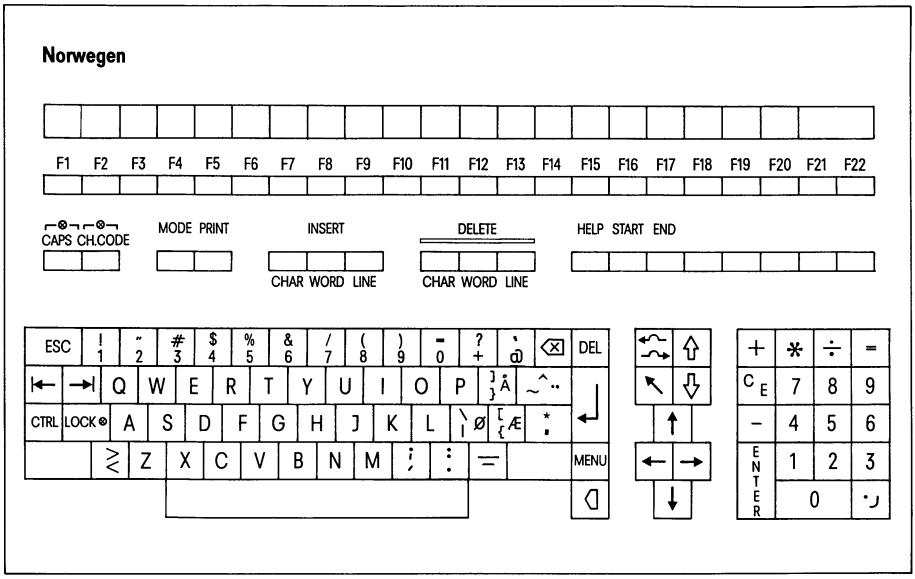
Italienisch (V10) 97801-131/97811-131 und Tastenkappensatz 97801-150/97811-150



Britisch (V10) 97801-131/97811-131 und Tastenkappensatz 97801-153/97811-153



Norwegisch (V10) 97801-131/97811-131 und Tastenkappensatz 97801-154/97811-154



8.2 Tastaturbelegungstabelle

Die Tastatur sendet nur noch Platzcodes für die Betätigung einer Taste an die Steuereinheit (siehe Kapitel 1 und 2):

- Beim Drücken einer Taste sendet die Tastatur den Make-Code der betätigten Taste.
- Beim Loslassen einer Taste sendet die Tastatur den Break-Code der Taste, die losgelassen wurde.

Alle Funktionen, die mit der Tastaturbelegung zusammenhängen, werden von der Steuereinheit erzeugt.
Die Tastaturbelegung mit den Umschaltebenen und das Sperren der AutorepeatFunktion für einzelne Tasten sind frei wählbar.

Achtung

Die Tasten können nur mit Einzelzeichen belegt werden.

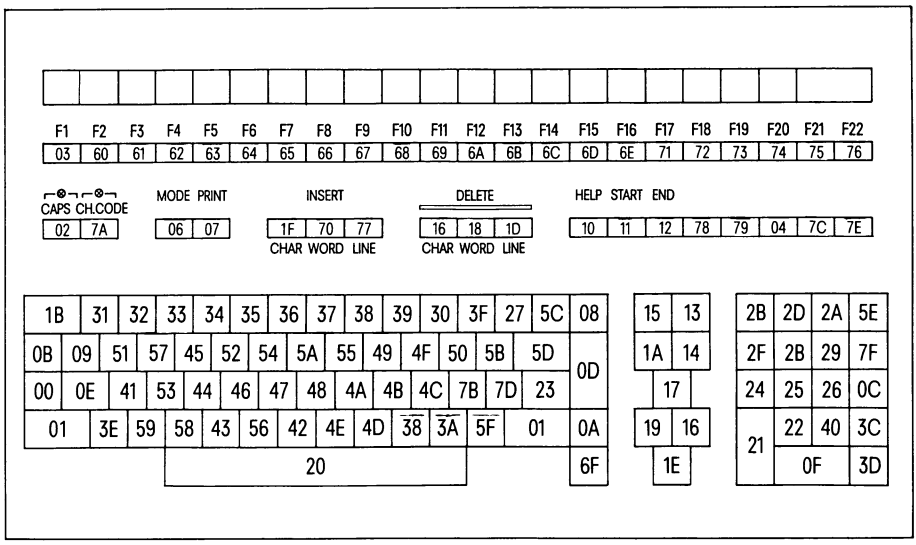


Bild 8-1 Tastatur-Platzcodes

Make- und Break-Codes sind wie folgt aufgebaut:

- Die Tastatur sendet Acht-Bit-Datenwörter zur Steuereinheit.
- Jeder Taste der Tastatur ist ein Sieben-Bit-Platzcode zugeordnet.
- Make- und Break-Code enthalten den Platzcode der betätigten Taste in Bit 0 bis Bit 6 .
- Bit 7 ist beim Make-Code gelöscht.
- Bit 7 ist beim Break-Code gesetzt.

Beispiel Leertaste:

```
Platzcode     0x20

Make-Code     0x20    (wird beim Drücken der Taste gesendet)

Break-Code    0xA0    (0x20 + 0x80
                       wird beim Lösen der Taste gesendet)
```

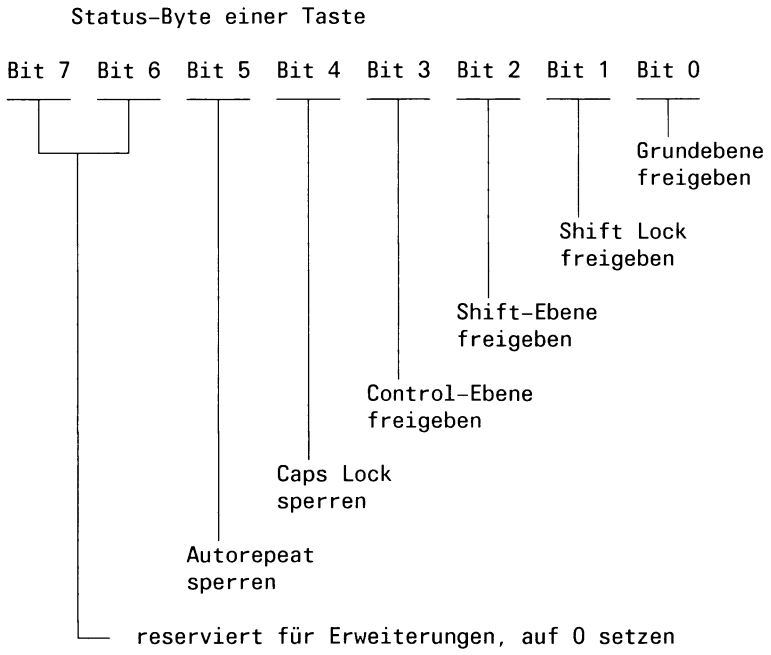
8.2.1 Erstellen der Tastaturbelegungstabelle

Die Tastaturbelegungstabelle besteht aus mehreren Untertabellen.
Jede dieser Untertabellen ist 128 Byte lang (siehe auch 8.2.2). Jedes Byte innerhalb einer Untertabelle wird über den Platzcode der betätigten Taste angesprochen.

Die Tastaturbelegungstabelle des PC-MX2 besteht aus 4 Untertabellen:

Status

Diese Untertabelle enthält die Bearbeitungsvorschriften für die von der Tastatur gesendeten Daten. Die Bearbeitungsvorschrift ist bitweise verschlüsselt:



Unshift (Grundebene)

Tastenbelegung ohne Umschalt- oder Control-Taste

Shift

Tastenbelegung bei gleichzeitig gedrückter Umschalttaste

Control

Tastenbelegung bei gleichzeitig gedrückter Control-Taste

Die Tastaturbelegungstabelle des PC-X10 besteht aus 5 Untertabellen:

lowercase (Grundebene)
Tastenbelegung ohne Umschalttaste oder Control-Taste

uppercase (Shift-Ebene)
Tastenbelegung bei gleichzeitig gedrückter Umschalttaste

normcaps
Tastenbelegung bei eingeschaltetem Caps-Lock

invcaps
Tastenbelegung bei eingeschaltetem Caps-Lock und gleichzeitig gedrückter Umschalttaste

control
Tastenbelegung bei gleichzeitig gedrückter Control-Taste

8.2.2 Laden der Tastaturbelegungstabelle

Die Systemeinheit übergibt die Tastaturbelegungstabelle als hexadezimal-codierte ASCII-Zeichenkette an die Bedieneinheit.
Diese Zeichenkette wird vor der Übertragung zur Bedieneinheit um einen Vorspann und eine Abschluß-Sequenz erweitert.
Zulässig innerhalb der Tastaturbelegungstabelle sind nur die ASCII-Zeichen 0 ... 9 (0x30 ... 0x39) und A ... F (0x41 ... 0x46).

Die Bedieneinheit wandelt die Hexadezimal-Zeichenkette in die binäre Darstellung der Tastaturbelegungstabelle um.
Die neue Tastaturbelegungstabelle wird erst wirksam, wenn die Übertragung erfolgreich abgeschlossen wurde.

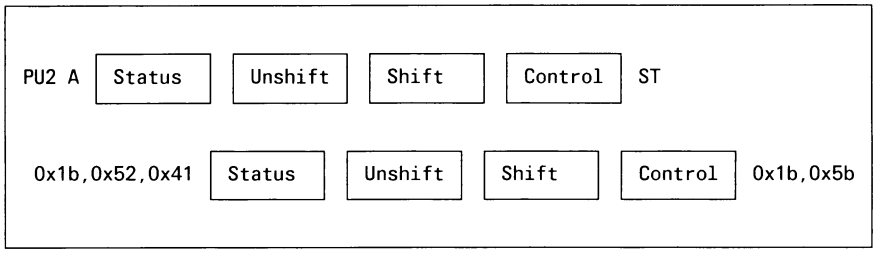
Hinweis

Die Tastaturbelegungstabelle kann über das Kommando

`/etc/keyload daen` ('daen' ist nur Beispiel für die Landerkennung)

geladen werden.

Lade-Kommando für PC-MX2



Ladefehler beim PC-MX2

Wenn beim Laden der Tastaturbelegungstabelle ein Fehler auftritt, sendet die Bedieneinheit keine Fehlermeldung zur Systemeinheit. Folgende Fehler können auftreten:

Timeout
Länger als 80 ms kein neues Zeichen der Tastaturbelegungstabelle empfangen.

Overrun
Die Tastaturbelegungstabelle ist einschließlich Vorspann und Abschluß-Sequenz länger als 1029 Zeichen.

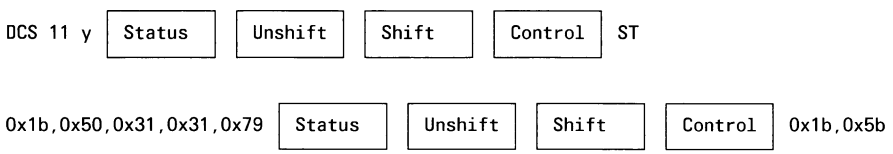
Übertragungsfehler
Falsche Parität, falscher Zeichenrahmen, unzulässige Zeichen

Falls erforderlich kann die aktuelle Tastaturbelegungstabelle zurückgelesen werden.

Rücklesekommando für PC-MX2

```
CSI 11 y  0x1b,0x5b,0x31,0x31,0x79
```

Die Bedieneinheit sendet auf dieses Kommando die aktuelle TastaturbelegungsTabelle an die Systemeinheit zurück. Die Tastaturbelegungstabelle wird im folgenden Format übertragen:



8.3 Beispiele

8.3.1 Beispiel 1:

Festlegung der Zeichensätze und Tastaturbelegungstabellen
in der Datei /etc/termcap

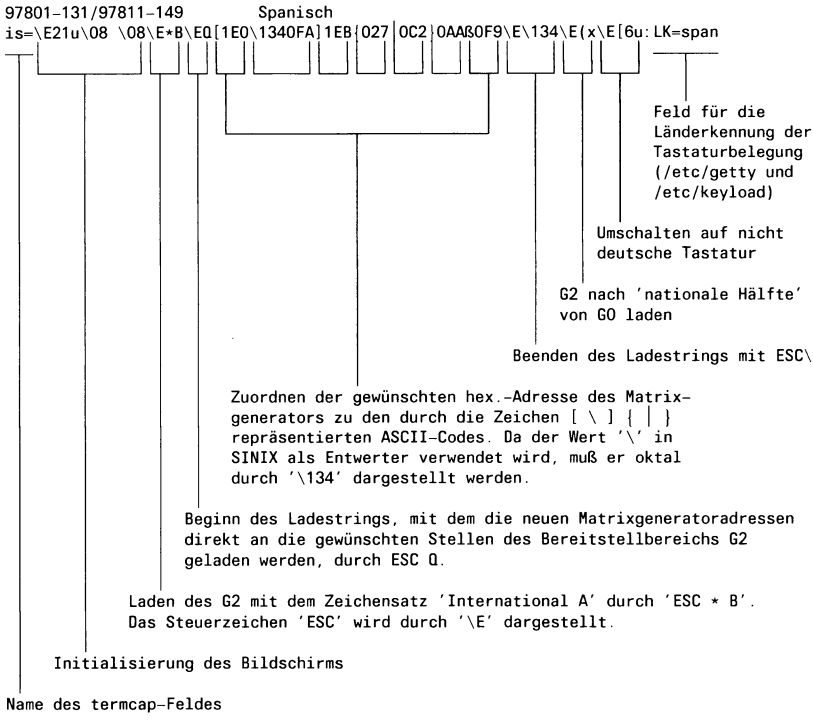
Beim Einspielen der SINIX-Software wird in Abhängigkeit der Angaben bei der Systeminstallation für jede Bedieneinheit ein Eintrag in der Datei /etc/termcap vorgenommen. Beim Einschalten des PC und bei Betätigung der Taste END wird dieser Eintrag zur Bedieneinheit gesendet und die Zeichensätze entsprechend geladen bzw. die Tastaturvariante vorgegeben.

z.B. Eintrag für PC-MX2 (4-Platz-System)

`console|:is=\E(B\E[6u:LK=inter`	internationale Tastatur
`tty00|:is=\E(B\E[6u:LK=inter`	internationale Tastatur
`tty01|:is=\E(K\E[7u:LK=deut`	deutsche Tastatur
`tty02|:is=\E(K\E[7u:LK=deut`	deutsche Tastatur

Die Einträge für die möglichen Tastaturvarianten sind wie folgt:

```
97801-131/97811-131        International
is=\E21u\08 \08\E(B\E[6u:LK=inter

97801-132/97811-132        Deutsch (Kurzform für PC-MX2, Tastaturbelegung
                           nach Einschalten oder Reset des Terminals)
is=\E21u\08 \08\E(K\E[7u:LK=deut

97801-132/97811-132        Deutsch (vollstandiger Eintrag für PC-MX500)
is=\E21u\08 \08\E*B\EQ@0EC[0E9\1340EA]0EB{0A4|0C6}0D7ß0D1\E\134\E(x\E[6u:LK=deut

97801-131/97811-145        Dänisch
is=\E21u\08 \08\E*B\EQ[0E1\1340E6]0E0\1360EB{0A90C8}0A5ß0D7\E\134\E(x\E[6u:\
LK=daen

97801-131/97811-146        Französisch
is=\E21u\08 \08\E*B\EQ@0A1[0F3\1340AA]0EC{0B1|0D4}0B0ß0F9\E\134\E(x\E[6u:\
LK=franz

97801-131/97811-150        Italienisch
is=\E21u\08 \08\E*B\EQ#0EE@0EC[0F3\1340AA]0B1«0D4{0A1|0C3}0B0ß0B9\E\134\E(x\E[6u:\
LK=ital

97801-131/97811-144        Schwedisch
is=\E21u\08 \08\E*B\EQ$024@0DF[0E9\1340EA]0E0\1360EB«0B1{0A4|0C6}0A5ß0D7\E\134\E(x\E[6u:\
LK=schwed

97801-131/97811-154        Norwegisch
is=\E21u\08 \08\E*B\EQ[0E1\1340E6]0E9{0A9|0C8}0A5ß0F9\E\134\E(x\E[6u:LK=norweg

97901-131/97811-153        Britisch
is=\E21u\08 \08\E*B\EQ#0EEß07E\E\134\E(x\E[6u:LK=brit
```


97801-131/97811-149 Spanisch
is=\E21u\08 \08\E*B\EQ[1EO\134OFA]1EB|027|OC2|OAAßOF9\E\134\E(x\E[6u:LK=span
Feld für die Länderkennung der Tastaturbelegung (/etc/getty und /etc/keyload)
Umschalten auf nicht deutsche Tastatur
G2 nach 'nationale Hälfte' von GO laden
Beenden des Ladestrings mit ESC\
Zuordnen der gewünschten hex.-Adresse des Matrix-generators zu den durch die Zeichen [\ ] { | } repräsentierten ASCII-Codes. Da der Wert '\' in SINIX als Entwerter verwendet wird, muß er oktal durch '\134' dargestellt werden.
Beginn des Ladestrings, mit dem die neuen Matrixgeneratoradressen direkt an die gewünschten Stellen des Bereitstellbereichs G2 geladen werden, durch ESC Q.
Laden des G2 mit dem Zeichensatz 'International A' durch 'ESC * B'. Das Steuerzeichen 'ESC' wird durch '\E' dargestellt.
Initialisierung des Bildschirms
Name des termcap-Feldes

8.3.2 Beispiel 2: Tastaturbelegungstabellen

Die Tastaturbelegungstabellen sind hier Abdrucke editierbarer Quelldateien. Die Quelldateien müssen durch ein Programm in das Tastaturladeformat umgewandelt werden.
Die umgewandelten Tastaturbelegungstabellen müssen im Dateiverzeichnis /etc unter der Länderkennung mit der Namenserweiterung **.new** abgelegt werden.

Hinweis

Einplatzsysteme unter SINIX V1.2A besitzen im Dateiverzeichnis /etc zusätzliche Tastaturbelegungstabellen.
Der Name dieser Tabellen besteht aus der Länderkennung und der Namenserweiterung **.old** .

Urtabelle für Mehrplatzsysteme

```
/* ******************** i n t e r . n k ****************** */
/* status */
/*         00 01 02 03 04 05 06 07   08 09 0A 0B 0C 0D 0E 0F */

/* 00 /*   30 30 30 37 17 30 17 37   3f 17 17 17 17 37 30 17
/* 10 /*   17 17 17 17 17 15 17 17   17 1f 37 17 1f 17 1f 17
/* 20 /*   17 37 17 1f 17 17 17 1f   17 17 17 17 30 17 30 17
/* 30 /*   1f 17 17 1f 17 17 17 17   17 17 17 17 17 17 1f 1f
/* 40 /*   17 0f 0f 0f 0f 0f 0f 0f   0f 0f 0f 0f 0f 0f 0f 0f
/* 50 /*   0f 0f 0f 0f 0f 0f 0f 0f   0f 0f 0f 1f 17 1f 17 1f
/* 60 /*   37 37 37 37 37 37 37 37   37 37 37 37 37 37 37 17
/* 70 /*   17 37 37 37 37 37 37 17   17 17 37 1f 17 1f 17 17

/* unshift */
/*         00 01 02 03 04 05 06 07   08 09 0A 0B 0C 0D 0E 0F */

/* 00 /*   00 00 00 a0 98 00 94 f8   7f 99 9a ca e6 cd 00 e0
/* 10 /*   9e ba b7 c3 c4 b5 c0 b1   bf b4 b8 db b3 bd b2 b0
/* 20 /*   20 ed e1 5d ea e4 e5 5e   e7 e8 c6 d1 00 c5 00 eb
/* 30 /*   30 31 32 33 34 35 36 37   38 38 2e 2c e3 2e 5c 2d
/* 40 /*   e2 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
/* 50 /*   70 71 72 73 74 75 76 77   78 7a 79 40 ee 5b c7 2f
/* 60 /*   a1 a2 a3 a4 a5 a6 a7 a8   a9 c9 ab ac ad ae af da
/* 70 /*   be 90 f0 f1 ce d0 d2 bc   96 97 9c 3b b9 3a d4 e9

/* shift */
/*         00 01 02 03 04 05 06 07   08 09 0A 0B 0C 0D 0E 0F */
/* 00 /*   00 00 00 80 98 00 95 f8   7f 99 9a ca e6 cd 00 e0
/* 10 /*   9f bb b7 c3 c4 c8 c0 b1   bf b4 d8 db b3 bd b2 b0
/* 20 /*   20 ed e1 7d ea e4 e5 7e   e7 e8 c6 c1 00 c5 00 eb
/* 30 /*   5f 21 22 23 24 25 26 27   28 29 3e 3c e3 2e 7c 3d
/* 40 /*   e2 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
/* 50 /*   50 51 52 53 54 55 56 57   58 5a 59 60 ee 7b c7 3f
/* 60 /*   81 82 83 84 85 86 87 88   89 8a 8b 8c 8d 8e 8f da
/* 70 /*   be 91 92 93 cf d1 d3 bc   96 97 9d 2b b9 2a d4 e9
/* control */
/*         00 01 02 03 04 05 06 07   08 09 0A 0B 0C 0D 0E 0F */

/* 00 /*   80 80 80 80 80 80 80 80   1f 80 80 80 80 80 80 80
/* 10 /*   80 80 80 80 80 80 80 80   80 80 80 80 80 80 80 80
/* 20 /*   80 80 80 1d 80 80 80 1e   80 80 80 80 80 80 80 80
/* 30 /*   1f 80 80 80 80 80 80 80   80 80 80 80 80 80 1c 80
/* 40 /*   80 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
/* 40 /*   10 11 12 13 14 15 16 17   18 1a 19 00 80 1b 80 80
/* 60 /*   80 80 80 80 80 80 80 80   80 80 80 80 80 80 80 80
/* 70 /*   80 80 80 80 80 80 80 80   80 80 80 80 80 80 80 80
```

Urtabelle für Einplatzsysteme

```
/* ********************** d e u t . n k ******************** */
/* lowercase */
/*          00 01 02 03 04 05 06 07   08 09 0A 0B 0C 0D 0E 0F */

/* 00 /*    00 00 00 8c 80 80 80 b4   7f 09 0a 8b 36 0d 00 30
/* 10 /*    b5 b6 04 87 86 88 b9 85   bb 82 8a 1b 84 bd 83 b8
/* 20 /*    20 0d 31 23 2d 34 35 27   37 38 2f 2b 30 2a 0d b7
/* 30 /*    30 31 32 33 34 35 36 37   38 39 2e 2c 33 2c 3c 7e
/* 40 /*    32 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
/* 50 /*    70 71 72 73 74 75 76 77   78 79 7a 7d 08 2b 3d 2d
/* 60 /*    8d 8e 8f 90 91 92 93 94   95 96 97 98 99 9a 9b 80
/* 70 /*    ba 9c 9d 9e 9f 80 80 bc   80 80 00 7c 80 7b 80 39

/* uppercase */
/*          00 01 02 03 04 05 06 07   08 09 0A 0B 0C 0D 0E 0F */

/* 00 /*    00 00 00 a0 80 80 80 b4   7f 09 0a 8b 36 0d 00 30
/* 10 /*    b5 b6 04 87 86 88 b9 85   bb 82 8a 1b 84 bd 83 b8
/* 20 /*    20 0d 31 5e 2d 34 35 60   37 38 2f 2b 30 2a 0d b7
/* 30 /*    3d 21 22 40 24 25 26 27   28 29 3a 3b 33 2e 3e 3f
/* 40 /*    32 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
/* 50 /*    50 51 52 53 54 55 56 57   58 59 5a 5d 08 2a 3d 5f
/* 60 /*    a1 a2 a3 a4 a5 a6 a7 a8   a9 aa ab ac ad ae af 80
/* 70 /*    ba b0 b1 b2 b3 80 80 bc   80 80 00 5c 80 5b 80 39

/* normcaps  */
/*          00 01 02 03 04 05 06 07   08 09 0A 0B 0C 0D 0E 0F */
/* 00 /*    00 00 00 8c 80 80 80 b4   7f 09 0a 8b 36 0d 00 30
/* 10 /*    b5 b6 04 87 86 88 b9 85   bb 82 8a 1b 84 bd 83 b8
/* 20 /*    20 0d 31 23 2d 34 35 27   37 38 2f 2b 30 2a 0d b7
/* 30 /*    30 31 32 33 34 35 36 37   38 39 2e 2c 33 2c 3c 7e
/* 40 /*    32 41 42 43 44 45 46 47   48 49 4a 4b 4c 4d 4e 4f
/* 50 /*    50 51 52 53 54 55 56 57   58 59 5a 5d 08 2b 3d 2d
/* 60 /*    8d 8e 8f 90 91 92 93 94   95 96 97 98 99 9a 9b 80
/* 70 /*    ba 9c 9d 9e 9f 80 80 bc   80 80 00 5c 80 5b 80 39
/* invcaps   */
/*          00 01 02 03 04 05 06 07   08 09 0A 0B 0C 0D 0E 0F */

/* 00 /*    00 00 00 a0 80 80 80 b4   7f 09 0a 80 36 0d 00 30
/* 10 /*    b5 b6 04 87 86 89 b9 85   bb 82 8a 1b 84 bd 83 b8
/* 20 /*    20 0d 31 5e 2d 34 35 60   37 38 2f 2b 30 2a 0d b7
/* 30 /*    3d 21 22 40 24 25 26 2f   28 29 3a 3b 33 2e 3e 3f
/* 40 /*    32 61 62 63 64 65 66 67   68 69 6a 6b 6c 6d 6e 6f
/* 40 /*    70 71 72 73 74 75 76 77   78 79 7a 7d 08 2a 3d 5f
/* 60 /*    a1 a2 a3 a4 a5 a6 a7 a8   a9 aa ab ac ad ae af 80
/* 70 /*    ba b0 b1 b2 b3 80 80 bc   80 80 00 7c 80 7b 80 39

/* control   */
/*          00 01 02 03 04 05 06 07   08 09 0A 0B 0C 0D 0E 0F */

/* 00 /*    80 80 80 80 80 80 80 80   80 80 80 80 80 80 80 80
/* 10 /*    80 80 80 80 80 80 80 80   80 80 80 80 80 80 80 80
/* 20 /*    80 80 80 1e 80 80 80 00   80 80 80 80 80 80 80 80
/* 30 /*    80 80 80 80 80 80 80 80   80 80 80 80 80 80 80 80
/* 40 /*    80 01 02 03 04 05 06 07   08 09 0a 0b 0c 0d 0e 0f
/* 40 /*    10 11 12 13 14 15 16 17   18 19 1a 1d 80 80 80 1f
/* 60 /*    80 80 80 80 80 80 80 80   80 80 80 80 80 80 80 80
/* 70 /*    80 80 80 80 80 80 80 80   80 80 80 1c 80 1b 80 80
```

8.3.3 Beispiel 3: Setzen der CH-Code-Taste und Laden der Zeichensätze

Das folgende Quellprogramm finden Sie auf der Diskette 'SSHB' unter dem Dateinamen chcode.c.

```
static char SCCSID[] = "@(#)chcode.c 1.3 85/06/13";

/*------------------------------------------------------------------------
 * chcode.c
 *
 * Programm zum Setzen der CH-CODE-Taste und Laden der Zeichensaetze.
 *
 *------------------------------------------------------------------------
 *
 * Die CH-CODE-Taste ist nach dem Einschalten immer in Stellung <INT> und kann
 * dann manuell oder per Programm in die jeweils andere Stellung "gekippt"
 * werden.
 * Mit diesem Beispielprogramm besteht die Moeglichkeit, die Taste gezielt in
 * eine Stellung zu bringen und den Zeichenvorrat evtl. auch gleich zu laden.
 *
 * \E  im String wird als <ESC> ausgegeben,
 * \\  im String wird als   \   ausgegeben.
 *
 * Beispiele:   chcode int              CH-CODE-Taste in Stellung international
 *              chcode nat              CH-CODE-Taste in Stellung national
 *              chcode nat "\E(K\E[7u"  zusaetzlich deutschen Zeichensatz laden
 *                                      und deutsche Tastatur einstellen.
 *
 *------------------------------------------------------------------------
 */

#include        <stdio.h>

#define         ESC     0x1b
#define         CHCODE  "%c[5v",ESC                     /* Code umschalten */
#define         CHCDEN  "%c[11v",ESC                    /* CHCODE freigeben */
#define         ZVTEST  "%c[13v",ESC                    /* akt. ZV abfragen */

/*------------------------------------------------------------------------*/

main (argp, argv)

int     argp;
char    *argv[];

{
switch (argp)
    {
    case  3 : prints (argv[2]);                         /* fall through */
    case  2 : if  (tcode () != strcmp ("nat",argv[1]))     printf (CHCODE);
              vexit (0);
    default : fprintf (stderr, "\nusage: chcode nat/int [string]");
              fprintf (stderr, "\n(\\E  in the string means  <ESC>, ");
              fprintf (stderr, "\\\\  in the string means  \\)\n");
              exit (1);
    }
}

/*------------------------------------------------------------------------*/
```

```
prints (s)                                      /* String ausgeben, \E == ESC */

register char   *s;

{
register char   c, old;

while (c = *s++)
    {
    if  (old == '\\')       { putchar ((c == 'E') ? ESC : c); c = 0; }
    else if (c != '\\')     putchar (c);
    old = c;
    }
}

/*-------------------------------------------------------------------------*/

tcode ()                                            /* CH-CODE-Taste abfragen */

{
register char   c;

system ("stty cbreak -echo");
printf (CHCDEN); printf (ZVTEST);
if  (getchar () != ESC)
    if  (getchar () != ESC)     vexit (1);
if  (getchar () != 'P')         vexit (1);
if  (getchar () != '1')         vexit (1);
if  (getchar () != '3')         vexit (1);
if  (getchar () != 'v')         vexit (1);
c = getchar ();
if  (c < '0' || c > '3')        vexit (1);
if  (getchar () != ESC)         vexit (1);
if  (getchar () != '\\')        vexit (1);
return (++c & 1);
}

/*-------------------------------------------------------------------------*/

vexit (n)

int     n;

{
system ("stty -cbreak echo");
exit (n);
}

/*-------------------------------------------------------------------------*/
```

9 Beschreibung der TTY-Schnittstelle

Wichtiger Hinweis:
Da sich diese SINIX-Schnittstelle in letzter Zeit in Weiterentwicklung befunden hat und sich teilweise noch befindet, ist diese Beschreibung als provisorisch zu betrachten.

Die TTY-Schnittstelle ist die asynchrone Terminal-Schnittstelle des SINIX-Systems. Da die anzuschließenden Terminals unterschiedlichster Natur sein können, muß die TTY-Schnittstelle sehr flexibel, das heißt in weitem Rahmen parametrisierbar sein.

Unter Terminals versteht man nicht nur Endgeräte, sondern Datenquellen und Datensenken ganz allgemein, also auch einen anderen Rechner.

Bei den Terminals handelt es sich also um

- Steuer-Terminals (Bildschirme mit Tastaturen)
- Drucker
- sonstige Endgeräte wie Plotter, Waagen usw.
- andere SINIX-Rechner
- TRANSDATA-Rechner
- Datenkassen-Systeme bzw. Bedienplätze
- sonstige Rechner, wie CP/M-Systeme, MS/DOS-Systeme

9.1 Auswahl einer TTY-Schnittstelle

Jede TTY-Schnittstelle kann über eine Geräte-Datei im Dateisverzeichnis **/dev** angesprochen werden.
Vorher muß die Geräte-Datei mit dem Kommando **/etc/mknod** eingerichtet werden, wobei die entsprechenden Parameter wie Majornummer und Minornummer angegeben werden müssen, wodurch die hardwaremäßige Zuordnung der Schnittstelle zu einem Stecker im Anschlußfeld erfolgt.
Wie bei allen anderen Dateien wirken auch hier die entsprechenden Datei-Schutz-Attribute wie Leseschutz, Schreibschutz usw.

Beispiel für das Dateiverzeichnis /dev eines PC-MX2 (ls -l /dev, gekürzt):

```
crw--w--w- 1 holger   6,  0 Nov  5 14:54 console   }
crw--w--w- 1 solf     6,  1 Nov  5 14:55 tty00     }  Control-
crw--w--w- 1 grw      6,  3 Nov  5 14:56 tty01     }  Terminals
crw--w--w- 1 roat     6,  4 Nov  5 14:57 tty02     }

crw-rw-rw- 1 root     3,  0 Jul 30 14:01 tty   prozeß-spez. Contr.-Terminal

crw-rw-rw- 1 root     6,110 Nov  5 11:54 lp9001-D06     Drucker 9001
crw-rw-rw- 1 gast     6,111 Nov  5 12:10 plotter        Plotter
crw-rw-rw- 1 root     1,  2 Nov  5 13:46 null           Biteimer

brw-rw-rw- 1 root     0, 20 Okt 22 10:45 fl0
brw-rw-rw- 1 root     0, 21 Okt 22 10:45 fl1
brw-rw-rw- 2 root     0, 22 Nov  5 11:30 fl2
|  |       | |        |  |  |__________| |
|  |       | |        |  |       |       |__ Dateiname
|  |       | |        |  |       |
|  |       | |        |  |       |__ Datum und Uhrzeit der Erstellung
|  |       | |        |  |
|  |       | |        |  |__ Minornummer  2)
|  |       | |        |
|  |       | |        |__ Majornummer  1)
|  |       | |
|  |       | |__ Benutzerkennung des Eigentümers
|  |       |
|  |       |__ Anzahl der Verweise
|  |
|  |__ Datei-Attribute
|
|__ Character-/Block-Device
```

Geräteadressen (Major- und Minor-Nummern)

1) Die Majornummer definiert den Flachbaugruppentyp

```
3   PC-X/PC-X10-Systemboard oder

6   E/A-Prozessor SERAD oder
    E/A-Prozessor SERAG
```

2) Die Minornummer definiert den Anschluß, ein OFFSET ermöglicht evtl. die volle Parametrisierbarkeit der Schnittstelle:

PC-X/PC-X10-Systemboard (alle Schnittstellen voll parametrisierbar)

```
131   Stecker-1  (RS232)
  1   Stecker-2  (V.11)     vorgesehen für Drucker
130   Stecker-3  (RS232)
  0   Stecker-4  (V.11)
```

E/A-Prozessor SERAD und E/AProzessor SERAG:

Die Minornummer ist dezimal-3-stellig:

```
H Z E   (Hunderter-, Zehner-, Einer-Stelle)
|  |  |
|  |  0 ... 5  definiert die Anschluß-Schnittstelle
|  |
|  0 :  erster  E/A-Prozessor
|  1 :  zweiter E/A-Prozessor
|
0 :  Übertragungsrate und Zeichenrahmen nicht änderbar *)
1 :  alle Parameter einstellbar
```

Die Minor-Nummern werden von 0 bis 5 durchgezählt. Bei Druckern oder Fremdterminals muß der Offset 100 zur normalen Minor-Nummenr addiert werden.
Die Gerätedateien für die Drucker D01 bis D05 erhalten daher die Minor-Nummern 101 bis 105.
Für Standardterminals ist die Übertragungs-Geschwindigkeit auf 38400 Baud festgelegt, während die Einstellung der Kanäle mit einer 'Hunderter-Nummer' verändert werden kann.
Die Standardeinstellung ist hier 9600 Baud.

Geräteadressen des PC-X/PC-X10

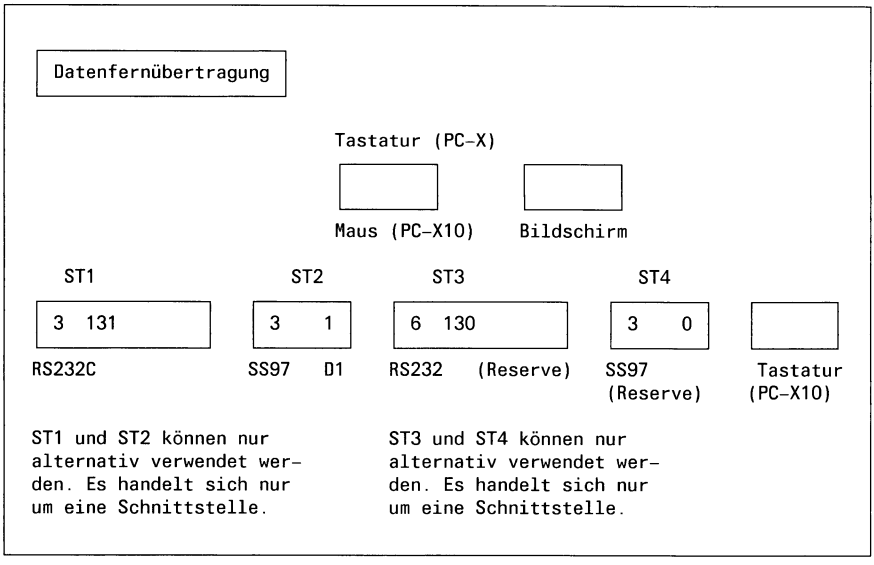


Bild 9-1 PC-X/PC-X10 Geräteadressen

Geräteadressen des PC-MX2

Die angegebenen Minor-Nummern gelten für Drucker oder Fremdgeräte. Bei Standardterminals gelten nur die letzten beiden Stellen.

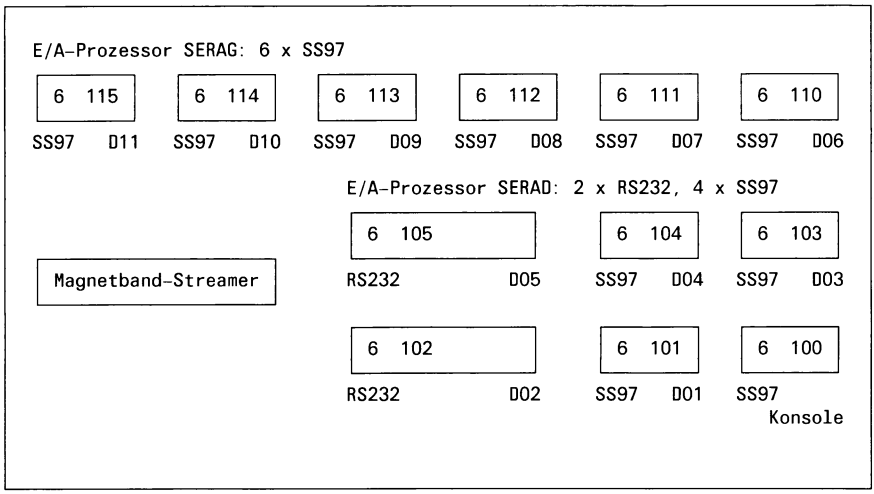


Bild 9-2 PC-MX2 Geräteadressen

Geräteadressen des PC-2000

Die angegebenen Minor-Nummern gelten für Drucker oder Fremdgeräte. Bei Standardterminals gelten nur die letzten beiden Stellen.

Für jedes Terminal werden neben der üblichen Gerätedatei novh bis zu sechs weitere Gerätedateien (virtuelle Terminals, Sessions) erzeugt.
Die Minor-Nummern dieser Gerätedateien (tty??-?) für virtuelle Terminals werden noch folgender Regel erzeugt:

tty??-n = tty?? + n * 6 (n = 1,2,3,4,5,6)

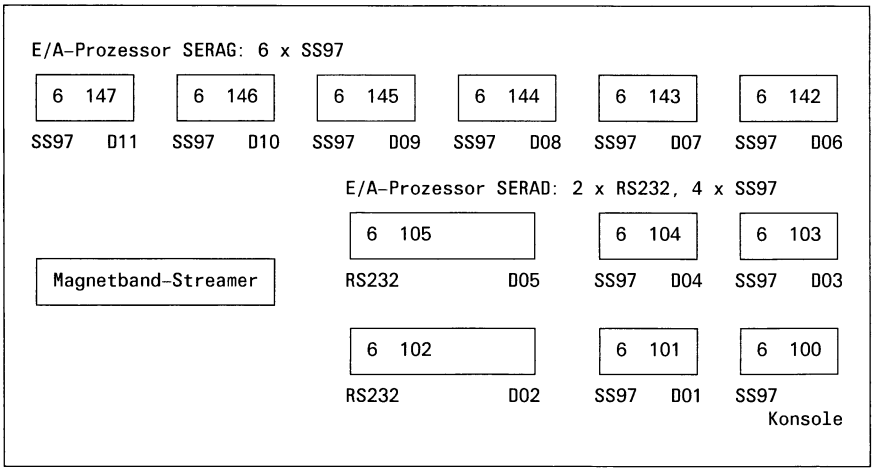


Bild 9-3 PC-2000 Geräteadressen

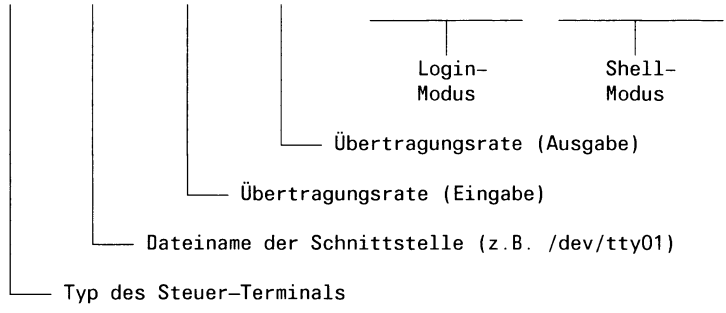
9.2 Die Anpassung der TTY-Schnittstelle

9.2.1 Anpassung beim System-Start-Up

Beim System-Start-Up wird die Datei **/etc/ttytype** ausgewertet.
In dieser Datei sind die für die Steuer-Terminals erforderlichen Parameter hinterlegt.

Der Inhalt der Datei sieht bei einem Vierplatzsystem wie folgt aus:

```
(cat /etc/ttytype)

97801  console 38400   38400   PO;RW;NL1;CR1   PO;EC;CM;FO;TO
97801  tty00   38400   38400   PO;RW;NL1;CR1   PO;EC;CM;FO;TO
97801  tty01   38400   38400   PO;RW;NL1;CR1   PO;EC;CM;FO;TO
97801  tty02   38400   38400   PO;RW;NL1;CR1   PO;EC;CM;FO;TO
```



PO	ungerades Paritätsbit	CM	CRMOD-Modus
RW	RAW-Modus	FO	keine Wartezeit nach Form Feed
NL1	Wartezeit nach New Line	TO	keine Wartezeit nach Tabulator
CR1	Wartezeit nach Carriage Return	TD	TANDEM-Modus
EC	ECHO-Modus	CB	CBREAK-Modus

9.2.2 Anpassung durch das Kommando stty

Das Kommando ohne Parameter zeigt die wichtigsten Eigenschaften der eigenen Schnittstelle an:

Beispiel:

`stty` (Ausgabe auf **stderr**)

```
Übertragungsgeschwindigkeit 38400 bit/s
erase = 'H'; kill = 'X'
odd -nl echo
```

Durch Umleiten von **stdout** auf den entsprechenden Dateinamen können die Einstellungen einer anderen Schnittstelle angezeigt bzw. verändert werden:

Beispiel:

`stty > /dev/lp9001-003` (Anzeige Druckerparameter)

```
Übertragungsgeschwindigkeit 9600 bit/s
erase = 'H'; kill = 'X'
odd -nl -tabs cbreak
```

Durch Angabe von entsprechenden Schaltern kann die Einstellung einer Schnittstelle verändert werden:

Beispiel:

```
          stty -echo cbreak
Kontrolle:
          stty

          Übertragungsgeschwindigkeit 38400 bit/s
          erase = 'H'; kill = 'X'
          odd -nl cbreak
```

9.2.3 Anpassung durch den System-Aufruf ioctl

Dies ist die umfassendste Möglichkeit der Parametrisierung einer TTY-Schnittstelle und soll im folgenden beschrieben werden.

Es wird darauf hingewiesen, daß diese Schnittstelle sich in letzter Zeit in Weiterentwicklung befunden hat und teilweise noch befindet. Je nach eingesetzter SINIX-Version ist es daher möglich, daß einige der beschriebenen Funktionen noch nicht oder nur eingeschränkt funktionieren. Dies betrifft vor allem die Datenfluß-Steuerung in Eingaberichtung. Hier können unter ungünstigen Bedingungen Zeichen verlorengehen.
Weiter werden aus Kompatibilitätsgründen 2 Möglichkeiten des **ioctl**-Systemaufrufes zur Verfügung gestellt, die sich durch unterschiedliche Header-Dateien mit unterschiedlichen Strukturen unterscheiden.
Die 'alte' Schnittstelle entspricht XENIX-V7 und ist in allen SINIX-Versionen enthalten.
Die 'neue' Schnittstelle entspricht UNIX-System-III und wird ab SINIX V2.0 unterstützt.

9.3 TTY-Schnittstelle entsprechend XENIX-V7

Die im Dateiverzeichnis **/dev** eingetragenen **Geräte-Dateien** müssen wie normale Dateien geöffnet und geschlossen werden und auch das Lesen und Schreiben erfolgt wie bei normalen Dateien durch die Aufrufe **read**, **write** oder **getc**, **putc**, **fprintf** usw.

In der Praxis werden die Gerätedateien der Control-Terminals (Bildschirme) nicht durch Anwenderprogramme geöffnet, dies übernimmt das System entsprechend den Dateien **/etc/ttys** und **/etc/ttytype**.

Das erste in einem Prozeß geöffnete Terminal wird das Control-Terminal dieses Prozesses.
Das Control-Terminal spielt eine besondere Rolle durch Behandlung der **QUIT** und **INTR**- Signale.
Das Control-Terminal wird bei einer **fork**-Operation an den Kind-Prozeß weitervererbt, selbst wenn das Control-Terminal geschlossen ist.
Alle Prozesse, die auf diese Weise ein Control-Terminal teilen, werden eine **Prozeß-Gruppe** genannt.
Alle Mitglieder einer Prozeßgruppe erhalten bestimmte Signale gemeinsam, wie zum Beispiel **INTR** und **KILL**.
Die Datei **/dev/tty** ist in jedem Prozeß ein Synonym für das diesem Prozeß zugeordnete Control-Terminal. über diese Gerätedatei kann ein Programm oder ein Shell-Script eine Nachricht auf das Control-Terminal schicken, unabhängig davon, ob **stdout** bzw. **stderr** umgeleitet wurden.
Weiterhin kann diese Datei verwendet werden, wenn Ausgaben auf das Control-Terminal erfolgen sollen, und es zu mühsam ist, das gerade verwendete Terminal zu ermitteln.

Wenn ein Prozeß schneller Zeichen produziert, als über die TTY-Schnittstelle ausgegeben werden können, wird dieser Prozeß gestoppt, sobald die Ausgabe-Warteschlange ein bestimmtes Limit erreicht.

Der Prozeß wird wieder fortgesetzt, sobald die Ausgabe-Warteschlange entsprechend geleert wurde.
Das EOT-Zeichen wird im COOKED-Modus nicht übertragen.

Normalerweise arbeiten die Control-Terminals im Voll-Duplex-Modus. Eingaben können zu jeder Zeit erfolgen, auch während eine Ausgabe läuft. Zeichen gehen nur verloren, wenn der System-Zeichen-Eingabepuffer überläuft, was sehr selten ist, oder wenn der Anwender die maximal mögliche Anzahl von Zeichen aufgesammelt hat und diese nicht rechtzeitig durch das Programm gelesen werden. Wenn dieses Limit von zur Zeit 256

Zeichen erreicht wird, werden alle Zeichen ohne Meldung verworfen.
Es kann jedoch, falls größere Datenblöcke gelesen werden sollen, eine Eingabe-Daten-Flußsteuerung eingeschaltet werden (TANDEM-Modus).

Wenn ein Prozeß Zeichen von einer Gerätedatei lesen will, so wird der Prozeß gestoppt, bis mindestens ein Zeichen zur Verfügung steht.
Wenn ein Hänge-Zustand vermieden werden soll, kann mit dem Kommando **rdchk (fd)** geprüft werden, ob ein Zeichen zur Verfügung steht. Dies ist jedoch nur dann sinnvoll, wenn zeichenweise im CBREAK-Modus gelesen wird. Eine andere Möglichkeit, einen Hänge-Zustand zu vermeiden, ist die Möglichkeit, sich nach einer definierten Zeit wecken zu lassen. Dies erfolgt durch den Aufruf **alarm (sec)** über das Signal **SIGALRM**, das dann entsprechend abgefangen und bearbeitet werden muß.

Normalerweise werden Terminal-Eingaben zeilenweise verarbeitet (COOKED-Modus). Das bedeutet, daß ein Programm bei einer Lese-Operation wartet, bis eine ganze Zeile eingegeben wurde.
Das Zeilen-Ende-Kriterium ist normalerweise das Zeichen NL und auch EOT. Es kann jedoch zusätzlich auch das Zeichen CR verwendet werden (CRMOD-Modus). Weiterhin kann noch ein beliebiges Zeichen als **Eingabe-Begrenzungs-Zeichen** mit derselben Wirkung definiert werden, wenn die Eingabezeilen zu lang sind (Struktur **tchars**, Zeichen: **t_brkc**).

Es wird auch unabhängig von der Anzahl der angeforderten Zeichen maximal eine Zeile übergeben. Es ist aber nicht notwendig, eine ganze Zeile auf einmal zu lesen; dies kann auch zeichenweise erfolgen, ohne daß Information verloren geht.
Es gibt auch andere Betriebsmodi, wobei jedes Zeichen sofort nach dem Eintreffen an das Anwenderprogramm übergeben wird (CBREAK-Modus, RAW-Modus).

Wenn von einem Anwenderprogramm Zeichen an ein Terminal gesendet werden, so werden diese sofort geschrieben, wenn die Ausgabe von vorher gesendeten Zeichen beendet ist.
Eingegebene Zeichen werden wieder ausgegeben, indem sie in die AusgabeWarteschlange geschrieben werden, sobald sie eintreffen.
Dieser ECHO-Modus kann ausgeschaltet werden.

Während der Eingabe werden ERASE- und KILL-Funktionen normal durchgeführt. Folgende ASCII-Steuerzeichen werden standardmäßig dafür verwendet, können aber geändert werden (Struktur **sgttyb**).
Diese Zeichen werden im COOKED-Modus nicht an das Anwenderprogramm übergeben.

CTRL H Das ERASE-Zeichen löscht das letzte eingegebene Zeichen, aber nicht über den Zeilenanfang hinaus.
Bei den Siemens-PC entspricht das Standard-ERASE-Zeichen der Taste ⌫.

CTRL X Das KILL-Zeichen löscht die aktuelle Eingabezeile.

Das ERASE- und das KILL-Zeichen können jedoch durchgereicht werden, indem ihnen das Backslash-Zeichen | vorangestellt wird; das Backslash-Zeichen wird in diesem Fall nicht gelesen und dient nur dem Zweck, die Funktion des ERASE- bzw. KILL-Zeichens aufzuheben.
Weiterhin haben folgende ASCII-Steuerzeichen eine besondere Bedeutung:

- Im COOKED-Modus wird nur das NL an ein Anwenderprogramm übergeben.
- Im CBREAK-Modus werden DEL, FS, DC3 und DC1 nicht übergeben.
- Im RAW-Modus werden alle Zeichen übergeben.

Auch diese Zeichen können geändert werden (Struktur **tchars**).

CTRL D Das EOT-Zeichen wird verwendet, um das Datei-Ende von einem Terminal zu bewirken. In diesem Fall werden alle wartenden Zeichen sofort an das Anwenderprogramm übergeben, ohne auf das Zeilenende zu warten. Das EOT-Zeichen selbst wird nicht übertragen.
Wenn keine Zeichen auf einen Lesebefehl warten, das heißt, wenn das EOT-Zeichen am Zeilenanfang eingegeben wurde, werden keine (NULL) Zeichen übergeben, was der Standard-Dateiende-Indikator ist. Das EOT-Zeichen wirkt also nur am Zeilenanfang.
Bei den Siemens-PC entspricht das Standard-EOT-Zeichen der Taste END.

0x7f Das DEL-Zeichen erzeugt das INTR - Signal, welches an alle Prozesse mit diesem Control-Terminal gesendet wird.
Normalerweise werden alle diese Prozesse beendet, wenn nicht durch geeignete Programmierung entsprechende Vorsorge getroffen wurde. So kann das Signal zum Beispiel ignoriert werden oder einer Behandlung zugeführt werden.
Bei den Siemens-PC entspricht das Standard-DEL-Zeichen der Taste DEL.

CTRL | Das FS-Zeichen erzeugt das QUIT-Signal, welches an alle
CTRL \ Prozesse mit diesem Control-Terminal gesendet wird. Normalerweise werden alle diese Prozesse beendet und ein **core**-Dump erzeugt, wenn nicht durch geeignete Programmierung entsprechende Vorsorge getroffen wurde.
So kann das Signal zum Beispiel ignoriert werden oder einer speziellen Behandlung zugeführt werden.

CTRl S Das DC3-Zeichen (XOFF) stoppt die weitere Ausgabe an dem Terminal. Wenn die Ausgabe gestoppt ist, werden DC3-Zeichen ignoriert.

CTRL Q Das DC1-Zeichen (XON) startet eine evtl. gestoppte Ausgabe wieder. Wenn die Ausgabe nicht gestoppt ist, werden DC1-Zeichen ignoriert.

CTRL J Das Newline-Zeichen (NL) ist das normale Zeilen-Ende-Zeichen. Das NL-Zeichen kann nicht geändert oder verhindert werden.

Der Aufruf **ioctl** verwendet folgende Strukturen, die in **sgtty.h** definiert wurden:

```
struct  sgttyb  {
                char    sg_ispeed;      /* Eingabe-Datenrate */
                char    sg_ospeed;      /* Ausgabe-Datenrate */
                char    sg_ospeed;      /* ERASE-Zeichen */
                char    sg_kill;        /* KILL-Zeichen */
                char    sg_flags;       /* Betriebsart */
                };

struct  tchars  {
                char    t_intrc;        /* INTR-Zeichen */
                char    t_quitc;        /* QUIT-Zeichen */
                char    t_startc;       /* START-Zeichen */
                char    t_stopc;        /* STOP-Zeichen */
                char    t_eofc;         /* EOF-Zeichen */
                char    t_brkc;         /* Eingabe-Begrenzungs-Zeichen */
                };
```

Wenn ein Zeichen den Code **-1** aufweist, ist die entsprechende Funktion nicht wirksam.

Dies trifft standardmäßig für das Eingabe-Begrenzungs-Zeichen zu. Damit kann ein Zeichen definiert werden, das wie das Zeilenende-Zeichen NL bzw. CR wirkt, also die Übergabe an das Anwenderprogramm im COOKED-Modus auslöst.

9.3.1 Beschreibung des System-Aufrufes ioctl

```
#include <sgtty.h>

ioctl (fildes, code, argp)
```

fildes ist ein Filedescriptor, den ein **open**-Aufruf liefert.

code definiert die auszuführende Operation. Entsprechende symbolische Werte werden in **sgtty.h** definiert und später beschrieben.

argp verweist auf eine in **sgtty.h** definierte Struktur:

```
struct sgttyb *argp;
```

Vereinfachte, eingeschränkte Formen sind:

```
stty (fildes, argp);  entspricht  ioctl (fildes, TIOCSETP, argp);
gtty (fildes, argp);  entspricht  ioctl (fildes, TIOCGETP, argp);
```

Zwei Sonderformen des ioctl-Aufrufes sind:

```
ioctl (fildes, FIOCLEX, NULL);
```

Die Datei, beschrieben durch **fildes**, wird automatisch geschlossen während einer erfolgreichen **exec**-Operation.

```
ioctl (fildes, FIONCLEX, NULL);
```

Die Datei beschrieben, durch **fildes**, bleibt geöffnet während einer erfolgreichen **exec**-Operation.

Ein erfolgreicher **ioctl** liefert den Wert 0, im Fehlerfall -1.

9.3.2 Schnittstellen-Parameter

Das Setzen der entsprechenden Schnittstellen-Parameter erfolgt durch entsprechende **#define-Symbole**, die in **sgtty.h** definiert sind.

Für **sg_ispeed** und **sg_ospeed**:

B0	Verbindung abbauen	(nicht implementiert)
B50	50 bit/s	
B75	75 bit/s	
B110	110 bit/s	
B134	134,5 bit/s	
B150	150 bit/s	
B200	200 bit/s	(nicht implementiert)
B300	300 bit/s	
B600	600 bit/s	
B1200	1200 bit/s	
B2400	2400 bit/s	
B4800	4800 bit/s	
B9600	9600 bit/s	
EXTA	19200 bit/s	
B19200	19200 bit/s	
EXTB	38400 bit/s	
B38400	38400 bit/s	

sg_erase ist mit CTRL H vorbelegt.
sg_kill ist mit CTRL X vorbelegt.

In **sg_flags** müssen die einzelnen Parameter bit-gerecht versorgt werden, also mit logisch ODER gesetzt werden.

ALLDELAY	Maske zum Löschen aller Verzögerungsbits
BSDELAY	Maske zum Löschen des BS-Verzögerungsbit
(BS0	Keine Back-Space-Verzögerung)
BS1	Back-Space-Verzögerung (nicht implementiert)
VTDELAY	Maske zum Löschen des VT/FF-Verzögerungsbits
(FF0	Keine Form-Feed-Verzögerung)
FF1	Form-Feed-Verzögerung (2 Sekunden)
CRDELAY	Maske zum Löschen der CR-Verzögerungsbits

(CR0	Keine Carriage-Return-Verzögerung)
CR1	Carriage-Return-Verzögerung-1 (0,08 Sekunden)
CR2	Carriage-Return-Verzögerung-2 (0,16 Sekunden)
CR3	Carriage-Return-Verzögerung-3 (nicht implementiert)
TBDELAY	Maske zum Löschen der TAB-Verzögerungsbits
(TAB0	Keine Tabulator-Verzögerung)
TAB1	Tabulator-Verzögerung-1 (variabel)
TAB2	Tabulator-Verzögerung-2 (nicht implementiert)
XTABS	TAB-Zeichen durch entsprechende Anzahl Spaces ersetzen
NLDELAY	Maske zum Löschen der NL-Verzögerungsbits
(NL0	Keine New-line-Verzögerung)
NL1	New-Line-Verzögerung-1 (variabel)
NL2	New-Line-Verzögerung-2 (0,1 Sekunden)
NL3	New-Line-Verzögerung-3 (nicht implementiert)
EVENP	Gerade Zeichenparität
ODDP	Ungerade Zeichenparität
TANDEM	Datenflußsteuerung bei Eingabe durch Senden von DC3/DC1
RAW	RAW-Modus
CBREAK	CBREAK-Modus
ECHO	Eingegebene Zeichen automatisch wieder ausgeben
CRMOD	Eingabe: CR ändern in LF, wenn ECHO, dann CR+LF Ausgabe: CR oder LF gibt CR+LF

LCASE Großbuchstaben ändern in Kleinbuchstaben bei Eingabe. Ein eingegebener Großbuchstabe wird akzeptiert, wenn vorher das Backslash-Zeichen (\) eingegeben wurde, und wenn ein Großbuchstabe ausgegeben werden soll, wird automatisch vorher ein Backslash-Zeichen (\) ausgegeben. Weiter werden folgende Backslash-Folgen akzeptiert bzw. ausgegeben:

«	\'
|	\!
ß	\^
{	\(
}	\)
\	\\

Beispiele:

A	\a
\n	\\n
\N	\\\n

Die Verzögerungs-Bits bewirken eine automatische Sendepause bei bestimmten Steuerzeichen, um Verzögerungen bei mechanischen Geräten auszugleichen.

Zeichen mit falschem Paritätsbit werden ignoriert bzw. als Schmierzeichen durchgereicht.

Wenn das TANDEM-Bit gesetzt ist, führt das System eine Datenflußsteuerung bei der Eingabe durch.
Wenn die Eingabe-Warteschlange ein bestimmtes Limit erreicht, wird das STOP-Zeichen (DC3) ausgegeben.
Wenn die Eingabe-Warteschlange wieder entsprechend geleert ist, wird das START-Zeichen (DC1) ausgegeben.
Voraussetzung ist natürlich, daß die Datenquelle diese Art der Flußsteuerung versteht.

Achtung

Das TANDEM-Bit wirkt bei SINIX-V1.0B nicht !

Der COOKED-Modus

Terminal-Eingaben werden zeilenweise verarbeitet. Das bedeutet, daß ein Programm bei einer Lese-Operation wartet, bis eine ganze Zeile eingegeben wurde.
Es wird auch unabhängig von der Anzahl der angeforderten Zeichen maximal eine Zeile übergeben. Es ist aber nicht notwendig, eine ganze Zeile auf einmal zu lesen; dies kann auch zeichenweise erfolgen, ohne daß Information verloren geht.
Alle oben beschriebenen Steuerzeichen und Funktionen wie ERASE, KILL, EOT, INTR, QUIT, START, STOP, Verzögerungen usw. werden unterstützt.
In Ausgaberichtung wird Datenflußsteuerung durchgeführt.
Es kann ein Zeichenparitätsbit verlangt werden.

Der CBREAK-Modus

Im CBREAK-Modus wird jedes eingegebene Zeichen sofort an das Anwenderprogramm übergeben.
Die Funktionen ERASE, KILL, EOT und \ wirken nicht, sonst werden alle Funktionen wie im COOKED-Modus unterstützt.

Der RAW-Modus

Im RAW-Modus wird jedes eingegebene Zeichen sofort an das Anwenderprogramm übergeben.
Es werden keine Steuerzeichen behandelt, es gibt also keine Funktionen wie ERASE, KILL, EOT, INTR, QUIT, START und STOP.
Weiterhin wirken keine Verzögerungen und keine Paritätsbit-Bewertung.
Normalerweise wird in Ausgaberichtung keine Datenflußsteuerung durchgeführt.

Ausnahme
Wenn **EVENP** eingestellt ist, erfolgt Datenflußsteuerung bei Ausgabe, aber keine Paritätsbitbewertung.

Alle gewünschten Funktionen müssen durch das Anwendungsprogramm durchgeführt werden.

Folgende **ioctl**-Aufrufe stehen zur Verfügung, wobei die entsprechenden Codes durch symbolische Werte angegeben werden, die in **sgtty.h** definiert sind:

```
ioctl (fildes, TIOCGETP, argp)
```

Alle gerade gültigen Schnittstellen-Parameter werden in der Struktur ***argp** vom Typ **sgttyb** gespeichert.

```
ioctl (fildes, TIOCSETP, argp)
```

Die Schnittstelle wird mit den in der Struktur ***argp** vom Typ **sgttyb** festgelegten Parametern parametrisiert.
Vor der Parametrisierung wird gewartet, bis die Ausgabewarteschlange leer ist, außerdem wird die Eingabewarteschlange gelöscht.

```
ioctl (fildes, TIOCSETN, argp)
```

Die Schnittstelle wird mit den in der Struktur ***argp** vom Typ **sgttyb** festgelegten Parametern parametrisiert, ohne auf die Ausgabe- bzw. Eingabe-Warteschlange Rücksicht zu nehmen. Dadurch können unsinnige Eingabezeichen verursacht werden.

```
ioctl (fildes, TIOCEXCL, NULL)
```

Es wird der **exclusive-use**-Modus gesetzt, keine weiteren **open**-Aufrufe werden zugelassen.

```
ioctl (fildes, TIOCNXCL, NULL)
```

Der **exclusive-use**-Modus wird wieder zurückgesetzt.

```
ioctl (fildes, TIOCHPCL, NULL)
```

Wenn die Gerätedatei das letzte Mal geschlossen wird, das heißt von keinem Prozeß mehr angesprochen wird, wird die Verbindung zum Terminal automatisch abgebaut.

```
ioctl (fildes, TIOCFLUSH, NULL)
```

Die Ausgabewarteschlange und die Eingabewarteschlange werden gelöscht.

```
ioctl (fildes, TIOCGETC, argp)
```

Alle gerade gültigen Schnittstellen-Steuerzeichen werden in der Struktur **argp** vom Typ **tchars** gespeichert.

```
ioctl (fildes, TIOCSETC, argp)
```

Die Schnittstelle wird mit den in der Struktur ***argp** vom Typ **tchars** festgelegten Steuerzeichen parametrisiert.
Wenn ein Zeichen den Code **-1** aufweist, ist die entsprechende Funktion nicht verfügbar.

Beispiel

Die Schnittstelle des Control-Terminals soll durch ein Anwenderprogramm in den CBREAK-Modus geschaltet werden.
Die alten Parameter sollen gesichert und anschließend wieder eingestellt werden.

```
#include        <stdio.h>
#include        <sgtty.h>          /* ioctl - Parameter und Struktur */

#define     FDSTDIN     0          /* File-Descriptor der Standard-Eingabe */

main ()

{
struct  sgttyb  ttypar;                     /* TTY-Struktur */
char            oldflags;                   /* zum Sichern der TTY-Flags */
.
.
.
ioctl (FDSTDIN, TIOCGETP, &ttypar);         /* TTY-Parameter lesen */
oldflags = ttypar.sg_flags;                 /* alte Flags sichern */
ttypar.sg_flags &= ßRAW;                    /* praeventiv ruecksetzen */
ttypar.sg_flags |= CBREAK;                  /* CBREAK-Modus setzen */
ioctl (FDSTDIN, TIOCSETP, &ttypar);         /* TTY-Parameter einstellen */
.
.                       /*--------------------*/
.                       /* Anwendungsprogramm */
.                       /*--------------------*/
.
ttypar.sg_flags = oldflags;                 /* alte Flags einstellen */
ioctl (FDSTDIN, TIOCSETP, &ttypar);         /* TTY-Parameter einstellen */
}
```

9.4 TTY-Schnittstelle entsprechend UNIX-System-III

Die im Dateiverzeichnis **/dev** eingetragenen **Geräte-Dateien** müssen wie normale Dateien geöffnet und geschlossen werden und auch das Lesen und Schreiben erfolgt wie bei normalen Dateien durch die Aufrufe **read**, **write** oder **getc**,**putc**, **fprintf** usw.

In der Praxis werden die Gerätedateien der Control-Terminals (Bildschirme) nicht durch Anwenderprogramme geöffnet, dies übernimmt das System entsprechend den Dateien **/etc/ttys** und **/etc/ttytype**.

Das erste in einem Prozeß geöffnete Terminal wird das Control-Terminal dieses Prozesses.
Das Control-Terminal spielt eine besondere Rolle durch Behandlung der **QUIT** und **INTR**- Signale.
Das Control-Terminal wird bei einer **fork**-Operation an den Kind-Prozeß weitervererbt, selbst wenn das Control-Terminal geschlossen ist.
Alle Prozesse, die auf diese Weise ein Control-Terminal teilen, werden eine **Prozeß-Gruppe** genannt.
Alle Mitglieder einer Prozeßgruppe erhalten bestimmte Signale gemeinsam, wie zum Beispiel **INTR** und **KILL**.
Ein Prozeß kann diese Zuordnung durch den Aufruf **setpgrp** ändern.

Die Datei **/dev/tty** ist in jedem Prozeß ein Synonym für das diesem Prozeß zugeordnete Control-Terminal.
Über diese Gerätedatei kann ein Programm oder ein Shell-Script eine Nachricht auf das Control-Terminal schicken, unabhängig davon, ob **stdout** bzw. **stderr** umgeleitet wurden.

Weiterhin kann diese Datei verwendet werden, wenn Ausgaben auf das ControlTerminal erfolgen sollen, und es zu mühsam ist, das gerade verwendete Terminal zu ermitteln.

Wenn ein Prozeß schneller Zeichen produziert, als über die TTY-Schnittstelle ausgegeben werden können, wird dieser Prozeß gestoppt, sobald die AusgabeWarteschlange ein bestimmtes Limit erreicht.
Der Prozeß wird wieder fortgesetzt, sobald die Ausgabe-Warteschlange entsprechend geleert wurde.
Das EOT-Zeichen wird im ICANON-Modus nicht übertragen.

Normalerweise arbeiten die Control-Terminals im Vollduplex-Modus.
Eingaben können zu jeder Zeit erfolgen, auch während eine Ausgabe läuft.
Zeichen gehen nur verloren, wenn der System-Zeichen-Eingabepuffer

überläuft, was sehr selten ist, oder wenn der Anwender die maximal mögliche Anzahl von Zeichen aufgesammelt hat und diese nicht rechtzeitig durch das Programm gelesen werden. Wenn dieses Limit von zur Zeit **MAX_CHAR** Zeichen erreicht wird, werden alle Zeichen ohne Meldung verworfen.
Es kann jedoch, falls größere Datenblöcke gelesen werden sollen, eine Eingabedaten-Flußsteuerung eingeschaltet werden (Bit **IXOFF** in **c_iflag** gesetzt).

Wenn ein Prozeß Zeichen von einer Gerätedatei lesen will, so wird der Prozeß gestoppt, bis mindestens ein Zeichen zur Verfügung steht.
Wenn ein Hänge-Zustand vermieden werden soll, kann mit dem Aufruf **rdchk (fd)** geprüft werden, ob ein Zeichen zur Verfügung steht. Dies ist jedoch nur dann sinnvoll, wenn zeichenweise gelesen wird
(Bit **ICANON** in **c_lflag** nicht gesetzt).
Eine andere Möglichkeit, einen Hänge-Zustand zu vermeiden, ist die Möglichkeit sich nach einer definierten Zeit wecken zu lassen.
Dies erfolgt durch den Aufruf **alarm (sec)** über das Signal **SIGALRM**, das dann entsprechend abgefangen und bearbeitet werden muß.

Normalerweise werden Terminal-Eingaben zeilenweise verarbeitet
(COOKED-Modus, Bit **ICANON** in **c_lflag** gesetzt).
Das bedeutet, daß ein Programm bei einer Lese-Operation wartet, bis eine ganze Zeile eingegeben wurde.
Das Zeilen-Ende-Kriterium ist normalerweise das Zeichen NL und auch EOT. Es kann jedoch zusätzlich auch das Zeichen CR verwendet werden (Bit **ICRNL** in **c_iflag** gesetzt). Weiterhin kann noch ein beliebiges Zeichen als **Eingabe-Begrenzungs-Zeichen** mit derselben Wirkung definiert werden, wenn die Eingabezeilen zu lang sind (c_ccs[5] EOL, Standard = -1 = nicht verwendet).
Es wird auch unabhängig von der Anzahl der angeforderten Zeichen maximal eine Zeile übergeben.
Es ist aber nicht notwendig, eine ganze Zeile auf einmal zu lesen; dies kann auch zeichenweise erfolgen, ohne daß Information verloren geht.
Es gibt auch andere Betriebsmodi, wobei jedes Zeichen sofort nach dem Eintreffen an das Anwenderprogramm übergeben wird
(Bit **ICANON** in **c_lflag** nicht gesetzt).

Wenn von einem Anwenderprogramm Zeichen an ein Terminal gesendet werden, so werden diese sofort geschrieben, wenn die Ausgabe von vorher gesendeten Zeichen beendet ist.
Eingegebene Zeichen werden wieder ausgegeben, indem sie in die Ausga-

beWarteschlange geschrieben werden, sobald sie eintreffen. Dieser ECHO-Modus kann ausgeschaltet werden
(Bit **ECHO** in **c_lflag** nicht gesetzt).

Während der Eingabe werden ERASE- und KILL-Funktionen normal durchgeführt.
(Wenn Bit **ICANON** in **c_lflag** gesetzt ist).

Folgende ASCII-Steuerzeichen haben eine besondere Bedeutung und werden nicht an ein Anwenderprogramm übergeben (wenn Bit **ICANON** in **c_lflag** gesetzt ist):

CTRL H Das ERASE-Zeichen löscht das letzte eingegebene Zeichen, aber nicht über den Zeilenanfang hinaus.
Bei den Siemens-PC entspricht das Standard-ERASE-Zeichen der Taste [⌫].

CTRL X Das KILL-Zeichen löscht die aktuelle Eingabezeile.

CTRL D Das EOT-Zeichen wird verwendet, um das Datei-Ende von einem Terminal zu bewirken. In diesem Fall werden alle wartenden Zeichen sofort an das Anwenderprogramm übergeben ohne auf das Zeilenende zu warten. Das EOT-Zeichen selber wird verworfen.
Wenn keine Zeichen auf einen Lesebefehl warten, das heißt, wenn das EOT-Zeichen am Zeilenanfang eingegeben wurde, werden keine (NULL) Zeichen übergeben, was der Standard-Dateiende-Indikator ist.
Das EOT-Zeichen wirkt also nur am Zeilenanfang.
Bei den Siemens-PC entspricht das Standard-EOT-Zeichen der Taste [END].

0x7f Das DEL-Zeichen erzeugt das INTR-Signal, welches an alle Prozesse mit diesem Control-Terminal gesendet wird.
Normalerweise werden alle diese Prozesse beendet, wenn nicht durch geeignete Programmierung entsprechende Vorsorge getroffen wurde.
So kann das Signal zum Beispiel ignoriert werden oder einer speziellen Behandlung zugeführt werden.
Bei den Siemens-PC entspricht das Standard-DEL-Zeichen der Taste [DEL]. Bereits eingegebene Zeichen werden vergessen.

CTRL | Das FS-Zeichen erzeugt das QUIT-Signal, welches an alle Prozesse
CTRL \ mit diesem Control-Terminal gesendet wird.
Normalerweise werden alle diese Prozesse beendet und ein **core**-Dump erzeugt, wenn nicht durch geeignete Programmierung entsprechende Vorsorge getroffen wurde.
So kann das Signal zum Beispiel ignoriert werden oder einer speziellen Behandlung zugeführt werden.

CTRL S Das DC3 (XOFF)-Zeichen stoppt die weitere Ausgabe an dem Terminal. Wenn die Ausgabe gestoppt ist, werden DC3-Zeichen ignoriert.
DC3-Zeichen können nicht geändert und verhindert werden.

CTRL Q Das DC1 (XON)-Zeichen startet eine evtl. gestoppte Ausgabe wieder.
Wenn die Ausgabe nicht gestoppt ist, werden DC1-Zeichen ignoriert.
DC1-Zeichen können nicht geändert und verhindert werden.

CTRL J Das New-Line-Zeichen (NL) ist das normale Zeilen-Ende-Zeichen. Das NL-Zeichen kann nicht geändert oder verhindert werden.

CTRL @ Das NUL-Zeichen kann als ein zusätzliches Zeilen-Ende-Zeichen (EOL) verwendet werden, wird aber normalerweise nicht verwendet.

Die ERASE-, das KILL- und EOF-Zeichen können jedoch eingegeben werden, indem ihnen das Backslash-Zeichen vorangestellt wird; das BackslashZeichen wird in diesem Fall nicht gelesen und dient nur dem Zweck, die Funktion des ERASE-, KILL- bzw. EOF-Zeichens aufzuheben.

Die INTR, QUIT, ERASE, KILL, EOF, EOL und SWTCH-Zeichen können geändert werden, und zwar in der Struktur **termio** durch einen Eintrag im Array **c_ccs[];** an der entsprechenden Stelle:

```
c_ccs[VINTR]    INTR -Zeichen     Standard = 0x7f = Taste [DEL]
c_ccs[VQUIT]    QUIT -Zeichen     Standard = 0x1c = CTRL |
c_ccs[VERASE]   ERASE-Zeichen     Standard = 0x08 = Taste [⌫]
c_ccs[VKILL]    KILL-Zeichen      Standard = 0x18 = CTRL X
c_ccs[VEOF]     EOF-Zeichen       Standard = 0x04 = Taste [END]
c_ccs[VEOL]     EOL-Zeichen       Standard = -1 = nicht verwendet
```

Wenn ein Zeichen den Code **-1** aufweist ist die entsprechende Funktion nicht wirksam.

Der Aufruf **ioctl** verwendet folgende Struktur, die in **termio.h** definiert wurde:

```
struct  termio  {
                unsigned short  c_iflag;        /* Eingabe-Modus */
                unsigned short  c_oflag;        /* Ausgabe-Modus */
                unsigned short  c_cflag;        /* Hardware-Steuerung */
                unsigned short  c_lflag;        /* Terminal-Steuerung */
                char            c_line;         /* z. Zt. immer 0 */
                char            c_ccs[NCC];  /* Steuer-Zeichen */
                }                           /* bzw. Non-canonical-Parameter */
```

Die zu verwendende Header-Datei **sys/termio.h** sieht folgendermaßen aus:

```
(cat /usr/include/sys/termio.h)

/*
 * @(#)termio.h 5.3 of 86/02/24
 *
 * SINIX V2.0 include file
 *
 */

#define NCC     8

/* control characters */

#define VINTR   0           /* Interrupt */
#define VQUIT   1           /* Quit */
#define VERASE  2           /* Erase */
#define VKILL   3           /* Kill */
#define VEOF    4           /* End of file */
#define VEOL    5           /* Additional end of line */

                            /* for non-canonical-mode */
#define VMIN    4           /* Min. input characters to satisfy read */
#define VTIME   5           /* Max. time (.1 seconds) to satisfy read */

#define VCEOF   NCC         /* true EOF char (V7 compatibility) */
#define VCEOL   (NCC + 1)   /* true EOL char */

#define CNUL    0
#define CDEL    0377

/* default control chars */
#define CESC    '\\'
#define CINTR   0177        /* DEL */
#define CQUIT   034         /* FS, cntl | */
#define CERASE  '\010'      /* backsp */
#define CKILL   '\030'      /* cntl x */
#define CEOF    04          /* cntl d */
#define CSTART  021         /* cntl q */
#define CSTOP   023         /* cntl s */
```

```
/* input modes */
#define IGNBRK  0000001     /* Ignore break condition */
#define BRKINT  0000002     /* Signal interrupt on break */
#define IGNPAR  0000004     /* Ignore characters with parity errors */
#define PARMRK  0000010     /* Mark parity errors */
#define INPCK   0000020     /* Enable input parity check */
#define ISTRIP  0000040     /* Strip characters */
#define INLCR   0000100     /* Map NL to CR on input */
#define IGNCR   0000200     /* Ignore CR */
#define ICRNL   0000400     /* Map CR to NL on input */
#define IUCLC   0001000     /* Map upper-case to lower-case on input */
#define IXON    0002000     /* Enable start/stop output control */
#define IXANY   0004000     /* Enable any character to restart output */
#define IXOFF   0010000     /* Enable start/stop input control */

/* output modes */
#define OPOST   0000001    /* Postprocess output */
#define OLCUC   0000002    /* Map lower-case to upper-case on output   */
#define ONLCR   0000004    /* Map NL to CR-NL on output */
#define OCRNL   0000010    /* Map CR to NL on output */
#define ONOCR   0000020    /* No CR output at column 0 */
#define ONLRET  0000040    /* NL performs CR function */
#define OFILL   0000100    /* Use fill characters for delay */
#define OFDEL   0000200    /* Fill is DEL, else is NUL */
#define NLDLY   0000400    /* Select new line delays */
#define NL0     0
#define NL1     0000400
#define CRDLY   0003000    /* Select carriage return delays */
#define CR0     0
#define CR1     0001000
#define CR2     0002000
#define CR3     0003000
#define TABDLY  0014000    /* Select horizontal tab delays */
#define TAB0    0
#define TAB1    0004000
#define TAB2    0010000
#define TAB3    0014000    /* Expand tabs to spaces */
#define BSDLY   0020000    /* Select backspace delays */
#define BS0     0
#define BS1     0020000
#define VTDLY   0040000    /* Select vertical tab delays */
#define VT0     0
#define VT1     0040000
#define FFDLY   0100000    /* Select form feed delays */
#define FF0     0
#define FF1     0100000
```

```
/* control modes */
#define CBAUD   0000017    /* Baud rate */
#define B0      0          /* Hang up line */   (nicht implementiert)
#define B50     0000001    /* 50 baud */
#define B75     0000002    /* 75 baud */
#define B110    0000003    /* 110 baud */
#define B134    0000004    /* 134.5 baud */
#define B150    0000005    /* 150 baud */
#define B200    0000006    /* 200 baud */       (nicht implementiert)
#define B300    0000007    /* 300 baud */
#define B600    0000010    /* 600 baud */
#define B1200   0000011    /* 1200 baud */
#define B1800   0000012    /* 1800 baud */
#define B2400   0000013    /* 2400 baud */
#define B4800   0000014    /* 4800 baud */
#define B9600   0000015    /* 9600 baud */
#define B19200  0000016    /* 19200 baud */
#define B38400  0000017    /* 38400 baud */
#define CSIZE   0000060    /* Character size (excl. parity bit) */
#define CS5     0          /* 5 bits */
#define CS6     0000020    /* 6 bits */
#define CS7     0000040    /* 7 bits */
#define CS8     0000060    /* 8 bits */
#define CSTOPB  0000100    /* Send two stop bits, else send one */
#define CREAD   0000200    /* Enable receiver */
#define PARENB  0000400    /* Parity enable */
#define PARODD  0001000    /* Odd parity, else even */
#define HUPCL   0002000    /* Hang up on last close */
#define CLOCAL  0004000    /* Local line, else dial-up */

/* line discipline 0 modes */
#define ISIG    0000001    /* Enable signals */
#define ICANON  0000002    /* Canonical input */
#define XCASE   0000004    /* Canonical upper/lower presentation */
#define ECHO    0000010    /* Enable echo */
#define ECHOE   0000020    /* Echo erase character as BS-SP-BS */
#define ECHOK   0000040    /* Echo NL after kill character */
#define ECHONL  0000100    /* Echo NL */
#define NOFLSH  0000200    /* Disable flush after interrupt or quit */
#define XCLUDE  0100000    /* *V7* exclusive use */

#define SSPEED  13         /* default speed: 7=300, 13=9600 baud */

/*
 * Ioctl control packet
 */
struct termio {
        unsigned short  c_iflag;        /* input modes */
        unsigned short  c_oflag;        /* output modes */
        unsigned short  c_cflag;        /* control modes */
        unsigned short  c_lflag;        /* line discipline modes */
        char            c_line;         /* line discipline */
        uchar_t         c_cc[NCC];      /* control chars */
};
```

9.4.1 Beschreibung des System-Aufrufes ioctl

```
#include <sys/types.h>
#include <sys/ioctl.h>
#include <sys/termio.h>

ioctl (fildes, code, argp)
```

fildes ist ein Filedescriptor, den uns ein **open**-Aufruf liefert.

code definiert die auszuführende Operation. Entsprechende symbolische Werte werden in **sys/ioctl.h** definiert und später beschrieben.

argp verweist auf eine in **termio.h** definierte Struktur:

```
struct termio *argp;
```

oder auf ein **integer**, siehe Code-Beschreibung:

```
int       argp;
```

Ein erfolgreicher **ioctl** liefert den Wert 0, im Fehlerfall -1.

9.4.2 Schnittstellen-Parameter

Das Setzen der entsprechenden Schnittstellen-Parameter erfolgt durch entsprechende **define-Symbole**, die ebenfalls in **termio.h** definiert sind.

Daten-Eingabe-Steuerung c_iflag

Folgende Flags entsprechen je einem Bit in diesem Feld und werden in **termio.h** definiert:

IGNBRK Ignoriere die Break-Bedingung, das heißt, ein ZeichenrahmenFehler mit allen Bits auf 0 wird ignoriert.

BRKINT Setze das Signal **INTR** bei Erkennen der Break-Bedingung, das heißt, bei einem Zeichenrahmen-Fehler mit allen Bits auf 0. Außerdem werden die Eingabewarteschlange und die AusgabeWarteschlange gelöscht.

IGNPAR Ignoriere Zeichen mit anderen Zeichenrahmen-Fehlern bzw. Paritäts-Fehlern.

PARMRK Markiere Zeichen mit Zeichenrahmenfehlern bzw. Paritätsfehlern. Wenn das fehlerhafte Zeichen nicht ignoriert wird, wird es als 3-Zeichen-Folge in den Eingabepuffer geschrieben: 0x7f, 0, x, wobei **x** das fehlerhafte Zeichen darstellt.
Um Mehrdeutigkeit in dem Fall zu vermeiden, wo der Parameter ISTRIP nicht gesetzt ist, wird ein gültiges Zeichen 0x7f verdoppelt.
Wenn der Parameter PARMRK nicht gesetzt ist, und ein fehlerhaftes Zeichen nicht ignoriert wird, wird es als NULL (0) in den Eingabepuffer geschrieben.

INPCK Ermögliche die Paritätsprüfung der gelesenen Zeichen. Der Parameter hat keinen Einfluß auf die Paritätsbit-Generierung bei den auszugebenden Zeichen.

ISTRIP Setze das Bit-8 bei allen gelesenen gültigen Zeichen auf 0.

INLCR Ändere alle gelesenen NL-Zeichen in CR-Zeichen.

IGNCR Ignoriere alle gelesenen CR-Zeichen.

ICRNL Ändere alle gelesenen CR-Zeichen in NL-Zeichen.

IUCLC Ändere alle gelesenen Großbuchstaben in Kleinbuchstaben. Siehe auch die Parameter ICANON und XCASE in **c lflag**.

IXON Ermögliche Daten-Ausgabe-Flußsteuerung. Wenn ein STOP-Zeichen (DC3 = XOFF) erkannt wird, wird die Ausgabe gestoppt. Die Ausgabe wird wieder gestartet, sobald ein START-Zeichen (DC1 = XON), oder bei gesetztem Parameter **XANY** ein beliebiges weiteres Eingabe-Zeichen erkannt wird.

IXANY Jedes Zeichen startet gestoppte Ausgabe.

IXOFF Ermögliche Daten-Eingabe-Flußsteuerung. Wenn die Eingabe-Warteschlange ein bestimmtes Limit erreicht, wird **DC3 = XOFF** ausgegeben.
Wenn die Eingabe-Warteschlange wieder entsprechend geleert ist, wird **DC1 = XON** ausgegeben.

Daten-Ausgabe-Steuerung c_oflag

Alle Verzögerungen sind abhängig von der Datenrate und der Systembelastung. Folgende Flags entsprechen je einem Bit in diesem Feld und werden in **termio.h** definiert:

OPOST Ausgabe-Zeichen-Behandlung. Wenn das Flag gesetzt ist, erfolgt eine Zeichen-Behandlung entsprechend den folgenden Flags. Wenn das Flag nicht gesetzt ist, werden alle folgenden Flags ignoriert und das Zeichen wird unverändert ausgegeben.

OLCUC Auszugebende Kleinbuchstaben werden in Großbuchstaben umgewandelt. Siehe auch das Flag **IUCLC**.

ONLCR Ein auszugebendes NL wird als CR-NL ausgegeben.

OCRNL Ein auszugebendes CR wird als NL ausgegeben.

ONOCR Ein auszugebendes CR wird an der ersten Stelle einer Zeile, also in Spalte 0, verhindert.

ONLRET Ein auszugebendes NL führt auch alle CR-Funktionen durch. Der Zeichenzähler wird auf 0 gesetzt und evtl. definierte Verzögerungen **CR1**, **CR2** oder **CR3** werden ausgeführt.
Wenn das Flag nicht gesetzt ist, wird nur die NL-Funktion durchgeführt, der Zeichenzähler bleibt also unverändert.

OFILL Wenn das Flag gesetzt ist, wird anstelle von Sende-Verzögerungen eine entsprechende Anzahl von Füllzeichen ausgegeben. Das Füllzeichen ist normalerweise NULL (0); wenn das Flag **OFDEL** gesetzt ist, ist das Füllzeichen DEL (0x7f).

OFDEL Ein evtl. auszugebendes Füllzeichen ist DEL (0x7f). Wenn das Flag nicht gesetzt ist, sind evtl. auszugebende Füllzeichen NULL (0).

NLDLY Maske für das NL-Verzögerungsbit zum Löschen.

NL0 Keine Verzögerung bei auszugebendem NL-Zeichen.

NL1 Verzögerung von 0.10 Sekunden bei auszugebendem NL-Zeichen. Wenn das Flag **ONLRET** gesetzt ist, wird die CR-Verzögerung ausgeführt. Wenn das Flag **OFILL** gesetzt ist werden 2 Füllzeichen gesendet.

CRDLY Maske für die CR-Verzögerungsbits zum Löschen.

CR0	Keine Verzögerung bei auszugebendem CR-Zeichen.
CR1	Spaltenabhängige Verzögerung bei auszugebendem CR-Zeichen. Wenn das Flag **OFILL** gesetzt ist, werden 2 Füllzeichen gesendet.
CR2	Verzögerung von 0.10 Sekunden bei auszugebendem CR-Zeichen. Wenn das Flag **OFILL** gesetzt ist, werden 4 Füllzeichen gesendet.
CR3	Verzögerung von 0.15 Sekunden bei auszugebendem CR-Zeichen.
TABDLY	Maske für die TAB-Verzögerungsbits zum Löschen.
TAB0	Keine Verzögerung bei auszugebendem TAB-Zeichen.
TAB1	Spaltenabhängige Verzögerung bei auszugebendem TAB-Zeichen. Wenn das Flag **OFILL** gesetzt ist, werden 2 Füllzeichen gesendet.
TAB2	Verzögerung von 0.10 Sekunden bei auszugebendem TAB-Zeichen. Wenn das Flag **OFILL** gesetzt ist, werden 2 Füllzeichen gesendet.
TAB3	Anstelle dem TAB-Zeichen wird eine entsprechende Anzahl von Blanks ausgegeben.
BSDLY	Maske für das BS-Verzögerungsbit zum Löschen.
BS0	Keine Verzögerung bei auszugebendem BS-Zeichen.
BS1	Verzögerung von 0.05 Sekunden bei auszugebendem BS-Zeichen. Wenn das Flag **OFILL** gesetzt ist, wird 1 Füllzeichen gesendet.
VTDLY	Maske für das VT-Verzögerungsbit zum Löschen.
V	Keine Verzögerung bei auszugebendem VT-Zeichen.
VT1	Verzögerung von 2 Sekunden bei auszugebendem VT-Zeichen.
FFDLY	Maske für das FF-Verzögerungsbit zum Löschen.
FF0	Keine Verzögerung bei auszugebendem FF-Zeichen.
FF1	Verzögerung von 2 Sekunden bei auszugebendem FF-Zeichen.

Hardware-Steuerung des Terminals c_cflag

Folgende Flags entsprechen je einem Bit oder einer Bit-Kombination in diesem Feld und werden in **termio.h** definiert:

CBAUD	Maske für die Baud-Raten-Bits zum Löschen.	
B0	Verbindung auflösen	(nicht implementiert)
B50	50 bit/s	
B75	75 bit/s	
B110	110 bit/s	
B134	134.5 bit/s	
B150	150 bit/s	
B200	200 bit/s	(nicht implementiert)
B300	300 bit/s	
B600	600 bit/s	
B1200	1200 bit/s	
B1800	1800 bit/s	
B2400	2400 bit/s	
B4800	4800 bit/s	
B9600	9600 bit/s	
B19200	19200 bit/s	
EXTA	19200 bit/s	
B38400	38400 bit/s	
EXTB	38400 bit/s	

CSIZE Maske für CS-Bits zum Löschen

CS5 Zeichenrahmen ist 5 Bits ohne Paritätsbit für Ein/Ausgabe.
CS6 Zeichenrahmen ist 6 Bits ohne Paritätsbit für Ein/Ausgabe.
CS7 Zeichenrahmen ist 7 Bits ohne Paritätsbit für Ein/Ausgabe.
CS8 Zeichenrahmen ist 8 Bits ohne Paritätsbit für Ein/Ausgabe.

CSTOPB Wenn das Flag gesetzt ist, werden 2 Stopbits gesendet, sonst nur eines.

CREAD Wenn das Flag gesetzt ist, ist der Datenempfang freigegeben, sonst können keine Daten empfangen werden.

PARENB Wenn das Flag gesetzt ist, wird die Paritätsbit-Generierung und Prüfung unterstützt. Das Flag **PARODD** definiert die gewünschte Zeichenparität.

PARODD Dieses Flag ist nur bei gesetztem **PARENB**-Flag wirksam. Wenn das Flag gesetzt ist, ist die Zeichenparität ungerade. Wenn das Flag nicht gesetzt ist, ist die Zeichenparität gerade.

HUPCL Wenn das Flag gesetzt ist, wird die Verbindung automatisch abgebaut, indem das DTR-Leitung weggenommen wird, sobald der letzte Prozeß die Gerätedatei schließt (nicht implementiert).

CLOCAL Wenn das Flag gesetzt ist, wird ein lokaler Direktanschluß ohne Modem-Steuersignale angenommen.
Wenn das Flag nicht gesetzt ist, werden die Modem-Steuersignale unterstützt.

LOBLK Wenn das Flag gesetzt ist, wird die Ausgabe anderer Prozesse gestoppt, bis das Flag wieder rückgesetzt wird.
Bei rückgesetztem Flag werden die Ausgaben aller Prozesse gleichzeitig im Multiplex-Modus ausgegeben (nicht implementiert).

Steuerung der Terminal-Funktionen c_lflag

Folgende Flags entsprechen je einem Bit in diesem Feld und werden in **termio.h** definiert:

ISIG Ermögliche das Setzen von Signalen. Alle eingegebenen Zeichen werden auf die Steuerzeichen **INTR = Taste** [DEL] **SWTCH** und **QUIT = CTRL** | überprüft.
Wenn ein entsprechender Code erkannt wird, wird die betreffende Funktion ausgeführt und das Zeichen nicht weitergereicht.
Einzelne Signale können trotz gesetztem Flag
ISIG verhindert werden, indem als auszulösender Zeichencode **-1** in das Array **c_ccs[]** eingetragen wird.

SWTCH ist derzeit nicht implementiert.

ICANON Wenn zusätzlich das Flag ISIG gesetzt ist entspricht das dem **COOKED-Modus**.
Die ERASE- und KILL-Funktionen werden ausgeführt.
Alle Eingaben werden zeilenweise bearbeitet, entsprechend den Steuerzeichen NL, EOT und evtl. EOL
und CR.

Wenn das Flag nicht gesetzt ist, entspricht das je nach den anderen Parametern dem **RAW-Modus** oder dem **CBREAK-Modus**. Eingabezeichen kommen direkt aus der Eingabewarteschlange. Ein Leseauftrag ist erfüllt, wenn entweder **VMIN** Zeichen verfügbar sind oder zwischen dem Empfang von 2 Zeichen die Zeit **VTIME** in Zehntel-Sekunden abläuft.
Die entsprechenden Werte für **VMIN** und
VTIME sind im Array **c_ccs[]** an der entsprechenden Stelle einzutragen; (in diesem Fall werden die anderen Einträge im Array nicht bewertet).

c_ccs [VMIN] = Anzahl der blockweise zu lesenden Zeichen.

c_ccs [VTIME] = Timeout in Zehntel-Sekunden.

Dadurch werden schnelle **Burst**-Lese-Operationen ermöglicht. Durch geeignete Kombination von
VMIN und **VTIME** ergeben sich 3 mögliche Verhaltensweisen:

VMIN > 0 und VTIME > 0:
Der Timer wird gestartet, nachdem das erste Zeichen empfangen wurde (und bei jedem weiteren Zeichen). Der Leseauftrag ist erfüllt, wenn VMIN Zeichen empfangen wurden oder der Timer vorher abläuft
(derzeit nicht implementiert).

VMIN > 0 und VTIME = 0:
Der Leseauftrag ist erfüllt, wenn VMIN Zeichen empfangen wurden.

VMIN = 0 und VTIME = 0:
Der Leseauftrag ist sofort erfüllt. Alle verfügbaren Zeichen bekommt der Auftraggeber
(derzeit nicht implementiert).

XCASE Wenn dieses und das **ICANON**-Flag gesetzt sind, wird ein eingegebener Großbuchstabe akzeptiert, wenn vorher das Backslash-Zeichen (\) eingegeben wurde, und wenn ein Großbuchstabe ausgegeben werden soll, wird automatisch vorher ein Backslash-Zeichen (\) ausgegeben.

Weiterhin werden folgende Backslash-Folgen akzeptiert bzw. ausgegeben:

`	\'
|	\!
~	\^
{	\(
}	\)
\	\\

Beispiele:

A	\a
\n	\\n
\N	\\\n

ECHO Alle empfangenen gültigen Zeichen werden sofort wieder automatisch ausgegeben.

ECHOE Wenn zusätzlich die Flags **ICANON** und **ECHO** gesetzt sind, wird das ERASE-Zeichen (Taste ⌫) als 3-Zeichen-Folge ge-echoed
BS-SP-BS.
Dadurch wird das zu löschende Zeichen auch auf dem Bildschirm gelöscht.
Wenn das Flag **ECHO** nicht gesetzt ist, wird das ERASE-Zeichen (Taste ⌫) als 2-Zeichen-Folge ge-echoed
SP-BS.

ECHOK Wenn zusätzlich das Flag **ICANON** gesetzt ist, wird nach einem erkannten KILL-Zeichen (CTRL X) zusätzlich das Zeichen NL ge-echoed, um das Löschen der Zeile anzuzeigen.

ECHONL Wenn zusätzlich das Flag **ICANON** gesetzt ist, wird das NL-Zeichen in jedem Fall ge-echoed, auch wenn das Flag **ECHO** nicht gesetzt ist.

NOFLSH Wenn dieses Flag gesetzt ist, werden die Eingabe- und die Ausgabewarteschlange nicht gelöscht, sobald ein QUIT- oder INTR-Zeichen erkannt wurde.

9.4.4 ioctl-Aufrufe

Folgende **ioctl**-Aufrufe stehen zur Verfügung, wobei die entsprechenden Codes durch Texte angegeben werden, die in **termio.h** definiert sind:

```
ioctl (fildes, TCGETA, argp)
```

Alle gerade gültigen Schnittstellen-Parameter werden in der Struktur ***argp** vom Typ **termio** gespeichert.

```
ioctl (fildes, TCSETA, argp)
```

Die Schnittstelle wird sofort mit den in der Struktur ***argp** vom Typ **termio** festgelegten Parametern parametrisiert.

```
ioctl (fildes, TCSETAW, argp)
```

Es wird gewartet, bis die Ausgabewarteschlange leer ist, dann wird die Schnittstelle mit den in der Struktur ***argp** vom Typ **termio** festgelegten Parametern parametrisiert.

`ioctl (fildes, TCSETAF, argp)`

Es wird gewartet, bis die Ausgabewarteschlange leer ist, dann wird die Eingabewarteschlange gelöscht und dann die Schnittstelle mit den in der Struktur ***argp** vom Typ **sys/ioctl.h** festgelegten Parametern parametrisiert.

`ioctl (fildes, TCSBRK, 1)`

Es wird gewartet, bis die Ausgabewarteschlange leer ist.

`ioctl (fildes, TCSBRK, 0)` nicht implementiert.

Es wird gewartet, bis die Ausgabewarteschlange leer ist, dann werden 0.25 Sekunden lang 0-Bits gesendet, was einem **break** entspricht.

`ioctl (fildes, TCXONC, 0)`	Stoppe Datenausgabe
`ioctl (fildes, TCXONC, 1)`	Starte gestoppte Datenausgabe wieder
`ioctl (fildes, TCFLSH, 0)`	Lösche die Eingabewarteschlange
`ioctl (fildes, TCFLSH, 1)`	Lösche die Ausgabewarteschlange
`ioctl (fildes, TCFLSH, 2)`	Lösche die Eingabe- und Ausgabewarteschlange

```
Vorsicht:

In der Include-Datei /usr/include/sys/ioctl.h sowohl sind Codes für
die UNIX-System-III-Schnittstelle als auch Codes für die XENIX-V7-
Schnittstelle definiert.
Die Codes der XENIX-V7-Schnittstelle sind teilweise gekennzeichnet.

Der C-Programmierer darf auf keinen Fall XENIX-V7-Codes mit einem
Argument vom Typ 'struct termio *' oder UNIX-System-III-Codes mit
einem Argument vom Typ 'struct sgttyb *' verwenden.

Vermischen beider Schnittstellen ergibt beliebigen Unsinn!
```

10 Spool-System

10.1 Druckerverwaltung der SINIX-Versionen 2.0 und 1.2

Die Druckverwaltung kontrolliert die konfigurierten Drucker laufend. Auch wenn kein Druckauftrag bearbeitet wird, überprüft sie ca. alle 80 Sekunden mit Hilfe des ETX/ACK-Protokolls oder einer Statusabfrage, ob die konfigurierten Drucker wirklich ansprechbar sind. Eine Zustandsabfrage ("lpr -q") liefert daher (fast) immer den wirklich aktuellen Zustand der Drucker. Bei jeder solchen Überprüfung wird der Ausgabekanal zum Drucker gelöscht.

Falls die Zustandsabfragen für einen Drucker immer die Zustände GESTOERT oder POLL ergeben, so ist dieser Drucker vermutlich nicht eingeschaltet, OFF LINE, oder die Verbindung zwischen Rechner und Drucker ist nicht in Ordnung. Der Fehler kann auch in einer falschen Einstellung der Codierschalter des Druckers begründet liegen. Bei den Druckern 9001, 9004 und 9013 muß ungerade Parität und XON/XOFF(DC1/DC3)-Protokoll eingestellt sein. Nachdem der Fehler behoben ist, versetzt die Spoolverwaltung den Drucker innerhalb von ca. 80 Sekunden automatisch in den Zustand BEREIT .

Im Gegensatz zum Spoolsystem der SINIX-Version 1.0C wird ein unterbrochener Druckauftrag später mit der Seite wieder begonnen, bei der er unterbrochen wurde. Treten während eines Druckauftrages Störungen auf, so muß der Druckauftrag anschließend nicht wieder von vorn begonnen werden.

Im neuen Spoolsystem können mehrere Drucker zu Druckergruppen zusammengefaßt werden. Wenn ein Druckauftrag abgegeben wird, kann (über die "lpr"-Schalter "-dru=" oder "-ws=") festgelegt werden, welche Druckergruppe zur Ausgabe dieses Auftrags in Frage kommt. Die Ausgabe erfolgt auf dem ersten freien Drucker aus der angegebenen Gruppe. Jeder Drucker kann zu mehreren Gruppen gehören.

Bei der Konfigurierung des lokalen Systems wird standardmäßig für jeden Drucker eine Gruppe eingerichtet, die genau diesen Drucker enthält. Außerdem wird eine Gruppe eingerichtet, in der alle Drucker vereinigt sind (Gruppe: "ALLE") .

Die Einteilung der konfigurierten Drucker in geeignetere Druckergruppen kann aus dem Standardmenüsystem heraus vorgenommen werden.
(Auswahl: s s d, Definition von Druckergruppen)

Konfiguration von Druckern:

Die Konfiguration von Standarddruckern (z.B. 9001, 9004, 9013) sollte nur über das Menüsystem vorgenommen werden. Jede Konfiguration muß an der Konsole erfolgen, wobei kein anderer Anwender am System arbeiten darf. Nach dem "Einloggen" unter der Benutzerkennung "admin" trifft man die Auswahlen:

```
"s  - Systemverwaltung"
"k  - Konfiguierung des lokalen Systems"  und
"a  - Aktuelle Konfigurierung zeigen" .
```

Die aktuelle Konfigurierung muß nun vollständig notiert werden. Anschließend kann man mit der Konfiguration von Bildschirmarbeitsplätzen und Druckern beginnen (siehe SINIX-Buch 2). Beim PC-MX2 müssen bei jeder Änderung immer alle Arbeitsplätze und Drucker neu konfiguriert werden. (Daher ist die genaue Kenntnis der aktuellen Konfigurierung so wichtig.)
Beim PC-MX4 können einzelne Drucker und Terminals nachträglich hinzugefügt oder umkonfiguriert werden.

10.1.1 Dateien und Dateiverzeichnisse

Folgende Dateien und Dateiverzeichnisse sind für die Druckerverwaltung notwendig:

/usr/spool/tmp/sp.*.*

Ruft der Anwender den "lpr" über Pipe oder mit einem der Schalter "-cp" oder "+co" auf, so kopiert der "lpr" die zu druckende Datei das Dateiverzeichnis /usr/spool/tmp . Diese temporäre Kopie erhält einen eindeutigen Namen, der mit "sp." beginnt.
Nach dem erfolgreichen Ausdruck wird die temporäre Datei wieder gelöscht.
Das Dateiverzeichnis /usr/spool/tmp muß immer existieren und die Zugriffsrechte "drwxrwxrwx" besitzen.

/usr/spool/tmp/at.* (ab SINIX V1.2A)

Ab der SINIX-Version 1.2A werden im Dateiverzeichnis /usr/spool/tmp auch Auftragsdateien für Druckaufträge angelegt (für jeden Auftrag eine Datei).
Die Dateinamen diese Auftragsdateien beginnen mit "at.".

/usr/spool/tmp/st.*

Interne Statusdateien für "alte" Backends. Diese Dateien werden in späteren SINIX-Versionen nicht mehr vorhanden sein.

/usr/spool/APOOL (nur SINIX V2.0)

Auftragspuffer des Spoolsytems der SINIX-Version 2.0A .
Im Anschluß an einen Header von 256 Bytes werden alle notwendigen Informationen für jeden Druckauftrag in einer 256 Bytes großen Struktur in dieser Datei abgelegt. Die Dateigröße ist daher von der Anzahl der Druckaufträge abhängig.
Eine defekte APOOL-Datei kann das ganze Spoolsystem blockieren und muß gelöscht werden.
Die APOOL-Datei wird nicht automatisch zurückgesetzt!
Ein große APOOL-Datei belastet das Spoolsystem, auch wenn alle Druckaufträge erledigt sind.
Es ist daher sinnvoll, von Zeit zu Zeit die APOOL-Datei zu löschen und das Spoolsystem mit /usr/spool/startup neu zu starten.
Falls keine APOOL-Datei existiert, wird beim Start des Spoolsystems automatisch eine leere APOOL-Datei (256-Bytes) erzeugt.

/usr/spool/POOLDAT (ab SINIX V1.2A)

Neuer Auftragspuffer des Spoolsystems ab der V1.2A.
Ab der SINIX-Version 1.2A werden in dieser Datei gespeichert:

– die anliegenden Druckaufträge in Kurzform (nur Auftragsnummern)
in der Reihenfolge des späteren Ausdrucks

– alle dynamischen Informationen des Spoolsystems (Druckerstati, Anzahl gedruckter Seiten, usw.)

Die notwendigen Informationen über die Druckaufträge werden in zusätzlichen Auftragsdateien **/usr/spool/tmp/at.*** gespeichert.

/usr/spool/CONFIG

Textversion der Drucker- und Druckergruppen-Konfi-gurationsdatei.
Im ersten Teil dieser Datei werden für jeden verwalteten Drucker der

Name des Druckers, die Pfadnamen des Backends und der Gerätedatei angegeben.
In der gleichen Zeile müssen jeweils alle Schalter aufgelistet sein, die für diesen Drucker zulässig sind.
Im zweiten Teil werden die Druckergruppen definiert und im dritten Teil die "Druckerverwalter" angegeben.
Die drei Teile der CONFIG-Datei sind jeweils durch eine Leerzeile getrennt (siehe 10.1.5).

/usr/spool/CONFIG.bin

Binäre (übersetzte) Version der CONFIG-Datei.
Diese Datei enthält die aktuelle Druckerkonfiguration in Binärform (einschließlich Druckergruppen).

In der SINIX-Version 2.0A wird sie auch zum Träger von Statusangaben und weiteren dynamischen Informationen über die konfigurierten Drucker. In dieser SINIX-Version kann eine defekte CONFIG.bin die ganze Druckerverwaltung blockieren.
In einem solchen Fall muß die CONFIG.bin gelöscht und die Druckerverwaltung neu gestartet werden.
Der "daemon" ruft dann automatisch das Programm "/usr/spool/digest" auf.

/usr/spool/BTRACE.*
/usr/spool/TRACE

Trace-Dateien zur Fehlersuche. Diese Dateien sind nur vorhanden, wenn die TRACE-Funktionen beim Start der Druckerverwaltung aktiviert wurden (siehe 10.1.9).

/usr/spool/daemtrc

Startprotokoll des "daemon". Sollte sich die Druckerverwaltung mit dem Shellscript
"/usr/spool/startup" nicht starten lassen, findet man in dieser Datei meist eine Fehlermeldung, aus der die Ursache des Fehlers abzulesen ist (siehe 10.1.4).

Folgende Dateien werden zusätzlich vom SINIX-Menüsystem verwaltet:

/usr/spool/gruppen

Diese Datei enthält zeilenweise alle Druckergruppen bzw. "-ws" oder "-dru" Werte, die dem Menüsystem bekannt sind. Druckergruppen, die nicht über das Menüsystem eingerichtet wurden, sollten hier auch eingetragen werden.

/usr/spool/drucker

Diese Datei enthält alle Drucker, die dem Menüsystem bekannt sind.

10.1.2 Programme für die Druckerverwaltung

/bin/lpr

Das Programm "lpr" realisiert die Benutzerschnittstelle zum Spoolsystem. Der Benutzer kann Druckaufträge mit entsprechenden Parametern stellen, modifizieren und löschen und die Lage der Drucker und Druckaufträge abfragen. Benutzer mit besonderen Berechtigungen haben die Möglichkeit, das Verhalten der Drucker direkt zu beeinflussen (siehe 10.1.3). Der "lpr" notiert die Aufträge im Auftragspuffer, wo sie vom "daemon" gelesen und bearbeitet werden.

/usr/spool/startup

Diese Shellprozedur dient zum Starten der Druckerverwaltung. Sie führt zunächst einige Aufräumungsarbeiten durch, setzt die Variable BTRACE und startet den "daemon". Der "daemon" sollte nie direkt aufgerufen werden. Zum Aktivieren der TRACE-Funktion kann man in dieser Prozedur die Variable BTRACE=y setzen und den "daemon" mit dem Schalter "-trace" aufrufen (siehe 10.1.9). /usr/spool/startup wird normalerweise bereits in der "/etc/rc"-Prozedur aufgerufen.

/etc/daemon

Das Programm "daemon" ist die eigentliche Druckerverwaltung (siehe 10.1.4). Der "daemon" verwaltet die Druckerwarteschlangen und sorgt dafür, daß auf jedem Drucker immer nur ein Auftrag gleichzeitig ausgegeben wird. Druckaufträge werden der Reihe nach an das entsprechende "Backend" weitergegeben. Das "Backend" übernimmt die druckerspezifische Ausführung des Druckauftrages und wird dabei vom "daemon" kontrolliert.
Falls für einen Drucker kein Druckauftrag anliegt, wird dieser Drukker trotzdem etwa alle 80 Sekunden vom "Backend" überprüft (ETX-/ACKProtokoll oder Statusabfrage). Das Programm "daemon" wird über die Shellprozedur "/usr/spool/startup" gestartet. Falls mindestens ein Drucker konfiguriert ist, sollte der "daemon" immer aktiv sein. Der "daemon" kann mit dem "lpr"-Befehl "lpr -dg" kontrolliert beendet werden.

/usr/spool/lp9001, /usr/spool/lp9001-b
/usr/spool/lp9004
/usr/spool/lp9013
/usr/spool/lp9022 (ab SINIX V1.2A)
/usr/spool/lp9645 (nur PC-MX4 !)

Diese Programme (die "Backends") sind druckerspezifisch. Sie sorgen dafür, daß die Gerätedatei (der Terminaltreiber) für den Drucker richtig eingestellt wird und setzen den Drucker in den Grundzustand. Sie überprüfen, ob der Drucker richtig arbeitet, werten druckerspezifische "lpr"-Optionen aus und senden die entsprechenden Steuerzeichen zum Drucker. Kopf- und Anhangseiten werden ebenfalls von diesen Programmen erstellt. Für jeden konfigurierten Drucker ist das zugehörige Backend immer aktiv.
Backends sind C-Anwenderprogramme; spezielle Kenntnisse des Systemkerns sind zur Erstellung eines solchen Programms nicht notwendig, jedoch genaue Kenntnisse der Schnittstellen zum "daemon" (siehe 10.1.7).

/usr/spool/interface

Interface-Backend zur Einbindung einfacher eigener Backends (z.B. für Fremddrucker) in die Druckverwaltung (siehe 10.1.7). Das InterfaceBackend übernimmt die Kommunikation mit dem "daemon".

/dev/lp*

Gerätedateien (character-devices) zur Ausgabe auf die konfigurierten Drucker. Diese Gerätedateien werden vom Menüsystem oder vom Anwender mit dem Kommando "/etc/mknod" (siehe SINIX Buch 1) erzeugt. Die richtigen Major- und Minor-Device-Nummern sind in der SINIXFreigabemitteilung für Ihren PC angegeben (siehe auch 10.2.1).

/usr/spool/digest

Falls keine Datei **CONFIG.bin** existiert oder die Datei **CONFIG** verändert wurde, so übersetzt das Programm "digest" die **CONFIG** und erzeugt eine neue **CONFIG.bin** . "digest" wird automatisch beim Start der Druckerverwaltung vom "daemon" aufgerufen.

10.1.3 Die Funktionen des lpr

Das Programm "lpr" realisiert die Benutzerschnittstelle zum Spoolsystem. Der Benutzer kann Druckaufträge mit entsprechenden Parametern stellen, modifizieren und löschen und die Lage der Drucker und Druckaufträge abfragen. Benutzer mit besonderen Berechtigungen haben die Möglichkeit, das Verhalten der Drucker direkt zu beeinflussen.
Der "lpr" wertet die angegebenen Parameter (Schalter und Dateinamen) von links nach rechts aus. Die Reihenfolge der Parameter spielt dabei eine wesentliche Rolle.
Alle Optionen (Schalter) für einen Druckauftrag müssen vor dem Dateinamen angegeben werden. Es ist möglich, mehrere verschiedene Druckaufträge mit einem "lpr"-Befehl abzusetzen. In diesem Fall gilt ein "lpr"-Schalter für alle Dateien, die rechts von ihm stehen. Aus diesem Grund können viele "lpr"-Optionen wieder rückgängig gemacht (ausgeschaltet) werden.

Beispiel:

```
lpr -ws=GRUPPE1 +del DAT1 -del +hd -ws=GRUPPE2 DAT2 -hd DAT3 -pb3 DAT4 -q
```

Der "lpr" notiert die Druckaufträge im Auftragspuffer, wo sie vom "daemon" gelesen und bearbeitet werden.
Bei Statusabfragen ("lpr -q") unter SINIX V2.0A liest der "lpr" die Dateien APOOL und CONFIG.bin.
Ab SINIX V1.2A wertet der "lpr" die Datei POOLDAT und alle Auftragsdateien "/usr/spool/tmp/at.*" aus.

Von den Schaltern, die zusammen mit dem "lpr"-Befehl angegeben werden können (siehe SINIX-Buch 1), werden viele direkt vom lpr oder vom "daemon" erkannt und verarbeitet. Alle anderen werden an das Backend weitergegeben.

"lpr" und "daemon" verarbeiten folgende Parameter:

-ws=, -dru=, -pr=, -tl=, -ap=, -ca, -cp, +co, -co, -nc=, -no, -to=, -q, -qdru, -mp, -id=, -su=, -dd=, -dk=, -du=, -of=, -rr, -tst=, -dg, -ex=, -ld=, -su=, -rm, +del, -del, +v, -v,

Diese Schalter müssen nicht in der CONFIG-Datei angegeben werden.

Alle Parameter, die der "lpr" nicht kennt und die in der CONFIG-Datei für den ausgewählten Drucker erlaubt sind (druckerspez. Optionen), werden an das Backend weitergegeben. Ein druckerspezifischer "lpr"Schalter wird

nur akzeptiert, wenn er für alle Drucker der ausgewählten Druckergruppe zugelassen ist.

Hinweis zum Auftragspuffer /usr/spool/APOOL unter SINIX V2.0A:

Alle "lpr"-Prozesse und der "daemon"-Prozeß greifen auf die APOOL-Datei zu. Um zu verhindern, daß mehrere Prozesse gleichzeitig Veränderungen an dieser Datei vornehmen, muß die Datei vor jeder Veränderung für alle anderen Prozesse gesperrt werden ("locking"). Eigene Anwenderprogramme sollten diese kritische Datei nicht verändern. Falls sehr viele Druckaufträge gestellt und abgearbeitet werden, kann es zu langen Verzögerungen beim "lpr"-Kommando kommen. Die APOOL-Datei ist dann häufig gesperrt und ein neuer "lpr"-Prozess (z.B. "lpr -q") kann nicht zugreifen. Die Druckverwaltung "wehrt" sich gegen neue Aufträge.
Aus diesem Grund ist es unter SINIX V2.0A nicht sinnvoll mehr als ca. 50 Druckaufträge vom Spoolsystem verwalten zu lassen.

Ab SINIX V1.2A für PC-X/PC-X10 wird diese Anzahl erheblich vergrösert.
Für den PC-MX2 wird es eine Korrekturversion geben.

Im neuesten Spoolsystem (ab SINIX V1.2A) wurden anlagenspezifische Grenzen für die Anzahl der Drucker und die Anzahl der Druckaufträge eingeführt, die das Spoolsystem verwalten kann.

Ungefähre Werte:

	Drucker	Druckauträge
PC-X/PC-X10	2	256
PC-MX2	8	ca. 500 (Korrekturversion)
PC-MX2	8	510

Die ANtwortzeiten für das Kommandoe **lpr -q** sind im neuesten Spoolsystem auch bei 256- Druckaufträgen noch akzeptabel.

10.1.4 Die Funktion des "daemon"

Der "daemon" wird über das Shellscript /usr/spool/startup bereits in der Datei /etc/rc gestartet. Er ist das zentrale Programm der Druckerverwaltung. Er sorgt dafür, daß immer nur ein Druckauftrag an einem Drucker ausgegeben wird. Die im Auftragspuffer gespeicherten Aufträge werden gemäß ihrer Priorität und ihrer Auftragszeit abgearbeitet.

Der "daemon" gibt ein Startprotokoll aus, das von der startupProzedur in die Datei /usr/spool/daemtrc umgelenkt wird. Sollte der Start der Druckverwaltung mißlingen, so kann man in dieser Datei nachlesen, welche Aktionen noch erfolgreich ausgeführt wurden.

Beim Start des "daemon" werden folgende Aktionen ausgeführt:

- Ist keine Datei CONFIG.bin vorhanden oder wurde die Datei CONFIG verändert, so wird zunächst das Programm /usr/spool/digest aufgerufen, das eine neue Datei CONFIG.bin erzeugt.
  Falls beim Start der Druckerverwaltung gleich zu Beginn Probleme auftreten, ist es sinnvoll, die Datei CONFIG.bin zu löschen und die CONFIG-Datei auf mögliche Fehler zu untersuchen.

- Gemäß der in der Datei CONFIG.bin angegebenen Druckerkonfiguration
  wird nun für jeden Drucker das entsprechende "Backend" aktiviert. Die Kommunikation zwischen "daemon" und "Backends" geschieht über je zwei Pipes und über das Signal SIGALRM, wobei ein festes Protokoll eingehalten wird (siehe 10.1.7). Sollte kein Drucker konfiguriert sein, beendet sich der "daemon" wieder.

- Falls kein Auftragspuffer existiert, so wird er (/usr/spool/APOOL oder /usr/spool/POOLDAT) automatisch erzeugt.
  Andernfalls wird der vorhandene Auftragspuffer verwendet und nach unerledigten Aufträgen durchsucht. Falls es beim Start des Spoolsystems (nach der Umsetzung der CONFIG-Datei) Probleme gibt, sollte der Auftragspuffer gelöscht werden.

Der "daemon" beendet sich automatisch, wenn die Zahl der zu verwaltenden Drucker auf Null sinkt. Dies ist normalerweise nach einem "lpr -dg"Kommando der Fall.

10.1.5 Die CONFIG - Datei

Die Zeilen der CONFIG-Datei sind oft so lang, daß sie mit dem "ced"Editor nicht bearbeitet werden kann. Mit Hilfe des "ed"-Editors (siehe SINIX Buch 1) kann die Datei immer verändert werden. Es ist jedoch Vorsicht geboten, da eine fehlerhafte CONFIG-Datei die Druckerverwaltung blokkieren und beim "Hochfahren" des SINIX-Systems sogar zum Systemhänger führen kann. Vor jeder Veränderung sollte eine Sicherheitskopie der CONFIG angelegt werden.

Im neuen Spoolsystem kann der Systemverwalter für bestimmte Drucker jeweils einen "Druckerverwalter" in die CONFIG-Datei eintragen. Dieser ist dann berechtigt, "seine" Drucker zu sperren, freizugeben, die Probedruckfunktion aufzurufen und den Schwellwert für die Ausgabepriorität zwischen 0 und 20 festzulegen.

Bei dem folgenden Beispiel handelt es sich um eine spezielle CONFIGDatei, bei der in den ersten Zeilen nicht alle zulässigen Schalter angegeben wurden (die Zeilen sind daher kürzer) .

Beispiel /usr/spool/CONFIG:

```
D03 /usr/spool/lp9013 /dev/lp9013-D03 -zs= -dt -int -pb1 -pb2 -pb3 -pb= -pl= +hd -hd +trl -trl -font= -ab= -bis=
D06 '/usr/spool/lp9004 -hdgrp' /dev/lp9004-D06 -pb2 -ps -pb= -pl= +hd -hd +trl -trl -nff -ab= -bis=
D07 '/usr/spool/lp9001 -pb2 -dt' /dev/lp9001-D07 -zs= -dt -int -zb= -pb1 -pb2 -pb3 +hd -hd +trl -trl -ab= -bis=
DF '/usr/spool/interface -prog=/usr/xaver/backend +cbreak +odd +even -speed=B9600' /dev/lp-fremd -dt -pb2

GRUPPE3 ( D03 ) 'Multifunktionsdrucker 9013'
GRUPPE6 ( D06 ) 'Typenraddrucker 9004'
GRUPPE7 ( D07 ) 'Nadeldrucker 9001'
SCHNELL ( D03 D07 ) 'Alle schnellen Drucker'
FREMD   ( DF  )  'Fremddrucker, ueber Privatbackend angesprochen'
ALLE ( D03 D06 D07 DF ) 'Alle vorhandenen Drucker'

admin ( D03 D06 D07 DF )
franz ( D03 )
xaver ( D07 DF )
nicole ( D06 )
```

Die Leerzeilen sind Trennzeilen zwischen den drei Teilen der CONFIG-Datei. Zusätzliche Leerzeilen dürfen nicht vorkommen.

Die drei Teile der Datei besitzen folgenden Aufbau:

1. Teil Druckerbeschreibungen (Aufbau):

   "Druckername Backendname Gerätedatei ....." "Druckername Bakkendname Gerätedatei ....."

   .
   .
   .

   Backendname und Gerätedatei müssen als absolute Pfadnamen angege-

ben werden! Anstelle der "....." müssen die für diesen Drucker zugelassenen Schalter stehen. Der "lpr" akzeptiert nur druckerspezifische Schalter, die in der CONFIG-Datei angegeben wurden. Falls der Backendname in Apostrophe eingeschlossen wird, kann man innerhalb der Apostrophe noch Standardschalter für dieses Backend eintragen.

Beispiel: `D07 '/usr/spool/lp9001 -pb2 -dt' /dev/lp9001-D07`

Diese Standardschalter werden nun immer automatisch gesetzt.

2. Teil Definition der Druckergruppen (Aufbau):

   "Druckergruppenname (.....) ' Kommentar ' " "Druckergruppenname (.....) ' Kommentar ' "

   .
   .
   .

   Anstelle der "....." müssen die Namen der Drucker angegeben werden, die in dieser Gruppe zusammengefaßt werden sollen.

3. Teil Angabe der "Druckerverwalter" (Aufbau):

   "Benutzername (.....) " "Benutzername (.....) "

   .
   .
   .

   Anstelle der "....." müssen die Namen der Drucker angegeben werden, die der Benutzer verwalten soll.

Nach jeder Änderung der Datei "/usr/spool/CONFIG" muß der Befehl
"lpr -rr"
abgegeben werden, um der Spoolverwaltung die Änderung mitzuteilen. Die Druckverwaltung benötigt einige Zeit, um diesen Befehl auszuführen.

Falls die Spoolverwaltung anschließend nicht mehr aktiv ist, kann das Shellscript "/usr/spool/startup" gestartet werden (Dauer ca. 20 Sekunden). Sollte die Spoolverwaltung danach immer noch nicht aktiv sein oder sollten einige Drucker im Zustand UNBEKANNT verbleiben, so ist die CONFIG-Datei fehlerhaft.

Es ist wichtig, daß alle Backends und Gerätedateien, die in der CONFIG-Datei angegeben sind, auch wirklich existieren.

Die Meldungen, die der "daemon" beim Start abgibt, werden in die Datei /usr/spool/daemtrc umgelenkt und sind bei der Fehlersuche hilfreich.

Eine fehlerhafte CONFIG-Datei muß vor dem Abschalten des Systems unbedingt berichtigt werden.

10.1.6 Aufbau der Datei CONFIG.bin

Die Binärversion der Konfigurationsdatei wird vom Programm "digest" aus der Textversion erzeugt. Unter SINIX V2.0A wird sie auch zum Träger von Statusangaben und weiteren dynamischen Informationen über die konfigurierten Drucker.

Jedem Drucker ist ein Druckerbeschreibungsfeld mit der Struktur "dlist" zugeordnet.
Unter SINIX V2.0A enthält die Struktur "dlist" auch zwei Komponenten für dynamische Daten.

Alle Druckerbeschreibungsfelder sind über die Zeigerkomponente "dnext" miteinander zu einer linearen Liste verkettet. Der Wert "NULL" in dieser Zeigerkomponente kennzeichnet das Ende der Liste.

```
struct dlist {
              char *dnam;              /* Name des Druckers          */
              char *dexec;             /* Aufrufform des Backends    */
              char *dfile;             /* Drucker special-file       */
              char msg[MESLEN];        /* Mitteilung vom backend      (nur SINIX V2.0A) */
              struct dstat dstatus;    /* aktuelle Statusinformation (nur SINIX V2.0A) */
              struct dlist *dnext;     /* naechstes Listenelement    */
             };

/* nur SINIX V2.0 */

struct dstat {
              char cstate;       /* current state (S_RDY, S_RUN,.. )*/
              char admreq;       /* adm.request (du,dd,dk,tst,dg,..)*/
              int  anr;          /* Auftragsnummer (APOOL-Position) */
              int  cmdnr;        /* laufende Kommandonummer         */
              int  ofence;       /* output fence value              */
              int  excode;       /* exit-code des Backends          */
              int  bpid;         /* pid des Backends                */
              int  lprpid;       /* lpr pid fuer Probedruckfunktion */
              int  pip[2];       /* Pipe fids zu  Backend           */
              long wecker;       /* naechstes Weckerlaeuten         */
             };
```

In ähnlicher Weise gibt es für jede Gruppe und für jeden Administrator ein entsprechendes Beschreibungsfeld. Für jede Druckergruppe ist eine Liste von Schaltern (flags) vorhanden, in der die druckerspezifischen "lpr"-Schalter angegeben sind, die für alle Drucker dieser Gruppe erlaubt sind.

Einzelheiten des Aufbaus der Datei CONFIG.bin:

Unter SINIX V2.0A ist die Prozeßnummer des "daemon" in den ersten 4 Bytes der CONFIG.bin abgespeichert ("int"-Format des MX2/MX4).

Die nächsten 4 Bytes enthalten die Anfangsadresse der ersten Druckerbeschreibungsstruktur "dlist" (abgespeichert als "Pointer").

Es folgt die Anfangsadresse des ersten Gruppenbeschreibungsfeldes ("glist") und die Anfangsadresse des ersten Administratorbeschreibungsfeldes ("alist").

Falls keine CONFIG.bin existiert, wird beim Start des "daemon" das Programm "digest" aufgerufen und eine neue CONFIG.bin erzeugt.

Hinweis zu SINIX V2.0A

Viele "lpr"-Prozesse und der "daemon"-Prozeß greifen auf die Datei CONFIG.bin zu. Um zu verhindern, daß mehrere Prozesse gleichzeitig Veränderungen an dieser Datei vornehmen, muß die Datei vor jeder Veränderung für alle anderen Prozesse gesperrt werden ("locking"). Eigene Anwenderprogramme sollten diese kritische Datei nicht verändern. Eine defekte CONFIG.bin kann die Druckerverwaltung blockieren und muß gelöscht werden.

Falls sehr viele Druckaufträge gestellt und abgearbeitet werden, kann es zu langen Verzögerungen beim "lpr"-Kommando kommen. Die CONFIG.bin ist dann häufig gesperrt und ein neuer "lpr"-Prozess (z.B. "lpr -q") kann nicht zugreifen.

Dieses Problem tritt nicht mehr unter SINIX V1.2A auf, da unter SINIX V1.2A keine dynamischen Informationen mehr in der CONFIG.bin abgespeichert werden.

10.1.7 Die Funktion der Backends ("Druckertreiber")

Jedem konfigurierten Drucker ist ein Backend und eine Gerätedatei zugeordnet (siehe CONFIG-Datei). Beim Start der Druckerverwaltung ("daemon") wird für jeden konfigurierten Drucker das zugehörige Backend geladen und bleibt aktiv. Die Standard-Ein/Ausgabe-Kanäle der Bakkends werden bereits vom "daemon" auf die Gerätedatei des betreffenden Druckers umgelenkt.

Die Grundaufgabe eines Backends besteht darin, die Gerätedatei (den tty-Treiber) richtig einzustellen (Parität, Baudrate), alle druckerspezifischen "lpr"-Schalter (z.B. -dt, -zs=, -pb3, -zb=, -pl=) und einige weitere "lpr"-Parameter (z.B. -ab=, -bis=, -pb=, +trl, +hd) auszuwerten, die Druckdatei zu lesen und druckerspezifisch aufbereitet auf die Gerätedatei (Standardausgabe) auszugeben. Zusätzlich sollte ein Backend die Bereitschaft des Druckers überprüfen (ETX/ACK-Protokoll oder Statusabfrage) und alle Rückmeldungen des Druckers (z.B. Statusmeldungen) lesen und auswerten. Die Rückmeldungen des Druckers gelangen in den Standardeingabekanal des Backends.

Die Backends sind die einzigen druckerspezifischen Programme in der gesamten Druckerverwaltung. Da ein Backend Steuerzeichen zum Drucker sendet (z.B. zur Einstellung des Zeichensatzes) und die Rückmeldungen des Druckers verstehen muß, kann es i.a. nur für einen bestimmten Drukkertyp verwendet werden. Eine Ausnahme bildet hier das Interface-Bakkend "/usr/spool/interface", das z.B. in Verbindung mit dem Programm "/bin/cat" ein primitives Universalbackend darstellt (siehe Abschnitt 10.2.3).

Jedes Backend ist über je zwei Kommunikationskanäle (Pipes) mit dem "daemon" verbunden. Über eine dieser Pipes sendet der "daemon" Kommandos (Testaufträge, Druckaufträge) und Texte (z.B. Headerinformationen) zum Backend. Das Backend antwortet über die zweite Pipe mit positiven oder negativen Annahme- oder Ausführungsquittungen, liefert Fortschritts- und Statusberichte und sendet Fehlermeldungen an den "daemon". Um einem Kommando Nachdruck zu verleihen und einem eventuell laufenden Auftrag zu unterbrechen, sendet der "daemaon" im Anschluß an ein Kommando noch das Signal SIGALRM zum Backend. Das Backend bricht dann seine augenblickliche Tatigkeit sofort ab, liest das Kommando aus der Pipe und führt es aus. Die Kommunikation zwischen "daemon" und Backend wird durch ein genau festgelegtes Protokoll geregelt (siehe backprot.h und Abschnitt 10.2.3).

In den SINIX-Versionen V 1.2 und V 2.0 bestehen einige Backends aus zwei Teilen, einem "alten" (abgeleitet aus Version 1.0C) und einem "neuen" Teil, der das neue Spoolsystem unterstützt. Bei jedem Druckauftrag führt das neue Backend einen "fork" aus und ruft das alte Backend auf. Es sind dann zwei Backends aktiv.
Das "alte" Backend wertet die Statusdatei /usr/spool/tmp/st.* aus.

"Alte" private Backends, die für die SINIX-Version 1.0C oder 1.0B entwikkelt wurden, können in leicht veränderter Form mit Hilfe des Interface-Backends (/usr/spool/interface) in die neue Druckerverwaltung eingebunden werden (siehe Abschnitt 10.2.3).

Die Gerätedatei (der tty-Treiber) wird von allen Standard-Backends so eingestellt, daß das Datenflußprotokoll XON/XOFF vom Treiber übernommen wird. Diese beiden Steuerzeichen werden dann nicht an die Bakkends weitergeleitet. Normalerweise wird ein Zeichenrahmen von 7 Bit + Paritätsbit bei ungerader (odd) Parität gewählt. Eine Ausnahme ist z.B. der Grafikmode beim Drucker 9001 (Bit-ImageMode); hier werden 8 Datenbits ohne Paritätsbit übertragen. Es ist wichtig, daß die Drucker 9001, 9004 und 9013 immer auf ungerade Parität eingestellt sind, damit die Rückmeldungen (z.B. XOFF oder ACK) der Drucker mit der richtigen Parität erfolgen (ansonsten bewirken sie einen Paritätsfehler und gehen verloren!).

Der Drucker 9022 wird mit 8 Datenbits ohne Parität betrieben!

10.1.8 Die Funktion des tty-Treibers

Alle Ein- und Ausgaben laufen in SINIX über den Terminaltreiber (tty-Treiber). Der tty-Treiber ist, wie alle Gerätetreiber, ein Teil des Systemkerns. Seine Aufgabe besteht darin, Daten an eine der seriellen Schnittstellen SS97 (V.11) oder RS 232 C (V.24) weiterzugeben und diese Hardware-Schnittstelle einzustellen. Die möglichen Einstellungen sind im Kapitel über die tty-Schnittstelle in diesem Manual beschrieben. Die Bezeichnung "tty-Schnittstelle" sollte nicht mit dem Hardware-Interface gleichen Namens verwechselt werden.

Zum Betrieb eines Druckers sind die Einstellungen "cbreak" und "raw" interessant.

Im cbreak-Mode führt der Treiber ein XON-XOFF-Protokoll (Datenflußsteuerung). Die Parität ist wahlweise auf "odd" (ungerade), "even" (gerade) oder "even odd" (kein Paritätsbit, 8 Datenbits) einstellbar. Falls sich ein Drucker im Zustand "UNBEKANNT" befindet ("lpr -ex=..."), und kein Backend auf die Gerätedatei zugreift, so kann der Drucker mit dem "cat"-Kommando zu Testzwecken direkt angesprochen werden (siehe 10.1.9).

Im raw-Mode werden alle Zeichen (auch XON-XOFF) vom Treiber weitergegeben, und der Benutzer muß selbst für die Datenflußsteuerung sorgen (z.B. mit dem ETX/ACK-Protokoll). Es werden 8 Datenbits übertragen.

10.1.9 Verhalten im Fehlerfall

Falls ein richtig konfigurierter Drucker nicht ansprechbar ist (Zustand GESTOERT oder POLL), so kann man ihn mit "lpr -ex=..." in den Zustand UNBEKANNT versetzen und versuchen, eine Datei mit dem "cat"-Kommando direkt auf die Gerätedatei des Druckers auszugeben.

Sollte beispielsweise der Drucker 9001 als D03 konfiguriert sein, so ist folgende Befehlsfolge empfehlenswert (nur Superuser !):

```
lpr -ex=D03                    # Drucker in Zustand UNBEKANNT versetzen
  (ca. 10 Sekunden warten)

stty cbreak -nl -tabs > /dev/lp9001-D03  # Schnittstelle einstellen

cat DATEI > /dev/lp9001-D03          # Datei direkt auf Schnittstelle
                                     # ausgeben
```

Mit Hilfe des "cat"-Kommandos kann überprüft werden, ob der Drucker richtig angeschlossen ist und ob das E/A-Board bzw. das MFA-Board funktioniert. Es kann getestet werden, ob das Datenflußprotokoll (XON/XOFF bzw. DC1/DC3) korrekt durchgeführt wird.

Wenn sich ein Drucker, der mit dem "cat"-Kommando fehlerfrei ansprechbar ist, immer im Zustand UNBEKANNT befindet und auch mit "lpr -ld=..." nicht aktivierbar ist, so liegt ein Fehler in der betreffenden Zeile der CONFIG-Datei vor (siehe Beschreibung der CONFIGDatei). Solche Fehler treten nur auf, wenn die Konfigurierung nicht über das Menüsystem durchgeführt wurde, oder nachträgliche Veränderungen im System vorgenommen wurden.

Falls die Druckerverwaltung nicht aktiv ist, so kann man versuchen, sie über das Menüsystem (Auswahl s s i: "Spoolsystem initialisieren") zu aktivieren. Es ist sinnvoll, dabei alle alten Aufträge zu löschen. Eine zweite Möglichkeit besteht in der Eingabe der folgenden ShellKommandos (nur Superuser !):

```
cd /usr/spool        # in das richtige Dateiverz. wechseln
rm CONFIG.bin APOOL  # CONFIG.bin und Auftragspuffer löschen (nur SINIX V2.0A)
startup &            # Druckverwaltung starten
sleep 10             # 10 Sekunden warten
```

Wenn die Spoolverwaltung sich nicht auf diese Weise starten läßt, so ist die CONFIG-Datei fehlerhaft (siehe oben).

Sollte das SINIX-System bereits beim "Hochfahren" hängenbleiben, so kann dies an einem Fehler in der Spoolverwaltung liegen. Es gibt die

Möglichkeit, mit geeigneten Mitteln (z.B. spezielle SINIX0-Diskette) trotzdem in das System "einzubrechen" und die "rc"-Datei der Festplatte so zu verändern, daß die Prozedur "/usr/spool/startup" nicht mehr aufgerufen wird. Zu diesem Zweck muß am Anfang der betreffenden Zeile ein Kommentarzeichen "#" eingefügt werden. Vorsicht: Der "ced"-Editor kann dabei nicht ohne weiteres verwendet werden, da die Shellvariablen TERM und TERMCAP nicht richtig gesetzt sind.

Fehlersuche mit Hilfe der TRACE-Funktion:

Zu Testzwecken ist es möglich, für die Druckverwaltung ("daemon") und die Backends eine Trace-Funktion einzuschalten. Alle Zustände der Bakkends und des "daemon" werden dann in den Trace-Dateien BTRACE.??? und TRACE protokolliert. Ein Kenner des Spoolsystems kann aus diesen Dateien Rückschlüsse auf mögliche Fehlerursachen ziehen.

Die Trace-Funktion für "alte" Backends wird eingeschaltet, indem vor dem Start des "daemon" die Variable BTRACE = y gesetzt und exportiert wird. Bei neuen Backends (z.B. lp9022) wird die Trace-Funktion durch den Schalter "-trace" in der CONFIG-Datei aktiviert:

```
D05 '/usr/spool/lp9022 -trace /dev/lp9022-d05 -dt -int
```

Die Trace-Funktion des "daemon" wird mit dem Schalter "-trace" beim Aufruf des "daemon" aktiviert:

```
"/etc/daemon -trace > /usr/spool/daemtrc"
```

Diese Änderungen werden am besten in der Shellprozedur "/usr/spool/startup" vorgenommen.

```
Informationen zur Auswertung der Trace-Datei '/usr/spool/TRACE' der Druckerverwaltung:
```

```
Zustände des Druckers:
( config.h )

S_VOID     UNBEKANNT
S_RDY      BEREIT
S_LOCK     GESPERRT
S_INOP     GESTOERT
S_POLL     POLL
S_START    START AUSGABE
S_RUN      LAEUFT
S_CANC     ABBRUCH
S_WAIT1    WARTET
S_WAIT2    WARTET
S_TEST     PROBEDRUCK 1 (S_START)
S_RTEST    PROBEDRUCK 2 (S_RUN)
```

```
Ereignisse, die im 'daemon' eintreten können:
( daemev.h )

E_AFN    Ausführungsquittung negativ ist angekommen
E_AFP    Ausführungsquittung positiv ist angekommen
E_ANN    Annahmequittung negativ ist angekommen
E_ANP    Annahmequittung positiv ist angekommen
E_BDIE   das 'backend' hat sich beendigt
E_CANC   das 'backend' soll laufende Ausgabe abbrechen
E_DD     -dd Flag in 'lpr' Aufruf war gesetzt
E_DG     der backend eines Druckers soll sich beenden
E_DK     -dk Flag in 'lpr' Aufruf war gesetzt
E_DU     -du Flag in 'lpr' Aufruf war gesetzt
E_FILE   Druckauftrag ist für einen Drucker bereit
E_LD     Ladeanforderung für backend eines Druckers
E_NOP    Nullereignis
E_TEST
E_TIME   der Wecker für diesen 'backend' läutet
E_TXT1   Ein BP_TXT1 Protokollelement ist angekommen
```

Eine Meldung der Form: @v"8s von s skipped, haben=%d, soll=%d" wird von der Routine 'daemev.c' ausgegeben und besagt, daß die vom Backend gemeldete Laufnummer pro Kommandobehandlungszyklus nicht mit der laufenden Kommandonummer des 'daemon' übereinstimmt.

```
Mögliche Aktionen des 'daemon' (daemon.c):

bdie     cleanup nach exit von backend
canc     erzeugt BP_TXT2 (TX2_CANC) Protokollelement
dg       erzeugt BP_KDO  (KDO_DIE)  Protokollelement
err      Fehlermeldung im Diagnoseprotokoll
file     erzeugt BP_KDO  (KDO_OUT)  Protokollelement
kickl    kick 'backend'
ld       lädt einen backend
loesch   löscht Druckauftrag, zum aktuellen Drucker
nop      Nulloperation
poll     erzeugt BP_KDO  (KDO_TST)  Protokollelement
prmeld   verständigt lpr: Probedrucks ist fertig !
rpoll    Auftrag von Drucker lösen, Polling starten
rwakel   Auftrag von Drucker lösen, Wecker aufziehen
rwakeno  Auftrag von Drucker lösen, Wecker stoppen
tst      erzeugt BP_KDO   (KDO_PRO) Protokollelement
txt1l    verarbeitet BP_TXT1        Protokollelement
txt1s    verarbeitet BP_TXT1        Protokollelement
txt1skp  verwirft BP_TXT1 Protokollelement
wakel    Wecker läutet in T_LONG Sekunden              T_LONG    Wartezeit 'lang'
wakeno   Wecker läutet nicht mehr
wakes    Wecker läutet in T_SHORT Sekunden             T_SHORT   Wartezeit 'kurz'
```

```
Informationen zur Auswertung der Trace-Dateien '/usr/spool/BTRACE.*' der Backends

Funktion:           Bedeutung:

bp_opcom            Eröffnen der Kommunikation: 'backend'<—>'daemon'
bp_clcom            Schließen der Kommunikation 'backend'<—>'daemon'
bp_kdo              Uebernahme von Kommandos vom 'daemon'
bp_anp              sendet positive Annahmequittung zum 'daemon'
bp_ann              sendet negative Annahmequittung zum 'daemon'
bp_afp              sendet positive Ausführungsquittung zum 'daemon'
bp_afn              sendet negative Ausführungsquittung zum 'daemon'
bp_hdrq             Anforderung von Headerinformation
bp_rept             Fortschrittsbericht zum 'daemon' senden
bp_stat             Druckerstatusmeldung zum 'daemon' senden
bp_err              Nachricht an einen Benutzer senden (mail)
```

10.2 Betrieb von verschiedenen Druckern am PC-MX2/PC-MX4

Im Prinzip kann jeder Drucker mit einer Schnittstelle SS97 oder RS 232 C (V24), der ein XON/XOFF-(DC1/DC3-) Protokoll zur Datenflußsteuerung führt, verwendet werden. Standardmäßig wird augenblicklich nur der Anschluß der Siemens-Drucker 9001, 9013, 9004 und 9022 unterstützt. Am PC-MX4 ist mit Hilfe zusätzlicher Hardware-Baugruppen der Kettendrukker 9645 anschließbar (dieser Drucker verwendet nicht die Standardschnittstellen). Die Unterstützung des Seitendruckers 9025 ist geplant.

Es ergibt sich oft das Problem, daß ein neuer Drucker mit anderen Steuerzeichen (Escape-Sequenzen) arbeitet, andere Statusmeldungen zurückliefert, eine andere Parität oder Baudrate erfordert als die vier Standard-Drucker. In einem solchen Fall läßt sich dieser Drucker nicht ohne weiteres über die Spool-Druckerverwaltung betreiben (dies gilt auch für neue Siemens-Drucker).

In diesem Kapitel sollen die Anschlußmöglichkeiten für Drucker erläutert werden, die dem SINIX-System noch nicht bekannt sind. Der Programmieraufwand für eine "logische" Unterstützung eines neuen Druckers schwankt zwischen einem Tag und zwei Monaten, je nachdem, für welche Anwendungen der Drucker eingesetzt werden soll.
Bei Grafikanwendungen ist der Programmieraufwand besonders hoch.

10.2.1 Einrichten der Gerätedatei

Eine Gerätedatei wird mit dem /etc/mknod-Kommando eingerichtet. Gerätedateien für Drucker und Terminals sind immer "character-devices".

Der Name einer Gerätedatei spielt dabei eine untergeordnete Rolle, wichtig sind die Angaben der Major- und Minor-Device Nummern. Mit diesen Nummern wird festgelegt, welcher Gerätetreiber und welcher physikalische Anschluß (Stecker) dieser Gerätedatei zugeordnet ist. Die richtigen Nummern können z.B. der SINIX-Freigabemitteilung für den entsprechenden PC entnommen werden.

Falls beispielsweise der Drucker 9001 am PC-MX2 als Drucker D03 konfiguriert wurde, sollte man in dem Dateiverzeichnis "/dev" mit dem Befehl "ll lp*" etwa folgende Gerätedatei finden:

```
crw-rw-r-- 1 root          6,  103 Mar 20 13:30 lp9001-D03
```

Diese Gerätedatei besitzt die Majornummer 6 und die Minornummer 103.

Falls ein Drucker an der unteren RS 232 C Schnittstelle des PC-MX2 betrieben werden soll, so kann die Gerätedatei wie folgt eingerichtet werden:

```
/etc/mknod /dev/lp-fremd c 6 102
```

Eine Gerätedatei für die zweite Schnittstelle an der ersten MFABaugruppe (1.ATX) des PC-MX4 wird folgendermaßen erzeugt:

```
/etc/mknod /dev/lp-fremd c 3 193
```

Diese Gerätedatei am MX4 kann nur sinnvoll verwendet werden, wenn für die angesprochene Schnittstelle kein Terminal und kein weiterer Drucker konfiguriert wurde. Es darf also keine Gerätedatei mit der Major-, Minornummer 3, 1 und keine weitere mit der Major-, Minornummer 3, 193 existieren.

Eine Gerätedatei ist eine "Tür" zum Gerätetreiber im Systemkern, der die Daten an die physikalische Schnittstelle weiterleitet.

10.2.2 Umgehung der Standarddruckverwaltung

Es besteht die Möglichkeit, direkt Daten zu diesem Treiber und damit zu einem angeschlossenen Fremddrucker zu senden. Der Befehl

```
cat DATEINAME > /dev/lp-fremd
```

bewirkt beispielsweise die Ausgabe einer Datei auf dem Drucker, falls der Gerätetreiber richtig eingestellt ist und der Drucker ein XON/XOFFProtokoll führt. Die Gerätedatei "/dev/lp-fremd" muß natürlich die richtigen Zugriffsberechtigungen besitzen.

Achtung:
Eine direkte Ausgabe mit "cat" auf einen Drucker ist nur empfehlenswert, wenn kein anderer Prozeß gleichzeitig auf die Schnittstelle zugreift. Es darf insbesondere nur dann auf die Gerätedatei ausgegeben werden, wenn der Drucker nicht vom Spoolsystem verwaltet wird. Ein Drucker, der dem Spoolsystem bekannt ist, muß zuvor mit

```
lpr -ex=...
```

in den Zustand UNBEKANNT versetzt werden.

Die Einstellung des Treibers kann mit dem Kommando

```
stty > /dev/lp-fremd
```

abgefragt werden. Falls keine Schreibberechtigung für "other" besteht, muß das Kommando vom super-user abgegeben werden.

Es kommt darauf an, daß Baudrate und Parität mit den Einstellungen des Druckers übereinstimmen und der Treiber im "cbreak"-Mode arbeitet.

Bedeutung weiterer empfehlenswerter "stty"-Einstellungen:

```
"-nl"     Umwandlung von LF in CR + LF (vor jedem Zeilenvorschub
          wird auch ein Wagenrücklauf durchgeführt)
"-tabs"   Umwandlung von Tabulatoren in Blanks (dies ist sinnvoll,
          falls am Drucker keine Standardtabulatoren gesetzt sind)
```

Beispiele zur Veränderung der Treiber-Einstellung:

```
stty cbreak odd -nl -tabs > /dev/lp-fremd
  (ungerade Parität)

stty cbreak odd even -nl -tabs > /dev/lp-fremd
  (8 Datenbits, kein Paritätsbit)
```

Mit Hilfe des "cat"-Kommandos kann überprüft werden, ob der Drucker richtig angeschlossen ist und ob das E/A-Board bzw. das MFA-Board funktioniert. Es kann getestet werden, ob das Datenflußprotokoll (XON/XOFF bzw. DC1/DC3) korrekt durchgeführt wird. Eventuelle Rückmeldungen des Druckers kann man mit dem Befehl

```
cat -u /dev/lp-fremd > DATEI &
```

in einer Datei protokollieren. Die Angabe der "-u"-Option ist dabei sehr wichtig !

Anwender, die eine komfortablere Ausgabe auf den Drucker wünschen, können sich zu diesem Zweck eine kleine Shell-Prozedur schreiben, die diese Ausgabe durchführt.

Beispiel:

```
# ------------------------------------------------------------------------------
# Einfaches Shell-Script zum direkten Ausdrucken von Dateien, unter Umgehung der
# Standarddruckverwaltung .
# Als Voraussetzung fuer diese Prozedur muss eine Geraetedatei /dev/lp-fremd
# und ein Dateiverzeichniss /usr/dlpr  mit Schreib- und Leseberechtigungen fuer
# "a" (alle) eingerichtet sein.
# In diesem Beispiel sind die Steuerzeichen fuer die Drucker 9013 und 9001 angegeben.
# ------------------------------------------------------------------------------

while [ -s /usr/dlpr/lock.dat ]                 # Pruefen, ob gerade ein anderer Auftrag
do                                              # bearbeitet wird
    echo "Es wird gerade ein anderer Druckauftrag bearbeitet "
    echo "Bitte warten Sie !"
    sleep 10
done

echo $USER > /usr/dlpr/lock.dat
trap 'echo -n "\014" > /dev/lp-fremd; rm /usr/dlpr/lock.dat; exit' 2 3

# Jetzt wird der Geraetetreiber fuer den Drucker richtig eingestellt.
# Diese Angaben muessen fuer jeden Drucker angepasst werden:
stty 9600 cbreak odd -even -tabs -nl > /dev/lp-fremd
#
echo -n "\033c" > /dev/lp-fremd             # Drucker in den Grundzustand setzen
datei=
for schalter in $*                          # Schalter interpretieren
do                                          # Hier muessen die speziellen Drucker-
                                            # steuerzeichen eingesetzt werden.
    case $schalter in                       # ('\033' steht fuer ESC)
    -dt)  echo -n "\033(K" >/dev/lp-fremd;; # deutscher Zeichensatz
    -int) echo -n "\033(B" >/dev/lp-fremd;; # ASCII Zeichensatz
    -pb2) echo -n "\033[2w">/dev/lp-fremd;; # Schreibschritteinst. 12 Pitch
    -*) ;;                                  # unbekannter Schalter
    *)  datei=$schalter
        cat $datei     >  /dev/lp-fremd     # Datei auf den Drucker ausgeben
        echo -n "\014" >  /dev/lp-fremd ;;  # Formfeed zum Drucker senden
    esac
done
if [ "$datei" = "" ]
then
   cat  >  /dev/lp-fremd               # von Standardeingabe lesen
   echo -n "\014" >  /dev/lp-fremd ;;  # Formfeed zum Drucker senden
fi

rm /usr/dlpr/lock.dat
```

10.2.3 Betrieb eines neuen Druckers über die Standarddruckverwaltung

Um einen neuen Drucker mit dem üblichen "lpr"-Befehl ansprechen zu können, muß man ein individuelles Backend in die Druckverwaltung integrieren. Das Backend bildet eine speziell auf den jeweiligen Drucker angepaßte Verbindung zwischen der allgemeinen Druckverwaltung "lpr" bzw. "daemon" und den allgemeinen Treibern für die Ein/Ausgabe-Boards.

Die interne Struktur des neuen Spoolsystems unterscheidet sich grundlegend von der des Spoolsystems in der Version 1.0C. Die "echten" Backends sind über je zwei "Pipes" mit dem daemon verbunden und die Kommunikation erfolgt gemäß eines genauen festgelegten Protokolls (vgl. backprot.h).

10.2.3.1 Verwendung des Interface-Backends

Um zumindest eine beschränkte Kompatibilität zur Version 1.0C zu erreichen und den Anschluß von Fremddruckern zu erleichtern, wird ein Interface-Backend "/usr/spool/interface" ausgeliefert, das zur Anbindung von einfachen privaten Backend-Programmen dient. Das Interface-Backend übernimmt die Kommunikation mit dem daemon und ruft das private Backend auf, das (ähnlich wie in SINIX-Version 1.0C) die Ausgabe einer Datei auf den Drucker übernehmen muß. Die Standard-Ein/Ausgabe der Backends werden bereits vom daemon auf die Gerätedatei des Druckers umgelenkt.

Diese einfachen privaten Backends müssen daher lediglich eine Datei, deren Name als Parameter übergeben wird, auf ihre Standardausgabe kopieren und eventuell zuvor einige druckspezifische "lpr"-Schalter (z.B. "-pb2" oder "-dt") auswerten. "lpr"-Optionen, die nicht druckspezifisch sind (z.B. "-nc=") werden bereits vom Interface-Backend oder vom "daemon" ausgewertet. Im Gegensatz zur SINIX-Version 1.0C steht den Bakkends keine spezielle Statusdatei zur Verfügung (Vorsicht: Dateidescriptor 3 besitzt nun eine andere Bedeutung).

Eventuelle Rückmeldungen des Druckers können auch von diesen einfachen Backends aus der Standardeingabe gelesen werden.

Die Programme /bin/cat oder /bin/pr können als universelle primitive Backends mit Hilfe des Interface-Backends in die Druckverwaltung integriert werden. Diese primitiven Backends ignorieren allerdings alle druckspezifischen "lpr"-Parameter. Am Ende dieses Abschnitts ist der Quelltext eines weiteren "Universalbackends" abgedruckt.

Die richtige Einstellung des Gerätetreibers (Parität, Baudrate usw.) wird vom Interface-Backend vorgenommen, falls die entsprechenden Parameter in der CONFIG-Datei angegeben sind.

Im Gegensatz zu den "echten" Backends, wird die "Bereitschaft" des Drukkers vom Interface-Backend nicht überprüft. Der Drucker wird immer als "BEREIT" vorausgesetzt und es wird lediglich das Datenflußprotokoll XON/XOFF verwendet.

Das Interface-Backend wird wie folgt in die Datei /usr/spool/CONFIG eingetragen (anstelle des eigenen Backends ist hier /bin/cat angegeben):

```
DF '/usr/spool/interface -prog=/bin/cat +cbreak +odd +crmod -speed=B9600' /dev/lp-fremd
```

Folgende Schalter werden vom Interface-Backend ausgewertet:

-prog=, -odd, +odd, -even, +even, +cbreak, -crmod, +crmod, -speed=

Als Argumente für den Schalter "-speed=" sind u.a. zugelassen:

B110, B150, B200, B300, B600, B1200, B1800, B2400, B4800, B9600

Der Schalter -speed= sollte unbedingt angegeben werden.

Bedeutung der angegebenen Parameter für das Interface-Backend:

```
-prog=PROGRAMMNAME   hier kann der absolute Pfadname des eigenen
                     Backends angegeben werden.
+odd / -odd          ungerade Parität setzen / wegnehmen
+even / -even        gerade Parität setzen / wegnehmen

      Falls kein Paritätsbit gewünscht wird, so sollte "+odd +even"
      (odd und even) gesetzt werden.

+cbreak              für Druckerausgabe sollte immer der CBREAK-
                     Mode verwendet werden.
+crmod / -crmod      im "CRMOD" sorgt der Treiber dafür, daß vor
                     einem Zeilenvorschub auch ein Wagenrücklauf
                     durchgeführt wird.
```

Es ist leider nicht möglich, die Treibereinstellung "-tabs" ueber das Interface-Backend einzustellen. Bei Verwendung des "cat"-kommandos muß beim Systemstart daher noch der Befehl:

stty -tabs > /dev/lp-fremd

eingegeben werden.

Das folgende primitive "Universalbackend" übernimmt selbst die Umsetzung von Tabulatorzeichen und die Umwandlung von LF in CR + LF . Der Anwender muß es wie folgt in die CONFIG-Datei eingetragen:

```
DF '/usr/spool/interface -prog=/usr/xaver/back␣bsp +cbreak +odd -speed=B9600' /dev/lp-fremd
```

```
/* Primitives Universalprogramm, das anstelle des "cat"-Kommandos zur */
/* Ausgabe von Daten auf einen Drucker verwendet werden kann.         */
/* Dieses Programm setzt voraus, dass der Geraetetreiber richtig      */
/* eingestellt ist (CBREAK-Mode mit richtiger Paritaet).              */
/* Vorsicht:  Bei ESC-Folgen, die laenger als 2 Zeichen sind, stimmt  */
/* --------   die Spaltenzaehlung nicht mehr !!                       */
/* Im Anschluß an den Ausdruck wird ein Formularvorschub gemacht.     */

# include <stdio.h>
# include <signal.h>

# define HT '\011'      /* Horizontaltabulator          */
# define LF '\012'      /* Line Feed Zeichen            */
# define FF '\014'      /* Form Feed Zeichen            */
# define CR '\015'      /* Carriage Return Zeichen      */
# define ESC '\033'     /* ESC-Zeichen                  */

int abbruch() ;         /* Verweis auf die Abbruchfunktion */

main(argc,argv)
int argc;
char *argv[];
{
        FILE *fpein ;
        char c ;
        int nspalte = 1 ;
        register short nspalte ;
        char buffer[BUFSIZ] ;

        setbuf(stdout, buffer) ;  /* Puffer von 1KB bereitstellen          */

        signal(SIGALRM, abbruch) ;  /* Falls das Alarmsignal 15 eintrifft,   */
                                    /* wird das Backend kontrolliert beendet */

        /* Datei zum Lesen oeffnen */

        if ( ( fpein = fopen(argv[1] , "r") ) == NULL )
                exit(1) ;       /* Datei konnte nicht geoeffnet werden */

        putchar( CR ) ;                         /* Wagenruecklauf     */

        while ( ( c = getc(fpein) ) != EOF )    /* Datei ausgeben     */
        {
                switch( c )
                {
                case HT: do                     /* Tabulatoren durch  */
                                putchar(' ') ;  /* Blanks expandieren */
                         while ( nspalte++ % 8 ) ;
                         break ;
                case LF: putchar( CR ) ;        /* Wagenruecklauf     */
                case CR: nspalte = 1 ;          /* Spaltenzaehler     */
                         putchar( c ) ;
                         break ;
                case ESC:
                         putchar( c ) ;         /* ESC + 1 Zeichen !  */
                         if ( ( c = getc(fpein) ) != EOF )
                               putchar( c ) ;
                         break ;
                default: if ( c >= ' ' && c <= 'ß' ) /* nur abdruckbare */
                               nspalte++ ;           /* Zeichen zaehlen */
```

```
                        putchar( c ) ;
                        break ;
                }
        }
        putchar( FF ) ;       /* Formularvorschub am Ende des Ausdrucks */
        exit(0);
}

abbruch()  /* kontrollierter Abbruch nach Eintreffen des Alarm-Signals */
{
        clearerr( stdout ) ;
        putchar( FF ) ;
        putchar( CR ) ;
        exit( 64 ) ;
}
```

10.2.3.2 "Echte" eigene Backends

Wenn die Leistungsfähigkeit des neuen Spoolsystems voll ausgenutzt werden soll, so muß auf der Basis des festgelegten Kommunikationsprotokolls zwischen "daemon" und Backend ein neues Backend geschrieben werden. Das neue Backend muß die Kommandos des "daemon" verstehen, ausführen und die entsprechenden Ausführungsquittungen, Fortschrittsmeldungen und Mail-Meldungen zurückliefern.

Zur Kommunikation mit dem "daemon" stehen bereits eine Reihe von kleinen Funktionen zur Verfügung, die in der Datei backprot.o zusammengefaßt sind:

Funktion:	Argum.:	Bedeutung:
bp_anp	-	Sendet positive Annahmequittung zum "daemon"
bp_ann	-	Sendet negative Annahmequittung zum "daemon"
bp_afp	-	Sendet positive Ausführungsquittung zum "daemon"
bp_afn	-	Sendet negative Ausführungsquittung zum "daemon"
bp_rept	4 long, long, int, int	Fortschrittsbericht zum "daemon" senden
bp_stat	int	Druckerstatusmeldung zum "daemon" senden
bp_err	char *	Nachricht an einen Benutzer senden (mail)

Falls die Trace-Funktion eingeschaltet ist wird jede Ausführung einer dieser Funktionen in einer Datei /usr/spool/BTRACE.<pid> protokolliert. Bei neuen Backends wird die Trace-Funktion durch den Schalter '-trace' in der CONFIG-Datei aktiviert.

Jedes Backend besteht aus einer großen Anzahl von Funktionen, die nicht druckerspezifisch sind. Insbesondere sind das Hauptprogramm ("main") und alle Kommunikationsroutinen unabhängig vom Druckertyp. Auf der Diskette, die diesem Schnittstellenhandbuch beigelegt ist, befinden sich alle druckerunabhängigen Moduln als Objektfiles (übersetzte Moduln für PC-X/PC-X10 und PC-MX2/PC-MX4). Außerdem sind alle notwendigen "include"-Dateien auf der Diskette vorhanden.

Im Anschluß an diese Beschreibung sind die druckerspezifischen Unterprogramme eines einfachen Beispielbackends als Quelltexte abgedruckt. Für die einwandfreie Funktion dieses Beispiels wird jedoch keine Garantie übernommen.

In Anlehnung an diese Vorlage kann ein Anwender alle druckerspezifischen Unterprogramme für einen neuen Drucker erstellen. Mit Hilfe der "makefile"-Datei, die sich ebenfalls auf der Diskette befindet, können die C-Routinen übersetzt und zu einem kompletten eigenen Backend zusammengebunden werden.

Es müssen folgende eigene C-Funktionen geschrieben werden:

```
backflg(ac, av)    /* Einstellen von Standardwerten und Auswertung der */
int ac  ;          /* Uebergabeparameter (Schalter)                    */
char *av[] ;       /* Moegliche Returncodes ( vgl. backflg.h ):        */
                   /*       (BF_OK << 8  | i )                         */
                   /*       (BF_PERR << 8 | i )                        */
                   /*       (BF_PVERR << 8 | i )                       */
                   /* Die Variable i gibt dabei die Nummer des         */
                   /* Parameters an.                                   */
                   /* Diese Funktion wird beim Start des Backends und  */
                   /* vor Beginn jedes Druckauftrages aufgerufen.      */

druin()            /* "Drucker initialisieren", d.h. Geraetetreiber    */
                   /* einstellen, Variablen initialisieren             */
                   /* Diese Funktion wird i.a. nur einmal beim Start   */
                   /* des Backends im Anschluss an die Funktion        */
                   /* backflg() aufgerufen.                            */

dstat()            /* Druckerstati anfordern und auswerten. Diese      */
                   /* Routine prueft u.a. ob der Drucker ansprechbar   */
                   /* ist. Die Statusmeldungen des Druckers werden mit */
                   /* Hilfe der Funktion bp_stat( .. ) an den          */
                   /* "daemon" gemeldet ( siehe backmeld.h ).          */
                   /* Diese Funktion wird alle 80 Sekunden bei jedem   */
                   /* Polling und am Ende jeder Druckseite aufgerufen. */

anblock()          /* Die Funktion anblock() ruft die Funktion bp_anp()*/
                   /* auf, ermittelt die Groesse der auszudruckenden   */
                   /* Datei, kopiert ESC-Folgen fuer den Drucker in den*/
                   /* Ausgabepuffer, initialisiert interne Tabellen und*/
                   /* ruft die Funktion block() zum ersten Mal auf.    */
                   /* Diese Funktion wird zu Beginn jedes Druckauf-    */
                   /* trages von der Funktion bopen() aufgerufen.      */

block()            /* Einen Block von BLSIZ Zeichen druckerspezifisch  */
                   /* aufbereiten und auf die Standardausgabe ausgeben */
                   /* Am Ende jeder Seite wird die Funktion dstat()    */
                   /* aufgerufen.                                      */
                   /* ESC-Folgen, die im Text vorkommen sollten von    */
                   /* einem Unterprogramm seqsuch(escf) untersucht     */
                   /* werden ( damit die Spaltenzaehlung stimmt !)     */
```

```
npafn()          /* Aktionen bei Abbruch eines Druckauftrages        */
```

Die externen Variablen, die den druckerspezifischen Routinen zur Verfügung gestellt werden, sind im Beispielquelltext erklärt.

Die Arbeit des Backends wird vom "daemon" überwacht. Falls das Bakkend die Kommandos des "daemon" nicht innerhalb einer bestimmten Zeit erledigen kann, weil es auf einem "read"- oder "write"- Systemaufruf "hängenbleibt", so schickt der "daemon" das Signal SIGALRM zum Bakkend. Der Systemaufruf wird dadurch unterbrochen und eine festgelegte Signalbehandlungsroutine aufgerufen. Die "read" oder "write"-Funktion meldet sich anschließend mit einem Fehlercode (Returncode < 1) zurück.

In allen Unterprogrammen des Backends muß der Returncode eines Systemaufrufs unbedingt abgefragt werden. Im Fehlerfall kann man bei "read"- und "write"-Aufrufen davon ausgehen, daß eine Zeitüberschreitung eingetreten ist. Im druckerspezifischen Teil des Backends sollte keine eigene Signalbehandlung vereinbart werden, da sonst die genormte Signalbehandlung der Druckverwaltung gestört wird.

Das abgedruckte Beispielbackend (lp9012.c) für den Drucker 9012 wird nach der Übersetzung etwa wie folgt in die CONFIG-Datei eingetragen:

```
DO5 '/usr/spool/lp9012' /dev/lp-fremd -dt -int -pb1 -pb2 -pb3 -font= -ab= -bis= -pb= -pl=
```

Falls die Trace-Funktion aktiv sein soll, muß der Eintrag lauten:

```
DO5 '/usr/spool/lp9012 -trace' /dev/lp-fremd -dt -int -pb1 -pb2 -pb3 -font= -ab= -bis= -pb= -pl=
```

Es muß darauf geachtet werden, daß ein neues Backend auch in die Datei '/usr/spool/startup' eingetragen wird:

```
Beispiel für die ersten Zeilen dieser Datei:
# startup          Spoolverwaltung starten
cd /usr/spool
if [ "$1" != rc ]
then
/bin/ps -ef >/tmp/dat1.$$
/bin/fgrep '/etc/daemon
/usr/spool/lp9001
/usr/spool/lp9004
/usr/spool/lp9013
/usr/spool/lp9645
/usr/spool/lp9012
/usr/spool/interface' /tmp/dat1.$$ >/tmp/dat.$$
/bin/ed /tmp/dat.$$ <<EOF  >/dev/null
1,\$s//set /g
```

```
1,\$s/\$/\\
\\/bin\\/echo \$2/g
w
q
EOF
```

```
/*
  lp9012.c    Druckerspezfische Routinen fuer PT90-Tintenmatrixdrucker

  Dies ist ein einfaches Beispielbackend. Es wird keinerlei Garantie fuer
  eine fehlerfreie Funktion uebernommen.
*/

#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <sys/termio.h>
#include "backend.h"
#include "backmeld.h"
#include "backprot.h"
#include "excode.h"
#include "files.h"
#define BLSIZ  1000     /* Anzahl der Druckzeichen im Druckpuffer */

extern FILE *tfid;      /* Filepointer auf trace-File           */
extern FILE *fid;       /* Filepointer auf Druckfile            */
extern long seekp; /*bis zu dieser Adresse wurde Datei bereits gedruckt*/
extern long seeksav; /* laufender Zaehler */
extern long size;    /* Dateigroesse      */
extern int  copies;  /* Anzahl Kopien, die noch gedruckt werden sollen */
extern int  evstk[];    /* Ereignisstack des Backends           */
extern int  evstkp;     /* Zaehler fuer Ereignisstack           */
extern int  pages;      /* Anzahl bereits ausgedruckter Seiten  */
extern int  trace;      /* Flagge fuer TRACE-Funktion           */
extern char fname[];    /* Name der zu druckenden Datei         */
extern int   cancl;     /* Cancelkommando eingetroffen          */

short stack[52];        /* wird ev. von anderen Moduln benoetigt */

long   fseek();
char *strapd();         /* Private Routine zur Stringbehandlung */

/*      aktuelle Parameter  */

char  zsf;                      /* Charater fuer Zeichensatzauswahl */
int   spmax;                    /* Spalten pro Zeile            */
int   zmax;                     /* Zeilen pro Seite             */
int   ab;                       /* Dokumentbeginn               */
int   bis;                      /* Dokumentlaenge               */
int   font = 1;                 /* Font-Auswahl                 */
int   schr;                     /* Schreibschritt               */
char  dr;                       /* Druckrichtung                */

char buf[BLSIZ+LINSIZ+BLSIZ];   /* Druck-Ausgabepuffer          */
char *cp1;                      /* Zeiger auf buf               */
char *cpx;                      /* Zeiger auf buf               */
int   zz = 0;                   /* Zeilenzaehler                */
int   spz = 0;                  /* Spaltenzaehler               */
int   fontflag = 0;             /* Zeichensatzflag              */
int   print;                    /* Indikator fuer Druckvorgang  */
char  hd = 0 ;                /* Header (wird hier nicht unterstuetzt) */
char  trl = 0 ;               /* Trailer (wird hier nicht unterstuetzt)*/
```

```
char  hdgrp = 0 ;            /* Header (wird hier nicht unterstuetzt) */
char  trlgrp = 0 ;           /* Trailer (wird hier nicht unterstuetzt)*/

static char pty[256];           /* Paerity-Tabelle       */

static struct termio termio;    /* termio-Schnittstelle         */


/*************************************************************************/
/*


        Schnittstellenparameter:
*/
unsigned short iflag   = ( IXON | ISTRIP | IGNBRK | IGNPAR );
unsigned short cflag   = ( CREAD | CS8 | B9600 );
unsigned short oflag   = 0 ;    /* keine Behandlung der Ausgabezeichen */
unsigned short lflag   = 0;     /* immer Null                          */
char  ccsvt   = 0;
char  ccsvm   = 1;
char  ccsve   = 6;
char  cs  = 7 ;     /* fuer interne prty-Routine, die nicht aufgerufen wird */

/*
        Steuersequenzen
*/
static char  *bs    = "\b";          /* Backspace                   */
static char  *ht    = "\t";          /* Tabulator                   */
static char  *nl    = "\r\n";        /* Newline                     */
static char  *ff    = "\r\f";        /* Newpage                     */
static char  *cr    = "\r";          /* Carriage return             */

static char *GZZ    = "\033[x";       /* Zeilenabstand Grundzustand   */
static char *GZSTD  = "\033R";        /* Druckparameter Grundzustand  */
static char *fschrb = "\033[%dw";     /* Schreibschritte setzen       */
static char *ffont  = "\033,%d";      /* Font einstellen              */
static char *fzmax  = "\033[%d";      /* Seitenlaenge einstellen      */
static char *fzs    = "\033(%c";      /* Zeichensatz einstellen       */
static char *fdr    = "\033 %c";       /* Druckrichtung                */
/*
        Standardparameter
*/
int   spmaxstd = 4096;          /* Standard: keine Spaltenbegrenzung  */
int   zmaxstd  = 68;            /* Standard: 68 Zeilen pro Seite      */
char  zstd     = 'B';           /* Standard: ASCII - Zeichensatz      */
int   abstd    = 0;             /* Standard: Dokumentbeginn           */
int   bistd    = 1<<15-2;       /* Standard: beliebig langes Dokument */
int   schrstd  = 1;             /* Standard: 1/10 Zoll                */
char  drstd    = ':';           /* Standard bidirektional             */
int  emstd = 0 ;                /* Standard fuer em9001               */

static int grfbytes  = 0;  /* Anzahl Grafikbytes, die noch ausgegeben werden-
 muessen */
static int em9001 = 0 ;       /* Flagge fuer 9001-Grafik-Emulation */


/*************************************************************************/
```

```
backflg(ac,av)          /* Auswertung der Backendschalter */
int ac;                 /* wird beim Start des Backends und bei jedem  */
char *av[];             /* Druckauftrag aufgerufen                     */
{
        char * pschalt ;
        int  i = 0;
        int  j;
/*
        Einstellen der Standardwerte
*/
        spmax = spmaxstd;
        zmax = zmaxstd;
        zsf  = 'B';
        ab   = abstd;
        bis  = bistd;
        schr = schrstd;
        dr   = drstd;
        fontflag = 0;
        em9001 = emstd ;

        for(i=1;i<ac;i++)
        {
                pschalt = av[i]+1 ;

                if ( strncmp( pschalt , "ab=" , 3 ) == 0 )
                        ab = ( j = atoi( pschalt + 3 ) ) > 1 ? j : 1  ;
                else if ( strncmp( pschalt, "bis=" , 4 ) == 0 )
                        bis = (j = atoi( pschalt + 4 ) ) > ab ? j : bis ;
                else if ( strncmp( pschalt, "pl=", 3 ) == 0 )
                        zmax = (j = atoi( pschalt + 3) ) > 5 ? j : zmax ;
                else if ( strncmp( pschalt, "pb=", 3 ) == 0 )
                        spmax = (j = atoi(pschalt + 3)) > 5 ? j : spmax ;
                else if ( strncmp( pschalt, "font=", 5 ) == 0 )       {
                        fontflag = 1;
                        font = atoi( pschalt + 5 ) ;
                       }
                else if ( strcmp( pschalt, "pb1" ) == 0 ) {
                        schr = 1;
                       }
                else if ( strcmp( pschalt, "pb2" ) == 0 ) {
                         schr = 2;
                       }
                else if ( strcmp( pschalt, "pb3" ) == 0 )  {
                        fontflag = 1;
                        if ( font != 4 )      font = 2;
                        schr = 3;
                       }
                else if ( strcmp( pschalt, "pb4" ) == 0 )  {
                        fontflag = 1;
                        if ( font != 4 )      font = 2 ;
                        schr = 4;
                       }
                else if ( strcmp( pschalt, "dt" ) == 0 )
                        zsf = 'K';
                else if ( strcmp( pschalt, "int" ) == 0 )
                        zsf = 'B';
                else if ( strcmp( pschalt, "uni" ) == 0 )
                        dr = '=';
```

```
                else if ( strcmp( pschalt, "em91" ) == 0 )
                        em9001 = 1 ;
                else if ( strcmp( pschalt, "trace" ) == 0 )
                        trace = 1 ;
                else if ( strcmp( pschalt, "statusfile" ) == 0 )
                        ;       /* Nostalgieschalter ohne Bedeutung */
                else
                        return( (BF␣PERR<<8) | i ) ;  /* Fehler */
        }
        if (trace)    fprintf(tfid,"pschalt durchlaufen\n");
        return( BF␣OK << 8 ) ;
}


druin()                 /* Initialisierung des Backends */
{
        int n;

        ioctl(1,TCGETA,&termio);
        termio.c␣iflag      = iflag;
        termio.c␣cflag      = cflag;
        termio.c␣oflag      = oflag;
        termio.c␣lflag      = lflag;
        termio.c␣ccs[VTIME] = ccsvt;
        termio.c␣ccs[VMIN]  = ccsvm;
        termio.c␣ccs[VEOL]  = ccsve;
        ioctl(0,TCSETA,&termio);
        ioctl(1,TCSETA,&termio);


/*
      'echtes' (einmaliges) Einstellen der Standardwerte:
      Beim Start des Backends wird zunaechst die Funktion backflg()
      zur Auswertung der Standardschalter aufgerufen. Anschliessend wird
      die Funktion druin() durchlaufen, wo jetzt diese Standardwerte
      festgehalten werden.
*/

        parity();                       /* Initialisierung der Parity-
Tabelle */
                        /* Sicherheitshalber wird die eigene Routine aufgeru-
fen */

        spmaxstd = spmax ;
        zmaxstd  = zmax  ;
        zstd     = zsf   ;
        abstd    = ab    ;
        bistd    = bis   ;
        schrstd  = schr  ;
        drstd    = dr    ;
        emstd    = em9001 ;

        if(trace)
                fprintf(tfid,"\n, druin zmax=%d,spmax=%d,font=%d\n",
                                        zmax,spmax,font);

}
```

```
static int etxflag = 0 ;

dstat(bf3)                  /* Druckerstati auswerten       */
char *bf3 ;         /* wird von den druckerunabhaengigen Moduln verwendet */
{
        int statflag = 1, drst = 0 ;
        char c ;
        if(trace)   fprintf(tfid,"dstat: etx=%d, cancl=%d  ", etxflag, cancl);

        if ( etxflag == 0 ) {
                c = pty[ ETX & 0177 ];
                if( write( 1, &c, 1) != 1 ) {    /* Zeitablauf          */
                        if ( trace ) fprintf( tfid, " W-SIGNAL\n" ) ;
                        bp␣stat( B␣IDLE ) ;
                        if ( !cancl )  evstk[++evstkp] = E␣INTR ;
                        return( -1 ) ;
                }
        }
        else
              etxflag = 0 ;

        while( statflag || rdchk(0) > 0 )   /* Es werden alle Rueck-  */
        {                                   /* meldungen gelesen !    */
                if( read( 0, &c, 1 ) < 1 ) {
                        bp␣stat(B␣NPOLL);
                        drst = 1;
                        statflag = 0 ;   /* Schleife verlassen */
                        if (trace) fprintf(tfid, " R-SIGNAL\n" );
                }
                else {                  /* Zeichen wurde gelesen */
                        c = c & 0177 ;
                        if ( trace ) fprintf( tfid, "%o, ", c ) ;
                        if ( c == ACK )  statflag = 0 ;
                }
        }

        if ( drst )
                evstk[++evstkp] = E␣NAK ;
        else
                evstk[++evstkp] = E␣ACK;

        if ( trace ) fprintf( tfid, "dstat beendet : %d\n", drst ) ;
        return( drst );
}

char *strapd(cp1,cp2)           /* String anhaengen     */
char *cp1,*cp2;
{
        while(*cp2 && (*cp1++ = pty[*cp2++]));
        return(cp1);
}

anblock()                               /* Ausdruck starten */
{
        char cp2[10];
        struct stat statbuf;
        int    i;
```

```
        if(trace)   fprintf(tfid,"anblock ");
        bp␣anp();

        stat(fname,&statbuf);       /* Information aus der Inode       */
        size = statbuf.st␣size;
        fseek(fid,seekp,0);   /* Positionieren auf letzte Seitengrenze */
        cp1   = buf;
        if(seekp)   cp1 = strapd(cp1,ff);
        zz    = 0;
        spz   = 0;

/*      Drucker initialisieren   */
        if ( fontflag )       {
              sprintf(cp2,ffont,font);          /* Font-Auswahl       */
              cp1 = strapd(cp1,cp2);
        }
        sprintf(cp2,fzs,zsf);              /* Zeichenvorrat           */
        cp1 = strapd(cp1,cp2);
        sprintf(cp2,fschrb,schr);         /* passenden Schreibschritt */
        cp1 = strapd(cp1, cp2);           /* einschalten              */
        sprintf(cp2,fdr,dr);              /* Druckrichtung bestimmen  */
        cp1 = strapd(cp1, cp2);

        if ( em9001 ) {
                cp1 = strapd( cp1, "\033[3 w" ) ;
        }

        cpx = cp1;

        grfbytes = 0 ;  /* Anzahl Grafikbytes auf Null setzen  */

        if (trace)    fprintf(tfid,"anblock()-Ende\n");

        block();                                  /* 1. Block ausgeben  */
}

block()                          /* Datenblock ausgeben         */
{
        int     c;               /*  gelesenes Zeichen          */
        int     i;               /*  Zeichen im Ausgabepuffer   */
        int     j;
        if(trace)   fprintf(tfid,"block   ");
        print = (pages+1 >= ab);

        /* Eventuell noch Grafikbytes ausgeben */
        for ( i = cp1-buf; grfbytes > 0 && i < BLSIZ; i++, seeksav++, grfby-
tes--)
        {
                if ( ( c = fgetc( fid ) ) == EOF ) c = '\0';
                *cp1++ = c ;
        }

        for(i=cp1-buf;i<BLSIZ;i++,seeksav++) /* BLSIZ=1000 Druckzeichen*/
        {
                switch(c = fgetc(fid))  /* Druckfile lesen      */
                {
                case NL:
```

```
                        cp1 = strapd(cp1,nl);
                        spz = 0;
                        if(++zz >= zmax) {
                                if(++pages >= bis )
                                        c = EOF;
                                i = BLSIZ;
                                zz = 0;
                        }
                        break;
                case CR:
                        cp1 = strapd(cp1,cr);
                        spz = 0;
                        break;
                case FF:
                        if ( trace ) fprintf( tfid,"FF" ) ;
                        cp1 = strapd(cp1,ff);
                        if(++pages >= bis )  c = EOF;
                        i = BLSIZ;
                        zz = 0;
                        spz = 0;
                        break;
                case HT:  /* Tabulatoren werden in Blanks umgewandelt */
                        c = ' ';
                        do
                              *cp1++ = pty[c];
                        while ( spz < spmax && ++spz % 8 ) ;
                        i = cp1-buf; /* Zeichenzaehler aktualisieren */
                        break;
                case BS:
                        if(spz) {
                                cp1 = strapd(cp1,bs);
                                spz--;
                        }
                        break;
                case EOF:
                        if ( trace ) fprintf( tfid,"EOF" ) ;
                        seeksav--;
                        i = BLSIZ;
                        break;
                case ESC:
                        *cp1++ = pty[c] ;
                        if ( seqsuch() == EOF )  {
                                     i = BLSIZ ; c = EOF ;
                                     break ;
                        }
                    /* Falls grfbytes > 0, muessen Grafikdaten ausgegeben wer-
den */
                        for ( ; grfbytes>0 && i<BLSIZ; i++, seeksav++, grfby-
tes--)
                        {
                              if ( ( c = fgetc( fid ) ) == EOF ) c = '\0' ;
                              *cp1++ = c ;
                        }
                        break ;
                case '\0': break ;  /* um GKS-Fehler zu umgehen */
                default:
                        if(spz++ < spmax)
                                 *cp1++ = pty[c];
```

```
                        break;
                }
        }
        if(c == EOF)
        {
                if(zz || !pages)        cp1 = strapd(cp1,ff);
                zz = 0;
                if ( copies == 1 )      cp1 = strapd(cp1,GZSTD);
        }
/*
        Druckfileblock ausdrucken
*/
        if ( !zz && !grfbytes ) {  /* Falls Seitenende und keine Grafikda-
ten */
                *cp1++ = pty[ ETX & 0177 ] ; /* ETX anfuegen */
                etxflag =1 ;
        }

        if(print && write(1,buf,cp1-buf) == cp1-buf)
        {
                if( etxflag )
                {
                        if ( (i=dstat("\0")) != 0 ) /* Druckerstati abfra-
gen */
                                return ;
                        seekp = seeksav;
                        bp␣rept(seekp,size,pages,copies);
                }
                else   evstk[++evstkp] = E␣ACK;

                if(c == EOF)                    /* Druckfileende       */
                {
                        if(--copies && !cancl )   /* weitere Kopien      */
                        {
                                seeksav = seekp = 0L;
                                fseek(fid,0L,0);
                        }
                        else                /* Druckvorgang beendet */
                        {
                                evstk[evstkp] = E␣HDRQ;
                                fclose(fid);
                                fid = NULL;
                                seekp = seeksav;
                        }
                        bp␣rept(seekp,size,pages,copies);
                        print = 0;
                        sleep(5);              /* fuer Remote Reset   */
                }
                cp1 = buf;
        }
        else if(print)    /* kein Druckvorgang, 'CANCEL' oder 'offline' */
        {
                ioctl(1,TCFLSH,&termio);
                evstk[++evstkp] = E␣INTR;
                if ( trace ) fprintf( tfid, " W-INTR\n" ) ;
        }
        else
```

```
        {
                bp␣rept(seeksav,size,pages,copies);
                evstk[++evstkp] = E␣ACK;
                cp1 = cpx;
        }
}

parity()    /* Parity-Tabelle initialisieren, private Routine mit neuem Na-
men */
{
      int i = 0;
      register char a , p;
      pty[255] = '\0';

      while( i++ < 127 )
      {
              for( p = 0, pty[i] = a = i; a; a >>= 1)
                      if ( a & 01 )   p = p ? 0 : 1;          /* Flip-Flop */
              if (!p)      pty[i] |= 0200;              /* hoechstes Bit set-
zen */

      }
      if (trace)  fprintf(tfid,"parity-Ende  \n");

}

seqsuch()
{
        int n, c, escflag, esclaenge ;
        int nbyte ;
        char escfolg[10], *cpesc ;

        nbyte    = 0 ;
        escflag  = 1 ;
        esclaenge = 9999 ;    /* maximale Laenge einer ESC-Folge */

        while ( escflag )
        {
                if ( ( c = fgetc(fid) ) == EOF ) {
                        if (trace) fprintf(tfid, "EOF in ESC-Folge\n") ;
                        return( EOF ) ;
                }

                if ( escflag >= esclaenge ) { /* Ende der ESC-Folge er-
reicht */
                        *cp1++ = pty[c] ;
                        return( 0 ) ;
                }

                switch( escflag )
                {
                case 1: *cp1++ = pty[c] ;
                        switch( c )
                        {
                        case ' ':
                        case ',':
                        case '(':
                        case '!':
```

```
                                escflag   = 2 ;
                                esclaenge = 2 ;  /* ESC + 2 Zeichen */
                                break ;  /* Ende nach dem naechsten Zei-
chen */
                        case '[': escflag = 2 ;
                                cpesc = cp1 ;  /* Anfang der Zahlen mer-
ken */
                                break ;
                        default : escflag = 0; /* ESC + 1 Zei-
chen            */
                                break ;      /* Ende der ESC-Folge er-
reicht */
                        }
                        break ;
                case 2: /* dieser Fall kann nur eintreten, bei ESC [ .... */
                        *cp1++ = pty[c] ;
                        switch( c )
                        {
                        case 'p':
                        case '{':
                        case 's':
                        case 'x':
                                escflag = 0 ; /* sofortiges Ende */
                                break ;
                        case '<':
                        case '=':
                        case ' ':
                                esclaenge=3; /* Ende nach dem naechsten Zei-
chen */
                                break ;
                        case ';': escflag++ ;
                                break ;
                        default:
                                if ( c >= '0' && c <= '9' ) {
                                        escflag++ ;
                                        nbyte = 10*nbyte + c - '0' ;
                                }
                                else escflag = 0 ;
                                break ;
                        }
                        break ;
                case 3: /* dieser Fall kann nur bei ESC [ ZAHL oder nach ei-
nem ';' eintreten */
                case 4:
                case 5:
                case 6:
                        if ( c >= '0' && c <= '9' ) {
                                escflag++ ;
                                nbyte = 10*nbyte + c - '0' ;
                                *cp1++ = pty[c] ;
                                break ;
                        }
                case 7: *cp1++ = pty[c] ;
                        switch( c )
                        {
                        case ';': escflag = 3 ; /* Es kommen weitere Zahlen */
                                nbyte = 0 ;   /* Wieder bei Null beginnen */
                                break ;
```

```
                        case ' ': esclaenge = ++escflag ;
                                  break ; /* Ende der ESC-Folge nach dem na-
echsten Zeichen */
                        case 'y':         /* Grafikfolge wie bei 9001 */
                        case 'v':         /* Grafikfolge mit Wiederholungsfak-
tor */
                                  grfbytes = nbyte ;
                                  if (trace) fprintf(tfid, "grfby-
tes=%d ", grfbytes);
                                  escflag = 0 ;
                                  break ;
                        case 'x': if ( em9001 && nbyte ) {
                                        sprintf(escfolg, "%02dx",(nbyte*120)/
72 -1);
                                        cp1 = strapd( cpesc, escfolg ) ;
                                  }
                                  escflag = 0 ;
                                  break ;
                        default:
                                  escflag = 0 ;  /* Ende der ESC-Folge */
                                  break ;
                        }
                        break ;
                default: escflag = 0 ;
                         break ;
                }
        }
        return( 0 ) ;
}


header()
{
}

trailer()
{
}

npafn()                                         /* Abbruch eines Druckauf-
trags*/
{
        short i,j;

        if(trace)   fprintf(tfid,"npafn   ");

        if(print)
        {
                while( grfbytes > 0 )           /* Graphik-Mode verlas-
sen    */
                {
                        for(i=0,cp1=buf ; i<BLSIZ && grfbytes>0; grfbytes-
-, i++)
                                *cp1++ = '\0';
                        i = write(1,buf,cp1-buf);
                }
                cp1 = buf;
                cp1 = strapd(cp1,ff);
```

```
                i   = write(1,buf,cp1-buf);
                sleep(5);
                print = 0;
        }
        cancl = 0;

        bp_afn();
}
```

11 DÜ-Schnittstelle PC-X, PC-MX2

In diesem Kapitel möchten wir die für die Kommunikation mit dem übergeordneten Rechner wichtige Schnittstelle, die V.24, kurz und praxisbezogen erläutern.

Eine Kommunikationsschnittstelle ist ein komplexes Gebilde.
Will man es beschreiben, so ist es am einfachsten, es in einzelne Teilsysteme (Schichten) zu gliedern.

In der Praxis wird dies als ISO-7-Schichten-Modell dargestellt.
Die Schichten sind im Einzelnen:

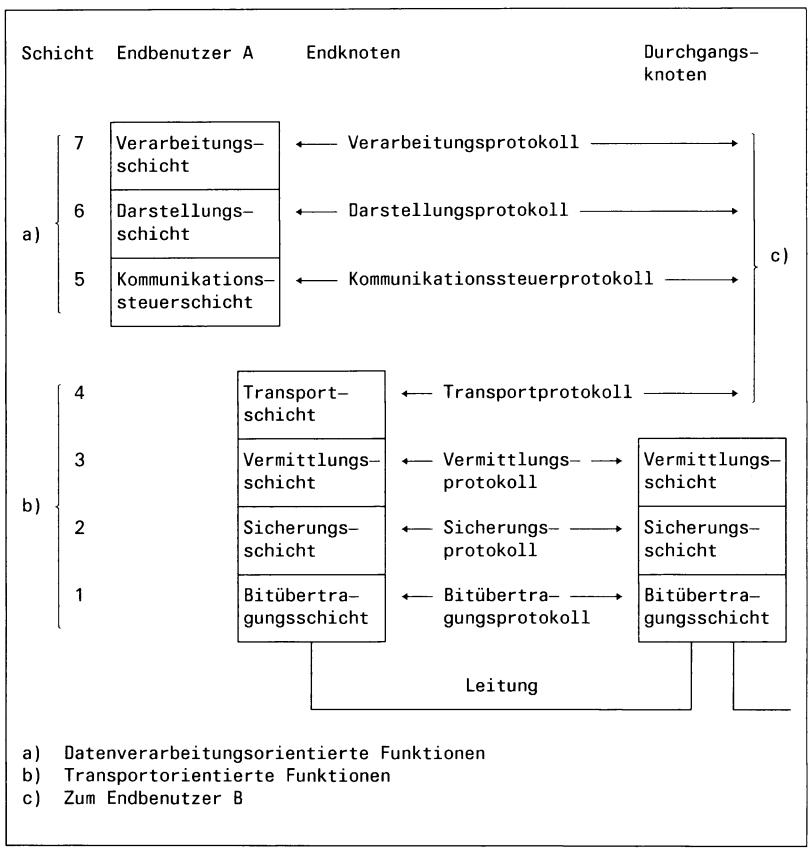


Bild 11-1 ISO-7-Schichten-Modell

Beschreibung der Schichten des ISO-7-Schichten-Modells

Aufgaben der Schicht 1 (Bitübertragungsschicht)

- Durchführung des Verbindungsaufbaues (wenn erforderlich, z.B. Anschalten des Modems an die Leitung beim Laden der Nachbildung der Datensichtstation 9750).
- Durchführung und Überwachung der physikalischen Verbindung (elektrische Anpassung).

Aufgaben der Schicht 2 (Sicherungsschicht)

- Informationsübertragung von Knoten zu Knoten, d.h. Übertragung der eingegebenen Daten vom PC zum nächsten Rechner, einschließlich der Sicherung der Daten und - wenn erforderlich - Fehlerkorrektur.
  Bei diesem PC ist das die Prozedur MSV1. Den Datenaustausch zwischen PC und dem nächsten Rechner kann man oft an den Anzeigelampen der Übertragungseinrichtungen verfolgen.

Aufgaben der Schicht 3 (Vermittlungsschicht)

- Datentransport von Endknoten zu Endknoten über alle Durchgangsrechner, z.B. wenn Sie mit einem Rechner Verbindung aufnehmen wollen, müssen Sie noch zusätzliche Kommandos geben (Adressierung), um die gewünschte Verbindung zu einem bestimmten Programm in einem bestimmten Rechner herzustellen.
  Diese Schicht besorgt den Transport der Nachricht über alle Knoten, oft mit unterschiedlichen Prozeduren bis zum Programm.

Aufgaben der Schicht 4 (Transportschicht)

- Überwachung des Datentransports zwischen den Endbenutzern, z.B. wenn Sie mit der Nachbildung der Datensichtstation 9750 arbeiten, erfolgt diese Kontrolle durch den Bediener. Arbeitet aber der Filetransfer, so übernimmt diese Überwachung im PC das Programm Filetransfer.

Aufgaben der Schicht 5 (Kommunikationssteuerschicht)

- Steuerung des Arbeitsablaufes während eines Dialogs mit dem Rechner, z.B. welche Dateien können und dürfen Sie vom übergeordneten Rechner übertragen.

Aufgaben der Schicht 6 (Darstellungsschicht)

- Loslösen der Anwendungen von der Hardware, z.B. an Ihrem PC soll ein Anwenderprogramm immer gleich zu bedienen sein, unabhängig, welcher Hersteller Ihren PC geliefert hat.

Aufgaben der Schicht 7 (Verarbeitungsschicht)

- Ihre Anwendung, mit der Sie nun endlich arbeiten können.

11.1 Die V.24-Schnittstelle

Die V.24-Schnittstelle (ein Teil der Schicht 1), die Schnittstelle zwischen Datenendeinrichtung (DEE, das ist Ihr PC) und Datenübertragungseinrichtungen (DÜE) wurde in der CCITT-Empfehlung V.24 festgelegt und als DIN 66020 genormt. Die elektrischen Eigenschaften sind in der Empfehlung V.28 festgelegt.

Es folgt die Beschreibung aller Leitungen im sog. V.24-Kabel mit allen Bezeichnungen (ohne Leitungen für die automatische Wahl).

11.1.1 Beschreibung der einzelnen Leitungen

11.1.1.1 Die Erdleitung E2

Die Erdleitung E2 ist der gemeinsame Rückleiter für die unsymmetrischen Doppelstromleitungen, d.h. der Bezugspunkt für alle Schnittstellenleitungen.

Hinweis zur Installation

PC's , die über eine geschirmte Schnittstellenleitung (wie z.B. das V.24-Kabel) angeschlossen sind, müssen am gleichen Netzverteiler angeschlossen sein wie die DÜE.

11.1.1.2 Betriebsbereitschaft S1, M1

a) Standleitung, auch HfD-Anschluß genannt (Hauptanschluß für Direktruf)
Die Übertragungsleitung ist immer geschaltet.

Wenn der PC die Schnittstelle S1 in den EIN-Zustand schaltet, wird die DÜE an die Übertragungsleitung angeschaltet und meldet M1 zurück. Dies ist z.B. der Fall, wenn die Nachbildung der Datensichtstation 9750 geladen wird.

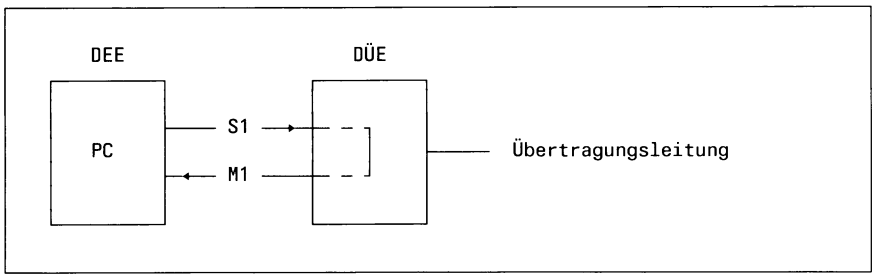


Bild 11-2 Standleitung

b) Wahlleitung
Die Übertragungsleitung wird nur für den Zeitraum eines Dialogs geschaltet.

Das Umschalten von Sprech- auf Datenbetrieb wird durch Drücken der Datentaste am Telefon ausgelöst.

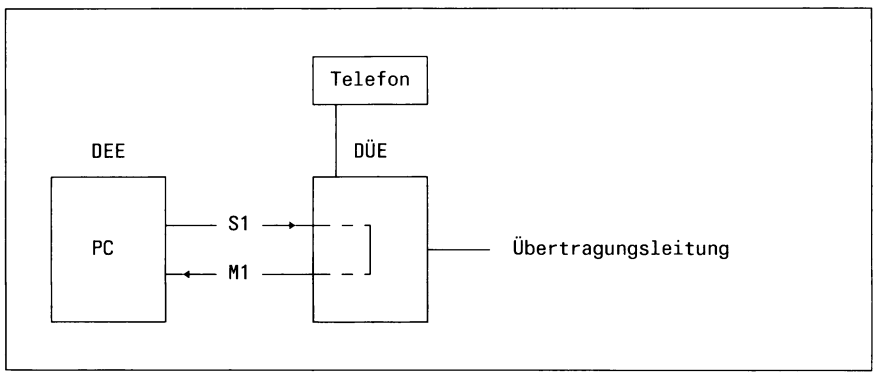


Bild 11-3 Wahlleitung

11.1.1.3 Sender steuern S2, M2, M5

Bei der MSV-1-Prozedur kann immer nur der eine oder andere Teilnehmer senden (Halbduplex). Trotzdem können auf den Übertragungsleitungen verschiedene Möglichkeiten der Übertragung gewählt werden.

a) Duplex-Betrieb
Die Sender und Empfänger an beiden Orten sind immer eingeschaltet. Die Übertragung könnte gleichzeitig in beiden Richtungen erfolgen. Dies ist bei Standleitungen immer der Fall und erspart die Richtungsumschaltepausen.

b) Halbduplex-Betrieb
Es ist immer nur ein Sender und der Empfänger an der Gegenstelle eingeschaltet.
Die Übertragung erfolgt zeitlich nacheinander, immer nur in einer Richtung.
Dies ist bei einer Wahlleitung der Fall.

Duplex-Betrieb

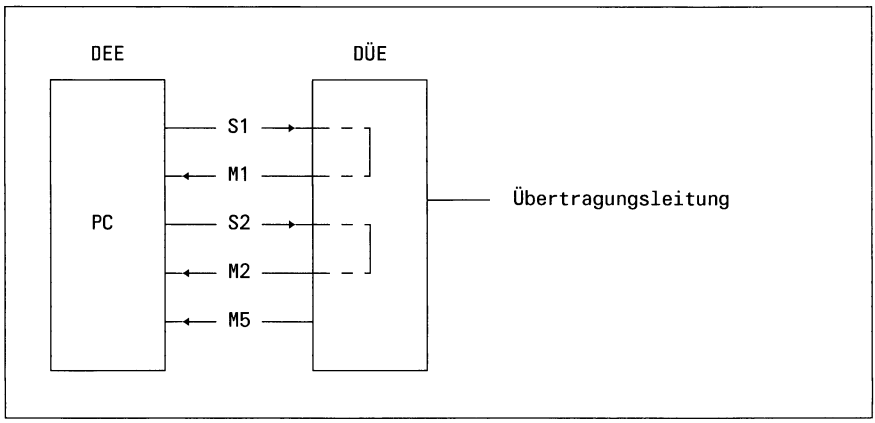


Bild 11-4 Duplex-Betrieb

Die Schnittstellenleitung S2 ist immer im EIN-Zustand.
Sie ist im PC durch die Auswahl 'Standleitung' fest vorgegeben.
Die Leitung M5 kann in den AUS-Zustand gehen, wenn die Übertragungsleitung gestört ist, wird aber von TRANSIN Version 1.0B und 1.0C nicht ausgewertet.

Halbduplex-Betrieb

Wer gerade senden darf, wird von Schicht 2 (Prozedur) bestimmt.

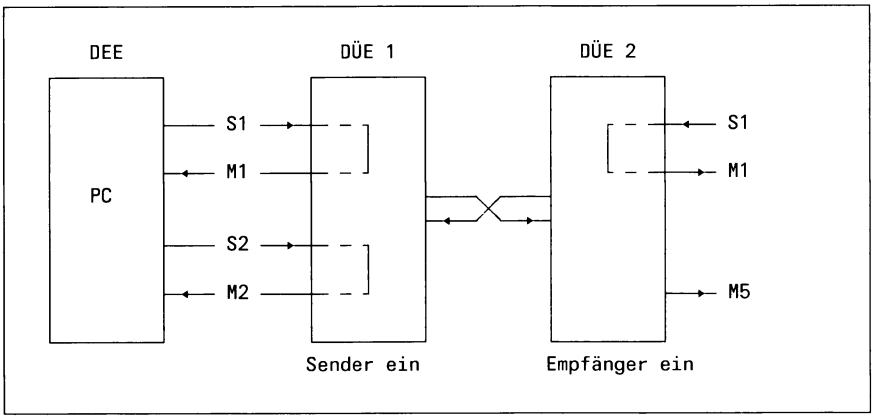


Bild 11-5 Halbduplex-Betrieb

Soll vom PC etwas gesendet werden, so wird die Leitung S2 in den EIN-Zustand geschaltet. Die Sendebereitschaft des Modems wird mit der Leitung M2 verzögert zurückgemeldet. Diese Verzögerungszeit stellt sicher, daß in der Gegenstelle der Empfänger eingeschaltet ist. Dies wird dem Rechner an der Gegenstelle mit der Leitung M5 gemeldet.

Bedingt durch diese Umschaltzeiten ist eine Übertragung von Daten - bei gleicher DÜ-Geschwindigkeit - auf einer Halbduplex-Strecke immer langsamer als bei Duplex-Betrieb.

Das Wechselspiel der Schnittstellenleitungen ist an vielen DÜE's an Anzeigelampen, die mit den Kurzbezeichnungen versehen sind, zu beobachten.

11.1.1.4 Daten D1, D2

D1 Sendedaten vom Rechner
D2 Empfangsdaten zum Rechner

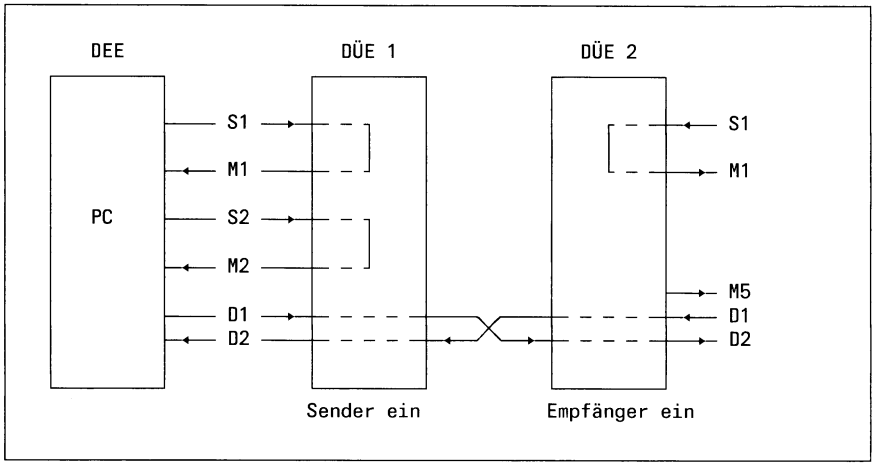


Bild 11-6 Daten D1, D2

Während des Sendens werden die Empfangsdaten nicht bewertet.
In den meisten DÜE's wird ein sog. Clamping eingestellt, d.h. wenn der Sender im EIN-Zustand ist, so werden die Leitungen M5 und D2 im AUS-Zustand gehalten.

11.1.1.5 Takte bei synchroner Übertragung

T2 Sendeschrittakt von der DÜE
T4 Empfangsschrittakt von der DÜE

Datenübertragung ist nur möglich, wenn beide Stellen im gleichen Takt arbeiten. Dies besorgen Taktgeneratoren in jeder DÜE.

Die Übertragungsgeschwindigkeit, mit welcher die DÜE's arbeiten, ist abhängig von der Leitungsart. Gebräuchlich sind für

- Wahlleitungen 1200, 2400 bit/s
- Standleitungen 4800, 9600 bit/s

11.2 Die Übertragungprozedur MSV1

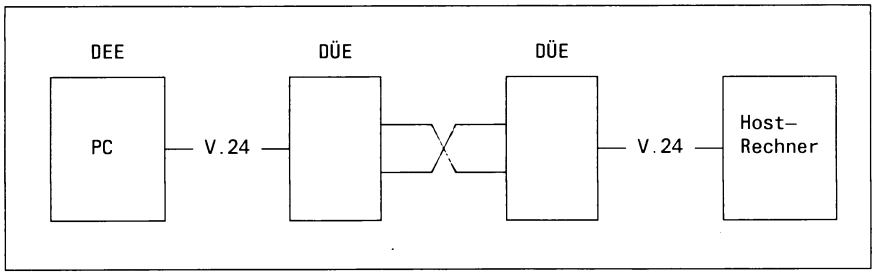


Bild 11-7 Übertragungsprozedur MSV1

Zur Übertragung von Daten mit der Prozedur MSV1 ist es notwendig, daß eine Station als Leitstation arbeitet. Das ist der Host. Alle angeschlossenen PC's sind Folgestationen.

Da an einen PC-MX2 mehrere Bedieneinheiten angeschlossen und an jeder Bedieneinheit mehrere On-Line-Programme gestartet werden können, hat jeder PC zwar nur eine Sendeaufforderungs-Adresse (General Poll- oder Knoten-Adresse), aber bis zu 12 Ausgabeadressen (Selecting- oder Empfangsaufforderungs-Adressen).

Aufgaben der Leitstation bei Standleitungen:

Der Leitstation müssen alle Trabantenstationen, mit denen sie Verbindung aufnehmen kann, bekannt sein. Dies geschieht mit Adressen und Identifikationszeichen, die für jeden PC vergeben werden. Auf einer Übertragungsstrecke darf eine Adresse und ein Identifikationszeichen nur einmal vorkommen, wie in einer Straße eine Hausnummer auch nur einmal vorkommt.
Das Einstellen von Adresse und Identifikationszeichen am PC wird beim Einspielen der TRANSIN-Software vorgenommen.
Die einzustellenden Daten sind vom Rechenzentrum zu erfragen.

Mehrere PC's können über einen sog. Schnittstellenvervielfacher an eine DÜE-Strecke angeschlossen werden.

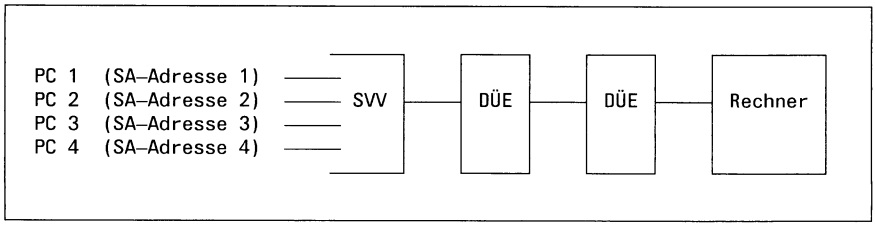


Bild 11-8 Schnittstellenvervielfacher (SVV)

Aufgaben der Leitstation bei Wahlleitungen:

Der Leitstation braucht nur ein PC mit seinen möglichen Empfangsaufforderungsadressen (Selecting-Adresse) bekannt sein.
Jeder PC, der an einer Wahlleitung Verbindung zu dem Host aufnimmt, hat diese gleichen Sende- und Empfangsaufforderungsadressen.
Dies ist möglich, da zur selben Zeit nur ein PC auf der Wahlleitung arbeiten kann.

Polling oder Sendeaufforderung

Die Leitstation spricht nacheinander jeden PC mit seiner Adresse an und prüft, ob Sendedaten anstehen (= Polling läuft).

An Ihrem PC erkennen Sie diesen Zustand, wenn bei Betrieb mit der Nachbildung 9750 die Anzeige **LTG** erscheint.

Stehen Sendedaten an, so werden sie sofort gesendet. Sind die Daten im Rechner angekommen, erhält der PC eine Quittung und der nächste PC wird abgefragt.
Stehen keine Daten an, sendet der PC ein EOT-Zeichen, um dies mitzuteilen und zu quittieren.

An Ihrem PC erkennen sie den Zustand **Sendedaten stehen an**, wenn bei Betrieb mit der Nachbildung 9750 die Anzeige **DÜE** erscheint.

11.3 Realisierung (SINIX 1.0B, 1.0C)

Die Prozedur MSV1 ist Bestandteil des SINIX-Kernel. Das Einbringen eigener Treiber oder das Verändern des bestehenden ist nicht möglich.

Die TRANSIN-Anwendungen, wie die Nachbildungungen DST 9750, 8122 und Filetransfer, sind als Prozesse realisiert, die über die Systemcall-Schnittstelle mit dem SINIX-Kernel kommunizieren.

Ein Offenlegen dieser Schnittstelle ist für obige Versionen nicht vorgesehen.

11.4 Technische Daten

Technische Daten SINIX 1.0B (ohne Kommunikations-Prozessor)

Synchrone Übertragung, normierter Modus : 3 oder 7 SYN Übertragungsgeschwindigkeit : max. 2400 bit/s Zeichenrahmen : 7 bit + Parity Parity : ungerade Übertragungscode : ISO-7-Bit-Code Zeichensicherung : Paritäts- und Blocksicherung (BCC) Taktversorgung : extern mit T2, T4, kein Eigentakt möglich

Unterstützte Schnittstellenleitungen

S1 (DTR,108) M1 (DSR,107) S2 (RTS,105) M2 (CTS,106) M3 (RI ,125)

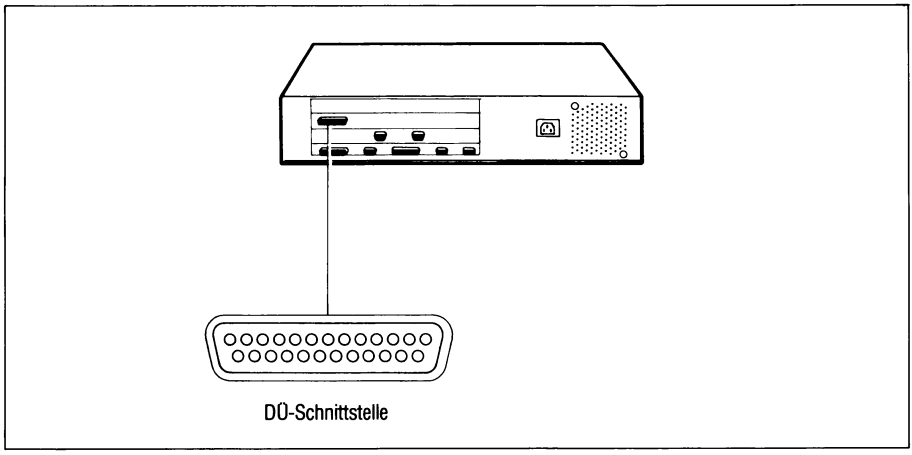


Bild 11-9 Anschlußfeld des PC-MX2

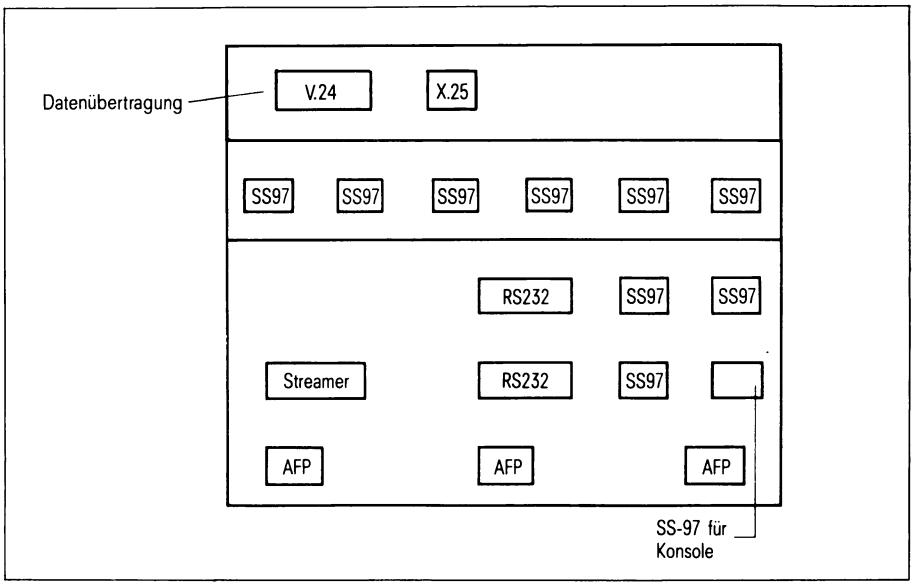


Bild 11-10 Anschlußfeld des PC-X/PC-X10

12 Kopplung

Fremd-PC besitzen oft eine Schnittstelle RS232. Über diese Schnittstelle ist eine Kopplung mit dem Siemens PC möglich.

Verfügt der Fremd-PC über eine Schnittstelle V.11 (entspricht in den elektrischen Eigenschaften der Siemens-Schnittstelle 97), so ist eine Kopplung über diese Schnittstelle ebenso möglich.

Bemerkungen zum PC-MX2

Der Standard-E/A-Prozessor des PC-MX2 hat 4 Schnittstellen SS97 und 2 Schnittstellen RS232.
Alle Schnittstellen können unabhängig voneinander bedient werden.

Bemerkungen zum PC-X

Bis zu einer Eingabegeschwindigkeit von 4800 bit/s funktioniert die Flußsteuerung. Bei höheren Übertragungsraten ist wieder der Anwender verantwortlich für die Datenkonstanz.

Diese Einschränkung entfällt beim PC-X10.

12.1 Kabel

Erfolgt eine Kopplung über die Schnittstelle RS232, so ist ein sog. Nullmodem erforderlich.

Beispiel:

Kabel für Schnittstelle RS232

PC-MX2 Schalterstellung Teil 2 beachten! Nur S4 auf Stellung 2
Fremd-PC Vorschriften beachten!

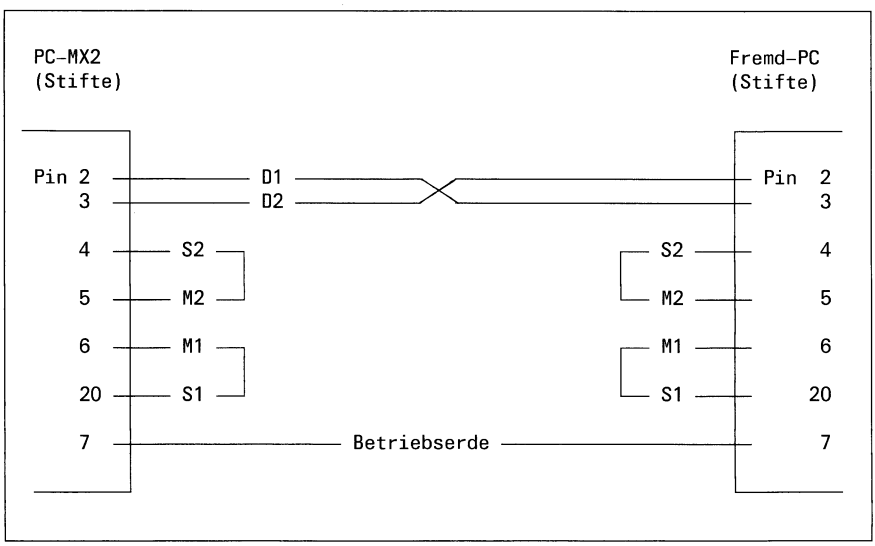


Bild 12-1 Nullmodem

Erfolgt eine Kopplung über die Schnittstelle SS97, so ist folgendes Kreuzungskabel erforderlich:

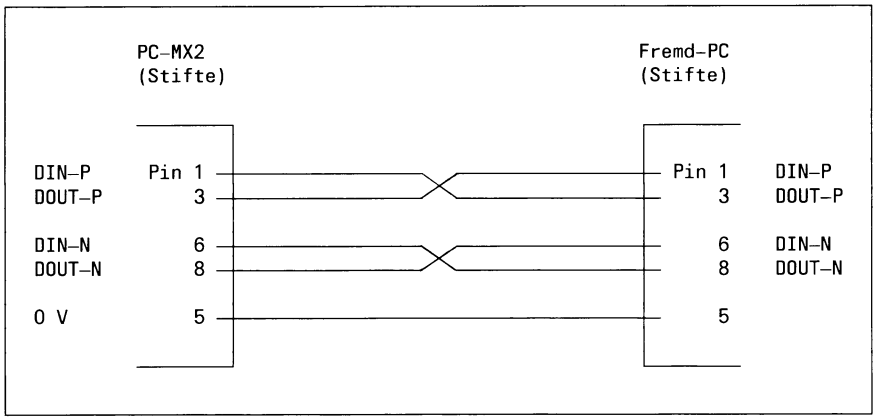


Bild 12-2 Kreuzungskabel für SS97

12.2 Beispiele

12.2.1 Shellprozedur zum Lesen von ASCII-Daten

Die folgende Prozedur finden Sie auf der Diskette 'SSHB' unter dem Dateinamen getrs232.

```
# @(#)getrs232 1.3 85/06/13
#####################################################################
#                                                                   #
# getrs232 - einfache Shellprozedur zum Lesen von ASCII-Daten von   #
#            einem anderen Rechner, der mit PC-MX/PC-X ueber die    #
#            Drucker-RS232-Schnittstelle gekoppelt ist.             #
#                                                                   #
#####################################################################
#                                                                   #
# Bemerkungen:                                                      #
#                                                                   #
#       Die Prozedur ist ablauffaehig unter SINIX V1.0B auf dem     #
#       Siemens PC-MX, ebenso unter SINIX V1.0 auf PC-X, duerfte    #
#       jedoch auch fuer SINIX V1.0C gelten.                        #
#                                                                   #
#       Es koennen nur ASCII-Daten (0x01 ... 0x7f) mit ungeradem    #
#       Parity-Bit uebertragen werden. Saemtliche <CR>-Zeichen      #
#       (0x0d) werden entfernt.                                     #
#                                                                   #
#       Die Uebertragungsgeschwindigkeit betraegt 1200 Baud (muss   #
#       auch am sendenden Rechner eingestellt werden!). Es wurde    #
#       deshalb eine so niedrige Rate gewaehlt, weil in SINIX V1.0B #
#       noch keine XON/XOFF-Flusssteuerung in Eingaberichtung       #
#       moeglich ist. In der Praxis hat sich gezeigt, dass bei 1200 #
#       Baud normalerweise kein Datenverlust auftritt. Voraussetzung #
#       ist jedoch, dass das System nicht anderweitig belastet ist. #
#       Trotzdem empfiehlt sich natuerlich eine Ueberpruefung der   #
#       angekommenen Daten.                                         #
#                                                                   #
#       Es wird pro Aufruf der Prozedur eine Datei erstellt. Die    #
#       Ausgabe geschieht ueber stdout, d.h. man muss beim Aufruf   #
#       eine geeignete Umlenkung vornehmen. Beispiel:               #
#                                                                   #
#                       getrs232 >zieldatei                         #
#                                                                   #
#       Da kein erkennbares Dateiende-Kriterium ueber die Leitung   #
#       kommt, muss nach dem Ende der Uebertragung mit der DEL-Taste #
#       das Lesen beendet werden.                                   #
#                                                                   #
#       Waehrend des Transfers darf kein als D1 generierter Drucker #
#       angesprochen werden.                                        #
#                                                                   #
#       Fuer einen korrekten Ablauf der Prozedur muss man unter der #
#       Benutzerkennung root oder admin arbeiten.                   #
#                                                                   #
#####################################################################
#                                                                   #
# Uebrigens:                                                        #
#                                                                   #
#       Verzichtet man auf jeglichen Komfort, so braucht die        #
#       Prozedur nur aus den folgenden drei Zeilen bestehen:        #
#                                                                   #
#           exec </dev/rs232                                        #
#           stty 1200 cbreak -echo nl -even odd tandem >/dev/rs232  #
#           cat -u                                                  #
#                                                                   #
#       Voraussetzung dafuer ist, dass beim Aufruf                  #
#       1.  unter root gearbeitet wird,                             #
#       2.  /dev/rs232 schon eingerichtet ist,                      #
#       3.  keine <CR>-Zeichen entfernt werden sollen.              #
#                                                                   #
#####################################################################
```

```
# Nachsehen, ob wir unter root arbeiten (nur dann duerfen wir /etc/passwd
# beschreiben):

if [ ! -w /etc/passwd ]
then
    display "Ablauf nur unter der Kennung root bzw. admin moeglich!"
    exit 1
fi

# Nachsehen, ob die Geraetedatei fuer die Drucker-RS232-Schnittstelle schon
# vorhanden ist. Wenn nicht, wird sie eingerichtet:

if [ ! -r /dev/rs232 ]
then
    /etc/mknod /dev/rs232 c 3 131
    XMX=`fgrep SYST=978 /usr/menus/sabin/header`
    case "$XMX" in
    *9780*) display -n "Schalter S4 (CONAC) bereits in RS232-Stellung? (j/n) > "
            read ANT
            case "$ANT" in
            j)      :;;

            *)      display "Bitte zuerst umschalten!"
                    exit 1;;
            esac;;
    esac
fi

# RS232-SS fuer stdin zuweisen und bis zum Prozedurende eroeffnet halten:

exec </dev/rs232

# RS232-SS einstellen:

stty 1200 cbreak -echo nl -even odd tandem >/dev/rs232

display "Nun wird der Daten-Empfang ueber die RS232-SS gestartet.
Bitte leiten Sie jetzt am Sende-Rechner die Uebertragung ein.
Nach dem Ende des Transfers (am Sende-Rechner erkenntlich) muss das
Empfangen durch Druecken der DEL-Taste beendet werden."

# Vorbereitung fuer das Uebertragungsende (ausgeloest durch DEL-Taste).
# Aus den uebertragenen Daten werden alle Zeichen <CR> entfernt, da manche
# Fremdrechner jede Zeile mit <CR><LF> abschliessen, in SINIX jedoch als
# Zeilenende nur <LF> verwendet wird:

trap 'trap "" 2
      tr -d "\015" <.tmprs232
      rm .tmprs232
      exit 0' 2

# Eigentliche Datenuebertragung (Lesen von /dev/rs232):

cat -u >.tmprs232
```

12.2.2 Programm zum Einlesen von ASCII-Dateien über einen TTY-Kanal

Das folgende Quellprogramm finden Sie auf der Diskette 'SSHB' unter dem Dateinamen gettty.c.

```
static char SCCSID[] = "@(#)gettty.c 1.3 85/06/13";

/*------------------------------------------------------------------------
 * gettty.c
 *
 * Programm zum Lesen von ASCII-Dateien ueber einen TTY-Kanal
 *
 *------------------------------------------------------------------------
 *
 * Immer wieder besteht die Notwendigkeit, Quell-Dateien im ASCII-Format von
 * anderen Rechnern zu uebernehmen.
 * Wenn die Uebernahme ueber Floppy-Disk nicht moeglich ist, bleibt noch die
 * Moeglicheit der direkten Kopplung der beiden Rechner ueber eine der verfueg-
 * baren Standard-Schnittstellen, also ueber V24/V28 oder SS97/V11.
 *
 * Diese Kopplung ist jedoch nicht ganz problemlos, da neben der rein physikali-
 * schen Kopplung (PIN-Belegung, Spezialkabel usw.) auch das Problem der Fluss-
 * Steuerung, der Datei-Ende-Erkennung und der Zeilen-Abschluss-Steuerzeichen
 * geloest werden muss.
 *
 * Fuer die Fluss-Steuerung gibt es folgende Loesungsansaetze:
 * -----------------------------------------------------------
 *
 * - Sehr langsame Uebertragungsrate, z.B. 300 Bd, oder Verzoegerungstimer
 *   nach jedem "Block" von der Groesse des Empfangspuffers:
 *
 *      Dies muss immer als Notloesung fuer kurze Dateien betrachtet werden.
 *      Ausserdem ist das Risiko vorhanden, dass doch Zeichen verlorengehen.
 *
 * - Fluss-Steuerung durch die Steuerzeichen DC3/DC1:
 *
 *      Dies setzt voraus, dass sowohl beim "Sender" als auch beim "Empfaenger"
 *      diese Steuerzeichen bedient werden.
 *      Bei der 9780/9781 trifft dies jedoch mit  SINIX V1.0B in Eingabe-
 *      Richtung nicht zu!
 *      Bei SINIX V1.0C ist diese Einschraenkung behoben.
 *
 *      Fuer den "Sender" ist dies haeufig eine sehr einfache Moeglichkeit, da
 *      meist Drucker-Anschluesse mit diesem Protokoll implementiert sind, und
 *      der "Empfaenger" einfach anstelle des Druckers angeschlossen wird.
 *
 *      Beispiel fuer CP/M-Systeme:     pip lst:=DATEI,eof[t8]
 *                                      (eof ist Datei mit SINIX-EOF-Zeichen)
 *
 *      Beim Empfaenger kann die Datei ebenso einfach empfangen werden, wenn
 *      eine entsprechende Kanal-Datei eingerichtet wurde.
 *      (/etc/mknod /dev/kdatei c 8 105 --> Beispiel fuer SERAC-Kanal-5).
 *
 *      Beispiel fuer SINIX-V-1.0C:  cat /dev/kdatei | tr -d '\015' > DATEI
 *
 *      Bei diesen Beispielen ist wichtig, dass das SINIX-EOF-Zeichen ueber die
 *      Leitung kommt, um die Uebertragung zu beenden. Bei einem Abbruch mit der
 *      DEL-Taste werden naemlich die noch vorhandenen Daten im Empfangspuffer
 *      geloescht und die uebertragene Datei ist daher unvollstaendig.
 *      Ausserdem ist darauf zu achten, dass evtl. eine Korrektur der Zeilen-
 *      ende-Steuerzeichen erforderlich ist. Diese sind bei CP/M z.B. <cr><lf>
 *      und bei SINIX nur <lf>. Eine Umsetzung kann in diesem Fall durch
 *      <tr -d '\015'> wie oben gezeigt erfolgen.
 *      Die Fluss-Steuerung ist jedoch noch keine Gewaehr fuer die richtige
 *      und vollstandige Uebertragung der Daten. Durch "missing-interrupts"
 *      verlorengegangene Zeichen werden so nicht erkannt, dies kann nur auf
 *      Anwendungsebene durch entsprechende Block-Pruef-Zeichen erfolgen.
 *
```

```
* - Fluss-Steuerung durch Protokoll auf Anwender-Ebene:
*
*      Dies setzt voraus, dass beim "Sender" und "Empfaenger" entsprechende
*      Anwender-Programme zur Verfuegung stehen.
*      Abgesehen vom erforderlichen Aufwand hat dies den Vorteil, dass alle
*      logischen Probleme, wie Fluss-Steuerung, Zeilenende-Kriterium und
*      Dateiende-Kriterium geloest werden koennen.
*      Ausserdem koennen noch die Uebertragung transparenter Dateien, die
*      Blockwiederholung im Fehlerfall und entsprechender Bedienungs-Komfort
*      relativ einfach implementiert werden.
*      Beispiele sind das Standard-Programm <uucp> zur Kopplung von unter-
*      schiedlichen UNIX-Systemen und das bei VS 1133 implementierte Programm
*      <ft> zur Kopplung von UNIX-Systemen mit dem Siemens CP/M-Rechner 9753.
*      Letzteres ermoeglicht nach entsprechender Umsetzung auch das Lesen
*      von IBM-BIF-Disketten.
*
*-------------------------------------------------------------------------
* Folgendes Beispiel zeigt prinzipiell das Lesen von Daten ueber einen
* TTY - Kanal unter der Voraussetzung, dass die Fluss-Steuerung mit DC3/DC1
* funktioniert.
* Es wird nochmals darauf hingewiesen, dass dies bei der SINIX-V-1.0B nicht
* der Fall ist, das Programm also reinen Test-Charakter hat.
* Es wird vorausgesetzt, dass eine entsprechende Kanal-Datei mit dem Kommando
* /etc/mknod eingerichtet wurde. Der Name der Kanal-Datei wird abgefragt, um
* Tests mit verschiedenen Kanaelen (Schnittstellen) zu ermoeglichen.
* Der Kanal wird entsprechend der XENIX-V7-TTY-Schnittstelle parametrisiert.
* Die entsprechende Struktur steht in </usr/include/sgtty.h>.
* Um fuer Tests flexibel zu sein, stehen die Kanal-Parameter in Form von
* Schluesselworten in einer Datei, deren Name ebenfalls abgefragt wird.
* Die gelesenen Daten werden nach <stdout> geschrieben und koennen daher ueber
* eine Pipe in eine beliebige Datei umgeleitet werden.
* Nach Beenden des Programmes werden die alten Kanal-Parameter wiederher-
* gestellt.
* Ein vorzeitiger Abbruch durch die DEL-Taste ist moeglich, dies ist auch
* notwendig, wenn kein EOF-Zeichen am Ende der Datei kommt, in diesem Fall
* gehen keine gelesenen Zeichen verloren.
*
*-------------------------------------------------------------------------
*/

#define         EOFL      4                                   /* Datei-Ende */
#define         ESC     0x1b
#define         ERR     (-1)
#define         MAX     128

#include    <stdio.h>
#include    <signal.h>
            <sgtty.h>

int             fdk;                            /* File-Descriptor Kanaldatei */

int             onintr ();

/*-------------------------------------------------------------------------*/
main ()

{
FILE            *fpp, *fopen ();          /* File-Pointer Kanalparameterdatei */
char            kdatei[64];                                 /* Kanal-Datei */
char            pdatei[64];                         /* Kanal-Parameter-Datei */
char            c;                               /* Puffer fuer <read>-Befehl */
int             n;

fprintf (stderr, "\n%c[7m Testprogramm zum Lesen von Daten ueber einen\
 TTY-Kanal: %c[m", ESC, ESC);
fprintf (stderr, "\n\nGeben Sie den Kanal-Datei-Namen ein              : ");
scanf ("%s", kdatei);
if  ((fdk = open (kdatei, 0)) == ERR)
    {
    fprintf (stderr, "\nKanal-Datei <%s> kann nicht geoeffnet werden\n",kdatei);
    exit (1);
    }
fprintf (stderr, "Geben Sie den Namen der Kanal-Parameter-Datei ein : ");
scanf ("%s", pdatei);
if  ((fpp = fopen (pdatei, "r")) == NULL)
    {
    fprintf (stderr, "\nKanal-Parameter-Datei <%s> kann nicht geoeffnet\
 werden\n", pdatei);
    exit (1);
    }
```

```
ssinit (fdk, fpp);                                  /* TTY-Kanal parametrisieren */

putc ('\n', stderr);
sprintf (pdatei, "stty >%s", kdatei); system (pdatei);  /* Parameter anzeigen */
putc ('\n', stderr);

while ((n = read (fdk, &c, 1)) == 1)
    {
    if  (c == EOFL)     break;                                  /* Datei-Ende */
    else                write (1, &c, 1);
    }
if  (n == ERR)          fprintf (stderr, "\nLesefehler !!\n");
else                    fprintf (stderr, "\nDatei-Ende   \n");
ssrein (fdk);
exit (n);
}

/*
 *-------------------------------------------------------------------------
 * ssinit (fdk, fpp)            Parametrisieren eines TTY-Kanals
 * ssrein (fdk)                 Alte Parameter wiederherstellen
 *
 * fdk : File-Descriptor der Kanal-Datei
 * fpp : File-Pointer der Kanal-Parameter-Datei
 *
 *-------------------------------------------------------------------------
 *
 * Beschreibung der Kanal-Parameter-Datei:
 *
 * Die Auswahl der Parameter erfolgt durch folgende Schluesselworte:
 *
 *       COOKED  CBREAK   RAW           Uebertragungs-Modus
 *       1200 ... 38400                 Uebertragungs-Geschwindigkeit
 *       ODD  EVEN  NO                  Paritätsbit
 *
 *-------------------------------------------------------------------------*/

struct  sgttyb  ttyoldp;                            /* TTY-Parameter-Struktur */

/*-------------------------------------------------------------------------*/
ssinit (fdk, fpp)                                   /* TTY-Kanal parametrisieren */
                                                    /*-------------------------*/
int     fdk;                                        /* File-Descriptor Kanal-Datei */
FILE    *fpp;                                       /* File-Pointer Kanal-Parameter-Datei */

{
#define BUF     128                                 /* Max Groesse der Parameterdatei */
#define SMAX     11                                 /* Anzahl der Schluesselwoerter */

int                     d;
register unsigned short i, j, mode, speed, par;
char                    c, rbuf[BUF], *str[SMAX+1];
struct sgttyb           ttypar;

str[0]  = "RAW";                                            /* Schluesselworte */
str[1]      = "CBREAK";
str[2]      = "COOKED";
str[3]  = "1200";
str[4]      = "2400";
str[5]      = "4800";
str[6]      = "9600";
str[7]      = "19200";
str[8]  = "38400";
str[9]  = "NO";
str[10] = "EVEN";
str[11] = "ODD";

                                                    /* Standards :          */
mode = CBREAK;                                      /* CBREAK - Modus       */
speed = B1200;                                      /* 1200 Baud            */
par = ODDP;                                         /* ungerade Paritaet    */
```

```
for (i = 0; i < BUF; i++)                          /* Parameter-Datei einlesen */
    {
    if  ((d = getc (fpp)) == ERR)       break;
    rbuf[i] = d;
    }
for (j = 0; j <= SMAX; j++)
    {
    for (i = 0; i < BUF; i++)
        {
        if  (*str[j] == rbuf[i])        break;
        }
    if  (i == BUF)                      continue;
    while ((c = *(++str[j])) > ' ')
        {
        if  (c != rbuf[++i])            break;
        }
    if  (c > ' ' || rbuf[i+1] > ' ')    continue;
    switch (j)
        {
        case 0  : mode = RAW; break;
        case 1  : mode = CBREAK; break;
        case 2  : mode = 0; break;
        case 3  : speed = B1200; break;
        case 4  : speed = B2400; break;
        case 5  : speed = B4800; break;
        case 6  : speed = B9600; break;
        case 7  : speed = EXTA; break;
        case 8  : speed = EXTB; break;
        case 9  : par = 0; break;
        case 10 : par = EVENP; break;
        case 11 : par = ODDP; break;
        default : break;
        }
    }

ioctl (fdk, TIOCGETP, &ttyoldp);          /* Parameter des TTY-Kanals sichern */

signal (SIGINT, onintr);                                      /* DEL-Taste */
signal (SIGQUIT, SIG_IGN);                                    /* QUIT-Taste */
ttypar.sg_ispeed =  speed;                        /* neue Parameter einstellen */
ttypar.sg_ospeed =  speed;                                    /* Baudrate */
ttypar.sg_kill   =  ttyoldp.sg_kill;
ttypar.sg_erase  =  ttyoldp.sg_erase;
ttypar.sg_flags  =  0;
ttypar.sg_flags  |= mode;                                     /* Modus */
ttypar.sg_flags  |= par;                                      /* Parity */
ttypar.sg_flags  |= TANDEM;                         /* DC1 - DC3 - Protokoll */

ioctl (fdk, TIOCSETP, &ttypar);                      /* TTY-Kanal umschalten */
ioctl (fdk, TIOCEXCL, &ttypar);                      /* Exclusiv-Modus setzen */
}

/*-------------------------------------------------------------------------*/
ssrein (fdk)                /* alte Parameter des TTY-Kanals wiederherstellen */
                            /*-----------------------------------------------*/
int     fdk;                                  /* File-Descriptor Kanal-Datei */

{
ioctl (fdk, TIOCSETP, &ttyoldp);          /* Parameter des TTY-Kanals sichern */
close (fdk);                                        /* Kanal-Datei schliessen */
}

/*-------------------------------------------------------------------------*/
onintr ()                                                     /* DEL - Taste */

{ ssrein (fdk); exit (1); }

/*-------------------------------------------------------------------------*/
```

12.2.3 Programm zum Lesen von ASCII-Dateien über einen TTY-Kanal

Das folgende Quellprogramm finden Sie auf der Diskette 'SSHB' unter dem Dateinamen eing.c.

```
static char SCCSID[] = "@(#)eing.c 1.2 85/05/07";
/* Programm zum Einlesen  von ASCII-Dateien ueber einen tty Kanal des
 * PC-MX.                                                    SINIX 1.0B
 *          **   Minimalversion fuer eilige  **
 * Zum Einlesen muss entsprechend dem gewuenschten Kanal eine Geraetedatei
 * eingerichtet sein.
 * Geraetedatei einrichten  im Dateiverzeichnis  /dev :
 * - einlesen ueber Fbg. CONAC    (Druckeranschluss)
 *   /etc/mknod datlese c 3 1              Geraetedatei "datlese"  einrichten
 * - einlesen ueber Fbg. SERAC    (Schnittst. 97  Kanal 5)
 *   /etc/mknod datlese c 8 105            Geraetedatei "datlese"  einrichten
 *
 *
*/

#define    PMODE  0666
#define    ETX    3
#include   <stdio.h>
#include   <sgtty.h>

main()
{
        int   fdfile, fdtty;
        char  n, fnam[9];
        char  modus;
        printf("       ***************************************************************");

        printf("\nSie wollen eine Datei empfangen ?\n");
        printf("Welchen Dateinamen soll die Datei am PC erhalten ?\n");
        printf("Bitte Dateiname eingeben.\n");

        scanf ("%s", fnam);

        if ((fdfile = creat(fnam,PMODE)) == -1)
        {
                printf ("\n\n%s kann nicht eingerichtet werden\n\007",fnam);
                exit(1);
        }

        if ((fdtty = open("/dev/datlese",2)) == -1)
        {
                printf("\n\nOPEN - Fehler auf /dev/datlese\n\007");
                exit(1);
        }

     ssinit (fdtty);

        while ((read (fdtty, &n, 1)) == 1)
        {
              n &= 0x7f;
                if (n == ETX) break;
                putchar(n);
                if ((write (fdfile, &n, 1)) != 1)
                {
                        printf ("\n\nSchreibfehler auf %s\n\007", fnam);
                        exit(1);
                }
        }
        printf ("\n\nDatei %s uebertragen\n\007", fnam);
        exit(0);
ende:
        printf("\n\nEnde\n\007");
}

ssinit (fdtty)
int fdtty;
{
        struct sgttyb ttypar;

        ttypar.sg_ispeed = B600;
        ttypar.sg_ospeed = B600;
        ttypar.sg_flags = 0;
        ttypar.sg_flags |= RAW;
        ttypar.sg_flags |= TANDEM;

        ioctl (fdtty, TIOCSETP, &ttypar);
}
```

13 Behandlung von Disketten beim PC-X/PC-X10 und PC-MX2

Nach dem SINIX-Standard ist eine Diskette in drei 'Partitions' unterteilt, die mit den Geräteeinträgen /dev/fl0, /dev/fl1 und /dev/fl2 gelesen bzw. beschrieben werden können. Die Beschreibung dieser 'Partitions' ist im SINIX-System-Kern in einer Tabelle enthalten. Charakteristische Größen für diese 'Partitions' sind z.B.: die Anzahl der Zylinder, die Anzahl der Sektoren pro Spur, die Anzahl der Bytes pro Sektor usw.

13.1 Gerätedateien

Außer den eben genannten Standard-Geräteeinträgen, sind vier weitere Einträge möglich:

/dev/fl3, /dev/fl4, /dev/fl5 und /dev/fl6.

/dev/fl3 - /dev/fl5 sind festgelegte Geräteeinträge,
/dev/fl6 ist ein Geräteeintrag, mit dem es dem Anwender möglich ist, den 'Controller' so zu programmieren, daß er verschiedene Disketten-Formate lesen und schreiben kann.

	fl0	fl1	fl2	fl3	fl4	fl5
Startzylinder	0	1	4	0	0	0
Anzahl der Zylinder	1	3	73	80	40	40
Zylinderoffset	1	1	1	1	2	2
Sektorgröße (B)	128	256	256	512	512	512
Sektoren/Spur	16	16	16	9	9	9
Anzahl Seiten: PC-X/PC-X10/PC-MX PC-MX2	 2 2	 2 2	 2 2	 2 2	 1 1	 2 2
Aufzeichnungdichte	0	1	1	1	1	1
Gap-Größe (B)	16	32	32	32	32	32

Dabei bedeutet

bei der Aufzeichnungsdichte — 0 = single density (fm) 1 = double density (mfm)

beim Zylinderoffset — 1 = 96 tpi 2 = 48 tpi

Folgende Gerätedateien sollten vorhanden sein, wenn Sie mit den Fremdformaten arbeiten wollen. Die Gerätedeteien müssen mit den angegebenen Zugriffsrechten versehen sein.

```
brw-rw-rw- 1 root        0,  3 Mar 22 13:34 /dev/fl3
brw-rw-rw- 1 root        0,  4 Mar 22 13:34 /dev/fl4
brw-rw-rw- 1 root        0,  5 Mar 22 13:34 /dev/fl5
brw-rw-rw- 1 root        0,  6 Mar 22 13:34 /dev/fl6
crw-rw-rw- 1 root        0,  6 Mar 22 13:36 /dev/rfl6
```

Diese Gerätedateien kann der Systemverwalter mit folgenden Kommandos einrichten:

```
/etc/mknod /dev/fl3 b 0 3
/etc/mknod /dev/fl4 b 0 4
/etc/mknod /dev/fl5 b 0 5
/etc/mknod /dev/fl6 b 0 6
/etc/mknod /dev/rfl6 c 0 6

chmod a+w /dev/fl3 /dev/fl4 /dev/fl5 /dev/fl6 /dev/rfl6
```

Nach dem Laden des Systems enthalten alle Einträge der Tabelle für /dev/fl6 den Wert Null. Jeder Anwender kann nun mit Hilfe eines C-Programmes die Inhalte dieser Tabelle für /dev/fl6 lesen und neu beschreiben.

Da jedoch diese Tabelle im System nur einmal vorhanden ist, gelten die evtl. veränderten Inhalte für alle Anwender, bis die Tabelle erneut verändert wird.

13.2 Disketten bearbeiten

Folgende Informationen, im Zusammenhang mit dem ioctl-Kommando und /dev/rfl6, sind nötig, um dem System das Format der Diskette bekanntzugeben.

```
#define     FLSET    (('S'<<8)|0)
#define     FLGET    (('G'<<8)|0)

struct ptab {
    int p_cylorg;   /* Anfangszylinder */                              (0)
    int p_cylct;    /* Anzahl der Zylinder */
    int p_cyloff;   /* Versatz zwischen Zylindern (1=96TPI, 2=48TPI) */ (1)
    int p_secsiz;   /* Anzahl der Bytes pro Sektor */
    int p_sectrk;   /* Anzahl der Sektoren pro Spur */
    int p_sides;    /* Anzahl der Seiten: 1 oder 2 */
    int p_mfm;      /* 1 bei MFM (doppelte Dichte,sonst 0 (FM) */
    int p_seclen;   /* Code für die Sektorlänge (0, 1, 2 oder 3) */    (2)
    int p_blkct;    /* Anzahl der BSIZE-Byte-Blöcke */                 (3)
    int p_gpl3;     /* Länge der Blocklücke (GAP) */
};
```

Erklärungen

(0) Erster zu lesender bzw. zu schreibender Zylinder. Der erste Zylinder auf der Diskette hat die Nummer Null.

(1) Dieser Wert muß auf 2 gesetzt werden, wenn eine Diskette mit 40 Zylindern (48 tpi) gelesen werden soll.

Hinweis

Disketten mit 40 Zylindern können nicht beschrieben, sondern nur gelesen werden.

(2)

Sektorenlänge	Code
128	0
256	1
512	2
1024	3

(3) Blockzahl = (Anzahl der Zylinder x Anzahl der Sektoren pro Spur x Anzahl der Seiten x Anzahl der Bytes pro Sektor) / BSIZE Die Größe von BSIZE ist in param.h festgelegt (1024).

13.2.1 Beispiele

Es folgen zwei Beispielprogramme, die das Lesen und Beschreiben der Tabelle zeigen. Diese Programme finden Sie auch auf der Diskette 'SSHB'.

Beispiel 1:

Mit diesem Programm und den oben beschriebenen Strukturen und Definitionen kann der Benutzer die Parameter, die für /dev/fl6 gesetzt werden sollen, in die Tabelle eintragen.

```
static char SCCSID[] = "@(#)fl6parset.c 1.1 85/05/02";

#define         BSIZE           1024L

#define         FLSET           (('S'<<8)|0)
#define         FLGET           (('G'<<8)|0)

struct ptab {
    int p_cylorg;       /* Anfangszylinder */
    int p_cylct;        /* Anzahl der Zylinder */
    int p_cyloff;       /* Versatz zwischen Zylindern (1=96TPI, 2=48TPI) */
    int p_secsiz;       /* Anzahl der Bytes pro Sektor */
    int p_sectrk;       /* Anzahl der Sektoren pro Spur */
    int p_sides;        /* Anzahl der Seiten: 1 oder 2 */
    int p_mfm;          /* 1 bei MFM (doppelte Dichte), sonst 0 (FM) */
    int p_seclen;       /* Code fuer die Sektorlaenge (0, 1, 2 oder 3) */
    int p_blkct;        /* Anzahl der BSIZE-Byte Bloecke */
    int p_gpl3;         /* Laenge der Blockluecke (Gap) */
};

struct ptab ptab = {
        0,                              /* Anf.zyl. */
        80,                             /* Zyl.anz. */
        1,                              /* Zyl.abstand */
        512,                            /* Bytes/Sektor */
        9,                              /* Sektoren/Spur */
        2,                              /* Seiten */
        1,                              /* Dichte */
        2,                              /* Sektorlaengencode */
        (int) ((80L * 512L * 9L * 2L) / BSIZE),         /* Anzahl Bloecke */
        0,                              /* Luecke (nicht verwendet) */
};

main()
{
        int fd;

        if ((fd = open ("/dev/rfl6", 2)) == -1)
        {
                perror ("open /dev/rfl6");
                exit(1);
        }

        if (ioctl (fd, FLSET, (char *) &ptab) == -1)
        {
                perror ("FLSET-ioctl /dev/rfl6");
                exit(1);
        }

        close (fd);
}
```

Beispiel 2:

Mit diesem Programm und den oben beschriebenen Strukturen und Definitionen kann der Benutzer der Tabelle /dev/fl6 lesen.

```
static char SCCSID[] = "@(#)fl6parget.c 1.1 85/05/02";

#define         FLSET           (('S'<<8)|0)
#define         FLGET           (('G'<<8)|0)

struct ptab {
    int p_cylorg;       /* Anfangszylinder */
    int p_cylct;        /* Anzahl der Zylinder */
    int p_cyloff;       /* Versatz zwischen Zylindern (1=96TPI, 2=48TPI) */
    int p_secsiz;       /* Anzahl der Bytes pro Sektor */
    int p_sectrk;       /* Anzahl der Sektoren pro Spur */
    int p_sides;        /* Anzahl der Seiten: 1 oder 2 */
    int p_mfm;          /* 1 bei MFM (doppelte Dichte), sonst 0 (FM) */
    int p_seclen;       /* Code fuer die Sektorlaenge (0, 1, 2 oder 3) */
    int p_blkct;        /* Anzahl der BSIZE-Byte Bloecke */
    int p_gpl3;         /* Laenge der Blockluecke (Gap) */
};

struct ptab ptab;

main()
{
        int fd;

        if ((fd = open ("/dev/rfl6", 2)) == -1)
        {
                perror ("open /dev/rfl6");
                exit(1);
        }

        if (ioctl (fd, FLGET, (char *) &ptab) == -1)
        {
                perror ("FLGET-ioctl /dev/rfl6");
                exit(1);
        }

        printf("Anfangszylinder: .... %d\n", ptab.p_cylorg);
        printf("Zylinderanzahl: ..... %d\n", ptab.p_cylct);
        printf("Zylinderabstand: .... %d TPI\n", ptab.p_cyloff == 1 ? 96 : 48);
        printf("Sektorlaenge: ....... %d\n", ptab.p_secsiz);
        printf("Sektoren/Spur: ...... %d\n", ptab.p_sectrk);
        printf("Seitenanzahl: ....... %d\n", ptab.p_sides);
        printf("Schreibdichte: ...... %s\n", ptab.p_mfm ? "doppelt (MFM)" : "einfach (FM)");
        printf("Gesamtkapazitaet: ... %d KB\n", ptab.p_blkct);

        close (fd);
}
```

13.2.2 Beschreibung einiger bekannter Disketten-Formate

	Siemens PC-D, Altos (fl3)	IBM-PC (fl4)	IBM-PC (fl5)	IBM-PC	IBM-PC	DEC RAINBOW
Startzylinder	0	0	0	0	0	0
Anzahl der Zyl.	80	40	40	40	40	80
Zylinderoffset	1	2	2	2	2	1
Sektorgröße (B)	512	512	512	512	512	512
Sektoren/Spur	9	9	9	8	8	10
Anzahl Seiten	2	1	2	1	2	1
Aufzeichnungdichte	1	1	1	1	1	1
Gap-Größe (B)	32	32	32	?	?	?

Wichtige Hinweise

Die hier beschriebene ioctl-Schnittstelle kann nicht langfristig garantiert werden. In künftigen SINIX-Versionen könnte sie abgeändert werden.

Formatieren von Disketten in SINIX-Fremdformaten ist nur auf dem PC-X möglich (vgl. Beschreibung von /etc/flformat).

13.2.3 Disketten-Formatierprogramm für den PC-X

Aufrufformat

```
/etc/flformat [-s n] [-b n] [-f (E/N)] [-pw]
```

Beschreibung

flformat formatiert eine Diskette und führt eine Lesekontrolle durch. Ohne Parameter aufgerufen impliziert flformat das SINIX-Standard-Diskettenformat (fl0, fl1, fl2).

Für andere Formate:

-s n n = Zahl der Sektoren/Spur

-b n n = Zahl der Bytes/Sektor

Die möglichen Kombinationen der beiden Optionen sind:

Sektoren/Spur	Bytes/Sektor
4	1024
8	512
9	512
10	512
16	256

-f (E/N)

Formatieren nach dem gewählten Format:

E = ECMA Standard (d.h. erster Zylinder fm). Dies gilt nur, falls 256 Bytes/Sektor und 16 Sektoren/Spur (3 Partitions) gewählt werden.

N = Neuer Standard (d.h. die ganze Diskette mfm)

-pw Außer der Lesekontrolle wird nochmals eine Schreib- und Lesekontrolle ausgeführt.

Bemerkungen

Dieses Formatierprogramm ist in SINIX V1.0B für den PC-X enthalten und ist **nur** auf dem PC-X ablauffähig.

Zur Schreib- und Lesekontrolle wird /dev/rfl6 auf die gewählten Parameter gesetzt.

Zur Zeit funktioniert die Kombination 10 x 512 nicht richtig (30.11.84).

13.2.4 Disketten lesen

Sind die Vorarbeiten (Gerätedatei einrichten, evtl. fl6-Parameter setzen) geleistet, können Disketten, die diesen Formaten entsprechen, physikalisch gelesen werden, z.B.:

```
cat /dev/flx          oder
xd /dev/flx           Disketteninhalt sedezimal ausgeben oder
cp /dev/flx datei     Disketteninhalt in eine Datei kopieren
```

Für Disketten, die logisch im tar-Format eingeteilt sind, ist dann auch über /dev/flx ein logisches Lesen möglich (tar xf /dev/flx, vgl. SINIX, Buch 1).

Für andere logische Formate ist es notwendig, ein Programm zu erstellen, das den Disketteninhalt entsprechend dem logischen Format aufbereitet.

MS-DOS-Disketten können Sie mit dem Software-Produkt **TRADOS** bearbeiten. Dieses Produkt ist für PC-X, PC-MX, und PC-MX2 erhältlich.

Warnung

Manche Fremdsysteme beginnen in jeder Spur mit Sektor Nr. 0 .
SINIX beginnt immer mit Sektor Nr.1, d.h. beim Lesen einer solchen Fremddiskette gehen die Daten von Sektoren mit der Nr.0 verloren.

A Anhang

Die mitgelieferte Diskette 'SSHB' ist im tar-Format erstellt. Sie kann nicht mit der Menü-Funktion 'Installation von Softwareprodukten' behandelt werden.
Die gewünschten Dateien müssen explizit mit dem tar- bzw. far-Kommando eingelesen werden.

Beispiel

```
far x c_*
```

Es werden die Dateien c_get.c, c_get.o, c_test.c, c_test.o, c_test und c_make eingelesen.

Hinweis

- Für die Programmbeispiele auf der mitgelieferten Diskette 'SSHB' wird keine Gewährleistung übernommen!
- Die mitgelieferte Diskette enthält Objektmodule und lauffähige Programme für PC-X/PC-X10 und PC-MX2.
  PC-X/PC-X10 und PC-MX2 sind nicht objektcodekompatibel.

A.1 Inhaltsverzeichnis der Diskette

Folgende Dateien sind auf der mitgelieferten Diskette 'SSHB' enthalten:

```
r--r--r-- 14/1     4818 Jun 13 16:39 1985 c_get.c
rw-r--r-- 14/1     1416 Jun 13 16:39 1985 c_get.o
r-xr-xr-x 14/1      173 May 23 14:50 1985 c_make
rwxr-xr-x 14/1     9400 Jun 13 16:39 1985 c_test
r--r--r-- 14/1     2349 May 23 14:50 1985 c_test.c
rw-r--r-- 14/1     2432 May 23 14:52 1985 c_test.o
rwxr-xr-x 14/1     6334 Jun 13 16:35 1985 chcode
r--r--r-- 14/1     3224 Jun 13 16:34 1985 chcode.c
rwxr-xr-x 14/1     7920 May  7 18:54 1985 eing
r--r--r-- 14/1     1879 May  7 18:53 1985 eing.c
rwxr-xr-x 14/1     6540 May  2 17:32 1985 fl6parget
r--r--r-- 14/1     1502 May  2 17:31 1985 fl6parget.c
rwxr-xr-x 14/1     1972 May  2 17:33 1985 fl6parset
r--r--r-- 14/1     1598 May  2 17:31 1985 fl6parset.c
r-xr-xr-x 14/1     6069 Jun 13 16:37 1985 getrs232
rwxr-xr-x 14/1     9500 Jun 13 16:43 1985 gettty
r--r--r-- 14/1    12135 Jun 13 16:42 1985 gettty.c
rw------- 14/1    12173 Oct 14 18:12 1986 ioctl.c
rw------- 14/1    17219 Nov 17 08:29 1986 lp9012.c
rw------- 14/1     2388 Oct 14 18:13 1986 lpFREMD.c
rw------- 14/1    10583 Nov 17 17:53 1986 pcmx2/backakt.o
rw------- 14/1     4947 Nov 17 17:53 1986 pcmx2/backend.o
```

```
rw------- 14/1     3339 Nov 17 17:54 1986 pcmx2/backprot.o
rw------- 14/1     4770 Aug  7 14:29 1986 pcmx2/backend.h
rw------- 14/1     2718 Aug  7 14:29 1986 pcmx2/backmeld.h
rw------- 14/1     2209 Aug  7 14:29 1986 pcmx2/backprot.h
rw------- 14/1     1518 Aug  7 14:29 1986 pcmx2/excode.h
rw------- 14/1     1519 Aug  7 14:25 1986 pcmx2/files.h
rw------- 14/1    17219 Nov 17 08:29 1986 pcmx2/lp9012.c verweist auf lp9012.c
rw------- 14/1      260 Nov 17 08:29 1986 pcmx2/makefile
rw------- 14/1     6220 Aug 11 14:15 1986 pcx/backakt.o
rw------- 14/1     4770 Aug  7 14:29 1986 pcx/backend.h verweist auf pcmx2/backend.h
rw------- 14/1     3884 Aug 11 15:06 1986 pcx/backend.o
rw------- 14/1     2718 Aug  7 14:29 1986 pcx/backmeld.h verweist auf pcmx2/backmeld.h
rw------- 14/1     2209 Aug  7 14:29 1986 pcx/backprot.h verweist auf pcmx2/backprot.h
rw------- 14/1     2828 Aug 11 16:59 1986 pcx/backprot.o
rw------- 14/1     1518 Aug  7 14:29 1986 pcx/excode.h verweist auf pcmx2/excode.h
rw------- 14/1     1519 Aug  7 14:25 1986 pcx/files.h verweist auf pcmx2/files.h
rw------- 14/1      260 Nov 17 08:29 1986 pcx/makefile verweist auf pcmx2/makefile
rw------- 14/1    17219 Nov 17 08:29 1986 pcx/lp9012.c verweist auf lp9012.c
r--r--r-- 14/1     2601 May  7 18:34 1985 tbuild.c
rw-r--r-- 14/1      984 May  7 18:35 1985 tbuild.o
r--r--r-- 14/1     9232 Oct  9 14:18 1985 termio.h
r-xr-xr-x 14/1      156 May  3 11:28 1985 tmake
rwxr-xr-x 14/1     8274 May  7 18:35 1985 txample
r--r--r-- 14/1     4641 May  7 18:34 1985 txample.c
rw-r--r-- 14/1     2536 May  7 18:35 1985 txample.o
rwxr-xr-x 14/1     7936 Jun 13 16:44 1985 zed
r--r--r-- 14/1    20185 Jun 13 16:43 1985 zed.c
```

A.1.1 Inhalt der Dateien

tbuild.c	Quellprogramm mit den Routinen tbuild und txtract
tbuild.o	daraus erzeugtes Objektmodul
txample.c	Quelle eines Beispielprogramms zur Verwendung von tbuild und txtract
txample.o	daraus erzeugtes Objektmodul
txample	lauffähiges Programm, aus tbuild.o und txample.o erzeugt
tmake	Prozedur zum Erzeugen von tbuild.o, txample.o und txample
c_get.c c_get.o c_test.c c_test.o c_test c_make	Diese Dateien entsprechen in etwa den Dateien tbuild.c,....,tmake, basieren aber auf verbesserten Routinen zur Verarbeitung von Eingabesequenzen (c_init, c_get).

chcode.c — Quellprogramm zum Setzen der CH-CODE-Taste

chcode — daraus erzeugtes lauffähiges Programm

lp9012.c — Quellprogramm eines einfachen Backends für den Drucker 9012

lp9012 — daraus erzeugtes lauffähiges Programm

lpFREMD.c — Quellprogramm eines einfachen Backends für einen Fremd-Drucker

lpFREMD — daraus erzeugtes lauffähiges Programm

```
pcmx2/backakt.o  ┐
pcmx2/backend.o  ├  Objektmodule für Drucker-Backends
pcmx2/backprot.o ┘  auf PC-MX2 und PC-MX4

pcx/backakt.o    ┐
pcx/backend.o    ├  Objektmodule für Drucker-Backends
pcx/backprot.o   ┘  auf dem PC-X/PC-X10
```

Achtung
Objektcode des PC-X/PC-X10 läuft nicht auf dem PC-MX2 und umgekehrt.

```
pcmx2/backend.h  ┐
pcx/backend.h    │
                 │
pcmx2/backmeld.h │
pcx/backmeld.h   │
                 │
pcmx2/backprot.h ├  Hinweis
pcx/backprot.h   │  Der Quellcode ist für PC-X/PC-X10 und PC-MX2
                 │
pcmx2/excode.h   │
pcx/excode.h     │
                 │
pcmx2/files.h    │
pcx/files.h      ┘
```

pcmx2/makefile
pcx/makefile

Steuerdatei für *make*, mit dieser Datei kann *make* lauffähige Drucker-Backends erzeugen.

fl6parset.c — Quellprogramm zum Setzen von Diskettenformat-Parametern

fl6parset	daraus erzeugtes lauffähiges Programm
fl6parget.c	Quellprogramm zum Lesen von Diskettenformat-Parametern
fl6parget	daraus erzeugtes lauffähiges Programm
ioctl.c	Quellprogramm für ein verbessertes stty-Kommando nach XENIX System III
getrs232	einfache Shell-Prozedur zum Empfangen von ASCII-Daten über die Schnittstelle RS232
eing.c	einfaches Quellprogramm zum Empfangen von ASCII-Daten
eing	daraus erzeugtes lauffähiges Programm
gettty.c	Quellprogramm zum Lesen von ASCII-Dateien über einen TTY-Kanal
gettty	daraus erzeugtes lauffähiges Programm
zed.c	Quellprogramm für einen einfachen Shell-ähnlichen Kommandointerpreter mit Editierfunktionen
zed	daraus erzeugtes lauffähiges Programm

A.1.2 Auflistung der Quellprogramme

A.1.2.1 tbuild.c

```
static char SCCSID[] = "@(#)tbuild.c 1.2 85/05/07";
/* Mods:
 *      m01     jgr     Endekriterium in Sequenzpuffer: '}200' statt ' '.
 */

/*
 *       Funktionen tbuild und txtract
 */

/*
 *       Ast des binaeren Baums
 */

typedef struct tnode {           /* Knoten */
       struct tnode *tnext;     /* Zeiger naechstes Element */
       struct tnode *talt;      /* Zeiger alternatives Element */
       int           tkey;      /* Funktionskode */
} *node;

/*
 *       Die Umsetzschiene und der Sequenzpuffer
 */

static node tbase[128];
static char tseqbuf[20], *tbp;

/*
 *       Anfordern vom Speicherplatz fuer einen Ast
 */

static node
getnode()
{
       register node tp;
       node malloc();

       if ( tp = malloc(sizeof (struct tnode)) ) {
              tp->tnext = (node)0;
              tp->talt  = (node)0;
              tp->tkey  = 0;
       }
       return tp;
}

/*
 *       Erstellen eines binaeren Baums
 */

tbuild(path, fkey)
register char *path;
int fkey;
{
```

```
        register node tp;

        if ( path == (char *)0 || *path == ' ' )
                return 0;
        if ( ( tp = tbase[*path & 0177] ) == (node)0 ) {
            if ( ( tp = getnode() ) == (node)0 )
                return -1;
        }
        tbase[*path & 0177] = tp;
        while ( *++path ) {
            if ( tp->tnext ) {
                if ( tp->tkey == *path )
                    tp = tp->tnext;
                else {
                    while ( tp->tkey != *path ) {
                        if ( tp->talt )
                            tp = tp->talt;
                        else {
                            if ((tp->talt=getnode()) == (node)0)
                                return -1;
                            tp = tp->talt;
                            tp->tkey = *path;
                            if ((tp->tnext=getnode()) == (node)0)
                                return -1;
                            break;
                        }
                    }
                    if ( tp->tnext )
                        tp = tp->tnext;
                }
            } else {
                if ( ( tp->tnext = getnode() ) == (node)0 )
                    return -1;
                tp->tkey = *path;
                tp = tp->tnext;
            }
        }
        tp->tkey = fkey;
        return 0;
}

/*
 *       Dekodieren einer Eingabesequenz
 */

txtract(tnextchar)

int (*tnextchar)();
{
        register node tp;
        register int c;

        if ( tbp ) {
                if ( *tbp != '\200' )                          /* m01 */
                        return *tbp++;
        }
```

```
        tbp = tseqbuf;
        c = (*tnextchar)();
        if ( c == -1 || ( tp = tbase[c & 0177] ) == (node)0 ) {
                tbp = (char *)0;
                return c;
        }
        *tbp++ = c;
        while ( tp->tnext ) {
                if ( ( c = (*tnextchar)() ) == -1 ) {
                        *tbp = '}200';                          /* m01 */
                        tbp = tseqbuf;
                        return *tbp++;
                }
                *tbp++ = c;
                if ( c == tp->tkey )
                        tp = tp->tnext;
                else {
                        while ( c != tp->tkey ) {
                                if ( tp->talt )
                                        tp = tp->talt;
                                else {
                                        *tbp = '}200';          /* m01 */
                                        tbp = tseqbuf;
                                        return *tbp++;
                                }
                        }
                        tp = tp->tnext;
                }
        }
        tbp = (char *)0;
        return tp->tkey;
}
```

A.1.2.2 txample.c

```
static char SCCSID[] = "@(#)txample.c 1.2 85/05/07";

/*
 * Beispielprogramm zur Demonstration der Funktionen tbuild und txtract.
 *
 * (Un-)Sinn des Programms: Beim Druecken der Tasten F1, F2, F3 oder HELP
 * kommt eine entsprechende Meldung, mit der END-Taste wird es beendet.
 * DEL bewirkt einen geregelten Abbruch (mit "normaler" Terminaleinstellung),
 * CTRL-\ erzeugt vorher noch einen core dump. Alle anderen Tasten haben
 * keine Wirkung. Allerdings: drueckt man z.B. ESC F1, so wird nicht wie
 * erwartet die Meldung "F1" ausgegeben (warum?)!
 *
 * Das Programm zeigt weiterhin, wie man sich termcap-Strings besorgen
 * kann und wie man erreicht, dass jedes eingetippte Zeichen sofort ans
 * lesende Programm uebergeben wird.
 *
 * Mit der Prozedur tmake kann das Programm auf einfache Weise uebersetzt
 * und gebunden werden.
 */

#include        <stdio.h>
#include        <sgtty.h>
#include        <signal.h>

#define END     0x04

#define F1      256
#define F2      257
#define F3      258
#define HELP    300

#define tF1     "P1"
#define tF2     "P2"
#define tF3     "P3"
#define tHELP   "10"

static int pid, savflags;
static struct sgttyb sy;
static char tcbuf[1024], cbuf[100], *pcbuf, *f1, *f2, *f3, *help;

int leave(), dump(), nextc();
char *getenv(), *tgetstr();

/*****************************************************************************/

main()
{
      register int c;

      if (isatty(0) == 0)   /* Terminal zugewiesen? */
              error("Kein Terminal zugewiesen (stdin)!");

      if (tgetent(tcbuf,getenv("TERM")) != 1) /* termcap-Eintrag besorgen */
              error("Fehler bei tgetent!");

      /* cbuf mit termcap-Strings fuellen: */
      pcbuf = cbuf;
      if ((f1=tgetstr(tF1,&pcbuf)) == NULL)           /* F1 */
```

```
        error("Kein F1-Eintrag!");
    if ((f2=tgetstr(tF2,&pcbuf)) == NULL)          /* F2 */
        error("Kein F2-Eintrag!");
    if ((f3=tgetstr(tF3,&pcbuf)) == NULL)          /* F3 */
        error("Kein F3-Eintrag!");
    if ((help=tgetstr(tHELP,&pcbuf)) == NULL)      /* HELP */
        error("Kein HELP-Eintrag!");

    pid = getpid();

    if (gtty(0,&sy) != 0)          /* Terminal-Status besorgen */
        error("Fehler bei gtty!");
    savflags = sy.sg_flags;        /* und sichern */

    signal(SIGINT,leave);
    signal(SIGQUIT,dump);
    signal(SIGTERM,leave);

    sy.sg_flags = (sy.sg_flags | CBREAK) & ~ECHO;
    if (stty(0,&sy) != 0)          /* cbreak und non-echo setzen */
        error("Fehler bei stty!");

    /* Erstellen des Baums: */

    if (tbuild(f1,F1) == -1 )
        error("Ueberlauf fuer F1 bei tbuild!");
    if (tbuild(f2,F2) == -1 )
        error("Ueberlauf fuer F2 bei tbuild!");
    if (tbuild(f3,F3) == -1 )
        error("Ueberlauf fuer F3 bei tbuild!");
    if (tbuild(help,HELP) == -1 )
        error("Ueberlauf fuer HELP bei tbuild!");

    /* Rueckgewinnung von Eingabesequenzen: */

    for ( ; ; ) {

        switch (c = txtract(nextc)) {

            case END:       reset();
                            exit(0);
            case F1:        printf("F1}n");
                            break;
            case F2:        printf("F2}n");
                            break;
            case F3:        printf("F3}n");
                            break;
            case HELP:      printf("HELP}n");
                            break;
            default:        break;
        }
    }
}

/*****************************************************************************/

/*
 *      Da getchar ein Macro ist,
 *      muss es in eine Funktion verpackt werden
 */
```

```
static nextc()
{
       return getchar();
}

/*****************************************************************************/

static leave()          /* INT- und TERM-Routine */
{
      reset();
      exit(0);
}

/*****************************************************************************/

static dump()           /* QUIT-Routine */
{
      reset();
      kill(pid,SIGQUIT);       /* Selbstmord mit Dump */
}

/*****************************************************************************/

static reset()          /* normalen Terminal-Status wiederherstellen */
{
      sy.sg_flags = savflags;
      if (stty(0,&sy) != 0)
             error("Fehler bei stty!");
}

/*****************************************************************************/

static error(s)         /* Fehlermeldung und Ende */
char *s;
{
      fprintf(stderr,"%s}n",s);
      exit(1);
}

/*****************************************************************************/

/* Aufloesung der "Warum-Frage" aus dem Anfangs-Kommentar:
 *
 * Die Funktion txtract arbeitet so, dass eine Eingabesequenz dann nicht
 * erkannt wird, wenn ihr eine "unvollstaendige" Eingabesequenz vorausge-
 * gangen ist.
 *
 * Beispiel:    Eingabe: ESC F1 (= ESC ESC @, ASCII-Codes 27 27 64).
 *              Ergebnis: txtract liefert 27 27 64 und nicht 27 256.
 *
 * Genausowenig koennen mit der Funktion tbuild zwei Sequenzen im Baum ein-
 * getragen werden, von denen die eine im Anfang der anderen enthalten ist.
 */
```

A.1.2.3 c_get.c

```
static char SCCSID[] = "@(#)c_get.c 1.3 85/06/13";

/*
 *    terminal input handling routines
 *      Author: mgs
 *      Modified by wjg
 */

#include <stdio.h>

/* values returned by c_init */
#define MULTIPLE 2
#define SUCCESS  1
#define NOMEM        0

#define MAXCHAR 256          /* number of initial characters */
#define MAXSEQL 128          /* maximal length of a sequence */
#define MASK 0377            /* mask for chars read. */

typedef struct t_node *node;

struct t_node {                        /* definition of an input node: */
      node t_next;            /* pointer to next node */
      node t_alt;             /* pointer to alternate node */
      int t_branch;           /* character sequence may differ in */
      int t_key;              /* code to return */
};

static node tbase[MAXCHAR];   /* baseline for sequence recognition */
static char tseqbuf[MAXSEQL]; /* buffer for sequence until recognition */
static char *tbp;             /* pointer into tseqbuf */

/*
 * static node
 * get_node()
 *
 *    DESCRIPTION:
 *            Allocates memory for a sequence node.
 *
 *    RETURNS:
 *            pointer to node allocated,
 *            (node)0 for no memory.
 */

static node
get_node()
{
      register node tp;
      node malloc();

      if (tp = malloc(sizeof(struct t_node)))
      {
            tp->t_next = (node)0;
            tp->t_alt = (node)0;
            tp->t_branch = 0;
            tp->t_key = 0;
      }
      return tp;
}

/*
 * int
 * c_init(seq, code)
```

```
 * char *seq;
 * int code;
 *
 *    DESCRIPTION:
 *            initialize input processor to return "code" whenever "seq" is
 *            encountered in input stream.
 *
 *    RETURNS:
 *            MULTIPLE for duplicate entry
 *              SUCCESS for successful initialization
 *            NOMEM for no more memory
 *
 */

int
c_init(seq, code)
register char *seq;
int code;
{
      register node tp;

      if (seq == (char *)0 || *seq == '}0')
              return SUCCESS;

      if ((tp = tbase[*seq & MASK]) == (node)0)
              if ((tp = get_node()) == (node)0)
                      return NOMEM;

      tbase[*seq & MASK] = tp;

      while (*++seq)
      {
              if (tp->t_next)
              {
                      if (tp->t_branch == *seq)
                              tp = tp->t_next;
                      else
                      {
                              while (tp->t_branch != *seq)
                              {
                                      if (tp->t_alt)
                                              tp = tp->t_alt;
                                      else
                                      {
                                              if ((tp->t_alt = get_node()) == (node)0)
                                                      return NOMEM;
                                              tp = tp->t_alt;
                                              tp->t_branch = *seq;
                                              if ((tp->t_next = get_node()) == (node)0)
                                                      return NOMEM;
                                              break;
                                      }
                              }
                              if (tp->t_next)
                                      tp = tp->t_next;
                      }
              }
              else
              {
                      if ((tp->t_next = get_node()) == (node)0)
                              return NOMEM;
                      tp->t_branch = *seq;
                      tp = tp->t_next;
              }
      }
      if (tp->t_key && tp->t_key != code)
```

```
                return MULTIPLE;
        tp->t_key = code;
        return SUCCESS;
}

/*
 * int
 * c_get(nextc)
 * int (*nextc)();
 *
 *    DESCRIPTION:
 *            read via nextc the next characters and check whether they
 *            construct a sequence. When encountering an illegal character
 *            in a sequence stop scanning and return characters one by one.
 *            "nextc" is the function used to read one character at a time
 *            and must be user supplied. It should return EOF for end of
 *            file, otherwise the character read. It is passed one parameter
 *            which is set (== 1) when "nextc" was called from within a
 *            sequence, cleared (== 0) otherwise. If set, "nextc" may return
 *            EOF to signalize an illegal sequence (i.e. timeout). "c_get"
 *            will then return all characters read up to that point.
 *
 *    RETURNS:
 *            Character or sequence code.
 *
 *    WARNINGS:
 *            EOF is only returned when it is the first character in a seq.
 *
 */

char *strcpy();
#define l_shift(to, from) tbp = strcpy(to, from)

int
c_get(nextc)
int (*nextc)();
{
        register node tp;               /* current node */
        register int c;                 /* current character */
        node remtp = (node)0;           /* last remebered valid node */
        char *remtbp = (char *)0;       /* and corresponding buffer pointer */

        /* if there are any characters buffered, return buffered chars */
        c = _m_nextc(nextc, 0);

        if (c == EOF || (tp = tbase[c & MASK]) == (node)0)
        {       /* not sequence introducer --> return char read */
                l_shift(tseqbuf, tbp);
                return c;
        }

        while (tp->t_next)
        {
                if (tp->t_key)
                {
                        remtp = tp;
                        remtbp = tbp;
                }
                if ((c = _m_nextc(nextc, 1)) == EOF)
                {       /* interrupted sequence: */
                        *--tbp = '\0';
                        if (remtp == tp)
                                break;
                        if (remtp)
                        {
                                l_shift(tseqbuf, remtbp);
```

```
                        return remtp->t_key;
                }
                tbp = tseqbuf;
                return *tbp++;
        }
        if (c == tp->t_branch)
                tp = tp->t_next;
        else
        {
                while (c != tp->t_branch)
                {
                        if (tp->t_alt)
                                tp = tp->t_alt;
                        else
                        {
                                if (remtp)
                                {
                                        l_shift(tseqbuf, remtbp);
                                        return remtp->t_key;
                                }
                                tbp = tseqbuf;
                                return *tbp++;
                        }
                }
                tp = tp->t_next;
        }
    }
    tbp = (char *)0;
    return tp->t_key;
}

static int
_m_nextc(nextc, inseq)
int (*nextc)();
int inseq;
{
    register int c;

    if (tbp && *tbp)
    {
        c = *tbp & MASK;
        *tbp++ = '\0';
    }
    else
    {
        if (!tbp)
                tbp = tseqbuf;
        *tbp++ = c = (*nextc)(inseq);
        *tbp = '\0';
    }
    return c;
}
```

A.1.2.4 c_test.c

```
static char SCCSID[] = "@(#)c_test.c 1.2 85/05/23";

#include <stdio.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/timeb.h>

#define OFF  0
#define ON   1

/*
 * test program for io functions
 */

unsigned delay;

alrmhand(on)
int on;       /* 0 for no alarm, 1 for alarm init, SIGALRM for event */
{
      static int (*lasthand)() = SIG_DFL;
      static unsigned alrm = 0;
      struct timeb tb;

      switch (on)
      {
      case OFF:       /* turn alarm handling off */
              /*VOID*/signal(SIGALRM, lasthand);
              /*VOID*/alarm(alrm);
              alrm = 0;
              break;
      case ON:        /* turn alarm handling on */
              lasthand = signal(SIGALRM, alrmhand);
              ftime(&tb);
              alrm = alarm((tb.millitm > 300) ? delay + 1 : delay);
              break;
      case SIGALRM:   /* received alarm signal: */
              /*VOID*/signal(SIGALRM, alrmhand);
              break;
      default:
              printf(" alrmhand: unexp argument '%d'}n", on);
              break;
      }
      return;
}

int
nextc(inseq)
int inseq;
{
      int c;

      if (delay && inseq)
      {
              alrmhand(ON);
              c = getchar();
              alrmhand(OFF);
      }
      else
              c = getchar();
      return c;
```

```
}

main(argc, argv)
int argc;
char **argv;
{
        int c;
        int i;
        char buf[BUFSIZ];

        system("stty cbreak");
        printf("%s READY: ", argv[0]);
        c_init("}4", EOF);                      /* END */
        c_init("}33P", -16);                    /* F16 */
        c_init("}33", -11);                     /* ESC */
        c_init("}33P0wP}33}}", -22);            /* turnkey on */
        c_init("}33P0wR}33}}", -33);            /* turnkey off */
        while ((c = c_get(nextc)) != EOF)
        {
                switch (c)
                {
                case 'a':       /* setup alarm handling: */
                        printf("}nEnter delay and RETURN: ");
                        if (scanf("%d", &delay) != 1)
                        {
                                printf("Illegal delay --> delay unchanged}n");
                                break;
                        }
                        printf("delay set to %d}n", delay);
                        getchar();      /* skip nl */
                        break;
                case 'n':       /* new entry: */
                        printf("}nPress key and RETURN: ");
                        for (i = 0; (c = getchar()) != '}n'; i++)
                                buf[i] = c;
                        buf[i] = '}0';
                        printf("Enter decimal code and RETURN: ");
                        if (scanf("%d", &c) != 1)
                        {
                                printf(" Illegal code --> aborted entry}n");
                                break;
                        }
                        printf("c_init returns %s}n", ((c = c_init(buf, c)) == 1
                                                        ? "SUCCESS"
                                                        : ((c == 0) ? "NOMEM"
                                                                    : "MULTIPLE")));
                        getchar();      /* skip nl */
                        break;
                default:        /* PLAYBACK */
                        printf(" -> %d}n", c);
                        break;
                }
                printf(" %s READY: ", argv[0]);
        }
        system("stty -cbreak");
        exit(0);
}
```

A.1.2.5 chcode.c

```
static char SCCSID[] = "@(#)chcode.c 1.3 85/06/13";

/*-----------------------------------------------------------------------------
 * chcode.c
 *
 * Programm zum Setzen der CH-CODE-Taste und Laden der Zeichensaetze.
 *
 *-----------------------------------------------------------------------------
 *
 * Die CH-CODE-Taste ist nach dem Einschalten immer in Stellung <INT> und kann
 * dann manuell oder per Programm in die jeweils andere Stellung "gekippt"
 * werden.
 * Mit diesem Beispielprogramm besteht die Moeglichkeit, die Taste gezielt in
 * eine Stellung zu bringen und den Zeichenvorrat evtl. auch gleich zu laden.
 *
 * }E  im String wird als <ESC> ausgegeben,
 * }}  im String wird als  }   ausgegeben.
 *
 * Beispiele:   chcode int              CH-CODE-Taste in Stellung international
 *              chcode nat              CH-CODE-Taste in Stellung national
 *              chcode nat "}E(K}E[7u"  zusaetzlich deutschen Zeichensatz laden
 *                                      und deutsche Tastatur einstellen.
 *
 *-----------------------------------------------------------------------------
 */

#include        <stdio.h>

#define         ESC     0x1b
#define         CHCODE  "%c[5v",ESC                     /* Code umschalten */
#define         CHCDEN  "%c[11v",ESC                    /* CHCODE freigeben */
#define         ZVTEST  "%c[13v",ESC                    /* akt. ZV abfragen */

/*-----------------------------------------------------------------------------*/

main (argp, argv)

int     argp;
char    *argv[];

{
switch (argp)
    {
    case  3 : prints (argv[2]);                             /* fall through */
    case  2 : if  (tcode () != strcmp ("nat",argv[1]))     printf (CHCODE);
              vexit (0);
    default : fprintf (stderr, "}nusage: chcode nat/int [string]");
              fprintf (stderr, "}n(}}E  in the string means  <ESC>, ");
              fprintf (stderr, "}}}}  in the string means  }})}n");
              exit (1);
    }
}

/*-----------------------------------------------------------------------------*/

prints (s)                                      /* String ausgeben, }E == ESC */

register char   *s;
```

```
{
register char   c, old;

while (c = *s++)
    {
    if  (old == '}}')       { putchar ((c == 'E') ? ESC : c); c = 0; }
    else if (c != '}}')     putchar (c);
    old = c;
    }
}

/*-------------------------------------------------------------------------*/

tcode ()                                        /* CH-CODE-Taste abfragen */

{
register char   c;

system ("stty cbreak -echo");
printf (CHCDEN); printf (ZVTEST);
if  (getchar () != ESC)
    if  (getchar () != ESC)     vexit (1);
if  (getchar () != 'P')         vexit (1);
if  (getchar () != '1')         vexit (1);
if  (getchar () != '3')         vexit (1);
if  (getchar () != 'v')         vexit (1);
c = getchar ();
if  (c < '0' || c > '3')        vexit (1);
if  (getchar () != ESC)         vexit (1);
if  (getchar () != '}}')        vexit (1);
return (++c & 1);
}

/*-------------------------------------------------------------------------*/

vexit (n)

int     n;

{
system ("stty -cbreak echo");
exit (n);
}

/*-------------------------------------------------------------------------*/
```

A.1.2.6 lp9012.c

```
/*
  lp9012.c    Druckerspezfische Routinen fuer PT90-Tintenmatrixdrucker

  Dies ist ein einfaches Beispielbackend. Es wird keinerlei Garantie fuer
  eine fehlerfreie Funktion uebernommen.
*/

#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <sys/termio.h>
#include "backend.h"
#include "backmeld.h"
#include "backprot.h"
#include "excode.h"
#include "files.h"
#define BLSIZ  1000     /* Anzahl der Druckzeichen im Druckpuffer */

extern FILE *tfid;      /* Filepointer auf trace-File           */
extern FILE *fid;       /* Filepointer auf Druckfile            */
extern long seekp; /*bis zu dieser Adresse wurde Datei bereits gedruckt*/
extern long seeksav; /* laufender Zaehler */
extern long size;    /* Dateigroesse      */
extern int  copies;  /* Anzahl Kopien, die noch gedruckt werden sollen */
extern int  evstk[];    /* Ereignisstack des Backends           */
extern int  evstkp;     /* Zaehler fuer Ereignisstack           */
extern int  pages;      /* Anzahl bereits ausgedruckter Seiten  */
extern int  trace;      /* Flagge fuer TRACE-Funktion           */
extern char fname[];    /* Name der zu druckenden Datei         */
extern int   cancl;     /* Cancelkommando eingetroffen          */

short stack[52];        /* wird ev. von anderen Moduln benoetigt */

long   fseek();
char *strapd();         /* Private Routine zur Stringbehandlung */

/*      aktuelle Parameter  */

char  zsf;                      /* Charater fuer Zeichensatzauswahl */
int   spmax;                    /* Spalten pro Zeile            */
int   zmax;                     /* Zeilen pro Seite             */
int   ab;                       /* Dokumentbeginn               */
int   bis;                      /* Dokumentlaenge               */
int   font = 1;                 /* Font-Auswahl                 */
int   schr;                     /* Schreibschritt               */
int   za;                       /* Zeilenabstand                */
char  dr;                       /* Druckrichtung                */

char buf[BLSIZ+LINSIZ];         /* Druck-Ausgabepuffer          */
char *cp1;                      /* Zeiger auf buf               */
char *cpx;                      /* Zeiger auf buf               */
int   zz = 0;                   /* Zeilenzaehler                */
int   spz = 0;                  /* Spaltenzaehler               */
int   fontflag = 0;             /* Zeichensatzflag              */
int   print;                    /* Indikator fuer Druckvorgang  */
char  hd = 0 ;                /* Header (wird hier nicht unterstuetzt) */
char  trl = 0 ;               /* Trailer (wird hier nicht unterstuetzt)*/
char  hdgrp = 0 ;             /* Header (wird hier nicht unterstuetzt) */
char  trlgrp = 0 ;            /* Trailer (wird hier nicht unterstuetzt)*/

static char pty[256];           /* Paerity-Tabelle      */

static struct termio termio;    /* termio-Schnittstelle         */
```

```
/*************************************************************************/
/*
      Druckerspezifische Parameter

      Schnittstellenparameter :
*/
unsigned short iflag   = ( IXON | ISTRIP | IGNBRK | IGNPAR );
unsigned short cflag   = ( CREAD | CS8 | B9600 );
unsigned short oflag   = 0 ;     /* keine Behandlung der Ausgabezeichen */
unsigned short lflag   = 0;      /* immer Null                          */
char  ccsvt   = 0;
char  ccsvm   = 1;
char  ccsve   = 6;
char  cs  = 7 ;      /* fuer interne prty-Routine, die nicht aufgerufen wird */

/*
      Steuersequenzen
*/
static char  *bs    = "}b";           /* Backspace                    */
static char  *ht    = "}t";           /* Tabulator                    */
static char  *nl    = "}r}n";         /* Newline                      */
static char  *ff    = "}r}f";         /* Newpage                      */
static char  *cr    = "}r";           /* Carriage return              */

static char *GZZ    = "}033[x";        /* Zeilenabstand Grundzustand   */
static char *GZSTD  = "}033R";         /* Druckparameter Grundzustand  */
static char *fschrb = "}033[%dw";      /* Schreibschritte setzen       */
static char *ffont  = "}033,%d";       /* Font einstellen              */
static char *fzmax  = "}033[%d";       /* Seitenlaenge einstellen      */
static char *fzs    = "}033(%c";       /* Zeichensatz einstellen       */
static char *fdr    = "}033 %c";        /* Druckrichtung                */
/*
      Standardparameter
*/
int   spmaxstd = 4096;          /* Standard: keine Spaltenbegrenzung  */
int   zmaxstd  = 68;            /* Standard: 68 Zeilen pro Seite      */
char  zstd     = 'B';           /* Standard: ASCII - Zeichensatz      */
int   abstd    = 0;             /* Standard: Dokumentbeginn           */
int   bistd    = 1<<15-2;       /* Standard: beliebig langes Dokument */
int   schrstd  = 1;             /* Standard: 1/10 Zoll                */
char  drstd    = ':';           /* Standard bidirektional             */
int  emstd = 0 ;                /* Standard fuer em9001               */

static int grfbytes  = 0;  /* Anzahl Grafikbytes, die noch ausgegeben werden muessen */
static int em9001 = 0 ;       /* Flagge fuer 9001-Grafik-Emulation */

/*************************************************************************/

backflg(ac,av)          /* Auswertung der Backendschalter */
int ac;                 /* wird beim Start des Backends und bei jedem  */
char *av[];             /* Druckauftrag aufgerufen                     */
{
      char * pschalt ;
      int  i = 0;
      int  j;
/*
      Einstellen der Standardwerte
*/
      spmax = spmaxstd;
      zmax = zmaxstd;
      zsf  = 'B';
      ab   = abstd;
      bis  = bistd;
      schr = schrstd;
      dr   = drstd;
      fontflag = 0;
```

```
        em9001 = emstd ;

        for(i=1;i<ac;i++)
        {
                pschalt = av[i]+1 ;

                if ( strncmp( pschalt , "ab=" , 3 ) == 0 )
                        ab = ( j = atoi( pschalt + 3 ) ) > 1 ? j : 1  ;
                else if ( strncmp( pschalt, "bis=" , 4 ) == 0 )
                        bis = (j = atoi( pschalt + 4 ) ) > ab ? j : bis ;
                else if ( strncmp( pschalt, "pl=", 3 ) == 0 )
                        zmax = (j = atoi( pschalt + 3) ) > 5 ? j : zmax ;
                else if ( strncmp( pschalt, "pb=", 3 ) == 0 )
                        spmax = (j = atoi(pschalt + 3)) > 5 ? j : spmax ;
                else if ( strncmp( pschalt, "font=", 5 ) == 0 )       {
                        fontflag = 1;
                        font = atoi( pschalt + 5 ) ;
                       }
                else if ( strcmp( pschalt, "pb1" ) == 0 ) {
                        schr = 1;
                       }
                else if ( strcmp( pschalt, "pb2" ) == 0 ) {
                         schr = 2;
                       }
                else if ( strcmp( pschalt, "pb3" ) == 0 )  {
                        fontflag = 1;
                        if ( font != 4 )      font = 2;
                        schr = 3;
                       }
                else if ( strcmp( pschalt, "pb4" ) == 0 )  {
                        fontflag = 1;
                        if ( font != 4 )      font = 2 ;
                        schr = 4;
                       }
                else if ( strcmp( pschalt, "dt" ) == 0 )
                        zsf = 'K';
                else if ( strcmp( pschalt, "int" ) == 0 )
                        zsf = 'B';
                else if ( strcmp( pschalt, "uni" ) == 0 )
                        dr = '=';
                else if ( strcmp( pschalt, "em91" ) == 0 )
                        em9001 = 1 ;
                else if ( strcmp( pschalt, "trace" ) == 0 )
                        trace = 1 ;
                else if ( strcmp( pschalt, "statusfile" ) == 0 )
                        ;     /* Nostalgieschalter ohne Bedeutung */
                else
                        return( (BF_PERR<<8) | i ) ;  /* Fehler */
        }
        if (trace)    fprintf(tfid,"pschalt durchlaufen}n");
        return( BF_OK << 8 ) ;
}

druin()                 /* Initialisierung des Backends */
{
        int n;

        ioctl(1,TCGETA,&termio);
        termio.c_iflag      = iflag;
        termio.c_cflag      = cflag;
        termio.c_oflag      = oflag;
        termio.c_lflag      = lflag;
        termio.c_ccs[VTIME] = ccsvt;
        termio.c_ccs[VMIN]  = ccsvm;
        termio.c_ccs[VEOL]  = ccsve;
        ioctl(0,TCSETA,&termio);
        ioctl(1,TCSETA,&termio);
```

```
        ioctl(1, TCFLSH, 2) ;

/*
        'echtes' (einmaliges) Einstellen der Standardwerte :
        Beim Start des Backends wird zunaechst die Funktion backflg()
        zur Auswertung der Standardschalter aufgerufen. Anschliessend wird
        die Funktion druin() durchlaufen, wo jetzt diese Standardwerte
        festgehalten werden.
*/

        parity();                       /* Initialisierung der Parity-Tabelle */
                    /* Sicherheitshalber wird die eigene Routine aufgerufen */

        spmaxstd = spmax ;
        zmaxstd  = zmax  ;
        zstd     = zsf   ;
        abstd    = ab    ;
        bistd    = bis   ;
        schrstd  = schr  ;
        drstd    = dr    ;
        emstd    = em9001 ;

        if(trace)
                fprintf(tfid,"}n, druin zmax=%d,spmax=%d,font=%d}n",
                                        zmax,spmax,font);

}

static int etxflag = 0 ;

dstat(bf3)                  /* Druckerstati auswerten       */
char *bf3 ;         /* wird von den druckerunabhaengigen Moduln verwendet */
{
        int statflag = 1, drst = 0 ;
        char c ;
        if(trace)   fprintf(tfid,"dstat: etx=%d, cancl=%d  ", etxflag, cancl);

        if ( etxflag == 0 ) {           /* Falls noch kein ETX gesendet wurde */
                c = pty[ ETX & 0177 ];  /* ETX senden                         */
                if( write( 1, &c, 1) != 1 ) {    /* Zeitablauf                */
                        if ( trace ) fprintf( tfid, " W-SIGNAL}n" ) ;
                        bp_stat( B_IDLE ) ;
                        if ( !cancl )  evstk[++evstkp] = E_INTR ;
                        return( -1 ) ;
                }
        }
        else
              etxflag = 0 ;

        while( statflag || rdchk(0) > 0 )   /* Es werden alle Rueck-  */
        {                                   /* meldungen gelesen !    */
                if( read( 0, &c, 1 ) < 1 ) {
                        bp_stat(B_NPOLL);
                        drst = 1;
                        statflag = 0 ;   /* Schleife verlassen */
                        if (trace) fprintf(tfid, " R-SIGNAL}n" );
                }
                else {                  /* Zeichen wurde gelesen */
                        c = c & 0177 ;
                        if ( trace ) fprintf( tfid, "%o, ", c ) ;
                        if ( c == ACK )  statflag = 0 ;
                }
        }

        if ( drst )
                evstk[++evstkp] = E_NAK ;
        else
```

```
            evstk[++evstkp] = E_ACK;

      if ( trace ) fprintf( tfid, "dstat beendet : %d\n", drst ) ;
      return( drst );
}

char *strapd(cp1,cp2)          /* String anhaengen     */
char *cp1,*cp2;
{
      while(*cp2 && (*cp1++ = pty[*cp2++]));
      return(cp1);
}

anblock()                           /* Ausdruck starten */
{
      char cp2[10];
      struct stat statbuf;
      int    i;

      if(trace)   fprintf(tfid,"anblock ");
      bp_anp();

      stat(fname,&statbuf);       /* Information aus der Inode       */
      size = statbuf.st_size;
      fseek(fid,seekp,0);   /* Positionieren auf letzte Seitengrenze */
      cp1   = buf;
      if(seekp)   cp1 = strapd(cp1,ff);
      zz    = 0;
      spz   = 0;

/*      Drucker initialisieren   */

      if ( fontflag )       {
            sprintf(cp2,ffont,font);          /* Font-Auswahl       */
            cp1 = strapd(cp1,cp2);
      }
      sprintf(cp2,fzs,zsf);              /* Zeichenvorrat            */
      cp1 = strapd(cp1,cp2);
      sprintf(cp2,fschrb,schr);         /* passenden Schreibschritt */
      cp1 = strapd(cp1, cp2);           /* einschalten              */
      sprintf(cp2,fdr,dr);              /* Druckrichtung bestimmen  */
      cp1 = strapd(cp1, cp2);

      if ( em9001 ) {
            cp1 = strapd( cp1, "}033[3 w" ) ;
      }

      cpx = cp1;

      grfbytes = 0 ;  /* Anzahl Grafikbytes auf Null setzen  */

      if (trace)    fprintf(tfid,"anblock()-Ende}n");

      block();                                  /* 1. Block ausgeben  */
}

block()                           /* Datenblock ausgeben         */
{
      int     c;               /*  gelesenes Zeichen          */
      int     i;               /*  Zeichen im Ausgabepuffer   */
      int     j;
      if(trace)   fprintf(tfid,"block   ");
      print = (pages+1 >= ab);

      /* Eventuell noch Grafikbytes ausgeben */
      for ( i = cp1-buf; grfbytes > 0 && i < BLSIZ; i++, seeksav++, grfbytes--)
      {
```

```
        if ( ( c = fgetc(fid) ) == EOF )  c = '}O' ;
        *cp1++ = c ;
}

for(i=cp1-buf;i<BLSIZ;i++,seeksav++) /* BLSIZ=2400 Druckzeichen*/
{
        switch(c = fgetc(fid))  /* Druckfile lesen      */
        {
        case NL:
                cp1 = strapd(cp1,nl);
                spz = 0;
                if(++zz >= zmax) {
                        if(++pages >= bis )
                                c = EOF;
                        i = BLSIZ;
                        zz = 0;
                }
                break;
        case CR:
                cp1 = strapd(cp1,cr);
                spz = 0;
                break;
        case FF:
                if ( trace ) fprintf( tfid,"FF" ) ;
                cp1 = strapd(cp1,ff);
                if(++pages >= bis )  c = EOF;
                i = BLSIZ;
                zz = 0;
                spz = 0;
                break;
        case HT:  /* Tabulatoren werden in Blanks umgewandelt */
                c = ' ';
                do
                     *cp1++ = pty[c];
                while ( spz < spmax && ++spz % 8 ) ;
                i = cp1-buf; /* Zeichenzaehler aktualisieren */
                break;
        case BS:
                if(spz) {
                        cp1 = strapd(cp1,bs);
                        spz--;
                }
                break;
        case EOF:
                if ( trace ) fprintf( tfid,"EOF" ) ;
                seeksav--;
                i = BLSIZ;
                break;
        case ESC:
                *cp1++ = pty[c] ;
                if ( seqsuch() == EOF )  {
                                i = BLSIZ ; c = EOF ;
                                break ;
                }
            /* Falls grfbytes > 0, muessen Grafikdaten ausgegeben werden */
                for ( ; grfbytes>0 && i<BLSIZ; i++, seeksav++, grfbytes--)
                {
                     if ( ( c = fgetc(fid) ) == EOF )  c = '}O' ;
                     *cp1++ = c ;
                }
                break ;
        case '}O': break ;  /* um GKS-Fehler zu umgehen */
        default:
                if(spz++ < spmax)
                        *cp1++ = pty[c];
                break;
        }
```

```
        }
        if(c == EOF)
        {
                if(zz || !pages)      cp1 = strapd(cp1,ff);
                zz = 0;
                if ( copies == 1 )       cp1 = strapd(cp1,GZSTD);
        }
/*
        Druckfileblock ausdrucken
*/
        if ( !zz && !grfbytes ) {  /* Falls Seitenende und keine Grafikdaten */
                *cp1++ = pty[ ETX & 0177 ] ; /* ETX anfuegen */
                etxflag =1 ;
        }

        if(print && write(1,buf,cp1-buf) == cp1-buf)
        {
                if( etxflag )
                {
                        if ( (i=dstat("}O")) != 0 ) /* Druckerstati abfragen */
                                return ;
                        seekp = seeksav;
                        bp_rept(seekp,size,pages,copies);
                }
                else   evstk[++evstkp] = E_ACK;

                if(c == EOF)                    /* Druckfileende       */
                {
                        if(--copies && !cancl )   /* weitere Kopien      */
                        {
                                seeksav = seekp = 0L;
                                fseek(fid,0L,0);
                        }
                        else                /* Druckvorgang beendet */
                        {
                                evstk[evstkp] = E_HDRQ;
                                fclose(fid);
                                fid = NULL;
                                seekp = seeksav;
                        }
                        bp_rept(seekp,size,pages,copies);
                        print = 0;
                }
                cp1 = buf;
        }
        else if(print)    /* kein Druckvorgang, 'CANCEL' oder 'offline' */
        {
                ioctl(1,TCFLSH, 2 );
                evstk[++evstkp] = E_INTR;
                if ( trace ) fprintf( tfid, " W-INTR}n" ) ;
        }
        else
        {
                bp_rept(seeksav,size,pages,copies);
                evstk[++evstkp] = E_ACK;
                cp1 = cpx;
        }
}

parity()    /* Parity-Tabelle initialisieren, private Routine mit neuem Namen */
{
        int i = 0;
        register char a , p;
        pty[255] = '}0';

        while( i++ < 127 )
        {
```

```
        for( p = 0, pty[i] = a = i; a; a >>= 1)
                if ( a & 01 )   p = p ? 0 : 1;          /* Flip-Flop */
        if (!p)       pty[i] |= 0200;             /* hoechstes Bit setzen */

    }
    if (trace)  fprintf(tfid,"parity-Ende  }n");

}

seqsuch()               /* Druckerspezifische Routine zur Erkennung von */
{                       /* ESC-Folgen                                   */
    int n, c, escflag, esclaenge ;
    int nbyte ;
    char escfolg[10], *cpesc ;

    nbyte    = 0 ;
    escflag  = 1 ;
    esclaenge = 9999 ;    /* maximale Laenge einer ESC-Folge */

    while ( escflag )
    {
        if ( ( c = fgetc(fid) ) == EOF ) {
            if (trace) fprintf(tfid, "EOF in ESC-Folge}n") ;
            return( EOF ) ;
        }

        if ( escflag >= esclaenge ) { /* Ende der ESC-Folge erreicht */
            *cp1++ = pty[c] ;
            return( 0 ) ;
        }

        switch( escflag )
        {
        case 1: *cp1++ = pty[c] ;
            switch( c )
            {
            case ' ':
            case ',':
            case '(':
            case '!':
                    escflag   = 2 ;
                    esclaenge = 2 ;  /* ESC + 2 Zeichen */
                    break ;  /* Ende nach dem naechsten Zeichen */
            case '[': escflag = 2 ;
                    cpesc = cp1 ;  /* Anfang der Zahlen merken */
                    break ;
            default : escflag = 0; /* ESC + 1 Zeichen             */
                    break ;      /* Ende der ESC-Folge erreicht */
            }
            break ;
        case 2: /* dieser Fall kann nur eintreten, bei ESC [ .... */
            *cp1++ = pty[c] ;
            switch( c )
            {
            case 'p':
            case '{':
            case 's':
            case 'x':
                    escflag = 0 ; /* sofortiges Ende */
                    break ;
            case '<':
            case '=':
            case ' ':
                    esclaenge=3; /* Ende nach dem naechsten Zeichen */
                    break ;
            case ';': escflag++ ;
                    break ;
```

```
                        default :
                                if ( c >= '0' && c <= '9' ) {
                                        escflag++ ;
                                        nbyte = 10*nbyte + c - '0' ;
                                }
                                else escflag = 0 ;
                                break ;
                        }
                        break ;
                case 3: /* dieser Fall kann nur bei ESC [ ZAHL oder nach einem ';' eintre-
ten */
                case 4:
                case 5:
                case 6:
                        if ( c >= '0' && c <= '9' ) {
                                escflag++ ;
                                nbyte = 10*nbyte + c - '0' ;
                                *cp1++ = pty[c] ;
                                break ;
                        }
                case 7: *cp1++ = pty[c] ;
                        switch( c )
                        {
                        case ';': escflag = 3 ; /* Es kommen weitere Zahlen */
                                  nbyte = 0 ;   /* Wieder bei Null beginnen */
                                  break ;
                        case ' ': esclaenge = ++escflag ;
                                  break ; /* Ende der ESC-Folge nach dem naechsten Zeichen */
                        case 'y':         /* Grafikfolge wie bei 9001 */
                        case 'v':         /* Grafikfolge mit Wiederholungsfaktor */
                                  grfbytes = nbyte ;
                                  if (trace) fprintf(tfid, "grfbytes=%d }n", grfbytes);
                                  escflag = 0 ;
                                  break ;
                        case 'x': if ( em9001 && nbyte ) {
                                        sprintf(escfolg, "%02dx",(nbyte*120)/72 -1);
                                        cp1 = strapd( cpesc, escfolg ) ;
                                  }
                                  escflag = 0 ;
                                  break ;
                        default:
                                  escflag = 0 ;  /* Ende der ESC-Folge */
                                  break ;
                        }
                        break ;
                default: escflag = 0 ;
                         break ;
                }
        }
        return( 0 ) ;
}

header()
{
}

trailer()
{
}

npafn()                                                 /* Abbruch eines Druckauftrags*/
{
        short i,j;

        if(trace)   fprintf(tfid,"npafn   ");
        if(print)
        {
```

```
            while( grfbytes > 0 )           /* Graphik-Mode verlassen    */
            {
                    for(i=0,cp1=buf ; i<BLSIZ && grfbytes>0; grfbytes--, i++)
                            *cp1++ = '}0';
                    i = write(1,buf,cp1-buf);
            }
            cp1 = buf;
            cp1 = strapd(cp1,ff);
            i   = write(1,buf,cp1-buf);
            sleep( 5 ) ;
            print = 0;
    }
    cancl = 0;
    bp_afn();
}
```

A.1.2.7 lpFREMD.c

```
/* Primitives Universalprogramm, das anstelle des "cat"-Kommandos zur */
/* Ausgabe von Daten auf einen Drucker verwendet werden kann.         */
/* Dieses Programm setzt voraus, dass der Geraetetreiber richtig      */
/* eingestellt ist ( CBREAK-Mode mit richtiger Paritaet ).            */
/* Vorsicht : Bei ESC-Folgen, die laenger als 2 Zeichen sind, stimmt  */
/* --------   die Spaltenzaehlung nicht mehr !!                       */
/* Im Anschluss an den Ausdruck wird ein Formularvorschub erzwungen.  */

#include <stdio.h>
#include <signal.h>

#define HT '\011'       /* Horizontaltabulator          */
#define LF '\012'       /* Line Feed Zeichen            */
#define FF '\014'       /* Form Feed Zeichen            */
#define CR '\015'       /* Carriage Return Zeichen      */
#define ESC '\033'      /* ESC-Zeichen                  */

int abbruch() ;         /* Verweis auf die Abbruchfunktion */

main(argc,argv)
int argc;
char *argv[];
{
      FILE *fpein ;
      int c ;
      register short nspalte ;
      char buffer[BUFSIZ] ;

      setbuf( stdout, buffer ) ;  /* Puffer von 1 KB bereitstellen        */

      signal(SIGALRM, abbruch) ;  /* Falls das Alarmsignal 15 eintrifft,  */
                                  /* wird das Backend kontrolliert beendet */

      /* Datei zum Lesen oeffnen */

      if ( ( fpein = fopen(argv[1] , "r") ) == NULL )
             exit(1) ;      /* Datei konnte nicht geoeffnet werden */

      putchar( CR ) ;                       /* Wagenruecklauf     */
      nspalte = 1 ;
      while ( ( c = getc(fpein) ) != EOF )  /* Datei ausgeben     */
      {
            switch( c )
            {
            case HT: do                     /* Tabulatoren durch  */
                           putchar(' ') ;   /* Blanks expandieren */
                     while ( nspalte++ % 8 ) ;
                     break ;
            case LF: putchar( CR ) ;        /* Wagenruecklauf     */
            case CR: nspalte = 1 ;          /* Spaltenzaehler     */
                     putchar( c ) ;
                     break ;
            case ESC:
                     putchar( c ) ;         /* ESC + 1 Zeichen ! */
                     if ( ( c = getc(fpein) ) != EOF )
                           putchar( c ) ;
                     break ;
```

```
            default: if ( c >= ' ' && c <= '~' ) /* nur abdruckbare */
                                nspalte++ ;      /* Zeichen zaehlen */
                     putchar( c ) ;
                     break ;
            }
      }
      putchar( FF ) ;      /* Formularvorschub am Ende des Ausdrucks */
      exit(0);
}

abbruch()    /* kontrollierter Abbruch nach Eintreffen des Alarm-Signals */
{
      clearerr( stdout ) ;
      putchar( FF ) ;
      putchar( CR ) ;
      exit( 64 ) ;
}
```

A.1.2.8 Include-Dateien für Drucker-Backends

pcx/backend.h / pcmx2/backend.h

```
/*
      backend.h       Parameter fuer Backends
*/

#ifndef  BACKEND_H
#define  BACKEND_H  1

#define   NU     0x00                /* ASCII - Steuerzeichentabelle */
#define   STX          0x02
#define   ETX          0x03
#define   ACK          0x06
#define   BEL          0x07
#define   BS   0x08
#define   HT   0x09
#define   NL   0x0a
#define   VT   0x0b
#define   FF   0x0c
#define   CR   0x0d
#define        SO      0x0e
#define        SI      0x0f
#define   DC1          0x11
#define   NAK          0x15
#define   EM   0x19
#define   SUB          0x1a
#define   ESC          0x1b
#define   FS      0x1c
#define   RS   0x1e
#define   US   0x1f
#define   NN   0x80
#define   CLT     0x81
#define   SHT     0x82
#define   NLF     0x83
#define   GR8     0x90
#define   GR16    0x91

#define   F_ZS             1                  /* zugelassene lpr - Schalter   */
#define   F_ZB             2
#define   F_PB        3
#define   F_PL        4
#define   F_HD        5
#define   F_TRL       6
#define   F_AB        7
#define   F_BIS       8
#define   F_PROBE     9
#define   F_BTRACE   10
#define   F_STAT     11
#define   F_PS       12
#define   F_HOP      13
#define   F_FONT     14
#define   F_TAB           15
#define   F_NK            16
#define   F_MAR           17

#define   ZS_XX       0                       /* zugelassene Sprachen         */
#define   ZS_INT      1
```

```
#define   ZS_ENGL    2
#define   ZS_ASCI    3
#define   ZS_DAEN    4
#define   ZS_FINN    5
#define   ZS_DTSH    6
#define   ZS_FRNZ    7
#define   ZS_SPAN    8

#define   INT       "INT"
#define   ENGL      "ENGL"
#define   ASCI      "ASCI"
#define   DAEN      "DAEN"
#define   FINN      "FINN"
#define   DTSH      "DT"
#define   FRNZ      "FRNZ"
#define   SPAN      "SPAN"

#define   ZPZ_XX     0                    /* Zeichenbreiten              */
#define   ZPZ_80     1
#define   ZPZ_96     2
#define   ZPZ_136    3

struct flag
{
      char *flagstr;     /* Zeiger auf Stringwert des Parameternamens */
      char flagtype;     /* Code fuer den Parameter (F_xx - defines ) */
      char para;         /* OBL, OPT oder NOPAR                       */
      char fevent;       /* Zusatzinformation                         */
      char *fval;        /* Zeiger auf Flagvalue                      */
};

struct ptab
{
      char  *val;        /* Zeiger auf Flagvalue                      */
      short nam;
};

struct stbyt
{
      short bnr;
      char  cnr;
      char  *meld;
};

/*
      Konstanten fuer Zustaende und Ereignisse im backend
*/

#define  S_BMAX   4     /* Gesamtzahl 'backend' Zustaende              */
#define  E_BMAX  10     /* max. moegliche 'backend' Ereignisse         */
#define  S_DATE  26  /* Feldbreite fuer Datum                       */

#define  S_GRND   0     /* Grundzustand                                */
#define  S_BTST   1     /* Druckertest laeuft                          */
#define  S_OPEN   2     /* Druckdatei eroeffnen                        */
#define  S_BLOCK  3     /* Block ausgeben                              */

#define  E_KTST   0     /* KDO_TST angekommen                          */
#define  E_KDIE   1     /* KDO_DIE angekommen                          */
#define  E_KOUT   2     /* KDO_OUT angekommen                          */
```

```
#define  E_TCNC   3     /* TX2_CANC angekommen                          */
#define  E_ACK    4     /* Druckeroperation war ok                      */
#define  E_NAK    5     /* Druckeroperation war nicht ok                */
#define  E_INTR   6     /* Zeitueberschreitung / Nachricht angekommen   */
#define  E_NOP    7     /* Nullereignis                                 */
#define  E_AERR   8     /* Fehler beim Auftrag (daemon muss A. loeschen)*/
#define  E_HDRQ   9     /* Header Info Request                          */

#define  LINSIZ  768    /* Druckpufferergaenzung fuer header + trailer  */

/*
      Schnittstellendefinitionen fuer backend Flaganalyse

   backflg.c liefert eine Wert vom Typ int
   die Makros  backret und backav liefern aus diesem Wert den Returncode
   und die Nummer eines fehlerhaften Arguments
*/

#define  OBL        1        /* Parameterwert  obligatorisch        */
#define  OPT        2        /* Parameterwert  optional             */
#define  NOPAR      4        /* Parameterwert nicht erlaubt         */
#define  STKLEN     5               /* maximale Stacktiefe                  */

#define  BF_OK      0           /* alles ok.                           */
#define  BF_OBL     1           /* Parameter braucht ein Argument      */
#define  BF_NOPAR   2           /* Parameter kennt kein Argument       */
#define  BF_PERR    3           /* unbekannter Parameter               */
#define  BF_PVERR   4           /* unzulaessiger Parameterwert         */
#define  BF_CDTB    5           /* Fehler beim oeffnen der Codetabelle */
#define  BF_VTAB    6           /* Fehler beim oeffnen der VT-Tabelle  */
#define  BF_HTAB    7           /* Fehler beim oeffnen der HT-Tabelle  */

#define backret(x) (x>>8)       /* linkes  Byte (maschinenunabhaengig) */
#define backav(x)  (x&255)      /* rechtes Byte (maschinenunabhaengig) */

#endif
```

pcx/backmeld.h / pcmx2/backmeld.h

```
/* @(#)backmeld.h    2.8 86/02/12 */
/*
      backmeld.h      Meldungen des PT88 backends
*/

#ifndef BACKMELD_H
#define BACKMELD_H  1

#define B_OPERR  "Das Dokument kann nicht gelesen werden."
#define B_PERR   "unbekannter Parameter %s"
#define B_PAROBL "Parameter %s benoetigt ein Argument"
#define B_NOPAR  "Parameter %s kennt kein Argument"
#define B_PVERR  "Parameter %s hat unzulaessiges Argument"
#define B_RIBEND "Farbband-Ende"
#define B_TONEMP "Toner leer"
#define B_RECER  "Empfangsfehler"
#define B_PPEND  "Papierende"
#define B_COPEN  "Schutzdeckel offen"
#define B_ASFERR "Fehler Einzelblattzufuehrung"
#define B_PAUSE  "Pause-Taste gedrueckt"
#define B_STOP   "Stop-Taste gedrueckt"
#define B_IBACX  "Backend %s kann nicht geladen werden"
#define B_STAT   "Statusdatei %s kann nicht geoeffnet werden}n"
#define B_TROPEN "Verfolgerdatei %s eroeffnet"
#define B_PRIDL  "Drucker ausser Betrieb"
#define B_NPOLL  "Drucker nicht ansprechbar"
#define B_DAUS   "Drucker nicht eingeschaltet"
#define B_IDLE       "Drucker offline"
#define B_PRICH  "Drucker im Checking"
#define B_CODE   "Drucker 9645 Fehlerbyte: %d"
#define B_CDTB   "Codetabelle %s kann nicht gelesen werden"
#define B_HTAB   "Horizontaltabulatortabelle %s kann nicht gelesen werden"
#define B_VTAB   "Vertikaltabulatortabelle %s kann nicht gelesen werden"
#define B_ATIM   "Auftragszeit:"
#define B_AUSEND "Ausgabe beendet:"
#define B_BOX1   "**********************************************************************}r}n"
#define B_BOX2   "**   %-22s%-36s**}r}n"
#define B_DTIM   "Druckzeit   :"
#define B_FROM   "Gedruckt von:"
#define B_FTIM   "Letzte Veraenderung:"
#define B_HASHM  "######################################################################}r}n"
#define B_TITEL  "Titel:"
#define B_TO     "Ausliefern an:"
#define B_ALEER  "Der gewaehlte Dateiausschnitt ist leer"
#define B_DRUCK  "Druckdatei %s ist unbekannt}n"
#define B_FESCF  "Fehlerhafte ESCAPE-Folge}n"
#define B_RBUFEM "Empfangspuffer leer}n"
#define B_IPROP  "Drucker aktiv}n"
#define B_LPERR      "Geraetefehler}n"
#define B_PPNRD  "Papier bereit}n"
#define B_PPERR  "Papier eingeklemmt oder leer}n"
#define B_PPJAM  "Papier verklemmt}n"
#define B_PPALR  "Papier Alarm}n"
#define B_RPUFO  "Empfangspuffer voll}n"
#define B_BRKRC  "Break-Signal empfangen}n"
#define B_PARER  "Paritaetsfehler}n"
#define B_WARMUP "Drucker heizt auf}n"
#define B_PPSER  "OP Message, Formatfehler"
#define B_OPRER  "Druckerueberlauf}n"
#define B_IDAER  "ungueltige Daten}n"
#define B_DOFER  "Datenueberlauf}n"
#define B_HOPP1  "Hopper 1 in Betrieb}n"
#define B_HOPP2  "Hopper 2 in Betrieb}n"
```

```
#define B_GRAPH  "Graphik-Mode ein\n"
#define B_PROGM  "Programm-Mode ein\n"
#define B_NAKCM  "NAK-Code Mode ein\n"

#endif
```

pcx/backprot.h / pcmx2/backprot.h

```
/* @(#)backprot.h    2.4 86/02/12 */
/*
      backprot.h      Protokoll zwischen daemon und backend
*/
#ifndef BACKPROT_H
#define BACKPROT_H  1

#define BP_KDO   0      /* Kommando                              */
#define BP_ANP   1      /* Annahmequittung positiv               */
#define BP_ANN   2      /* Annahmequittung negativ               */
#define BP_AFP   3      /* Ausfuehrungsquittung positiv          */
#define BP_AFN   4      /* Ausfuehrungsquittung negativ          */
#define BP_TXT1  5      /* Text in Richtung Kommandogebert       */
#define BP_TXT2  6      /* Text in Richtung Kommandoempfaenger   */
/*
      Spezifikatoren fuer BP_KDO-Elemente
*/
#define KDO_TST  0      /* Teste Drucker (Poll-Kommando)         */
#define KDO_OUT  1      /* Druckauftrag mit Begleitinformation   */
#define KDO_DIE  2      /* quittiere, und beendige dich          */
#define KDO_PRO  3      /* Probedruckkommando                    */
/*
      Spezifikatoren fuer BP_TXT1-Elemente
*/
#define TX1_REP  0      /* Fortschrittsmeldung vom Backend       */
#define TX1_STA  1      /* Statusmeldung vom Drucker             */
#define TX1_ERR  2      /* Mail an Auftragsteller                */
#define TX1_HDR  3      /* Anforderung von Headerinformation     */
/*
      Spezifikatoren fuer BP_TXT2-Elemente
*/
#define TX2_HDR  0      /* Fuellinformation fuer Header          */
#define TX2_CANC 1      /* Abbruch laufende Ausgabe              */

struct protel {
            int opcod;              /* Protokollkennzeichen          */
            int lfnr;               /* Laufnummer pro Kdo.Beh.Zyklus */
            int spez;               /* Operationscode                */
            int plen;               /* Laenge der nachfolgenden Info.*/
            char *cont;             /* Zeiger auf Info.              */
           };

#define BP_FMT   "%s %ld %d %d"         /* fname seekp page copies      */
#define BP_HDR   "%s %d %s %s %ld"      /* anam num unam to ctime       */
#define BP_RPT   "%ld %ld %d %d"        /* seek size pages nc           */

#define PROTSIZE sizeof(struct protel)
#define CONTSIZE 300

#define PIP_RD   2      /* Pipe-fid zum Lesen (fuer 'backend' ) */
#define PIP_WR   3      /* Pipe-fid zum Schreiben     -"-       */

#endif
```

A.1.2.9 fl6parset.c

```
static char SCCSID[] = "@(#)fl6parset.c 1.1 85/05/02";

#define         BSIZE           1024L

#define         FLSET           (('S'<<8)|0)
#define         FLGET           (('G'<<8)|0)

struct ptab {
    int p_cylorg;       /* Anfangszylinder */
    int p_cylct;        /* Anzahl der Zylinder */
    int p_cyloff;       /* Versatz zwischen Zylindern (1=96TPI, 2=48TPI) */
    int p_secsiz;       /* Anzahl der Bytes pro Sektor */
    int p_sectrk;       /* Anzahl der Sektoren pro Spur */
    int p_sides;        /* Anzahl der Seiten: 1 oder 2 */
    int p_mfm;          /* 1 bei MFM (doppelte Dichte), sonst 0 (FM) */
    int p_seclen;       /* Code fuer die Sektorlaenge (0, 1, 2 oder 3) */
    int p_blkct;        /* Anzahl der BSIZE-Byte Bloecke */
    int p_gpl3;         /* Laenge der Blockluecke (Gap) */
};

struct ptab ptab = {
      0,                                /* Anf.zyl. */
      80,                               /* Zyl.anz. */
      1,                                /* Zyl.abstand */
      512,                              /* Bytes/Sektor */
      9,                                /* Sektoren/Spur */
      2,                                /* Seiten */
      1,                                /* Dichte */
      2,                                /* Sektorlaengencode */
      (int) ((80L * 512L * 9L * 2L) / BSIZE),         /* Anzahl Bloecke */
      0,                                /* Luecke (nicht verwendet) */
};

main()
{
      int fd;

      if ((fd = open ("/dev/rfl6", 2)) == -1)
      {
              perror ("open /dev/rfl6");
              exit(1);
      }

      if (ioctl (fd, FLSET, (char *) &ptab) == -1)
      {
              perror ("FLSET-ioctl /dev/rfl6");
              exit(1);
      }

      close (fd);
}
```

A.1.2.10 fl6parget.c

static char SCCSID[] = "@(#)fl6parget.c 1.1 85/05/02";

```
#define         FLSET           (('S'<<8)|0)
#define         FLGET           (('G'<<8)|0)

struct ptab {
    int p_cylorg;       /* Anfangszylinder */
    int p_cylct;        /* Anzahl der Zylinder */
    int p_cyloff;       /* Versatz zwischen Zylindern (1=96TPI, 2=48TPI) */
    int p_secsiz;       /* Anzahl der Bytes pro Sektor */
    int p_sectrk;       /* Anzahl der Sektoren pro Spur */
    int p_sides;        /* Anzahl der Seiten: 1 oder 2 */
    int p_mfm;          /* 1 bei MFM (doppelte Dichte), sonst 0 (FM) */
    int p_seclen;       /* Code fuer die Sektorlaenge (0, 1, 2 oder 3) */
    int p_blkct;        /* Anzahl der BSIZE-Byte Bloecke */
    int p_gpl3;         /* Laenge der Blockluecke (Gap) */
};

struct ptab ptab;

main()
{
      int fd;

      if ((fd = open ("/dev/rfl6", 2)) == -1)
      {
              perror ("open /dev/rfl6");
              exit(1);
      }

      if (ioctl (fd, FLGET, (char *) &ptab) == -1)
      {
              perror ("FLGET-ioctl /dev/rfl6");
              exit(1);
      }

      printf("Anfangszylinder: .... %d\n", ptab.p_cylorg);
      printf("Zylinderanzahl: ..... %d\n", ptab.p_cylct);
      printf("Zylinderabstand: .... %d TPI\n", ptab.p_cyloff == 1 ? 96 : 48);
      printf("Sektorlaenge: ....... %d\n", ptab.p_secsiz);
      printf("Sektoren/Spur: ...... %d\n", ptab.p_sectrk);
      printf("Seitenanzahl: ....... %d\n", ptab.p_sides);
      printf("Schreibdichte: ...... %s\n", ptab.p_mfm ? "doppelt (MFM)" : "einfach (FM)");
      printf("Gesamtkapazitaet: ... %d KB\n", ptab.p_blkct);

      close (fd);
}
```

A.1.2.11 ioctl.c

```
/*
 * stty - Kommando gem. SVID Page 195 ff
 *
 * Autor: M. Weitzel
 *
 * Version 1.0 - 30.07.86
 * Version 1.1 - 04.09.86
 *   - Verwendung der IO-Kanaele geaendert
 *   - Hochkommas bei Ausgabe der CCHARS
 *     unter Option "-g" verhindert
 *   - Ausgabeformat komprimiert
*/

#include <stdio.h>

#ifdef MX
#define void int
#include <termio.h>
#else
#include <sys/types.h>
#include <sys/termio.h>
#include <sys/ioctl.h>
#endif

#define ERRFP stderr    /* File-Pointer fuer Fehlermeldungsausgabe */
#define OUTFP stdout    /* File-Pointer fuer normale Ausgabe */
#define IOCFP stdin     /* File-Pointer fuer ioctl-Einstellugen */

static struct termio wtbuff;
#  define IFLAG ((char *) &wtbuff.c_iflag)
#  define OFLAG ((char *) &wtbuff.c_oflag)
#  define CFLAG ((char *) &wtbuff.c_cflag)
#  define LFLAG ((char *) &wtbuff.c_lflag)
#  define LINE  (&wtbuff.c_line)
#  define CC(x) ((char *) &wtbuff.c_ccs[x])

#  define G_FLAG(b,p) (*(unsigned short *) ((char *) &b + (p-(char*)&wtbuff)))
#  define G_CC(b,p)   (*((char *) &b + (p-(char*)&wtbuff)))

struct pd { /* PARAMETER-Beschreibung */
  short ptype;              /* Parameter Typ, Werte wie folgt: */
#  define P_HDR (-1)           /* Boolscher Parameter */
#  define P_BOOL 1             /* Boolscher Parameter */
#  define P_ALTS 2             /* Alternativ Parameter */
#  define P_CHRS 3             /* Character-Wert */
#  define P_NUMS 4             /* Character-Wert */
  char *pname;              /* Name des Parameters */
  char *offset;             /* Offset innerhalb struct termio */
  unsigned short affmsk;    /* Maske fuer betroffene Bits */
  unsigned short pattern;   /* spezielles Muster */
};

static int ALL, GFORM;

struct pd params[] = {
  /*    ptype   pname           offset    affmsk    pattern  */
      { P_HDR,  "CONTROL MODES:"                             },
      { P_BOOL, "parenb",       CFLAG,  PARENB, PARENB },
```

```
        { P_BOOL, "parodd",       CFLAG,  PARODD, PARODD },
        { P_ALTS, "cs5",          CFLAG,  CSIZE,  CS5    },
        { P_ALTS, "cs6",          CFLAG,  CSIZE,  CS6    },
        { P_ALTS, "cs7",          CFLAG,  CSIZE,  CS7    },
        { P_ALTS, "cs8",          CFLAG,  CSIZE,  CS8    },
        { P_ALTS, "0",            CFLAG,  CBAUD,  B0     },
        { P_ALTS, "50",           CFLAG,  CBAUD,  B50    },
        { P_ALTS, "75",           CFLAG,  CBAUD,  B75    },
        { P_ALTS, "110",          CFLAG,  CBAUD,  B110   },
        { P_ALTS, "134",          CFLAG,  CBAUD,  B134   },
        { P_ALTS, "150",          CFLAG,  CBAUD,  B150   },
        { P_ALTS, "200",          CFLAG,  CBAUD,  B200   },
        { P_ALTS, "300",          CFLAG,  CBAUD,  B300   },
        { P_ALTS, "600",          CFLAG,  CBAUD,  B600   },
        { P_ALTS, "1200",         CFLAG,  CBAUD,  B1200  },
        { P_ALTS, "2400",         CFLAG,  CBAUD,  B2400  },
        { P_ALTS, "4800",         CFLAG,  CBAUD,  B4800  },
        { P_ALTS, "9600",         CFLAG,  CBAUD,  B9600  },
        { P_ALTS, "19200",        CFLAG,  CBAUD,  B19200 },
        { P_ALTS, "38400",        CFLAG,  CBAUD,  B38400 },
        { P_BOOL, "hupcl",        CFLAG,  HUPCL,  HUPCL  },
        { P_BOOL, "hup",          CFLAG,  HUPCL,  HUPCL  },
        { P_BOOL, "cstopb",       CFLAG,  CSTOPB, CSTOPB },
        { P_BOOL, "cread",        CFLAG,  CREAD,  CREAD  },
        { P_BOOL, "clocal",       CFLAG,  CLOCAL, CLOCAL },
/*      { P_BOOL, "loblk",          --- not for SINIX !! --- }, */
        { P_HDR,  "INPUT MODES:"                           },
        { P_BOOL, "ignbrk",       IFLAG,  IGNBRK, IGNBRK },
        { P_BOOL, "brkint",       IFLAG,  BRKINT, BRKINT },
        { P_BOOL, "ignpar",       IFLAG,  IGNPAR, IGNPAR },
        { P_BOOL, "parmrk",       IFLAG,  PARMRK, PARMRK },
        { P_BOOL, "inpck",        IFLAG,  INPCK,  INPCK  },
        { P_BOOL, "istrip",       IFLAG,  ISTRIP, ISTRIP },
        { P_BOOL, "inlcr",        IFLAG,  INLCR,  INLCR  },
        { P_BOOL, "igncr",        IFLAG,  IGNCR,  IGNCR  },
        { P_BOOL, "icrnl",        IFLAG,  ICRNL,  ICRNL  },
        { P_BOOL, "iuclc",        IFLAG,  IUCLC,  IUCLC  },
        { P_BOOL, "ixon",         IFLAG,  IXON,   IXON   },
        { P_BOOL, "ixany",        IFLAG,  IXANY,  IXANY  },
        { P_BOOL, "ixoff",        IFLAG,  IXOFF,  IXOFF  },
        { P_HDR,  "OUTPUT MODES:"                          },
        { P_BOOL, "opost",        OFLAG,  OPOST,  OPOST  },
        { P_BOOL, "olcuc",        OFLAG,  OLCUC,  OLCUC  },
        { P_BOOL, "onlcr",        OFLAG,  ONLCR,  ONLCR  },
        { P_BOOL, "ocrnl",        OFLAG,  OCRNL,  OCRNL  },
        { P_BOOL, "onocr",        OFLAG,  ONOCR,  ONOCR  },
        { P_BOOL, "onlret",       OFLAG,  ONLRET, ONLRET },
        { P_BOOL, "ofill",        OFLAG,  OFILL,  OFILL  },
        { P_BOOL, "ofdel",        OFLAG,  OFDEL,  OFDEL  },
        { P_ALTS, "cr0",          OFLAG,  CRDLY,  CR0    },
        { P_ALTS, "cr1",          OFLAG,  CRDLY,  CR1    },
        { P_ALTS, "cr2",          OFLAG,  CRDLY,  CR2    },
        { P_ALTS, "cr3",          OFLAG,  CRDLY,  CR3    },
        { P_ALTS, "nl0",          OFLAG,  NLDLY,  NL0    },
        { P_ALTS, "nl1",          OFLAG,  NLDLY,  NL1    },
        { P_ALTS, "tab0",         OFLAG,  TABDLY, TAB0   },
        { P_ALTS, "tab1",         OFLAG,  TABDLY, TAB1   },
        { P_ALTS, "tab2",         OFLAG,  TABDLY, TAB2   },
        { P_ALTS, "tab3",         OFLAG,  TABDLY, TAB3   },
        { P_ALTS, "bs0",          OFLAG,  BSDLY,  BS0    },
```

```
      { P_ALTS, "bs1",          OFLAG,  BSDLY,  BS1    },
      { P_ALTS, "ff0",          OFLAG,  FFDLY,  FF0    },
      { P_ALTS, "ff1",          OFLAG,  FFDLY,  FF1    },
      { P_ALTS, "vt0",          OFLAG,  VTDLY,  VT0    },
      { P_ALTS, "vt1",          OFLAG,  VTDLY,  VT1    },
      { P_HDR,  "LOCAL MODES:"                                 },
      { P_BOOL, "isig",         LFLAG,  ISIG,   ISIG   },
      { P_BOOL, "icanon",       LFLAG,  ICANON, ICANON },
      { P_BOOL, "xcase",        LFLAG,  XCASE,  XCASE  },
      { P_BOOL, "echo",         LFLAG,  ECHO,   ECHO   },
      { P_BOOL, "echoe",        LFLAG,  ECHOE,  ECHOE  },
      { P_BOOL, "echok",        LFLAG,  ECHOK,  ECHOK  },
      { P_BOOL, "lfck",         LFLAG,  ECHOK,  ECHOK  },
      { P_BOOL, "echonl",       LFLAG,  ECHONL, ECHONL },
      { P_BOOL, "noflsh",       LFLAG,  NOFLSH, NOFLSH },
      { P_HDR,  "CONTROL ASSIGNMENTS:"                         },
      { P_CHRS, "erase",        CC(VERASE)                   },
      { P_CHRS, "kill",         CC(VKILL)                    },
      { P_CHRS, "intr",         CC(VINTR)                    },
      { P_CHRS, "quit",         CC(VQUIT)                    },
/*    { P_CHRS, "swtch",        --- not for SINIX !! ---     }, */
      { P_CHRS, "eof",          CC(VEOF)                     },
      { P_CHRS, "eol",          CC(VEOL)                     },
      { P_NUMS, "min",          CC(VMIN)                     },
      { P_NUMS, "time",         CC(VTIME)                    },
      { P_NUMS, "line",         LINE                           }
    };
#define nPARAMS (sizeof params / sizeof *params)

static char *dispcc(c)  /* FUNKTIONIERT NUR FUER ASCII ! */
  int c;

{ static char buff[] = "^ ";

  buff[1] = (c & 0x80)    ? '-'     :
            (c == '}177') ? '?'     :
            (c < ' ')     ? c + '@' :
                            c;
  return ((c < ' ' || c >= '}177') ? buff : buff+1);
}

static char *dispbd(baud)
  unsigned short baud;

{
  struct pd *pp = params;

  for (pp = params; pp < params + nPARAMS; pp++)
    if (pp->offset == CFLAG && pp->affmsk == CBAUD && pp->pattern == baud)
      return (pp->pname);
  return ("???");
}

static int dispcs(cs)
  unsigned short cs;

{
  switch (cs) {
  case CS5: return (5);
  case CS6: return (6);
```

```
  case CS7: return (7);
  case CS8: return (8);
  }
  return (0);
}

static char *disppb(par)
  unsigned short par;

{
  return ((par & PARENB) ? ((par & PARODD) ? "odd" : "even") : "no");
}

static void gdisplay()

{
  static struct termio mrkbuff; /* Hilfsvariable */
  register int i;               /* Hilfsvariable */
  struct pd *pp;                /* Hilfsvariable */
  static int COLUMNS = 60;      /* Zeichen pro Zeile */
  static char obuff[100];

  /* Initialisierung der Hilfsvariablen */
  if (ALL) {
    mrkbuff.c_iflag =
    mrkbuff.c_oflag =
    mrkbuff.c_cflag =
    mrkbuff.c_lflag = ~0;
    mrkbuff.c_line  = ~0;
    for (i = 0; i < NCC; i++)
      mrkbuff.c_ccs[i] = ~0;
  }
  else
  { mrkbuff.c_iflag = ICRNL|IUCLC|IXON|IXOFF;
    mrkbuff.c_oflag = OLCUC;
    mrkbuff.c_cflag = CBAUD|CSIZE|PARENB|PARODD;
    mrkbuff.c_lflag = ISIG|ICANON|ECHO;
    mrkbuff.c_ccs[VERASE] = ~0;
    mrkbuff.c_ccs[VKILL] = ~0;
  }

  for (i = 0, pp = params; pp < params + nPARAMS; pp++) {
    switch (pp->ptype) {
    case P_HDR:
      if (!GFORM) {
      if (i > 0) { fprintf (OUTFP, "}n"); i = 0; }
      fprintf(OUTFP, "%s}n", pp->pname);
      }
      break;
    case P_BOOL:
      if (G_FLAG(mrkbuff, pp->offset) & pp->affmsk) {
      sprintf (obuff, (G_FLAG(wtbuff, pp->offset) & pp->affmsk) == pp->pattern
              ? "%s" : "-%s", pp->pname);
      G_FLAG(mrkbuff, pp->offset) &= ~pp->pattern;
      }
      break;
    case P_CHRS:
      if (G_CC(mrkbuff, pp->offset)) {
      sprintf (obuff, "%s %s%s%s", pp->pname, GFORM ? "" : "}'",
        dispcc(G_CC(wtbuff, pp->offset)), GFORM ? "" : "}'");
```

```
      G_CC(mrkbuff, pp->offset) = 0;
      }
      break;
    case P_NUMS:
      if (G_CC(mrkbuff, pp->offset)) {
      sprintf (obuff, "%s %d", pp->pname, G_CC(wtbuff, pp->offset));
      G_CC(mrkbuff, pp->offset) = 0;
      }
      break;
    case P_ALTS:
      if (G_FLAG(mrkbuff, pp->offset) & pp->affmsk) {
      if ((G_FLAG(wtbuff, pp->offset) & pp->affmsk) == pp->pattern) {
        sprintf (obuff, "%s", pp->pname);
        G_FLAG(mrkbuff, pp->offset) &= ~pp->affmsk;
      }
      }
      break;
    default:
      fprintf(ERRFP, "ioctl: internal error: params at pos %d}n", pp - params);
      abort();
    }
    if (obuff[0]) {
      if (i > 0) {
      if (i + strlen(obuff) + 1 >= COLUMNS && !GFORM) {
        fprintf(OUTFP, "}n");
        i = 0;
      }
      else {
        fprintf(OUTFP, " ");
        i++;
      }
      }
      fprintf(OUTFP, "%s", obuff);
      i += strlen(obuff);
      obuff[0] = '}0';
    }
  }
  if (i > 0) fprintf(OUTFP, "}n");
}

static int getvnu(s)
  char *s;

{ int val = 0;

  while (*s) {
    if (*s < '0' || *s > '9') return(-1);
    val = 10*val + *s++ - '0';
    if (val > 127) return (-1);
  }
  return (val);
}

static int getvcc(s)
  char *s;

{ int cntrl;

  if (cntrl = (*s == '^')) s++;
  if (s[1]) return (-1);
```

```
  if (!cntrl) return (*s);
  if (*s == '?') return (0177);
  if (*s == '-') return (0377);
  if ('a' <= *s && *s <= 'z') *s -= 'a'-'A';
  return ((*s -= '@') < 0 || *s > 31 ? -1 : *s);
}

static void setparms(av)
  char **av;

{
  struct pd *pp;                /* Hilfsvariable */

  while (*++av)
    if (**av) {
      int c, found;
      int swoff = (**av == '-');
      char *hp = swoff ? *av + 1 : *av;

      for (pp = params; pp < params + nPARAMS; pp++)
      if (found = !strcmp(pp->pname, hp)) break;
      if (found) {
      switch (pp->ptype) {
      case P_HDR: break;
      case P_CHRS:
        if ((c = getvcc(*++av)) == -1)
          fprintf(ERRFP, "ioctl: Bad Cchar <%s> '%s'}n", hp, av[0]);
        else
          G_CC(wtbuff, pp->offset) = c;
        break;
      case P_NUMS:
        if ((c = getvnu(*++av)) == -1)
          fprintf(ERRFP, "ioctl: Bad Number <%s> '%s'}n", hp, av[0]);
        else
          G_CC(wtbuff, pp->offset) = c;
        break;
      case P_ALTS:
        G_FLAG(wtbuff, pp->offset) &= ~pp->affmsk;
        G_FLAG(wtbuff, pp->offset) |= pp->pattern;
        break;
      case P_BOOL:
        G_FLAG(wtbuff, pp->offset) &= ~pp->affmsk;
        if (!swoff)
          G_FLAG(wtbuff, pp->offset) |= pp->pattern;
        break;
      } /* switch */
      } /* if */
      else
      fprintf(ERRFP, "ioctl: Unknow Option <%s>}n", *av);
    }
}

main (ac, av)
  int ac;
  char **av;

{ char **pp = av;

  while (*++pp) {
    if (!strcmp(*pp, "-g")) { GFORM = 1; **pp = '}0'; continue; }
```

```
    if (!strcmp(*pp, "-a")) { ALL   = 1; **pp = '}0'; continue; }
    if (!strcmp(*pp, "-ga")
     || !strcmp(*pp, "-ag")) { GFORM = ALL = 1; **pp = '}0'; continue; }
  }

  if (ioctl(fileno(IOCFP), TCGETA, &wtbuff) == -1) {
    perror(av[0]); exit(1);
  }

  if (ac > 1) {
    setparms(av);
    ioctl(fileno(IOCFP), TCSETA, &wtbuff);
  }

  if (ac == 1 || ALL || GFORM)
    gdisplay();

}
```

A.1.2.12 eing.c

```
static char SCCSID[] = "@(#)eing.c 1.2 85/05/07";
/* Programm zum Einlesen  von ASCII-Dateien ueber einen tty Kanal des
 * PC-MX.                                                          SINIX 1.0B
 *            **   Minimalversion fuer eilige  **
 * Zum Einlesen muss entsprechend dem gewuenschten Kanal eine Geraetedatei
 * eingerichtet sein.
 * Geraetedatei einrichten  im Dateiverzeichnis  /dev :
 * - einlesen ueber Fbg. CONAC    (Druckeranschluss)
 *   /etc/mknod datlese c 3 1              Geraetedatei "datlese"  einrichten
 * - einlesen ueber Fbg. SERAC    (Schnittst. 97  Kanal 5)
 *   /etc/mknod datlese c 8 105            Geraetedatei "datlese"  einrichten
 *
 *
*/

#define    PMODE  0666
#define    ETX    3
#include   <stdio.h>
#include   <sgtty.h>

main()
{
      int   fdfile, fdtty;
      char  n, fnam[9];
      char  modus;
      printf("        ************************************************************");

      printf("}nSie wollen eine Datei empfangen ?}n");
      printf("Welchen Dateinamen soll die Datei am PC erhalten ?}n");
      printf("Bitte Dateiname eingeben.}n");

      scanf ("%s", fnam);

      if ((fdfile = creat(fnam,PMODE)) == -1)
      {
             printf ("}n}n%s kann nicht eingerichtet werden}n}007",fnam);
             exit(1);
      }

      if ((fdtty = open("/dev/datlese",2)) == -1)
      {
             printf("}n}n0PEN - Fehler auf /dev/datlese}n}007");
             exit(1);
      }

     ssinit (fdtty);

      while ((read (fdtty, &n, 1)) == 1)
      {
            n &= 0x7f;
              if (n == ETX) break;
              putchar(n);
              if ((write (fdfile, &n, 1)) != 1)
              {
                     printf ("}n}nSchreibfehler auf %s}n}007", fnam);
                     exit(1);
              }
      }
      printf ("}n}nDatei %s uebertragen}n}007", fnam);
      exit(0);
ende:
      printf("}n}nEnde}n}007");
}
```

```
ssinit (fdtty)
int fdtty;
{
      struct sgttyb ttypar;

      ttypar.sg_ispeed = B600;
      ttypar.sg_ospeed = B600;
      ttypar.sg_flags = 0;
      ttypar.sg_flags |= RAW;
      ttypar.sg_flags |= TANDEM;

      ioctl (fdtty, TIOCSETP, &ttypar);
}
```

A.1.2.13 gettty.c

```
static char SCCSID[] = "@(#)gettty.c 1.3 85/06/13";

/*-----------------------------------------------------------------------------
 * gettty.c
 *
 * Programm zum Lesen von ASCII-Dateien ueber einen TTY-Kanal
 *
 *-----------------------------------------------------------------------------
 *
 * Immer wieder besteht die Notwendigkeit, Quell-Dateien im ASCII-Format von
 * anderen Rechnern zu uebernehmen.
 * Wenn die Uebernahme ueber Floppy-Disk nicht moeglich ist, bleibt noch die
 * Moeglicheit der direkten Kopplung der beiden Rechner ueber eine der verfueg-
 * baren Standard-Schnittstellen, also ueber V24/V28 oder SS97/V11.
 *
 * Diese Kopplung ist jedoch nicht ganz problemlos, da neben der rein physikali-
 * schen Kopplung (PIN-Belegung, Spezialkabel usw.) auch das Problem der Fluss-
 * Steuerung, der Datei-Ende-Erkennung und der Zeilen-Abschluss-Steuerzeichen
 * geloest werden muss.
 *
 * Fuer die Fluss-Steuerung gibt es folgende Loesungsansaetze:
 * ----------------------------------------------------------
 *
 * - Sehr langsame Uebertragungsrate, z.B. 300 Bd, oder Verzoegerungstimer
 *   nach jedem "Block" von der Groesse des Empfangspuffers:
 *
 *      Dies muss immer als Notloesung fuer kurze Dateien betrachtet werden.
 *      Ausserdem ist das Risiko vorhanden, dass doch Zeichen verlorengehen.
 *
 * - Fluss-Steuerung durch die Steuerzeichen DC3/DC1:
 *
 *      Dies setzt voraus, dass sowohl beim "Sender" als auch beim "Empfaenger"
 *      diese Steuerzeichen bedient werden.
 *      Bei der 9780/9781 trifft dies jedoch mit  SINIX V1.0B in Eingabe-
 *      Richtung nicht zu!
 *      Bei SINIX V1.0C ist diese Einschraenkung behoben.
 *
 *      Fuer den "Sender" ist dies haeufig eine sehr einfache Moeglichkeit, da
 *      meist Drucker-Anschluesse mit diesem Protokoll implementiert sind, und
 *      der "Empfaenger" einfach anstelle des Druckers angeschlossen wird.
 *
 *      Beispiel fuer CP/M-Systeme:     pip lst:=DATEI,eof[t8]
 *                                      (eof ist Datei mit SINIX-EOF-Zeichen)
 *
 *      Beim Empfaenger kann die Datei ebenso einfach empfangen werden, wenn
 *      eine entsprechende Kanal-Datei eingerichtet wurde.
 *      (/etc/mknod /dev/kdatei c 8 105 --> Beispiel fuer SERAC-Kanal-5).
 *
 *      Beispiel fuer SINIX-V-1.0C:  cat /dev/kdatei | tr -d '}015' > DATEI
 *
 *      Bei diesen Beispielen ist wichtig, dass das SINIX-EOF-Zeichen ueber die
 *      Leitung kommt, um die Uebertragung zu beenden. Bei einem Abbruch mit der
 *      DEL-Taste werden naemlich die noch vorhandenen Daten im Empfangspuffer
 *      geloescht und die uebertragene Datei ist daher unvollstaendig.
 *      Ausserdem ist darauf zu achten, dass evtl. eine Korrektur der Zeilen-
 *      ende-Steuerzeichen erforderlich ist. Diese sind bei CP/M z.B. <cr><lf>
 *      und bei SINIX nur <lf>. Eine Umsetzung kann in diesem Fall durch
 *      <tr -d '}015'> wie oben gezeigt erfolgen.
 *      Die Fluss-Steuerung ist jedoch noch keine Gewaehr fuer die richtige
```

```
 *      und vollstandige Uebertragung der Daten. Durch "missing-interrupts"
 *      verlorengegangene Zeichen werden so nicht erkannt, dies kann nur auf
 *      Anwendungsebene durch entsprechende Block-Pruef-Zeichen erfolgen.
 *
 * - Fluss-Steuerung durch Protokoll auf Anwender-Ebene:
 *
 *      Dies setzt voraus, dass beim "Sender" und "Empfaenger" entsprechende
 *      Anwender-Programme zur Verfuegung stehen.
 *      Abgesehen vom erforderlichen Aufwand hat dies den Vorteil, dass alle
 *      logischen Probleme, wie Fluss-Steuerung, Zeilenende-Kriterium und
 *      Dateiende-Kriterium geloest werden koennen.
 *      Ausserdem koennen noch die Uebertragung transparenter Dateien, die
 *      Blockwiederholung im Fehlerfall und entsprechender Bedienungs-Komfort
 *      relativ einfach implementiert werden.
 *      Beispiele sind das Standard-Programm <uucp> zur Kopplung von unter-
 *      schiedlichen UNIX-Systemen und das bei VS 1133 implementierte Programm
 *      <ft> zur Kopplung von UNIX-Systemen mit dem Siemens CP/M-Rechner 9753.
 *      Letzteres ermoeglicht nach entsprechender Umsetzung auch das Lesen
 *      von IBM-BIF-Disketten.
 *
 *-------------------------------------------------------------------------
 * Folgendes Beispiel zeigt prinzipiell das Lesen von Daten ueber einen
 * TTY - Kanal unter der Voraussetzung, dass die Fluss-Steuerung mit DC3/DC1
 * funktioniert.
 * Es wird nochmals darauf hingewiesen, dass dies bei der SINIX-V-1.0B nicht
 * der Fall ist, das Programm also reinen Test-Charakter hat.
 * Es wird vorausgesetzt, dass eine entsprechende Kanal-Datei mit dem Kommando
 * /etc/mknod eingerichtet wurde. Der Name der Kanal-Datei wird abgefragt, um
 * Tests mit verschiedenen Kanaelen (Schnittstellen) zu ermoeglichen.
 * Der Kanal wird entsprechend der XENIX-V7-TTY-Schnittstelle parametrisiert.
 * Die entsprechende Struktur steht in </usr/include/sgtty.h>.
 * Um fuer Tests flexibel zu sein, stehen die Kanal-Parameter in Form von
 * Schluesselworten in einer Datei, deren Name ebenfalls abgefragt wird.
 * Die gelesenen Daten werden nach <stdout> geschrieben und koennen daher ueber
 * eine Pipe in eine beliebige Datei umgeleitet werden.
 * Nach Beenden des Programmes werden die alten Kanal-Parameter wiederher-
 * gestellt.
 * Ein vorzeitiger Abbruch durch die DEL-Taste ist moeglich, dies ist auch
 * notwendig, wenn kein EOF-Zeichen am Ende der Datei kommt, in diesem Fall
 * gehen keine gelesenen Zeichen verloren.
 *
 *-------------------------------------------------------------------------
 */

#define         EOFL      4                                   /* Datei-Ende */
#define         ESC     0x1b
#define         ERR     (-1)
#define         MAX     128

#include     <stdio.h>
#include     <signal.h>
#include        <sgtty.h>

int             fdk;                            /* File-Descriptor Kanaldatei */

int             onintr ();

/*-------------------------------------------------------------------------*/
main ()

{
```

```
FILE            *fpp, *fopen ();          /* File-Pointer Kanalparameterdatei */
char            kdatei[64];                                 /* Kanal-Datei */
char            pdatei[64];                          /* Kanal-Parameter-Datei */
char            c;                                 /* Puffer fuer <read>-Befehl */
int             n;

fprintf (stderr, "}n%c[7m Testprogramm zum Lesen von Daten ueber einen}
 TTY-Kanal: %c[m", ESC, ESC);
fprintf (stderr, "}n}nGeben Sie den Kanal-Datei-Namen ein              : ");
scanf ("%s", kdatei);
if  ((fdk = open (kdatei, 0)) == ERR)
    {
    fprintf (stderr, "}nKanal-Datei <%s> kann nicht geoeffnet werden}n",kdatei);
    exit (1);
    }
fprintf (stderr, "Geben Sie den Namen der Kanal-Parameter-Datei ein : ");
scanf ("%s", pdatei);
if  ((fpp = fopen (pdatei, "r")) == NULL)
    {
    fprintf (stderr, "}nKanal-Parameter-Datei <%s> kann nicht geoeffnet}
 werden}n", pdatei);
    exit (1);
    }

ssinit (fdk, fpp);                              /* TTY-Kanal parametrisieren */

putc ('}n', stderr);
sprintf (pdatei, "stty >%s", kdatei); system (pdatei);  /* Parameter anzeigen */
putc ('}n', stderr);

while ((n = read (fdk, &c, 1)) == 1)
    {
    if  (c == EOFL)     break;                                  /* Datei-Ende */
    else                write (1, &c, 1);
    }
if  (n == ERR)          fprintf (stderr, "}nLesefehler !!}n");
else                    fprintf (stderr, "}nDatei-Ende   }n");
ssrein (fdk);
exit (n);
}

/*
 *-------------------------------------------------------------------------------
 * ssinit (fdk, fpp)            Parametrisieren eines TTY-Kanals
 * ssrein (fdk)                 Alte Parameter wiederherstellen
 *
 * fdk : File-Descriptor der Kanal-Datei
 * fpp : File-Pointer der Kanal-Parameter-Datei
 *
 *-------------------------------------------------------------------------------
 *
 * Beschreibung der Kanal-Parameter-Datei:
 *
 * Die Auswahl der Parameter erfolgt durch folgende Schluesselworte:
 *
 *       COOKED  CBREAK   RAW          Uebertragungs-Modus
 *       1200 ... 38400                Uebertragungs-Geschwindigkeit
 *       ODD  EVEN  NO                 Parit{tsbit
 *
 *-------------------------------------------------------------------------------*/
```

```
struct  sgttyb  ttyoldp;                                /* TTY-Parameter-Struktur */

/*-------------------------------------------------------------------------------*/
ssinit (fdk, fpp)                                  /* TTY-Kanal parametrisieren */
                                                  /*---------------------------*/
int     fdk;                                      /* File-Descriptor Kanal-Datei */
FILE    *fpp;                                /* File-Pointer Kanal-Parameter-Datei */

{
#define BUF     128                           /* Max Groesse der Parameterdatei */
#define SMAX     11                             /* Anzahl der Schluesselwoerter */

int                     d;
register unsigned short i, j, mode, speed, par;
char                    c, rbuf[BUF], *str[SMAX+1];
struct sgttyb           ttypar;

str[0]  = "RAW";                                                /* Schluesselworte */
str[1]        = "CBREAK";
str[2]        = "COOKED";
str[3]  = "1200";
str[4]        = "2400";
str[5]        = "4800";
str[6]        = "9600";
str[7]        = "19200";
str[8]  = "38400";
str[9]  = "NO";
str[10] = "EVEN";
str[11] = "ODD";
                                                    /* Standards :           */
mode = CBREAK;                                        /* CBREAK - Modus      */
speed = B1200;                                        /* 1200 Baud           */
par = ODDP;                                           /* ungerade Paritaet   */

for (i = 0; i < BUF; i++)                          /* Parameter-Datei einlesen */
    {
    if  ((d = getc (fpp)) == ERR)       break;
    rbuf[i] = d;
    }
for (j = 0; j <= SMAX; j++)
    {
    for (i = 0; i < BUF; i++)
      {
      if  (*str[j] == rbuf[i])        break;
      }
    if  (i == BUF)                      continue;
    while ((c = *(++str[j])) > ' ')
      {
      if  (c != rbuf[++i])            break;
      }
    if  (c > ' ' || rbuf[i+1] > ' ')    continue;
    switch (j)
      {
      case 0  : mode = RAW; break;
      case 1  : mode = CBREAK; break;
      case 2  : mode = 0; break;
      case 3  : speed = B1200; break;
      case 4  : speed = B2400; break;
      case 5  : speed = B4800; break;
      case 6  : speed = B9600; break;
      case 7  : speed = EXTA; break;
```

```
        case 8  : speed = EXTB; break;
        case 9  : par = 0; break;
        case 10 : par = EVENP; break;
        case 11 : par = ODDP; break;
        default : break;
        }
    }

ioctl (fdk, TIOCGETP, &ttyoldp);          /* Parameter des TTY-Kanals sichern */

signal (SIGINT, onintr);                                    /* DEL-Taste */
signal (SIGQUIT, SIG_IGN);                                  /* QUIT-Taste */

ttypar.sg_ispeed =  speed;                        /* neue Parameter einstellen */
ttypar.sg_ospeed =  speed;                                    /* Baudrate */
ttypar.sg_kill   =  ttyoldp.sg_kill;
ttypar.sg_erase  =  ttyoldp.sg_erase;
ttypar.sg_flags  =  0;
ttypar.sg_flags  |= mode;                                       /* Modus */
ttypar.sg_flags  |= par;                                       /* Parity */
ttypar.sg_flags  |= TANDEM;                          /* DC1 - DC3 - Protokoll */

ioctl (fdk, TIOCSETP, &ttypar);                       /* TTY-Kanal umschalten */
ioctl (fdk, TIOCEXCL, &ttypar);                       /* Exclusiv-Modus setzen */
}

/*-------------------------------------------------------------------------*/
ssrein (fdk)                /* alte Parameter des TTY-Kanals wiederherstellen */
                           /*-----------------------------------------------*/
int     fdk;                                  /* File-Descriptor Kanal-Datei */

{
ioctl (fdk, TIOCSETP, &ttyoldp);          /* Parameter des TTY-Kanals sichern */
close (fdk);                                        /* Kanal-Datei schliessen */
}

/*-------------------------------------------------------------------------*/
onintr ()                                                       /* DEL - Taste */

{ ssrein (fdk); exit (1); }

/*-------------------------------------------------------------------------*/
```

A.1.2.14 zed.c

```
static char SCCSID[] = "@(#)zed.c 1.3 85/06/13";

/*-----------------------------------------------------------------------------
 *
 * Programm-Beispiel : Zeilen-Editor
 *
 *-----------------------------------------------------------------------------
 *
 * Nach Ausgabe eines PROMPTS kann der Rest der Bildschirmzeile editiert werden.
 * Nach  Betaetigen der ENTER-Taste wird die Zeile als Kommando an eine Sub-Shell
 * uebergeben.
 *
 * Folgende Funktionstasten werden unterstuetzt:
 *
 * <Cursor-links>
 * <Cursor-rechts>
 * <Wort-links>
 * <Wort-rechts>
 * <Cursor-home>        Cursor auf Zeilenanfang.
 * <Zeichen-einfuegen>  An der Cursor-Position wird ein Zeichen eingefuegt.
 * <Zeichen-ausfuegen>  Das Zeichen an der Cursor-Position wird ausgefuegt.
 * <Backspace>          Das Zeichen links der Cursor-Position wird ausgefuegt.
 * <Wort-ausfuegen>     Das Wort bzw. der Wortrest ab dem Cursor rechts
 *                      wird ausgefuegt.
 * <Wort-einfuegen>     Alle nachfolgenden Zeichen inklusive dem ersten Space
 *                      werden eingefuegt.
 * <Zeile-loeschen>     Der Zeilenrest ab dem Cursor wird geloescht.
 * <DEL>                Eingabe abbrechen.
 * <END>                Der Zeileneditor wird beendet.
 * <ENTER>              Die Eingabezeile wird einer Sub-Shell als Kommando
 *                      uebergeben, wobei die Cursor-Position unerheblich ist.
 *
 * Es koennen auch Control-Codes eingegeben und editiert werden. Diese werden
 * invers dargestellt.
 * Soll ESC oder eine Funktionstaste in den Zeilenpuffer eingegeben werden,
 * so ist vorher die ESC-Taste einmal zusaetzlich zu betaetigen.
 * Alle nicht verwendeten Funktionstasten werden ignoriert, zur Warnung wird
 * <BEL> ausgegeben.
 *
 *-----------------------------------------------------------------------------
 *
 * Dieses Beispiel zeigt die Bedienung der Schnittstellen zur Tastatur,
 * dem Bildschirm, und zum TTY-Kanal.
 *
 * Bei der Tastatur wird die einfache Abfrage der Funktionstasten gezeigt,
 * die ja Code-Folgen absetzen koennen.
 *
 * Beim Bildschirm wird der sinnvolle Einsatz einiger Steuerzeichenfolgen
 * gezeigt.
 *
 * Der Einfachheit halber werden die Steuerzeichenfolgen durch <#define>-
 * Anweisungen definiert. Es soll jedoch darauf hingewiesen werden, dass diese
 * der Portabilitaet wegen aus der Datei </etc/termcap> geholt werden sollten,
 * wie an anderer Stelle des Buches ausfuehrlich beschrieben.
 *
 * Um einen Editor zu implementieren, ist die Umschaltung des TTY-Kanals erfor-
 * derlich, da standardmaessig der sogenannte <cooked-mode> eingestellt ist, der
 * ja mit Ausnahme der Backspace-Taste keinerlei Editiermoeglichkeit zulaesst
 * und erst nach Betaetigen der <ENTER>-Taste eine komplette Eingabezeile dem
```

```
 * Anwenderprogramm zur Verfuegung stellt. Dieses kann daher nicht auf jedes
 * eingegebene Zeichen sofort reagieren. Ausserdem muss das automatische <ECHO>
 * ausgeschaltet werden, um ungueltige Eingaben abfangen zu koennen.
 *
 * Ab der SINIX-Version 1.0B kann das Umschalten des TTY-Kanals auf zwei Arten
 * erfolgen, und zwar entsprechend XENIX-V7 oder entsprechend XENIX-S-III.
 * Die Beschreibung der TTY-Schnittstelle und der Umschaltmoeglichkeiten erfolgt
 * an anderer Stelle dieses Buches.
 * Bei diesem Beispiel erfolgt das Umschalten entsprechend der XENIX-S-III-
 * Schnittstelle.
 *
 * Da zum Zeitpunkt der Implementierung dieses Beispiels noch keine entspre-
 * chende Header-Datei <termio.h> in </usr/include> zur Verfuegung stand, wurden
 * die verwendeten Kommandos und Parameter und die entsprechende Struktur
 * explizit angegeben.
 */

#define         TCSETA          ('T'<<8|2)          /* Setzen TTY-Parameters */
#define         TCGETA          ('T'<<8|1)        /* Abfragen TTY-Parameters */
#define         T_ICRNL         0000400                         /* cr --> nl*/
#define         T_INLCR         0000100                         /* nl --> cr*/
#define         T_IGNCR         0000200                    /* ignorieren cr */
#define         T_IXON          0002000          /* Flussteuerung bei Ausgabe */
#define         T_IXOFF         0010000          /* Flussteuerung bei Eingabe */
#define         T_IUCLC         0001000 /* Umsetzen Gross- in Kleinbuchstaben */
#define         T_ISIG          0000001                  /* Signale zulassen */
#define         T_ICANON        0000002                    /* Canonical-Mode */
#define         T_ECHO          0000010                         /* Echo-Mode */
#define         T_VMIN          4  /* Distanz zu Zeichenanz. fuer read-Befehl */
#define         T_VTIME         5   /* Distanz zu max. Zeit. fuer read-Befehl */
#define         T_NCC           8                /* Anzahl der Steuerzeichen */

struct  termio  {               /* TTY-S-III-Struktur, Ergebnis des ioctl () */
            unsigned short  c_iflag;                     /* Eingabe-Modus */
            unsigned short  c_oflag;                     /* Ausgabe-Modus */
            unsigned short  c_cflag;                    /* Kontroll-Modus */
            unsigned short  c_lflag;                       /* Lokal-Modus */
            char            c_line;                   /* line-diszipline */
            char            c_cchar[T_NCC];             /* Steuerzeichen */
            } argp;

/*------------------------------------------------------------------------------*/

#include        <stdio.h>
#include        <signal.h>

/*------------------------------------------------------------------------------*/

#define         LNG       77                /* max Laenge der Eingabe-Zeile */
#define         AUS        0
#define         EIN        1
#define         ERROR    (-1)

/*------------------------------------------------------------------------------
 * Einige verwendete Steuerzeichen :
 */

#define         SO        14                       /* optionaler Zeichensatz */
#define         SI        15                     /* Standard-Zeichensatz */
#define         BEL        7                        /* akustischer Alarm */
#define         ESC     0x1b
```

```
/*-----------------------------------------------------------------------------
 * Die folgenden <#define>-Zuweisungen werden in <printf>-Kommandos verwendet
 * und dienen nur der besseren Lesbarkeit des Programmes.
 */

#define         INV     "%c[7m",ESC                         /* Attribut invers */
#define         NORMAL  "%c[m",ESC                          /* Attribut normal */
#define         G1LAD   "%c)w",ESC                  /* Klammernsatz in opt. ZV */
#define         PROMPT  "}r}n%ck%c ",SO,SI                  /* Pfeil als Prompt */
#define         ZRLOE   "%c[OK",ESC                     /* Zeilenrest loeschen */
#define         BRLOE   "%c[OJ",ESC                       /* Bildrest loeschen */
#define         CLEFT   "%c[%dD",ESC /* ,## */           /* printf (CLEFT,#); */
#define         CRIGHT  "%c[%dC",ESC /* ,## */          /* printf (CRIGHT,#); */
#define         CUP     "%c[A",ESC                          /* printf (CUP); */
#define         CINS    "%c[@",ESC                         /* printf (CINS); */
#define         CDEL    "%c[%dP",ESC /* ,## */           /* printf (CDEL,#); */
#define         RADIER  "}b }b"                                     /* Radierer */

/*-----------------------------------------------------------------------------
 * Die folgenden <#define>-Zuweisungen dienen der Definition der Steuertasten
 * in Verbindung mit der Routine <getchr ()>, wobei die Mehr-Zeichen-Codes
 * durch einen Code mit entsprechendem Offset ersetzt werden, um diese mit
 * einem <switch>-Kommando einfach auswerten zu koennen.
 */

#define         TENTER  '}r'                                    /* Taste ENTER */
#define         TBS       8                       /* Back-Space == Radierer */
#define         TEND      4                                 /* Taste END */
#define         TDEL    127                                 /* Taste DEL */

#define         ESCOFF  128                 /* Offset fuer Funktions-Tasten */
#define         CSIOFF  256                 /*             - " -            */

#define         TESC    ESCOFF + ESC        /* Taste ESC == 2 * ESC druecken */
#define         THOME   CSIOFF + 'H'                        /* Cursor home */
#define         TCLEFT  CSIOFF + 'D'                       /* Cursor links */
#define         TCRIGHT CSIOFF + 'C'                      /* Cursor rechts */
#define         TWLEFT  ESCOFF + '9'                        /* Wort links */
#define         TWRIGHT ESCOFF + ':'                       /* Wort rechts */
#define         TCINS   CSIOFF + '@'                 /* Zeichen einfuegen */
#define         TCDEL   CSIOFF + 'P'                 /* Zeichen ausfuegen */
#define         TWDEL   ESCOFF + 'p'                    /* Wort ausfuegen */
#define         TWINS   ESCOFF + 'o'                    /* Wort einfuegen */
#define         TLDEL   CSIOFF + 'M'                /* Zeilenrest loeschen */

/*-----------------------------------------------------------------------------*/
char            cdo[LNG+1];                                 /* Zeilenpuffer */
unsigned short  lng = 0;                                   /* Cursor-Position */
unsigned short  iflag, lflag;                         /* alte TTY-Parameter */
char            vmin, vtime;
unsigned short  iflag1, lflag1;                       /* neue TTY-Parameter */
char            vmin1, vtime1;

/*-----------------------------------------------------------------------------*/
main ()

{
unsigned short  c = 0;                                 /* aktuelles Zeichen */
unsigned short  old = 0;                               /* letztes Zeichen */
unsigned short  wi = AUS;                     /* Flag bei Einfuegen-Wort */
int             id = ERROR;                                 /* process-id */
int             stat;                                   /* wait - status */
```

```
unsigned short  i, j, k, l;                                  /* Hilfs-Integers */

signal (SIGINT, SIG_IGN);                              /* Signal-INT ignorieren */
signal (SIGQUIT, SIG_IGN);                            /* Signal-QUIT ignorieren */
ttysav ();                                             /* TTY-Parameter sichern */
ttyneu ();                                 /* TTY-umschalten : RAW-Modus ohne Echo */
printf (G1LAD);                           /* Klammern-Zeichensatz in optionalen ZV */
printf (BRLOE);                                 /* Rest des Bildschirmes loeschen */
cdol (lng = 0);                                       /* Eingabe-Puffer loeschen */
printf (PROMPT);

for ( ; ; old = c )                                                /* Hauptschleife */
    {                                                              /*---------------*/
    switch (c = getchr ())
      {
      case (TCLEFT) :                           /* Cursor 1 Zeichen nach links */
                                                /*----------------------------*/
          if  (!lng)          { putchar (BEL); continue; }
          --lng; putchar ('}b'); continue;

      case (TCRIGHT):                          /* Cursor 1 Zeichen nach rechts */
                                               /*-----------------------------*/
          if  (!cdo[lng])     { putchar (BEL); continue; }
          ++lng; printf (CRIGHT, 1); continue;

      case (TWLEFT) :                             /* Cursor 1 Wort nach links */
                                                  /*--------------------------*/
          if  (!lng)          { putchar (BEL); continue; }
          --lng; i = 1;
          while (lng && cdo[lng] == ' ')      { --lng; ++i; }
          while (lng && cdo[lng] != ' ')      { --lng; ++i; }
          if  (lng)                           { ++lng; --i; }
          printf (CLEFT, i); continue;

      case (TWRIGHT):                            /* Cursor 1 Wort nach rechts */
                                                 /*---------------------------*/
          if  (!cdo[lng])     { putchar (BEL); continue; }
          for   (i = 0; cdo[lng] && cdo[lng] != ' '; )    { ++lng; ++i; }
          while (cdo[lng] == ' ')                         { ++lng; ++i; }
          printf (CRIGHT, i); continue;

      case (TCINS)  :                                  /* 1 Zeichen einfuegen */
                                                       /*---------------------*/
          if  (cl () >= LNG || !(j = cdo[lng]))   { putchar (BEL); continue; }
          cdo[l=lng] = ' ';
          while (i = j)       { j = cdo[++l]; cdo[l] = i; }
          printf (CINS); continue;

      case (TBS)    :              /* 1 Zeichen links vom Cursor ausfuegen */
                                   /*--------------------------------------*/
          if  (!lng)             { putchar (BEL); continue; }
          if  (!cdo[lng--])      { cdo[lng] = 0; printf (RADIER); continue; }
          putchar ('}b');                                  /* break through */

      case (TCDEL)  :              /* Zeichen an Cursor-Position ausfuegen */
                                   /*--------------------------------------*/
          if  (!(cdo[lng]))          { putchar (BEL); continue; }
          for (j = lng+1; (i = cdo[j]) && (j < LNG); )    cdo[++j-2] = i;
          cdo[--j] = 0;
          printf (CDEL, 1); continue;

      case (TWDEL)  :                        /* Wort bzw. Wortrest ausfuegen */
```

```
                                        /*-------------------------------*/
    if  (!cdo[lng] || lng >= LNG)         { putchar (BEL); continue; }
    i = 0; k = l = lng;
    while (cdo[l] == ' ')                 { ++l; ++i; }
    while (cdo[l] && cdo[l] != ' ')       { ++l; ++i; }
    if  (!(lng && cdo[lng-1] > ' '))
        while (cdo[l] == ' ')             { ++l; ++i; }
    while (j = cdo[l])                    { cdo[k++] = j; ++l; }
    cdol (k); printf (CDEL, i); continue;
                                                    /* 1 Wort einfuegen */
case (TWINS)  :               /* == alle Zeichen incl. naechstes Space */
                              /*---------------------------------------*/
    wi = EIN; continue;

case (TDEL)   :                                     /* Eingabe abbrechen */
                                                    /*-------------------*/
    printf (PROMPT); cdol (lng = 0); continue;

case (TLDEL)  :                                  /* Rest der Zeile loeschen */
                                                 /*-------------------------*/
    cdol (lng); printf (ZRLOE); continue;

case (TEND)   :                                    /* Zeilen-Editor beenden */
                                                   /*-----------------------*/
    ttyalt (); putchar (SI); exit (0);

case (THOME)  :                                  /* Cursor an Zeilen-Anfang */
                                                 /*-------------------------*/
    printf (CLEFT, lng); lng = 0; continue;

case (TENTER) :                     /* Zeile als Kommando an Sub-Shell */
                                    /*---------------------------------*/
    if  (old == '}}' && !cdo[lng])
        { cdo[--lng] = 0; printf (RADIER); putchar (BEL); continue; }
    putchar ('}r'); putchar ('}n'); putchar ('}n');
    while ((id = fork ()) == ERROR) ;           /* Warten auf Prozess */
    /*-----------------------------------------------------------------*/
    if  (!id)                                                /* bei Sohn : */
        {                                                    /*------------*/
        ttyalt ();                    /* TTY-Parameter wiederherstellen */
        signal (SIGINT, SIG_DFL);        /* Standard-Signal-Behandlung */
        signal (SIGQUIT, SIG_DFL);
        execl ("/bin/sh", "sh", "-c", cdo, NULL);/* Sub-Shell starten */
        printf ("}rzed : </bin/sh> ???      }n}r"); exit (1);
        }
    /*-----------------------------------------------------------------*/
                                                         /* bei Vater : */
    signal (SIGINT, SIG_IGN);                            /*-------------*/
    signal (SIGQUIT, SIG_IGN);                    /* Signale ignorieren */
    while (wait (&stat) != id) ;             /* warten auf Prozess-Ende */
    ttyneu ();                      /* TTY-Parameter wieder umstellen */
    printf (G1LAD);
    putchar (SI);               /* Zeichenvorrat praeventiv einstellen */
    if  (stat)                                           /* wenn Fehler */
        {
        if  (c = (stat & 255))
            printf ("}n}rzed : Wait-Status %d  (0x%x)   }n}r", c, c);
        if  ((c = (stat >> 8) & 255) != 1)
            printf ("}n}rzed : Return-Value %d  (0x%x)  }n}r", c, c);
        }
    printf (PROMPT); cdol (lng = 0); continue;        /* weiter gehts */
```

```
        case (TESC)   :                         /* zweimal ESC --> ESC in Puffer */
                                                /*-------------------------------*/
            c = ESC;                                            /* fall through */

        default :
                                                          /* ungueltiges Zeichen */
            if  (c >= ESCOFF)                             /*---------------------*/
                { putchar (BEL); continue; }
            if  (wi)
                {                                           /* Zeichen einfuegen */
                if  (cl () >= LNG || !(j=cdo[lng]))       /*-------------------*/
                    { wi = AUS; putchar (BEL); continue; }
                cdo[l = lng++] = c;
                while (i=j)         { j = cdo[++l]; cdo[l] = i; }
                printf (CINS); putchr ((char)c);
                if  (c == ' ')      wi = AUS;
                }                                             /* normale Eingabe */
            else{                                             /*-----------------*/
                if  (lng >= LNG)    { putchar (BEL); continue; }
                cdo[lng++] = c; putchr ((char)c);
                }
            continue;
        }
    }
}

/*-------------------------------------------------------------------------------*/
putchr (c)              /* Zeichen ausgeben, Control-Codes invers darstellen */
                       /*---------------------------------------------------*/
register char   c;

{
if  (c >= ' ')          putchar (c);
else                    { printf (INV); putchar (c+'@'); printf (NORMAL); }
}

/*-------------------------------------------------------------------------------*/
getchr ()           /* Tasten-Code holen, Funktionstasten mit entspr. Offset, */
                  /* wenn diese aus Zeichenfolgen bestehen (ESC, CSI)       */
                  /*--------------------------------------------------------*/
{
register unsigned short     c;

if  ((c = getchar ()) != ESC)   return (c);               /* normales Zeichen */
if  ((c = getchar ()) != '[')   return (c + ESCOFF);         /* ESC + Zeichen */
return (getchar () + CSIOFF);                                /* CSI + Zeichen */
}

/*-------------------------------------------------------------------------------*/
cdol (l)                                    /* Kommandopufferrest loeschen */
                                           /*-----------------------------*/
register unsigned short     l;

{ while (l < LNG)         cdo[l++] = 0; }

/*-------------------------------------------------------------------------------*/
cl ()                                        /* Laenge des Kommandos ermitteln */
                                           /*--------------------------------*/
{
register char               *s;
register unsigned short     l;
```

```
for (s = cdo, l = 0; *s++; ++l) ;
return (l);
}

/*-------------------------------------------------------------------------*/
ttysav ()                  /* TTY-Parameter sichern und neue Parameter bilden */
                          /*-----------------------------------------------*/
{
ioctl (0, TCGETA, &argp);                              /* TTY-Parameter sichern */
iflag = argp.c_iflag;                                          /* input-flags */
lflag = argp.c_lflag;                                   /* local-mode-flags */
vmin = argp.c_cchar[T_VMIN];                            /* Nr. chars for read */
vtime = argp.c_cchar[T_VTIME];                               /* time for read */
                                                       /* neue TTY-Parameter : */
iflag1 = (iflag & ~(T_INLCR | T_ICRNL | T_IGNCR | T_IUCLC));
iflag1 |= (T_IXON | T_IXOFF);
lflag1 = (lflag & ~(T_ECHO | T_ISIG | T_ICANON));
vmin1 = 1;
vtime1 = 1;
}

/*-------------------------------------------------------------------------*/
ttyneu ()                                  /* neue TTY-Parameter einstellen */
                                          /*------------------------------*/
{
argp.c_lflag = lflag1;
argp.c_iflag = iflag1;
argp.c_cchar[T_VMIN] = vmin1;
argp.c_cchar[T_VTIME] = vtime1;
ioctl (0, TCSETA, &argp);
}

/*-------------------------------------------------------------------------*/
ttyalt ()                             /* alte TTY-Parameter wiederherstellen */
                                     /*-----------------------------------*/
{
argp.c_lflag = lflag;
argp.c_iflag = iflag;
argp.c_cchar[T_VMIN] = vmin;
argp.c_cchar[T_VTIME] = vtime;
ioctl (0, TCSETA, &argp);
}

/*-------------------------------------------------------------------------*/
/*      E N D E                                                            */
/*-------------------------------------------------------------------------*/
```

A.1.2.15 getrs232

```
# @(#)getrs232 1.3 85/06/13
#############################################################################
#                                                                           #
# getrs232 - einfache Shellprozedur zum Lesen von ASCII-Daten von           #
#            einem anderen Rechner, der mit PC-MX/PC-X ueber die            #
#            Drucker-RS232-Schnittstelle gekoppelt ist.                     #
#                                                                           #
#############################################################################
#                                                                           #
# Bemerkungen:                                                              #
#                                                                           #
#       Die Prozedur ist ablauffaehig unter SINIX V1.0B auf dem             #
#       Siemens PC-MX, ebenso unter SINIX V1.0 auf PC-X, duerfte            #
#       jedoch auch fuer SINIX V1.0C gelten.                                #
#                                                                           #
#       Es koennen nur ASCII-Daten (0x01 ... 0x7f) mit ungeradem            #
#       Parity-Bit uebertragen werden. Saemtliche <CR>-Zeichen              #
#       (0x0d) werden entfernt.                                             #
#                                                                           #
#       Die Uebertragungsgeschwindigkeit betraegt 1200 Baud (muss           #
#       auch am sendenden Rechner eingestellt werden!). Es wurde            #
#       deshalb eine so niedrige Rate gewaehlt, weil in SINIX V1.0B         #
#       noch keine XON/XOFF-Flusssteuerung in Eingaberichtung               #
#       moeglich ist. In der Praxis hat sich gezeigt, dass bei 1200         #
#       Baud normalerweise kein Datenverlust auftritt. Voraussetzung        #
#       ist jedoch, dass das System nicht anderweitig belastet ist.         #
#       Trotzdem empfiehlt sich natuerlich eine Ueberpruefung der           #
#       angekommenen Daten.                                                 #
#                                                                           #
#       Es wird pro Aufruf der Prozedur eine Datei erstellt. Die            #
#       Ausgabe geschieht ueber stdout, d.h. man muss beim Aufruf           #
#       eine geeignete Umlenkung vornehmen. Beispiel:                       #
#                                                                           #
#                       getrs232 >zieldatei                                 #
#                                                                           #
#       Da kein erkennbares Dateiende-Kriterium ueber die Leitung           #
#       kommt, muss nach dem Ende der Uebertragung mit der DEL-Taste        #
#       das Lesen beendet werden.                                           #
#                                                                           #
#       Waehrend des Transfers darf kein als D1 generierter Drucker         #
#       angesprochen werden.                                                #
#                                                                           #
#       Fuer einen korrekten Ablauf der Prozedur muss man unter der         #
#       Benutzerkennung root oder admin arbeiten.                           #
#                                                                           #
#############################################################################
#                                                                           #
# Uebrigens:                                                                #
#                                                                           #
#       Verzichtet man auf jeglichen Komfort, so braucht die                #
#       Prozedur nur aus den folgenden drei Zeilen bestehen:                #
#                                                                           #
#           exec </dev/rs232                                                #
#           stty 1200 cbreak -echo nl -even odd tandem >/dev/rs232          #
#           cat -u                                                          #
#                                                                           #
#       Voraussetzung dafuer ist, dass beim Aufruf                          #
#       1.  unter root gearbeitet wird,                                     #
#       2.  /dev/rs232 schon eingerichtet ist,                              #
#       3.  keine <CR>-Zeichen entfernt werden sollen.                      #
#                                                                           #
#############################################################################
```

```
# Nachsehen, ob wir unter root arbeiten (nur dann duerfen wir /etc/passwd
# beschreiben):

if [ ! -w /etc/passwd ]
then
    display "Ablauf nur unter der Kennung root bzw. admin moeglich!"
    exit 1
fi

# Nachsehen, ob die Geraetedatei fuer die Drucker-RS232-Schnittstelle schon
# vorhanden ist. Wenn nicht, wird sie eingerichtet:

if [ ! -r /dev/rs232 ]
then
    /etc/mknod /dev/rs232 c 3 131
    XMX=«fgrep SYST=978 /usr/menus/sabin/header«
    case "$XMX" in
    *9780*) display -n "Schalter S4 (CONAC) bereits in RS232-Stellung? (j/n) > "
          read ANT
          case "$ANT" in
          j)      :;;
          *)      display "Bitte zuerst umschalten!"
                  exit 1;;
          esac;;
    esac
fi

# RS232-SS fuer stdin zuweisen und bis zum Prozedurende eroeffnet halten:

exec </dev/rs232

# RS232-SS einstellen:

stty 1200 cbreak -echo nl -even odd tandem >/dev/rs232

display "Nun wird der Daten-Empfang ueber die RS232-SS gestartet.
Bitte leiten Sie jetzt am Sende-Rechner die Uebertragung ein.
Nach dem Ende des Transfers (am Sende-Rechner erkenntlich) muss das
Empfangen durch Druecken der DEL-Taste beendet werden."

# Vorbereitung fuer das Uebertragungsende (ausgeloest durch DEL-Taste).
# Aus den uebertragenen Daten werden alle Zeichen <CR> entfernt, da manche
# Fremdrechner jede Zeile mit <CR><LF> abschliessen, in SINIX jedoch als
# Zeilenende nur <LF> verwendet wird:

trap 'trap "" 2
      tr -d "}015" <.tmprs232
      rm .tmprs232
      exit 0' 2

# Eigentliche Datenuebertragung (Lesen von /dev/rs232):

cat -u >.tmprs232
```

A.2 Tabellen

A.2.1 ASCII-Tabelle (oktal)

000 NUL	001 SOH	002 STX	003 ETX	004 EOT	005 ENQ	006 ACK	007 BEL
010 BS	011 HT	012 NL	013 VT	014 NP	015 CR	016 SO	017 SI
020 DLE	021 DC1	022 DC2	023 DC3	024 DC4	025 NAK	026 SYN	027 ETB
030 CAN	031 EM	032 SUB	033 ESC	034 FS	035 GS	036 RS	037 US
040 SP	041 !	042 "	043 #	044 $	045 %	046 &	047 '
050 (	051)	052 *	053 +	054 ,	055 -	056 .	057 /
060 0	061 1	062 2	063 3	064 4	065 5	066 6	067 7
070 8	071 9	072 :	073 ;	074 <	075 =	076 >	077 ?
100 @	101 A	102 B	103 C	104 D	105 E	106 F	107 G
110 H	111 I	112 J	113 K	114 L	115 M	116 N	117 O
120 P	121 Q	122 R	123 S	124 T	125 U	126 V	127 W
130 X	131 Y	132 Z	133 [	134 \	135]	136 ^	137 _ O
140 `	141 a	142 b	143 c	144 d	145 e	146 f	147 g
150 h	151 i	152 j	153 k	154 l	155 m	156 n	157 o
160 p	161 q	162 r	163 s	164 t	165 u	166 v	167 w
170 x	171 y	172 z	173 {	174 |	175 }	176 ~	177 DEL

A.2.2 ASCII-Tabelle (hexadezimal)

00 NUL	01 SOH	02 STX	03 ETX	04 EOT	05 ENQ	06 ACK	07 BEL
08 BS	09 HT	0A NL	0B VT	0C NP	0D CR	0E SO	0F SI
10 DLE	11 DC1	12 DC2	13 DC3	14 DC4	15 NAK	16 SYN	17 ETB
18 CAN	19 EM	1A SUB	1B ESC	1C FS	1D GS	1E RS	1F US
20 SP	21 !	22 "	23 #	24 $	25 %	26 &	27 '
28 (	29)	2A *	2B +	2C ,	2D -	2E .	2F /
30 0	31 1	32 2	33 3	34 4	35 5	36 6	37 7
38 8	39 9	3A :	3B ;	3C <	3D =	3E >	3F ?
40 @	41 A	42 B	43 C	44 D	45 E	46 F	47 G
48 H	49 I	4A J	4B K	4C L	4D M	4E N	4F O
50 P	51 Q	52 R	53 S	54 T	55 U	56 V	57 W
58 X	59 Y	5A Z	5B [	5C \	5D]	5E ^	5F _ O
60 `	61 a	62 b	63 c	64 d	65 e	66 f	67 g
68 h	69 i	6A j	6B k	6C l	6D m	6E n	6F o
70 p	71 q	72 r	73 s	74 t	75 u	76 v	77 w
78 x	79 y	7A z	7B {	7C |	7D }	7E ~	7F DEL

A.2.3 Hexadezimale Vergleichstabelle

	No Parity ASCII	Even Parity ASCII	Odd Parity ASCII	EBCDIC		No Parity ASCII	Even Parity ASCII	Odd Parity ASCII	EBCDIC		No Parity ASCII	Even Parity ASCII	Odd Parity ASCII	EBCDIC
A	41	41	C1	C1	0	30	30	B0	F0	ACK	06	06	86	2E
B	42	42	C2	C2	1	31	B1	31	F1	BEL	07	87	07	2F
C	43	C3	43	C3	2	32	B2	32	F2	BS	08	88	08	16
D	44	44	C4	C4	3	33	33	B3	F3	BYP				24
E	45	C5	45	C5	4	34	B4	34	F4	CAN	18	18	98	18
F	46	C6	46	C6	5	35	35	B5	F5	CC				1A
G	47	47	C7	C7	6	36	36	B6	F6	CR	0D	8D	0D	0D
H	48	48	C8	C8	7	37	B7	37	F7	DC1	11	11	91	11
I	49	C9	49	C9	8	38	B8	38	F8	DC2	12	12	92	12
J	4A	CA	4A	D1	9	39	39	B9	F9	DC3	13	93	13	13
K	4B	4B	CB	D2	SP	20	A0	20	40	DC4	14	14	94	3C
L	4C	CC	4C	D3	!	21	21	A1	5A	DEL	7F	FF	7F	07
M	4D	4D	CD	D4	"	22	22	A2	7F	DLE	10	90	10	10
N	4E	4E	CE	D5	#	23	A3	23	7B	DS				20
O	4F	CF	4F	D6	$	24	24	A4	5B	EM	19	99	19	19
P	50	50	D0	D7	%	25	A5	25	6C	ENQ	05	05	85	2D
Q	51	D1	51	D8	&	26	A6	26	50	EOB				26
R	52	D2	52	D9	'	27	27	A7	7D	EOT	04	84	04	37
S	53	53	D3	E2	(	28	28	A8	4D	ESC	1B	1B	9B	27
T	54	D4	54	E3	)	29	A9	29	5D	ETB	17	17	97	26
U	55	55	D5	E4	*	2A	AA	2A	5C	ETX	03	03	83	03
V	56	56	D6	E5	+	2B	2B	AB	4E	FF	0C	0C	8C	0C
W	57	D7	57	E6	,	2C	AC	2C	6B	FS	1C	9C	1C	
X	58	D8	58	E7	-	2D	2D	AD	60	GS	1D	1D	9D	
Y	59	59	D9	E8	.	2E	2E	AE	4B	HT	09	09	89	05
Z	5A	5A	DA	E9	/	2F	AF	2F	61	IFS				1C
a	61	E1	61	81	:	3A	3A	BA	7A	IGS				1D
b	62	E2	62	82	;	3B	BB	3B	5E	IL				17
c	63	63	E3	83	<	3C	3C	BC	4C	IRS				1E
d	64	E4	64	84	=	3D	BD	3D	7E	IUS				1F
e	65	65	E5	85	>	3E	BE	3E	6E	LC				06
f	66	66	E6	86	?	3F	3F	BF	6F	LF	0A	0A	8A	25
g	67	E7	67	87	@	40	C0	40	7C	NAK	15	95	15	3D
h	68	E8	68	88	[	5B	DB	5B	BB	NL				15
i	69	69	E9	89	\	5C	5C	DC	BC	NUL	00	00	80	00
j	6A	6A	EA	91	]	5D	DD	5D	BD	PF				04
k	6B	EB	6B	92	^	5E	DE	5E	6A	PN				34
l	6C	6C	EC	93	_	5F	5F	DF	6D	PRE				27
m	6D	ED	6D	94	`	60	60	E0	4A	RES				14
n	6E	EE	6E	95	{	7B	7B	FB	FB	RLF				09
o	6F	6F	EF	96	|	7C	FC	7C	4F	RS	1E	1E	9E	35
p	70	F0	70	97	}	7D	7D	FD	FD	SI	0F	0F	8F	0F
q	71	71	F1	98	~	7E	7E	FE		SM				2A
r	72	72	F2	99	c				4A	SMM				0A
s	73	F3	73	A2	¬				5F	SO	0E	8E	0E	0E
t	74	74	F4	A3	◇				FF	SOH	01	81	01	01
u	75	F5	75	A4						SOS				21
v	76	F6	76	A5						STX	02	82	02	02
w	77	77	F7	A6						SUB	1A	9A	1A	3F
x	78	78	F8	A7						SYN	16	96	16	32
y	79	F9	79	A8						UC				36
z	7A	FA	7A	A9						US	1F	9F	1F	
										VT	0B	8B	0B	0B

Literatur

[1] Betriebssystem SINIX
Buch 1

[2] Betriebssystem SINIX
Buch 2
Menüs

[3] Betriebssystem SINIX
EMDS/FT-SINIX
Nachbildung Datensichtstation 9750,
Drucker 8122 und Dateiübertragung

[4] Drucker 9001
TRANSDATA
Betriebsanleitung

[5] Drucker 9004
TRANSDATA
Betriebsanleitung

[6] Drucker 9013
TRANSDATA
Betriebsanleitung

[7] Drucker 9022
TRANSDATA
Betriebsanleitung

[8] Betriebssystem SINIX
CCP-PC-X/X10