SIEMENS

SINIX-Schnittstellen

Benutzerhandbuch

SINIX

für Siemens PC

SINIX-Schnittstellen

Benutzerhandbuch

Ausgabe Dezember 1985 (SINIX Version 1.0)

Bestell-Nr. U 2300-J-Z95-1
Printed in the Federal Republic of Germany
9600 AG 12851. (12000)

Siemens Aktiengesellschaft

Inhalt

1	**Schnittstellen zu Bildschirm und Tastatur (PC-X, PC-MX)**	**1-1**
1.1	Hardware-Schnittstellen des PC-MX	1-1
1.1.1	Allgemeines (Bildschirm des PC-MX)	1-1
1.1.1.1	Funktionsübersicht	1-2
1.1.1.2	Bildschirmsteuerung	1-3
1.1.1.3	Stromversorgung	1-3
1.1.1.4	Tastatur	1-3
1.1.1.5	Übersicht über Bildschirm und Tastaturvarianten (PC-MX)	1-4
1.1.2	Belegung der Schnittstelle Bildschirm → Systemeinheit des PC-MX	1-5
1.1.2.1	Hardware-Schnittstelle	1-5
1.1.2.2	PIN-Belegung der SS97(Bedieneinheit → Systemeinheit)	1-6
1.1.2.3	PIN-Belegung der SS97 (Bildschirm → Tastatur)	1-7
1.2	Hardware-Schnittstellen des PC-X	1-8
1.2.1	Allgemeines (Bildschirm des PC-X)	1-8
1.2.1.1	Aufbau und Arbeitsweise	1-9
1.2.1.2	Funktionsübersicht	1-10
1.2.1.3	Bildschirmsteuerung	1-11
1.2.1.4	Tastatur	1-12
1.2.2	Belegung der Schnittstelle Bildschirm/Tastatur → Systemeinheit des PC-X	1-14
1.2.2.1	Hardware-Schnittstelle	1-14
1.2.2.2	PIN-Belegung des Bildschirm-Anschlusses	1-15
1.2.2.3	PIN-Belegung des Tastatur-Anschlusses	1-16
1.3	Software-Schnittstelle termcap	1-18
1.3.1	Einführung	1-18
1.3.1.1	Was ist termcap?	1-18
1.3.1.2	Woher kommt termcap?	1-18
1.3.1.3	Warum termcap?	1-18
1.3.1.4	Wie funktioniert termcap?	1-19
1.3.2	Die termcap-Datei	1-19
1.3.2.1	Aufbau und Inhalt einer termcap-Datei in SINIX	1-20
1.3.2.2	Definition von Steuerzeichenfolgen	1-22
1.3.2.3	Wartezeiten nach Terminalfunktionen	1-23
1.3.2.4	Cursor-Bewegungen	1-23
1.3.2.5	Funktionsfeldkennungen	1-24
1.3.3	Einträge in der /etc/termcap-Datei	1-26
1.3.4	Umgebungsvariablen	1-27
1.3.4.1	Variable TERM	1-27
1.3.4.2	Variable TERMCAP	1-28
1.3.5	Routinen in der termcap-Bibliothek	1-29
1.3.5.1	Funktion tgetent	1-29
1.3.5.2	Funktion tgetnum	1-31
1.3.5.3	Funktion tgetstr	1-31
1.3.5.4	Funktion tgetflag	1-32
1.3.5.5	Funktion tputs	1-33

1.3.5.6	Funktion tgoto	1-35
1.3.6	Terminalinitialisierung	1-36
1.3.6.1	/etc/getty	1-36
1.3.6.2	/bin/login	1-36
1.3.6.3	/etc/ttytype	1-36
1.3.7	Verarbeitung von Eingabesequenzen	1-37
1.3.7.1	Funktion tbuild	1-37
1.3.7.2	Funktion txtract	1-38
1.3.7.3	Anwendungsbeispiele	1-38
1.3.7.4	tbuild und txtract	1-40
1.4	Bildschirmfunktionen und Datenformate	1-43
1.4.1	Kommandoübersicht und Kurzbeschreibung der Normelemente	1-43
1.4.1.1	Kommandos zum Modifizieren der Zeichensätze	1-43
1.4.1.2	Kommandos zum Editieren, zur Cursor-Steuerung und zum Löschen	1-44
1.4.1.3	Kommandos zur Initialisierung des Bildschirms und Zeichendarstellung	1-45
1.4.1.4	Tastaturkommandos	1-45
1.4.1.5	Servicekommandos	1-46
1.4.1.6	Kurzbeschreibung der Normelemente (DIN 66254/ISO 6492.2)	1-47
1.4.2	Funktion und Befehle für BS	1-48
1.4.2.1	Codierung und Zeichendarstellung	1-48
1.4.2.2	Befehle zur Codiertabellen-Umschaltung	1-51
1.4.2.3	Anzeigedaten	1-53
1.4.2.4	Cursor-Befehle	1-54
1.4.2.5	Bildverschiebe-Befehle	1-60
1.4.2.6	Löschbefehle	1-61
1.4.2.7	Darstellungsarten (Attribute)	1-64
1.4.2.8	Bildschirm-Format und weitere Einstellung	1-66
1.4.2.9	Rücksetz-Befehl	1-69
1.4.3	Funktionen und Befehle für Tastatur	1-70
1.4.3.1	Tastaturbefehle	1-70
1.4.3.2	Tastenbelegung	1-74
1.4.4	Zusammenfassung von Tastenbelegung und Zeichensätzen	1-75
1.4.5	Diagnose-Funktionen	1-77
1.4.5.1	DIP-FIX-Schalter-Information abfragen (nur bei PC-MX)	1-77
1.4.5.2	Testmodus	1-78
1.4.5.3	Monitor-Funktion für Steuerzeichen	1-79
1.4.5.4	Systemtest (Bildschirm und Tastatur)	1-79

1.5	Zeichensätze und Adressen des Matrixgenerators	1-83
1.5.1	Übersicht der zu Zeichensätzen zusammengefaßten Zeichen	1-84
1.5.1.1	Zeichensatz: International A	1-85
1.5.1.2	Zeichensatz: International	1-86
1.5.1.3	Zeichensatz: Deutsch	1-87
1.5.1.4	Zeichensatz: Euro	1-88
1.5.1.5	Zeichensatz: Klammern	1-89
1.5.1.6	Zeichensatz: Mosaic	1-90
1.5.1.7	Zeichensatz: IBM	1-91
1.5.1.8	Zeichensatz: Mathematisch	1-92
1.5.2	Unterschiedliche Zeichen bei nationalen Tastatur-Varianten	1-93
1.5.3	Zeichen des Matrixgenerators mit internen Adressen	1-94
1.6	Tastaturbelegungen (PC-MX/PC-X)	1-96
1.6.1	International (V1) 97801-111/97811-111	1-96
1.6.2	Deutsch/International (V2) 97801-112/97811-112	1-96
1.6.3	Belgisch-Flämisch (V3) 97801-113/97811-113	1-97
1.6.4	Belgisch-Französisch (V7) 97801-117/97811-117	1-97
1.6.5	Schwedisch (V4) 97801-114/97811-114	1-97
1.6.6	Dänisch (V5) 97801-115/97811-115	1-98
1.6.7	Französisch (V6) 97801-116/97811-116	1-98
1.6.8	Schweizerisch (V8) 97801-118/97811-118	1-98
1.6.9	Spanisch (V9) 97801-119/97811-119	1-99
1.6.10	Italienisch (V10) 97801-120/97811-120	1-99
1.7	Beispiele	1-100
1.7.1	Beispiel 1: Festlegung der Zeichensätze in der Datei /etc/termcap	1-100
1.7.2	Beispiel 2: Setzen der CH-Code-Taste und Laden der Zeichensätze	1-102

2	**Weitere Schnittstellen des PC-MX**	**2-1**
2.1	Schnittstellen im Grundausbau (SS97/RS232)	2-1
2.1.1	Allgemeines	2-1
2.1.1.1	Diagnosehilfe	2-2
2.1.2	Umschaltung der Schnittstelle (SS97/RS232)	2-3
2.1.3	Umschaltung auf der Flachbaugruppe CONAC (SS97/RS232)	2-4
2.1.4	Pin-Belegung der Schnittstelle (SS97/RS232)	2-5
2.1.4.1	Schnittstelle SS97	2-5
2.1.4.2	Schnittstelle RS232	2-6
2.1.5	Beispiel	2-8
2.2	Ein-/Ausgabeprozessor	2-10
2.2.1	SERAC: 6 x SS97 (97802-201)	2-10
2.2.2	SERAD: 4 x SS97 und 2 x RS232 (97802-202)	2-11
2.2.3	Mehrere Ein-/Ausgabeprozessoren	2-12
2.2.4	Allgemeines über Gerätedateien	2-13
2.3	Schnittstellenbeschreibung	2-15
2.3.1	Schnittstelle SS97	2-15
2.3.1.1	Pin-Belegung	2-15
2.3.1.2	Elektrische Kennwerte	2-16
2.3.2	Schnittstelle RS232	2-23
2.3.2.1	Pin-Belegung	2-23
2.3.2.2	Elektrische Kennwerte	2-24
2.3.2.3	Bemerkungen	2-26
2.4	Konstruktiver Aufbau	2-28
2.4.1	Öffnen des Gehäuses beim PC-MX	2-28
2.4.2	Flachbaugruppenbelegung des PC-MX	2-30
3	**Weitere Schnittstellen des PC-X**	**3-1**
3.1	Schnittstellen im Grundausbau des PC-X	3-1
3.1.1	Allgemeines	3-1
3.1.2	Übersicht	3-1
3.1.3	Beeinflußbare externe Schnittstellen (Stecker 1 - 4)	3-4
3.1.4	Umschalten der Schnittstelle SS97/RS232	3-5
3.1.5	Beispiel	3-7
3.2	Pin-Belegung der Stecker am Anschlußfeld des Systemboards	3-8
3.2.1	Schnittstelle SS97 (Stecker 2 und 4)	3-8
3.2.2	Schnittstelle RS232 (Stecker 1 und 3)	3-9
3.2.3	Schnittstelle V.11 (Tastaturanschluß, Stecker 5)	3-11
3.3	Elektrische Kennwerte der Schnittstellen	3-12
3.3.1	Schnittstelle SS97	3-12
3.3.1.1	Signalzustände	3-12
3.3.1.2	Beschaltung der Leitung FE/PO und UH	3-15
3.3.1.3	Steckerbelegung	3-16
3.3.1.4	Bemerkungen	3-17
3.3.2	Schnittstelle RS232C	3-19
3.3.2.1	Elektrische Kennwerte	3-19
3.3.2.2	Bemerkungen	3-22

4	**Beschreibung der TTY-Schnittstelle**	**4-1**
4.1	Auswahl einer TTY-Schnittstelle	4-2
4.2	Die Anpassung der TTY-Schnittstelle	4-4
4.2.1	Anpassung beim System-Start-Up	4-4
4.2.2	Anpassung durch das Kommando	4-5
4.2.3	Anpassung durch den System-Aufruf	4-5
4.3	TTY-Schnittstelle entsprechend XENIX-V7	4-7
4.3.1	Beschreibung des System-Aufrufes	4-10
4.3.2	Allgemeines	4-11
4.3.3	Schnittstellen-Parameter	4-16
4.3.3.1	Der COOKED-Modus	4-19
4.3.3.2	Der CBREAK-Modus	4-19
4.3.3.3	Der RAW-Modus	4-19
4.4	TTY-Schnittstelle entsprechend UNIX-System-III	4-23
4.4.1	Beschreibung des System-Aufrufes	4-28
4.4.2	Allgemeines	4-28
4.4.3	Schnittstellen-Parameter	4-34
4.4.3.1	Daten-Eingabe-Steuerung <c_iflag>	4-34
4.4.3.2	Daten-Ausgabe-Steuerung <o_flag>	4-36
4.4.3.3	Hardware-Steuerung des Terminals <c_cflag>	4-39
4.4.3.4	Steuerung der Terminal-Funktionen <c_lflag>	4-41
4.4.4	ioctl-Aufrufe	4-44
5	**Spool-System**	**5-1**
5.1	Allgemeiner Überblick über die Druckerverwaltung unter SINIX	5-1
5.1.1	Dateien und Dateiverzeichnisse	5-1
5.1.2	Programme für die Druckerverwaltung	5-2
5.1.3	Die Funktionen des lpr	5-4
5.1.4	Die Funktionen des qdaemon	5-8
5.1.5	Die Funktionen des Backend	5-10
5.1.6	Die Funktion des Treibers	5-11
5.2	Betrieb von verschiedenen Druckern am PC-MX/PC-X	5-12
5.2.1	Auswahl der Treiber	5-12
5.2.2	Umgehung der Standarddruckverwaltung.	5-12
5.2.2.1	Spezielle Bemerkung für den PC-MX (SINIX Version 1.0B)	5-13
5.2.2.2	Spezielle Bemerkung für den PC-X	5-13
5.2.3	Betrieb eines Druckers über die Standarddruckverwaltung	5-14
5.2.3.1	Das primitive Backend /bin/cat	5-15
5.2.4	Eigenes Backend	5-15
5.2.5	Beispiele	5-18
5.2.5.1	Backend für den Drucker 9001	5-18
5.2.5.2	Backend für den Drucker 9004	5-25

6	**DÜ-Schnittstelle PC-X, PC-MX**	**6-1**
6.1	Allgemeines	6-1
6.1.1	Beschreibung der Schichten des ISO-7-Schichten-Modells	6-2
6.2	Die V.24-Schnittstelle	6-4
6.2.1	Beschreibung der einzelnen Leitungen	6-4
6.2.1.1	Die Erdleitung E2	6-4
6.2.1.2	Betriebsbereitschaft S1, M1	6-5
6.2.1.3	Sender steuern S2, M2, M5	6-6
6.2.1.4	Daten D1, D2	6-8
6.2.1.5	Takte bei synchroner Übertragung	6-8
6.3	Die Übertragungprozedur MSV1	6-9
6.4	Realisierung (SINIX 1.0B, 1.0C)	6-11
6.5	Technische Daten	6-11
7	**Behandlung von Disketten beim PC-X und PC-MX**	**7-1**
7.1	Allgemeines	7-1
7.2	Gerätedateien	7-1
7.3	Disketten bearbeiten	7-3
7.3.1	Beispiele	7-4
7.3.2	Beschreibung einiger bekannter Disketten-Formate	7-6
7.3.3	Disketten-Formatierprogramm für den PC-X	7-6
7.4	Disketten lesen	7-8
7.4.1	Allgemeines	7-8
7.4.2	Vorgehen bei MS-DOS-Disketten	7-9
7.4.2.1	Programm zur logischen Aufbereitung von MS-DOS-Disketten	7-10
8	**Kopplung**	**8-1**
8.1	Allgemein	8-1
8.2	Kabel	8-2
8.3	Beispiele	8-4
8.3.1	Shellprozedur zum Lesen von ASCII-Daten	8-4
8.3.2	Programm zum Einlesen von ASCII-Dateien über einen TTY-Kanal	8-7
8.3.3	Programm zum Lesen von ASCII-Dateien über einen TTY-Kanal	8-12

A	**Anhang**	**A-1**
A.1	Inhaltsverzeichnis der Diskette	A-1
A.1.1	Inhalt der Dateien	A-2
A.1.2	Auflistung der Quellprogramme	A-4
A.1.2.1	tbuild.c	A-4
A.1.2.2	txample.c	A-7
A.1.2.3	c_get.c	A-10
A.1.2.4	c_test.c	A-15
A.1.2.5	chcode.c	A-17
A.1.2.6	backend01.c	A-19
A.1.2.7	backend04.c	A-26
A.1.2.8	fl6parset.c	A-34
A.1.2.9	fl6parget.c	A-35
A.1.2.10	eing.c	A-36
A.1.2.11	gettty.c	A-38
A.1.2.12	zed.c	A-43
A.1.2.13	getrs232	A-51
A.2	Tabellen	A-53
A.2.1	ASCII-Tabelle (oktal)	A-53
A.2.2	ASCII-Tabelle (hexadezimal)	A-53
A.2.3	Hexadezimale Vergleichstabelle	A-54

Vorwort

Dieses Schnittstellenhandbuch haben wir für den geübten SINIX-Anwender und C-Programmierer erstellt. Wir möchten Ihnen mit diesem Buch nützliche Hinweise geben, wie Sie Ihren Siemens PC-X/PC-MX mit seinen Möglichkeiten besser nutzen können.

Alle Angaben und Beschreibungen beziehen sich (so weit keine anderen Angaben erfolgen) auf die

- Version 1.0B des PC-MX (9780) und die
- Version 1.0B des PC-X (9781).

Abweichende Funktionen in Version 1.0C sind, soweit bei Redaktionschluß bekannt, bereits berücksichtigt.

Mit einer neuen SINIX-Version werden wir auch dieses Handbuch ergänzen.

Auf der mitgelieferten Diskette 'SSHB' finden Sie Quellprogramme und lauffähige Programme, die Ihnen bei der Lösung kniffeliger Aufgaben helfen sollen.

Das Inhaltsverzeichnis dieser Diskette und die Quellprogramm-Dateien sind im Anhang aufgelistet.

Warnung

Benutzen Sie unter der Kennung root/admin keine Kommandos, deren Wirkungsweise Sie nicht kennen. Die Konsistenz Ihres SINIX-Systems kann dabei zerstört werden.

Siemens PC's entsprechen bei bestimmungsgemäßem Gebrauch der allgemeinen Genehmigung nach Postverfügung Nr.1115.

Alle an die PC's 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 keinerlei Haftung für die getroffenen Hinweise gewährt werden.

Änderungen, die dem technischen Fortschritt und der Fehlerbehebung dienen, vorbehalten.

Hinweis

System Development Toolset (SDT)

Wenn projektspezifisch Gerätetreiber entwickelt werden müssen, kann ein spezieller Satz Disketten zur Verfügung gestellt werden.
Dabei müssen folgende Voraussetzungen erfüllt sein:

- SDT wird nur an Siemens-Dienststellen weitergegeben.
- Die Entwicklung erfolgt am PC-X/PC-MX mit ordnungsgemäßer CES-Lizenz.
- Zielmaschinen sind PC-X/PC-MX mit ordnungsgemäßer SINIX-Lizenz.
- Die Entwicklungsabteilung verfügt über Erfahrungen beim Implementieren von UNIX/XENIX/SINIX-Gerätetreibern.
- Es wurde untersucht und festgestellt, daß das anzuschließende Gerät nicht über den standardmäßigen TTY-Treiber betrieben werden kann.
- Es wird akzeptiert, daß Siemens für SINIX-Systeme, die mit SDT erzeugt wurden, keinerlei Verpflichtung oder Gewährleistung übernimmt.

Was ist SDT?

SDT umfaßt alle Bibliotheken (in binärer Form) und Werkzeuge, die nötig sind, um einen SINIX-Betriebssystemkern zu binden.
Zu den Werkzeugen gehören insbesondere ein eigener C-Compiler, Assembler und Binder.

Die Disketten (derzeit drei) werden über die übliche Installation von Softwareprodukten eingelesen.
Die Bedienung von SDT wird in einem Dokument (in Englisch) beschrieben. Es wird nicht erklärt, wie man einen Treiber schreibt, sondern nur, wie man einen Treiber mit SDT in den Betriebssystemkern einbringt.

Die Werkzeuge (C-Compiler usw.) sind nicht offiziell, können allerdings von entsprechend erfahrenen Entwicklern eingesetzt werden, um 'middle model'-Anwenderprogramme zu entwickeln.
'middle model' heißt, daß ein Programm 64 kbyte Daten und Stack und mehr als 64 kbyte Code (derzeit 160 kbyte) umfassen kann.
Eine kurze englische Beschreibung über den Umgang mit dem 'middle model'-Compiler ist ebenfalls als Dokument vorhanden.

Floating Point für 'middle model' wird beim PC-MX erst ab Version 1.0C unterstützt.

Manualredaktion K D ST QM 2
Otto-Hahn-Ring 6, 8000 München 83

1 Schnittstellen zu Bildschirm und Tastatur (PC-X, PC-MX)

1.1 Hardware-Schnittstellen des PC-MX

1.1.1 Allgemeines (Bildschirm des PC-MX)

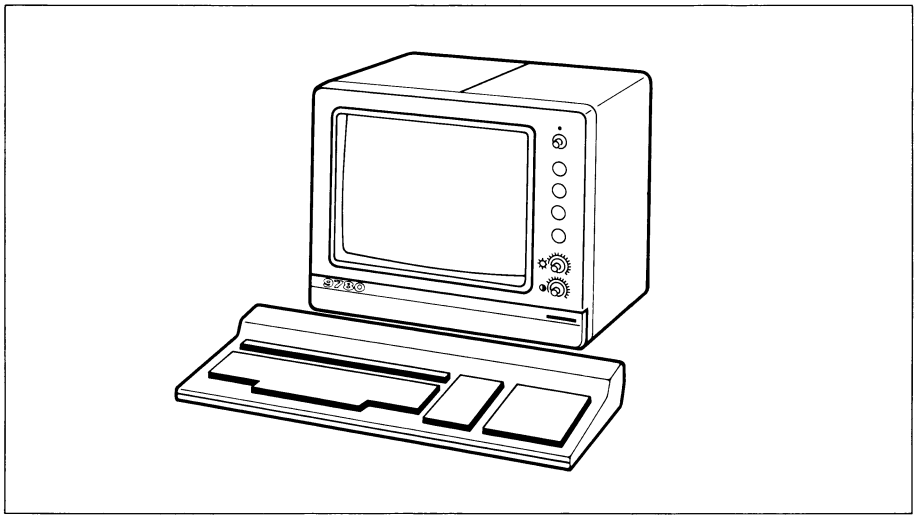


Bild 1-1 Bedieneinheit PC-MX (9780)

Die Tastatur (TAS) mit dem Bildschirm (BS) wird im weiteren als Bedieneinheit (BE) bezeichnet.
Der Bildschirm kann 25 x 80 Zeichen darstellen. Für jedes einzelne Zeichen kann die Darstellungsart (normal, invers, halbhell, blinken, unterstrichen) abgespeichert werden. Mit 2 Reglern an der Frontseite kann Helligkeit und Halbhelligkeit eingestellt werden.
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.

1.1.1.1 Funktionsübersicht

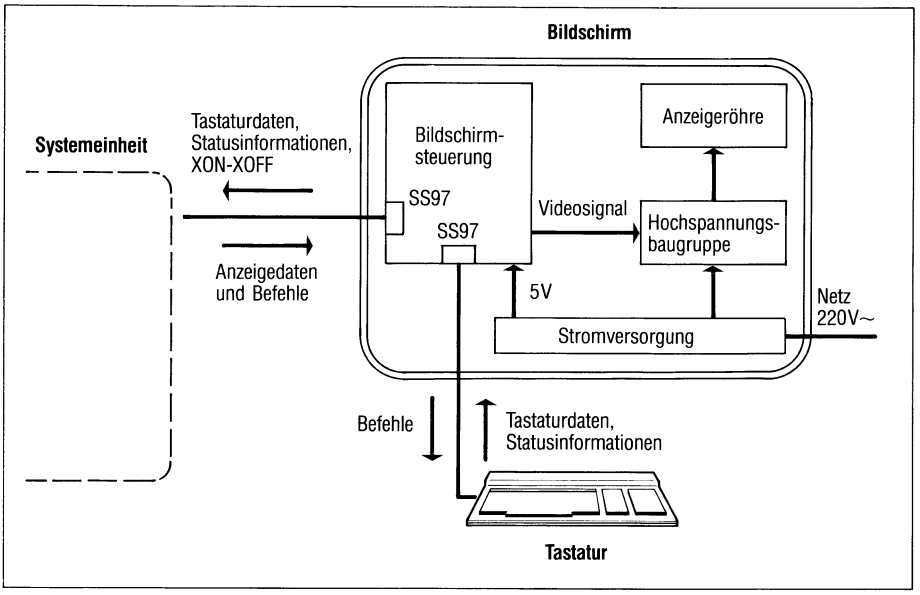


Bild 1-2 Funktionsübersicht Bedieneinheit

1.1.1.2 Bildschirmsteuerung

Die Bildschirmsteuerung ermöglicht

- das stellenrichtige Einschreiben der vom PC zurückübertragenen 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 60 Hz am Bildschirm angezeigt.

1.1.1.3 Stromversorgung

Die Stromversorgung liefert alle zum Betreiben der Bedieneinheit erforderlichen Spannungen. Auch 17000 Volt für die Bildröhre.
Also Reparaturen dem Service überlassen!

1.1.1.4 Tastatur

Jedes Betätigen einer Taste erzeugt einen Platzcode, der zur Bildschirmsteuerung übertragen wird. Hier erfolgt eine Umcodierung des Platzcodes entsprechend dem geladenen Zeichensatz in ASCII-Zeichen.

Hinweis

Die Firmware (Programm und Zeichenspeicher) im Bildschirm ist **immer** gleich, unabhängig von den länderspezifischen Tastaturen.
Die Firmware (Programm und Platzcodespeicher) in der Tastatur ist länderspezifisch.

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 terminalspezifisch hinterlegt (siehe 1.7: Beispiel 1).

1.1.1.5 Übersicht über Bildschirm und Tastaturvarianten (PC-MX)

Die unten aufgeführten Produkt-Nummern finden sie am Bildschirm hinter der Abdeckklappe und an der Tastatur auf der Unterseite.

Die Bedieneinheit PC-MX besteht aus

- Bildschirm 97803-301
- Tastatur, wahlweise mit folgenden Belegungen:
  - international 97801-111
  - international und deutsch 97801-112
  - international und belgisch QWERTY 97801-113
  - international und schwedisch 97801-114
  - international und dänisch 97801-115
  - international und französisch 97801-116
  - international und belgisch AZERTY 97801-117
  - international und schweizerisch 97801-118
  - international und spanisch 97801-119
  - international und italienisch 97801-120
- Tastatur optional mit integrierbarem Ausweisleser 97801-201
- Zusatz für Bildschirm:
  Ergolift
  (Höhenverstellung für Bildschirm 80 mm) 97506-4

1.1.2 Belegung der Schnittstelle Bildschirm → Systemeinheit des PC-MX

1.1.2.1 Hardware-Schnittstelle

Der Anschluß der Bedieneinheiten erfolgt über die Siemens-Schnitt stelle 97 (SS97), einer Schnittstelle mit den elektrischen Eigenschaften der genormten Schnittstelle V.11 (Informationen über Bauteile etc. siehe 2.3).

Alle Komponenten - ausgenommen die Konsole - können bis 30 m über konfektionierte Kabel angeschlossen werden. Längere Kabel müssen selbst hergestellt werden (siehe Aufbauplanung).

Wichtige Voraussetzung ist in jedem Fall, daß das 220V-Erdungspotential an PC und Komponente dasselbe ist, d.h. daß beide an einem Netz mit gemeinsamem Bezugspunkt (z.B. Stockwerkverteiler) angeschlossen werden.

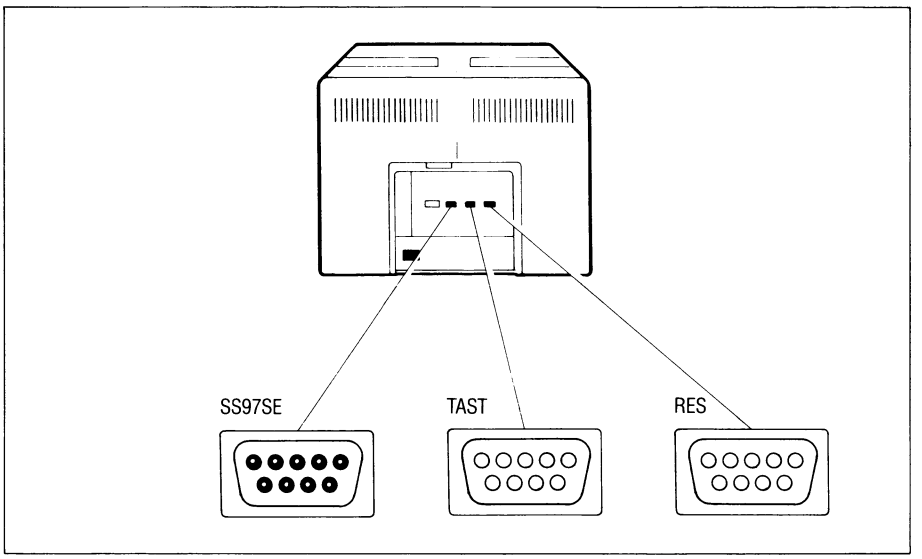


Bild 1-3 Anschlußfeld des Bildschirms

1.1.2.2 PIN-Belegung der SS97 (Bedieneinheit → Systemeinheit)

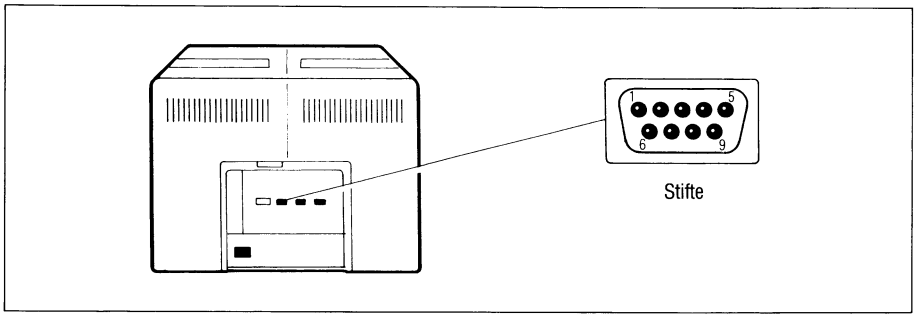


Bild 1-4 SS97 (Stifte) am Bildschirm

Stift	Bezeichnung	Erklärung
1	DOUT-P	Daten von BE zum PC
6	DOUT-N	
3	DIN-P	Daten vom PC zur BE
8	DIN-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		nicht belegt

Kennwerte der Übertragung

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

1.1.2.3 PIN-Belegung der SS97 (Bildschirm → Tastatur)

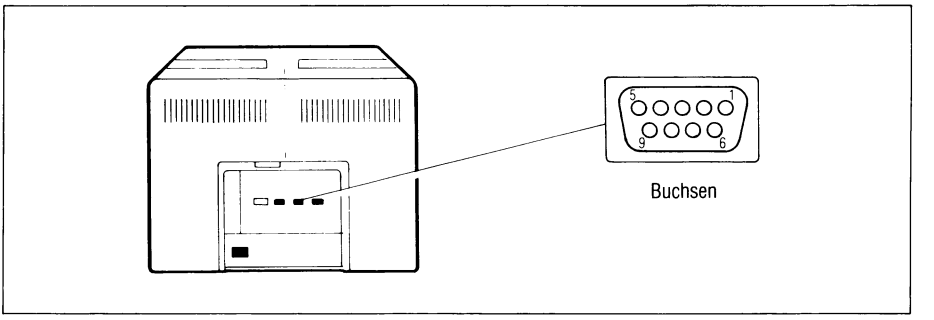


Bild 1-5 SS97 (Buchsen) am Bildschirm

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.

1.2 Hardware-Schnittstellen des PC-X

1.2.1 Allgemeines (Bildschirm des PC-X)

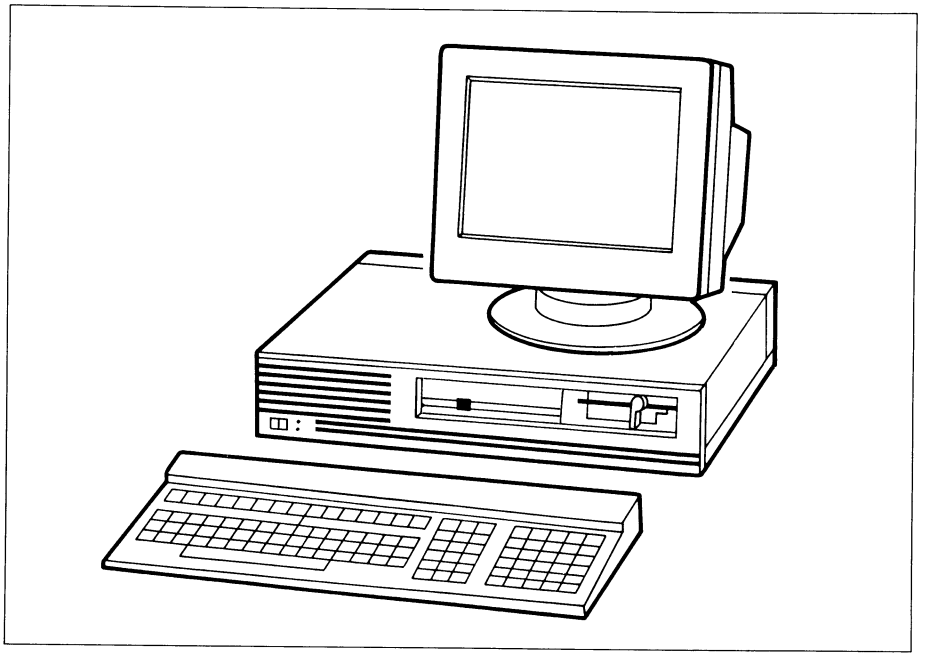


Bild 1-6 Personal Computer PC-X (9781)

1.2.1.1 Aufbau und Arbeitsweise

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 eingebauten Bildschirm-Steuerung angeschlossen. Die Ansteuerung des Bildschirms erfolgt über eine BAS-Schnittstelle (Bild-Austast-Synchronisiersignal). Seine Versorgungsspannung von +12 V liefert die Stromversorgung der Systemeinheit über das gleiche Anschlußkabel.

Die Bildhelligkeit läßt sich durch einen Regler am Bildschirm den Lichtverhältnissen am Arbeitsplatz anpassen. Der Pegel von Halbhelldarstellungen ist auf der Rückseite der Systemeinheit an der Bildschirm-Steuerung einstellbar.

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.

Seine 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-X. Er steuert und überträgt über den internen Erweiterungsbus die Ein- und Ausgaben zwischen Betriebssystem und Tastatur bzw. Bildschirm.

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

1.2.1.2 Funktionsübersicht

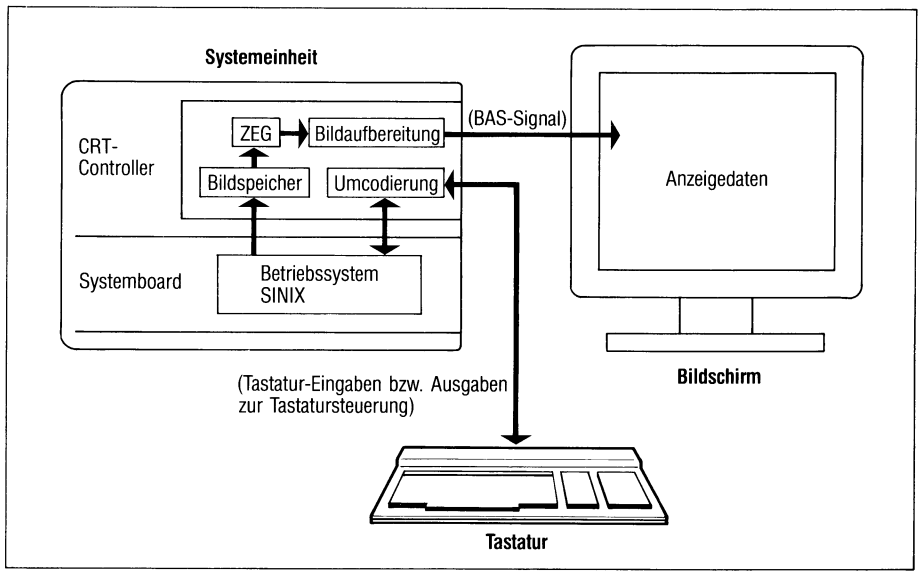


Bild 1-7 Funktionsübersicht

1.2.1.3 Bildschirmsteuerung

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.

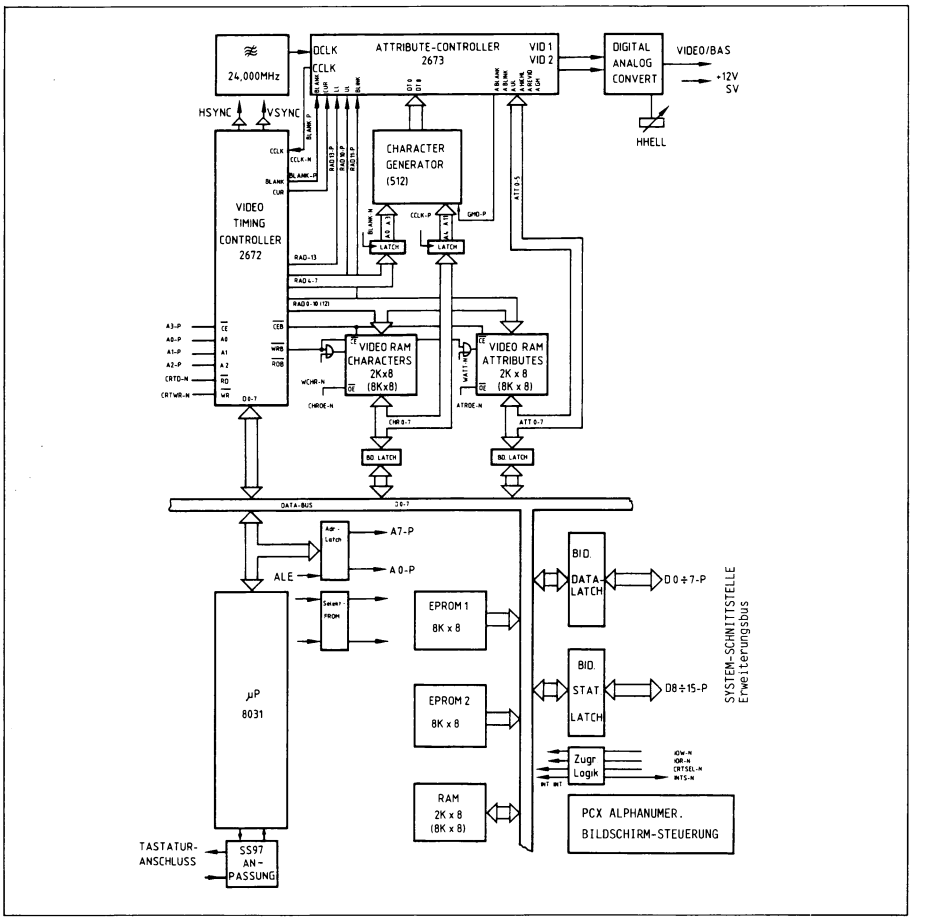


Bild 1-8 Blockschaltbild der Bildschirm-Steuerung

1.2.1.4 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.

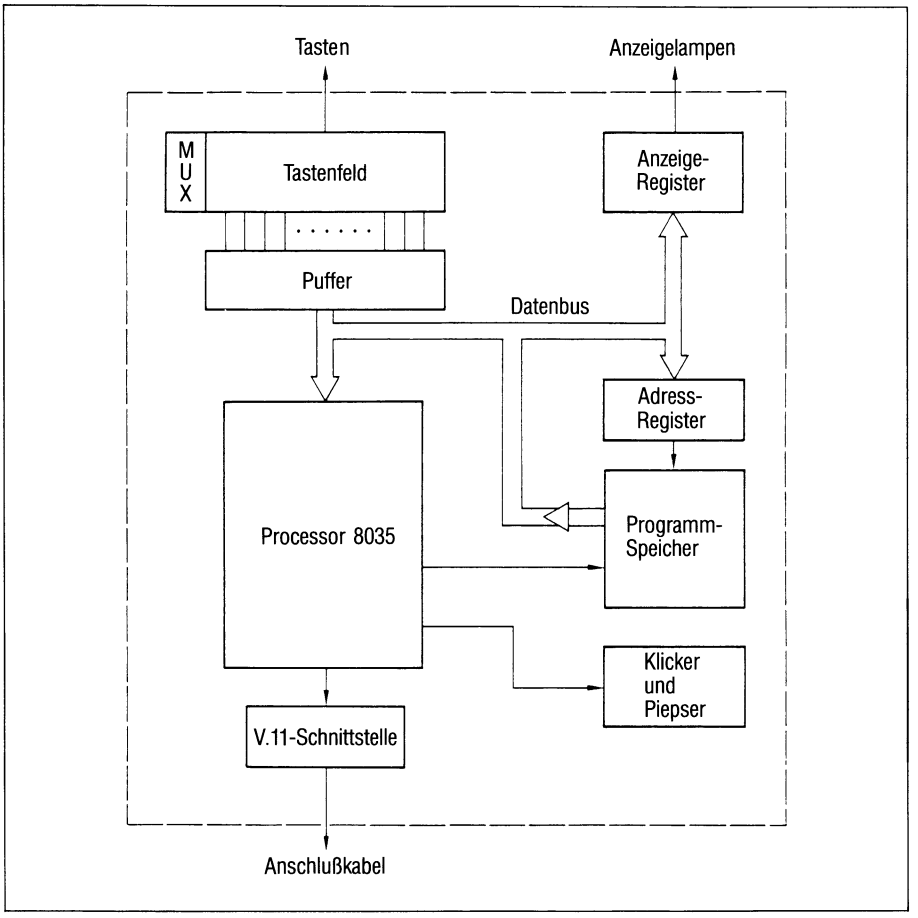


Bild 1-9 Blockschaltbild der Tastatur

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

– international	97811-111
– international und deutsch	97811-112
– international und belgisch QWERTY	97811-113
– international und schwedisch	97811-114
– international und dänisch	97811-115
– international und französisch	97811-116
– international und belgisch AZERTY	97811-117
– international und schweizerisch	97811-118
– international und spanisch	97811-119
– international und italienisch	97811-120

- Zusatz für Bildschirm:
  Höhenverstellung für Bildschirm um 80 mm 97811-390

1.2.2 Belegung der Schnittstelle Bildschirm/Tastatur → Systemeinheit des PC-X

1.2.2.1 Hardware-Schnittstelle

Der Anschluß von Tastatur und Bildschirm erfolgt über je einen 9-poligen Anschlußstecker.
Beide Stecker werden an der Rückseite der Systemeinheit am Anschlußfeld der Bildschirm-Steuerungs angeschlossen.
Die Bildschirm-Steuerung ist auf den Erweiterungsbus des Basissystems aufgesteckt.

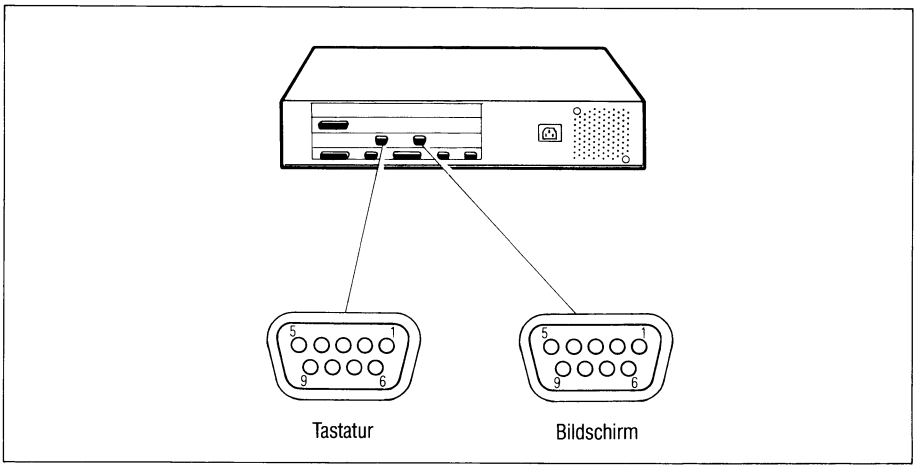


Bild 1-10 Anschlußfeld der Systemeinheit

1.2.2.2 PIN-Belegung des Bildschirm-Anschlusses

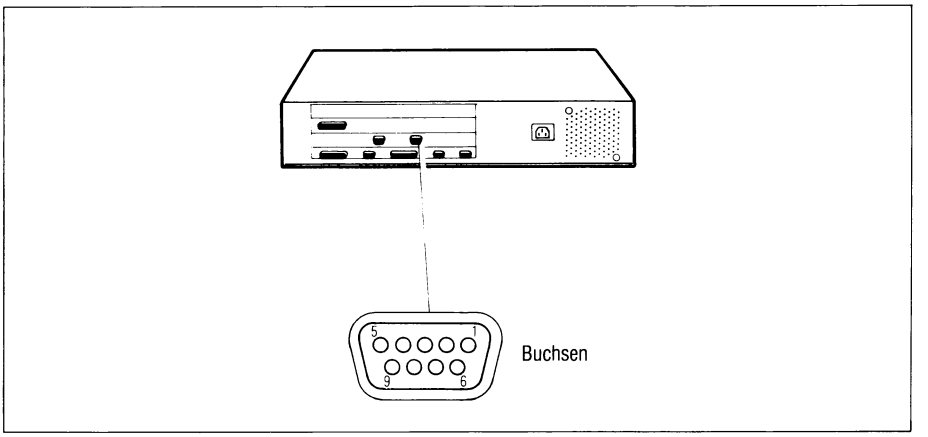


Bild 1-11 Bildschirm-Anschluß an der Systemeinheit

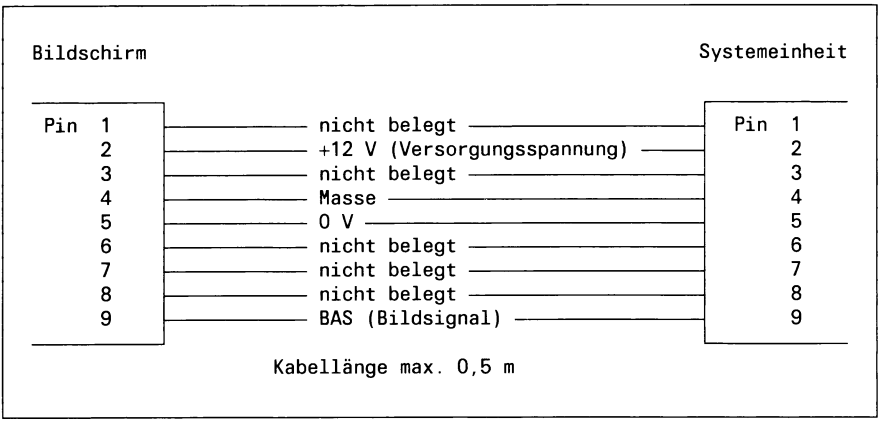


Bild 1-12 Pin-Belegung

1.2.2.3 PIN-Belegung des Tastatur-Anschlusses

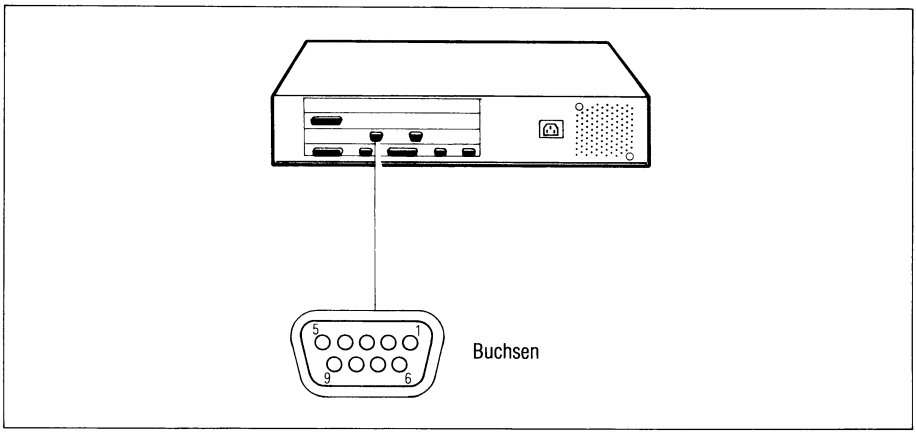


Bild 1-13 Tastatur-Anschluß an der Systemeinheit

Tastatur			Systemeinheit
Pin 1	———	Empfangsdaten ———	Pin 1
2	———	nicht belegt ———	2
3	———	Sendedaten ———	3
4	———	+5 V ———	4
5	———	0 V ———	5
6	———	Empfangsdaten ———	6
7	———	0 V ———	7
8	———	Sendedaten ———	8
9	———	+5 V ———	9

Bild 1-14 Pin-Belegung

Kennwerte der Übertragung (Tastatur → Bildschirm-Steuerung)

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

Hinweis

Über diese Schnittstelle werden die Platzcodes der Tasten und die Schlüsselschalterinformation übertragen.

1.3 Software-Schnittstelle termcap

1.3.1 Einführung

1.3.1.1 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.

1.3.1.2 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.

1.3.1.3 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 Anwendungen 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.

1.3.1.4 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).

1.3.2 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.

1.3.2.1 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.

1.3.2.2 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.

1.3.2.3 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.

1.3.2.4 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.

1.3.2.5 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

1.3.3 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.

1.3.4 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.

1.3.4.1 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.

1.3.4.2 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
```

1.3.5 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
```

1.3.5.1 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);
}
```

1.3.5.2 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);
}
```

1.3.5.3 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);
}
```

1.3.5.4 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);
        }
}
```

1.3.5.5 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);
}
```

1.3.5.6 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);
```

1.3.6 Terminalinitialisierung

1.3.6.1 /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.

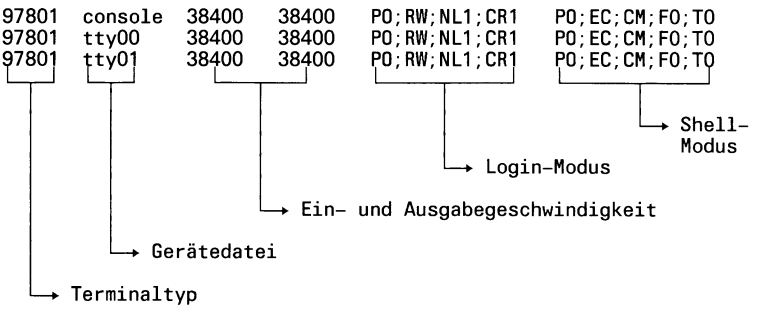
Nachdem der Typ des Terminals in /etc/ttytype festgestellt wurde, wird zusätzlich das 'ti'-Feld des für diesen Typ vorhandenen Eintrags, auf SINIX-Anlagen des Eintrag '97801', ausgesendet, da anschließend eine Cursor-Positionierung durchgeführt wird.

1.3.6.2 /bin/login

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

1.3.6.3 /etc/ttytype

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



1.3.7 Verarbeitung von Eingabesequenzen

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.

1.3.7.1 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.

1.3.7.2 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.

1.3.7.3 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();
}
```

1.3.7.4 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;
}
```

1.4 Bildschirmfunktionen und Datenformate

1.4.1 Kommandoübersicht und Kurzbeschreibung der Normelemente

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

1.4.1.1 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		Mosaik-Zeichensatz → G0 (ladbare Hälfte)
ESC) c		Mosaik-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 ...
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

1.4.1.2 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 [X] 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

1.4.1.3 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

1.4.1.4 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

1.4.1.5 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

1.4.1.6 Kurzbeschreibung der Normelemente (DIN 66254/ISO 6492.2)

a) Direkte Steuerzeichen aus dem Steuerzeichensatz C0 (01) bis (1F) z.B. BS, CR, usw.
b) Direkte erweiterte Steuerzeichen aus dem Steuerzeichensatz C1: ESC und Zeichen der Spalte 4 und 5: (1B)(40) bis (1B)(5F)
c) Steuerzeichenfolgen: Einleitungsfolgen, Parameter, Schlußzeichen

Einleitungsfolgen

Beispiel:

```
CSI  Control Sequence Introducer   ESC [  (1B)(5B)
DCS  Device Control String         ESC P  (1B)(50)
PU1  Private Use 1                 ESC Q  (1B)(51)
```

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. (38) bedeutet die Zahl 8
(36)(32) " 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 \ (1B)(5C)

F Final Code: ASCII-Zeichen aus den Spalten 4, 5, 6 oder 7 (40) bis (7E), das die Bedeutung der Steuerzeichenfolge festlegt.

1.4.2 Funktion und Befehle für BS

1.4.2.1 Codierung und Zeichendarstellung

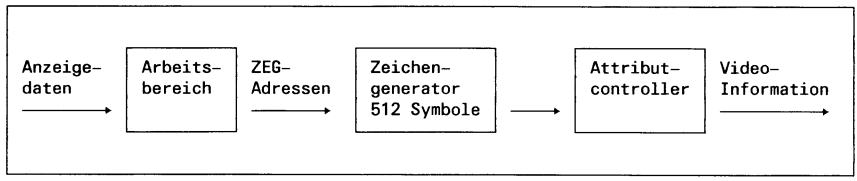


Bild 1-15 Schema der Zeichendarstellung

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 (hex. 0E, 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 (hex. 0F, CTRL O) wird wieder G0 zur Darstellung von Anzeigedaten am Bildschirm verwendet.

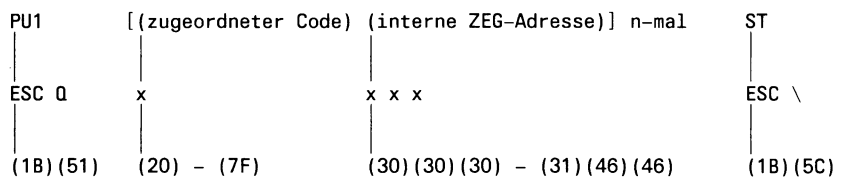
Die Bereitstellungsbereiche G0 und G1 können mit Zeichensätzen wie z.B. International, International A, Deutsch oder Graphik oder andere geladen werden. Das geschieht durch Übergabe der ESC-Folge 'ESC (F' bzw. 'ESC) F', wobei F ein für jeden Zeichensatz festgelegter Final Code ist.
In einem Pseudo-Bereitstellungsbereich G2 mit dem Final Code 'x' kann ein beliebiger Zeichensatz aus den 512 Möglichkeiten des Zeichengenerators zusammengestellt werden.
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.

Zeichensatz	Steuerfolgen zur Code-Übertragung in die Bereitstellungsbereiche	
	G0 nationale Hälfte	G1
International	ESC (@ (1B)(28)(40)	ESC) @ (1B)(29)(40)
International A	ESC (B (1B)(28)(42)	ESC) B (1B)(29)(42)
Deutsch	ESC (K (1B)(28)(4B)	ESC) K (1B)(29)(4B)
Klammern	ESC (w (1B)(28)(77)	ESC) w (1B)(29)(77)
Mosaik	ESC (c (1B)(28)(63)	ESC) c (1B)(29)(63)
IBM	ESC (v (1B)(28)(76)	ESC) v (1B)(29)(76)
Euro Symbole	ESC (u (1B)(28)(75)	ESC) u (1B)(29)(75)
Mathematische Symbole	ESC (t (1B)(28)(74)	ESC) t (1B)(29)(74)
Blanks	ESC (y (1B)(28)(79)	ESC) y (1B)(29)(79)
Pseudo Bereitstellungs-bereich G2	ESC (x (1B)(28)(78)	ESC) x (1B)(29)(78)

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

In diesen Bereich kann mit 'ESC * F', also (1B) (2A) F, jeder gewünschte Vorratsbereich geladen werden.
Mit dem Controllstring 'PU1, Ladeinformation, ST' können dann einzelne Zeichengeneratoradressen 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.

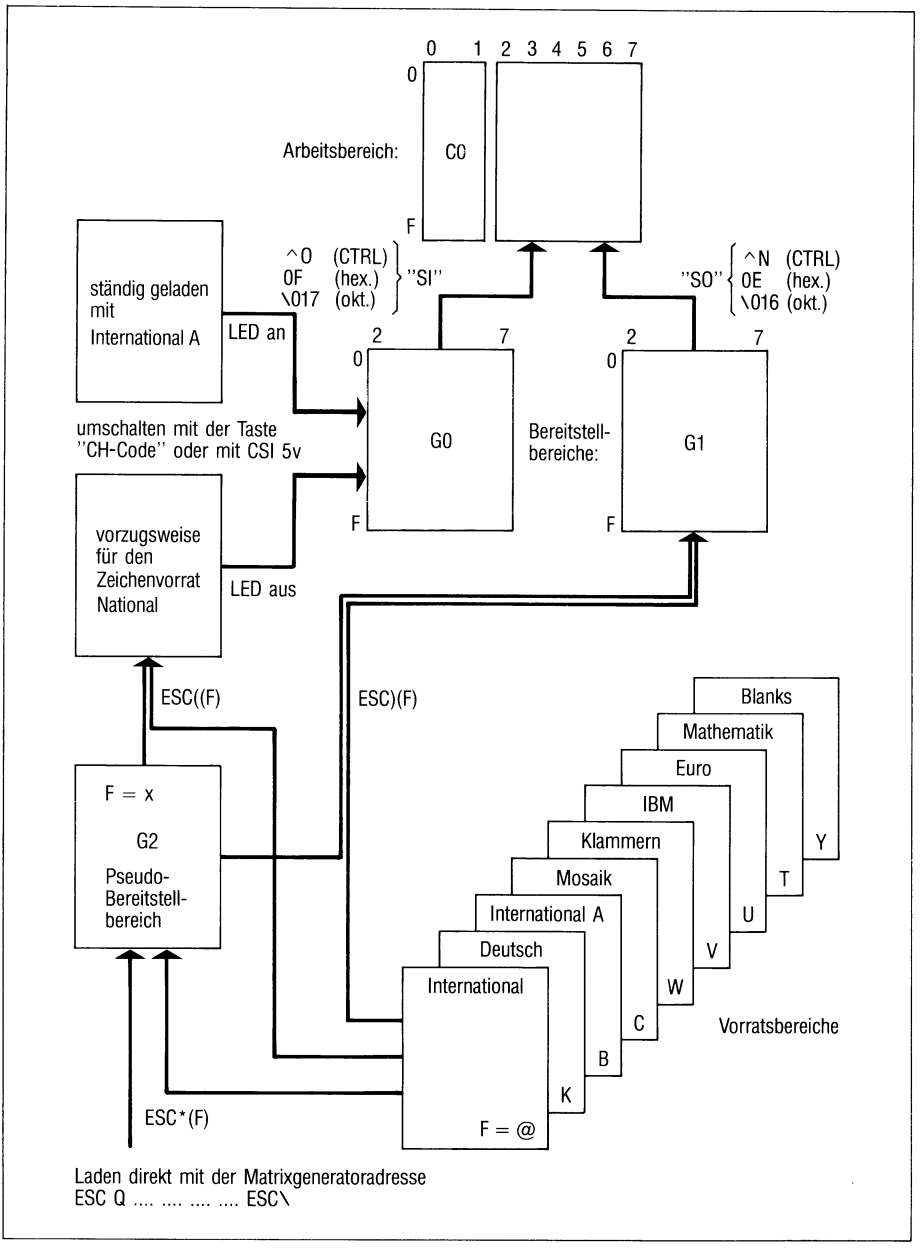


Bild 1-16 Übersicht der möglichen Codiertabellen

1.4.2.2 Befehle zur Codiertabellen-Umschaltung

Umschalten auf G0

SI (0F)	^O

Es wir die in G0 aktuelle Umsetztabelle verwendet.

Umschalten auf G1

SO (0E)	^N

Es wird entsprechend G1 umgesetzt.

Umschalten innerhalb G0

CSI 5 v (1B)(5B)(35)(76)

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 (1B)(5B)(31)(30)(76)

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

Freigeben der Taste CH CODE

```
CSI 11 v   (1B)(5B)(31)(31)(76)
```

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   (1B)(5B)(31)(33)(76)
```

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   (1B)(50)(31)(33)(76)(Pn)(1B)(5C)
```

Pn:	0	(30)	G0	International A
	1	(31)	G0	National
	2	(32)	G1	International A in G0
	3	(33)	G1	National in G0

1.4.2.3 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 @   (1B)(5B)(Pn)(40)
```

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   (1B)(5B)(Pn)(4C)
```

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

1.4.2.4 Cursor-Befehle

Beim Einschalten der PC-MX 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   (1B)(5B)(37)(70)
```

Standard nach RIS bzw. Netz-Ein.

Cursor dunkel **COF (Cursor off)**

```
CSI 6 p   (1B)(5B)(36)(70)
```

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

```
CSI Pn A  (1B)(5B)(Pn)(41)
```

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  (1B)(5B)(Pn)(42)
```

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  (1B)(5B)(Pn)(43)
```

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  (1B)(5B)(Pn)(44)
```

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  (1B)(5B)(Pl)(3B)(Pc)(48)
```

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 → BE)

```
CSI 6 n  (1B)(5B)(36)(6E)
```

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

Absolute Cursor-Position senden **CPR (Curser position report)**

```
CSI Pl;Pc R  (1B)(5B)(Pl)(3B)(Pc)(52)
```

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  (1C)(line + 20)(column + 20)
```

- Der Zeilenadressierung 0 bis 24 entsprechen die ASCII-Werte X'20' bis X'38' (Blank bis 8).
- Der Spaltenadressierung 0 bis 79 entsprechen die ASCII-Werte X'20' bis X'6F' (Blank bis o).

Kurz-Cursor-Positions-Abfrage

```
wie CPR:  CSI 5 p  (1B)(5B)(35)(70)
```

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

Achtung

Die Folge IS4 (1C) 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)**

```
(08)
```

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

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

(09)

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 (1B)(5B)(Pn)(5A)

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

(Standardwert-Annahme: 1)

Zeilenvorschub **LF (Line feed)**

(0A)

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)**

```
(0D)
```

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

Nächste Zeile **NEL (Next line)**

```
ESC E  (1B)(45)
```

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  (1B)(5B)(73)
```

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

Rücksprung **CRST (Cursor restore)**

```
CSI u  (1B)(5B)(75)
```

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

1.4.2.5 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   (1B)(5B)(31)(30)(75)
```

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   (1B)(5B)(31)(31)(75)
```

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   (1B)(5B)(Pn)(53)
```

Mit Pn wird die Anzahl der Zeilen angegeben.

(Standardwert-Annahme: 1)

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

```
CSI Pn T   (1B)(5B)(Pn)(54)
```

Mit Pn wird die Anzahl der Zeilen angegeben.

(Standardwert-Annahme: 1)

1.4.2.6 Löschbefehle

Zeichen löschen **DCH (Delete Character)**

```
CSI Pn P   (1B)(5B)(Pn)(50)
```

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   (1B)(5B)(Pn)(4D)
```

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   (1B)(5B)(Pn)(4B)
```

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

```
0 (30)        1 (31)        2 (32)        → Löschen mit Blank, Attribute
                                              werden gelöscht
3 (33)        4 (34)        5 (35)        → Löschen mit Nil, Attribute werden
                                              gelöscht
6 (36)        7 (37)        8 (38)        → Löschen mit Blank und mit dem zu-
                                              letzt gültigen Attribut vorbelegen
                                              (sofern Attribute nicht gespeichert
                                              sind)
9 (39)        10 (31)(30)   11 (31)(31)   → Löschen mit Nil und mit dem zu-
                                              letzt gültigen Attribut vorbelegen
                                              (sofern Attribute nicht gespeichert
                                              sind)
12 (31)(32)   13 (31)(33)   14 (31)(34)   → Löschen mit Blank, ohne Verändern
                                              der Attribute
15 (31)(35)   16 (31)(36)   17 (31)(37)   → Löschen mit Nil, ohne Verändern
                                              der Attribute

                            └→ Löschen aller Zeichen der Zeile

              └→ Löschen von Zeilenanfang bis einschließlich Cursor-Positon

└→ Löschen von Cursor-Position (einschließlich) bis Zeilenende

(Standardwert-Annahme: 0)
```

Bildschirm löschen **ED (Erase in display)**

CSI Pn J (1B)(5B)(Pn)(4A)

Der Befehl bezieht sich auf den Bildverschiebebereich.

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

0 (30)	1 (31)	2 (32)	→ Löschen mit Blank, Attribute werden gelöscht
3 (33)	4 (34)	5 (35)	→ Löschen mit Nil, Attribute werden gelöscht
6 (36)	7 (37)	8 (38)	→ Löschen mit Blank und mit dem zuletzt gültigen Attribut vorbelegen (sofern Attribute nicht gespeichert sind)
9 (39)	10 (31)(30)	11 (31)(31)	→ Löschen mit Nil und mit dem zuletzt gültigen Attribut vorbelegen (sofern Attribute nicht gespeichert sind)
12 (31)(32)	13 (31)(33)	14 (31)(34)	→ Löschen mit Blank, ohne Verändern der Attribute
15 (31)(35)	16 (31)(36)	17 (31)(37)	→ Löschen mit Nil, ohne Verändern der Attribute

→ Löschen aller Zeichen des Bildes; der Cursor geht zum Bildanfang (Verschiebebereich)

→ Löschen von Bildanfang bis einschließlich Cursor-Position

→ Löschen von Cursor-Position (einschließlich) bis Bildende (Verschiebebereich)

(Standardwert-Annahme: 0)

1.4.2.7 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.

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

```
CSI P1;...Pn m  (1B)(5B)(P1)(3B)...(Pn)(6D)
```

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

Parameter	Darstellungsart
0 (30)	Normal (Standardwert-Zuweisung)
2 (32)	Halbhell
4 (34)	Unterstrichen
5 (35)	Blinken
7 (37)	Invers
8 (38)	Dunkel (unterdrückte Darstellung)
50 (35)(30)	Attribute speichern *)

*) 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	(32)(3B)(34)	Halbhell, unterstrichen
2;4;7	(32)(3B)(34)(3B)(37)	Halbhell, invers, unterstrichen
2;5	(32)(3B)(35)	Blinken zwischen normal und halbhell
2;5;7	(32)(3B)(35)(3B)(37)	Blinken zwischen halbhell und halbhell invers
2;7	(32)(3B)(37)	Halbhell, invers
4;5	(34)(3B)(35)	Blinken zwischen normal und unterstrichen
4;5;7	(34)(3B)(35)(3B)(37)	Blinken invers, unterstrichen
4;7	(34)(3B)(37)	Unterstrichen, invers
5;7	(35)(3B)(37)	Blinken zwischen normal und invers
2;4;5	(32)(3B)(34)(3B)(35)	Blinken halbhell, unterstrichen

Blinken

```
CSI Pn p   (1B)(5B)(Pn)(70)
```

Pn: 0 (30) Blinken ein, Zeile 1 - 25 bzw. 1 - 24 (25/24-Zeilen-Mode)
1 (31) Blinken aus, Zeile 1 - 25 bzw. 1 - 24 (25/24-Zeilen-Mode)
2 (32) Blinken ein, Zeile 25
3 (33) Blinken aus, Zeile 25

Bildschirm dunkel steuern

```
CSI 8 p   (1B)(5B)(38)(70)
```

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

Bildschirm hell steuern

```
CSI 9 p   (1B)(5B)(39)(70)
```

Der Bildinhalt wird wieder angezeigt.

(Standard)

Hervorheben

```
CSI 10 p   (1B)(5B)(31)(30)(70)
```

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

1.4.2.8 Bildschirm-Format und weitere Einstellung

Bildverschiebebereich

```
CSI Pt;Pb r   (1B)(5B)(Pt)(3B)(Pb)(72)
```

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   (1B)(5B)(31)(75)
```

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   (1B)(5B)(30)(75)
```

Der Bildschirm besteht aus einem zusammenhängenden 25-Zeilen Feld.

Löschmuster

```
CSI 3 u   (1B)(5B)(33)(75)
```

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   (1B)(5B)(32)(75)
```

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

Video-Timeout

Einschalten

```
CSI 5 u   (1B)(5B)(35)(75)
```

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   (1B)(5B)(34)(75)
```

Ausschalten des 10-Minuten-Video-Timeouts.

Auto-Roll-Mode (Standard)

```
CSI 9 u   (1B)(5B)(39)(75)
```

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 (1B)(5B)(38)(75)

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.

1.4.2.9 Rücksetz-Befehl

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

ESC c (1B)(63)

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.

1.4.3 Funktionen und Befehle für Tastatur

1.4.3.1 Tastaturbefehle

DMI (Disable manual input)

ESC ` (1B)(60)

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

EMI (Enable manual input)

ESC b (1B)(62)

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 (1B)(5B)(Pn)(71)

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 (1B)(5B)(30)(71)

Die Kombination mit LED-an-Befehlen in einer Befehlsfolge ist zulässig.
Beispiel: CSI 0;1;3;5 q

Repeat aus/an

CSI Pn s (1B)(5B)(Pn)(73)

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**

(07)

Clicker aus/an

CSI Pn s (1B)(5B)(Pn)(73)

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.

Schlüsselschalter-Status senden

Standardmäßig ist nur ein Schlüsselschalter eingebaut. Ein nicht eingebauter Schlüsselschalter gilt als geschlossen.

```
CSI 0 w   (1B)(5B)(30)(77)
```

Nachdem ein Befehl 'Schlüsselschalter-Status senden' zur BE (bzw. Tastatur) übergeben worden ist, wird der Status wie folgt gesendet:

```
DCS 0 w XX ST   (1B)(50)(30)(77)(XX)(1B)(5C)
```

XX:			S1:	S2:
	P	(50)	geschlossen	geschlossen
	Q	(51)	geschlossen	offen
	R	(52)	offen	geschlossen
	S	(53)	offen	offen

Ändert sich der Schlüsselschalterstatus, wird obiger Device Control String automatisch ohne Anforderung zur SE gesendet.

Sicherungsfunktion des Schlüsselschalters 1:
Wird der Schlüsselschalter 'geöffnet', werden keine Tasten-Codes mehr von der BE zur SE gesendet.

Magnetkartenleser

```
CSI Pn t   (1B)(5B)(Pn)(74)
```

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  (1B)(50)(74)(XX)(1B)(5C)
```

XX: @ (40) Magnetkartenleser vorhanden
A (41) 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  (1B)(50)(74)(XX)(Kartendaten)(1B)(5C)
```

XX: C (43) Kartendaten im 5-Bit-ABA-Format wurden erfolgreich gelesen *)
D (44) Kartendaten im 7-Bit-SIPASS-Format wurden erfolgreich gelesen
E (45) Timeout-Fehler. Es wurde eine Karte im Ausweisleser aber nicht innerhalb 1 sec. gelesen
F (46) 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   (1B)(50)(74)(42)(1B)(5C)
```

1.4.3.2 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   (1B)(5B)(37)(75)
```

Alle anderen Tastaturen

```
CSI 6 u   (1B)(5B)(36)(75)
```

1.4.4 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 (1B)(28)(4B) 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' (0E) 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   (1B)(5B)(35)(76).
```

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   (1B)(28)(78)
```

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.

1.4.5 Diagnose-Funktionen

1.4.5.1 DIP-FIX-Schalter-Information abfragen (nur bei PC-MX)

```
CSI 0 y   (1B)(5B)(30)(79)
```

```
DCS 0 y Pn ST   (1B)(50)(30)(79)(Pn)(1B)(5C)
```

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 → 1
- Schalter geschlossen → 0

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

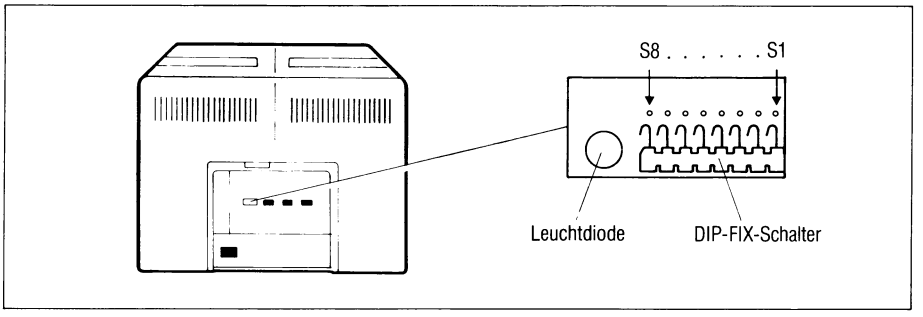


Bild 1-17 DIP-FIX-Schalter

1.4.5.2 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 (000) bis (1FF) 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.

1.4.5.3 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.

1.4.5.4 Systemtest (Bildschirm und Tastatur)

PC-MX:

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-MX 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   (1B)(5B)(Pn)(79)
```

Pn:
1	(31)	Bedieneinheit-Systemtest auslösen
2	(32)	Testergebnis abfragen *)
3	(33)	Bedieneinheit-Systemtest auslösen und Ergebnis abfragen (Kombination von 1 + 2)
4	(34)	Bildschirm Firmware-Version abfragen
5	(35)	Tastatur Firmware-Version abfragen

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

```
CSI 5 n  (1B)(5B)(35)(6E)
```

Folgende Reaktionen sind zu erwarten:

```
CSI Pn n  (1B)(5B)(Pn)(6E)
```

Pn:	0	(30)	BE betriebsbereit
	3	(33)	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  (1B)(50)(32)(79)(Pn1)(3B)(Pn2)...(1B)(5C)
```

Mögliche Codierungen für

Pn1...Pnn:	0	(30)	kein Fehler
	1	(31)	ROM1-Checksummenfehler
	2	(32)	ROM2-Checksummenfehler
	3	(33)	RAM-Fehler
	4	(34)	VIDEO-Fehler
	5	(35)	Tastatur ROM-Checksummenfehler
	6	(36)	Tastatur RAM-Fehler
	7	(37)	Tastatur Timeout (keine Tastatur angeschlossen)
	8	(38)	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   (1B)(50)(34)(79)(Pn)(1B)(5C)
```

Pn ist eine 6-stellige Zahl, z.B. 000017 oder 810113

Firmware-Version der Tastatur

Bei Anforderung der Firmware-Version erfolgt folgende Rückmeldung:

```
DCS 5 y Pn ST   (1B)(50)(35)(79)(Pn)(1B)(5C)
```

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

Bildschirm-Einstell-Funktion

```
CSI Pn v   (1B)(5B)(Pn)(76)
```

Pn = 8 (38) Der gesamte Bildverschiebebereich wird mit jedem Anzeige-Zeichen, das vom PC gesendet wird, beschrieben.
Ergänzung: Zu den horizontalen Verschiebegrenzen können auch vertikale Begrenzungen des Verschiebebereichs definiert werden. Befehlsfolge:
CSI Pn1;Pn2 z (1B)(5B)(Pn1)(3B)(Pn2)(7A)
Pn1 ist die linke, Pn2 die rechte Bildgrenzen-Spalte des so zu bildenden Fensters ($1 \leq Pn1 \leq Pn2 \leq 80$).

Pn = 7 (37) Jedes vom PC gesendete Anzeige-Zeichen wird wieder einfach dargestellt. Die linke/rechte Fenstergrenze wird automatisch wieder 1 bzw. 80.

Zeichenvorrat ausgeben und Dauertest

```
CSI 9 v  (1B)(5B)(39)(76)
```

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

Diese beiden Folgen sind kombinierbar und mit folgender Steuerfolge aufzurufen:

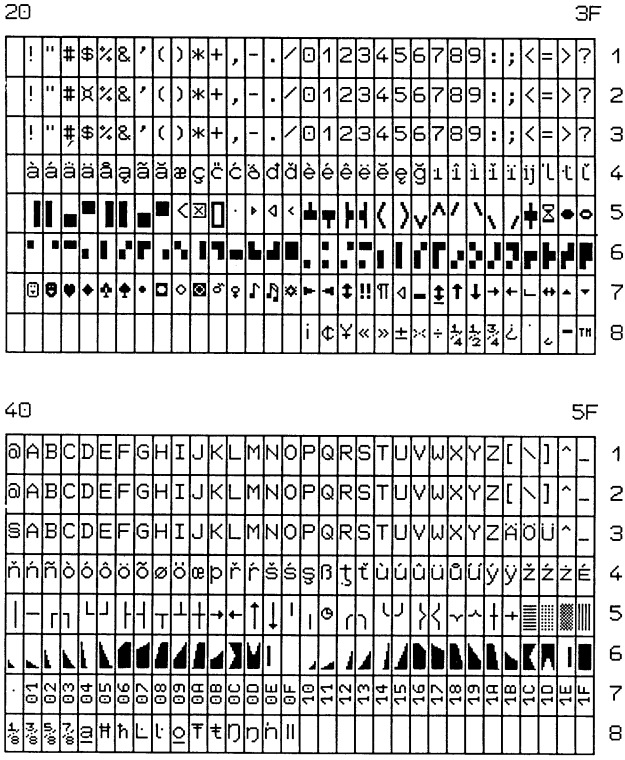
```
CSI 8 v CSI 9 v  (1B)(5B)(38)(76)(1B)(5B)(39)(76)
```

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

1.5 Zeichensätze und Adressen des Matrixgenerators

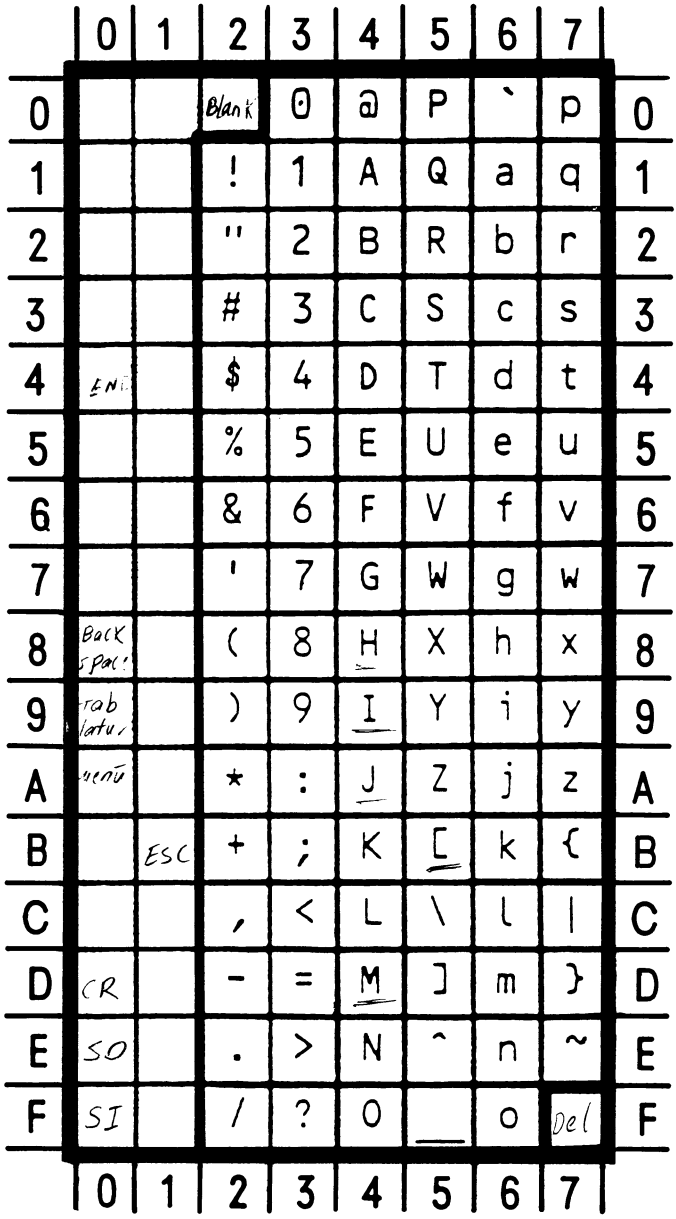
Der verwendete Matrixgenerator enthält 512 verschiedene Zeichen in einer 8x16-Punkt-Auflösung.
Diese 8x16-Punkt-Auflösung wird beim PC-MX direkt zur Anzeige auf den grünen 15"-Bildschirm gebracht.
Bei der Darstellung auf dem weißen 12"-Bildschirm des PC-X kommt eine 9x14-Punkt-Auflösung zur Anwendung.
Um dies mit der gleichen Matrixgenerator-Information zu ermöglichen, wird durch eine Zusatzschaltung die Information der untersten beiden Reihen (der Zeilenabstand) ausgeblendet und 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.

1.5.1 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

1.5.1.1 Zeichensatz: International A



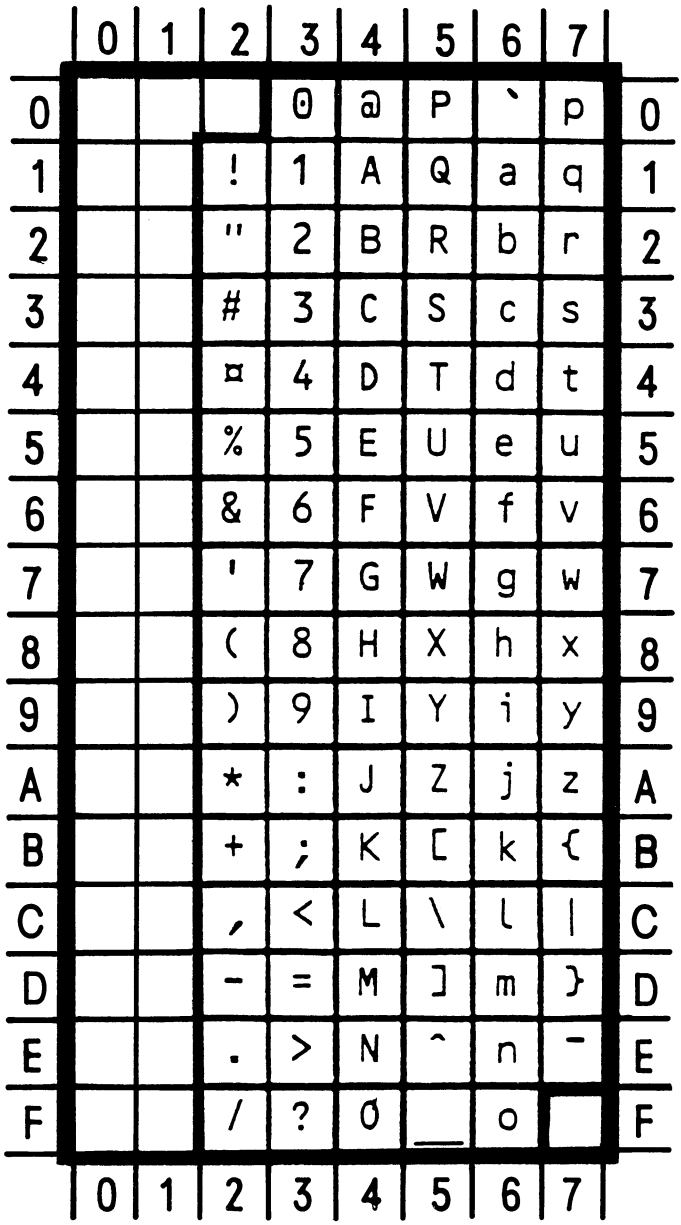
BACK ≙ ^H

TAB → ≙ ^I

MENU ≙ ^J

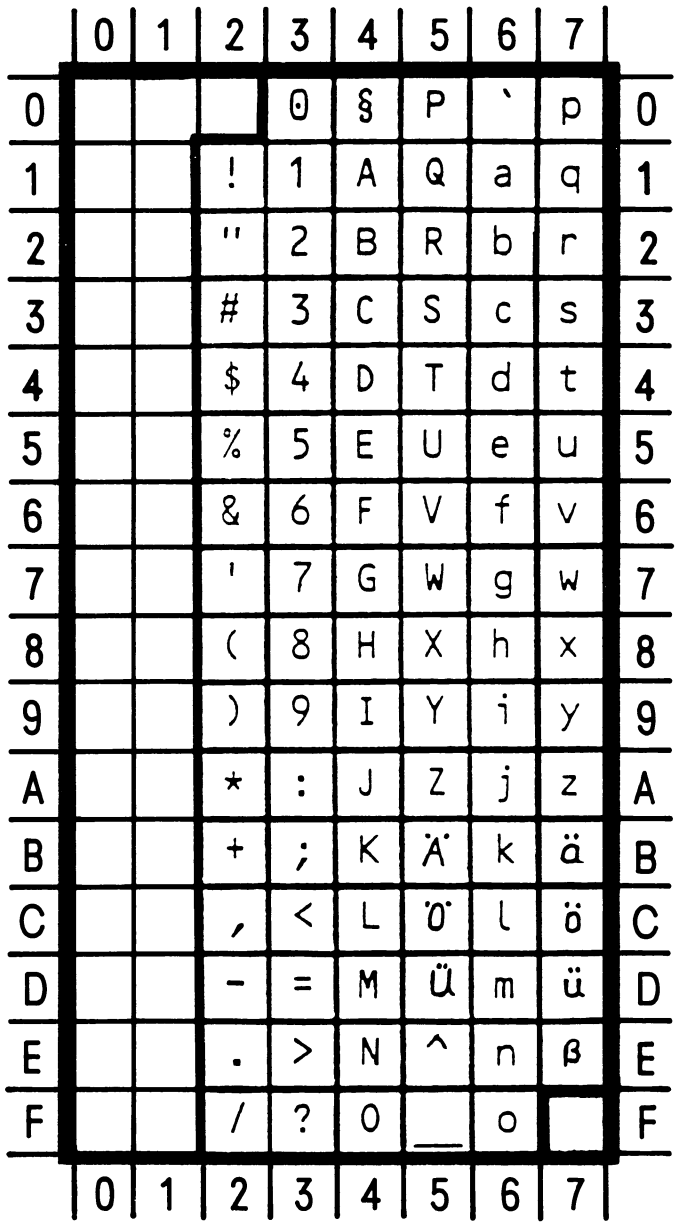
ESC ≙ ^[

1.5.1.2 Zeichensatz: International



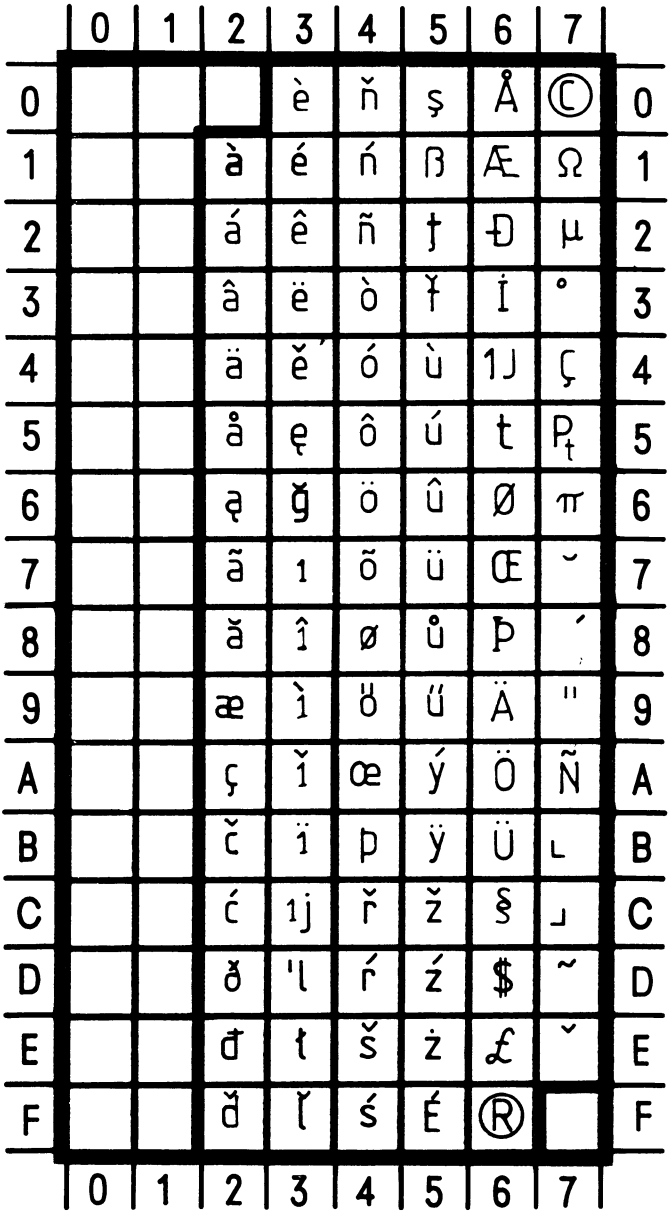
	0	1	2	3	4	5	6	7	
0				0	@	P	`	p	0
1			!	1	A	Q	a	q	1
2			"	2	B	R	b	r	2
3			#	3	C	S	c	s	3
4			¤	4	D	T	d	t	4
5			%	5	E	U	e	u	5
6			&	6	F	V	f	v	6
7			'	7	G	W	g	w	7
8			(	8	H	X	h	x	8
9			)	9	I	Y	i	y	9
A			*	:	J	Z	j	z	A
B			+	;	K	[	k	{	B
C			,	<	L	\	l	|	C
D			-	=	M	]	m	}	D
E			.	>	N	^	n	~	E
F			/	?	O	_	o	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

1.5.1.3 Zeichensatz: Deutsch



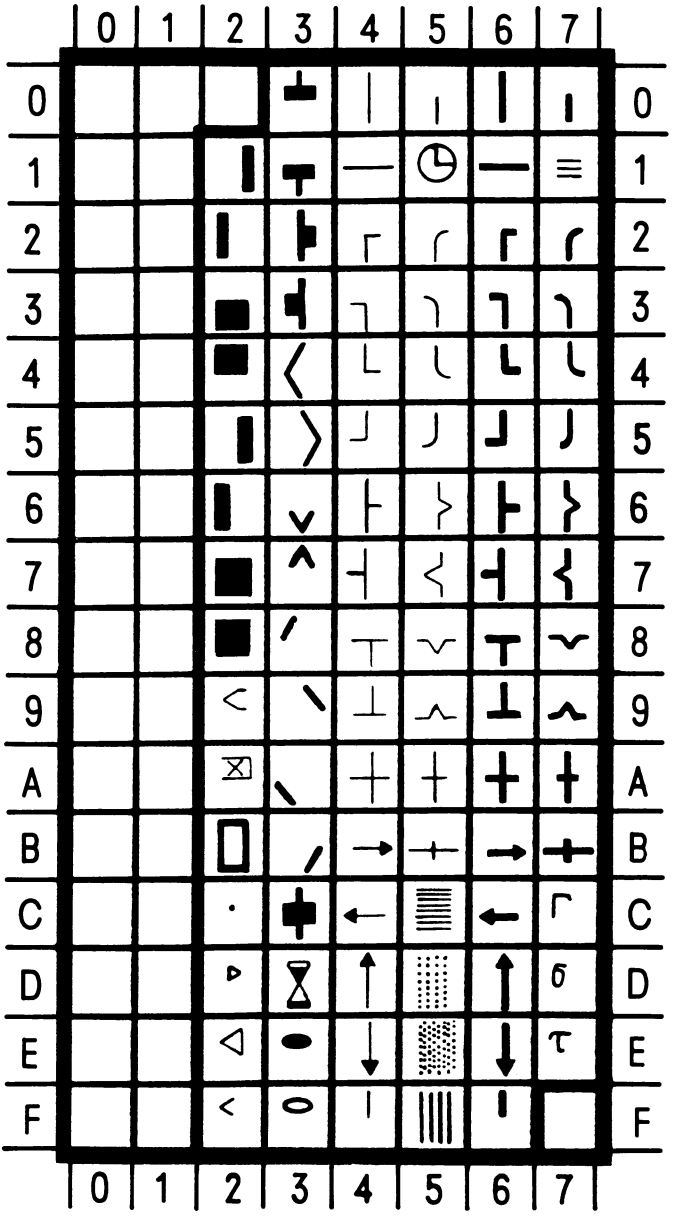
	0	1	2	3	4	5	6	7	
0				0	§	P	`	p	0
1			!	1	A	Q	a	q	1
2			"	2	B	R	b	r	2
3			#	3	C	S	c	s	3
4			$	4	D	T	d	t	4
5			%	5	E	U	e	u	5
6			&	6	F	V	f	v	6
7			'	7	G	W	g	w	7
8			(	8	H	X	h	x	8
9			)	9	I	Y	i	y	9
A			*	:	J	Z	j	z	A
B			+	;	K	Ä	k	ä	B
C			,	<	L	Ö	l	ö	C
D			-	=	M	Ü	m	ü	D
E			.	>	N	^	n	ß	E
F			/	?	O	_	o	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

1.5.1.4 Zeichensatz: Euro

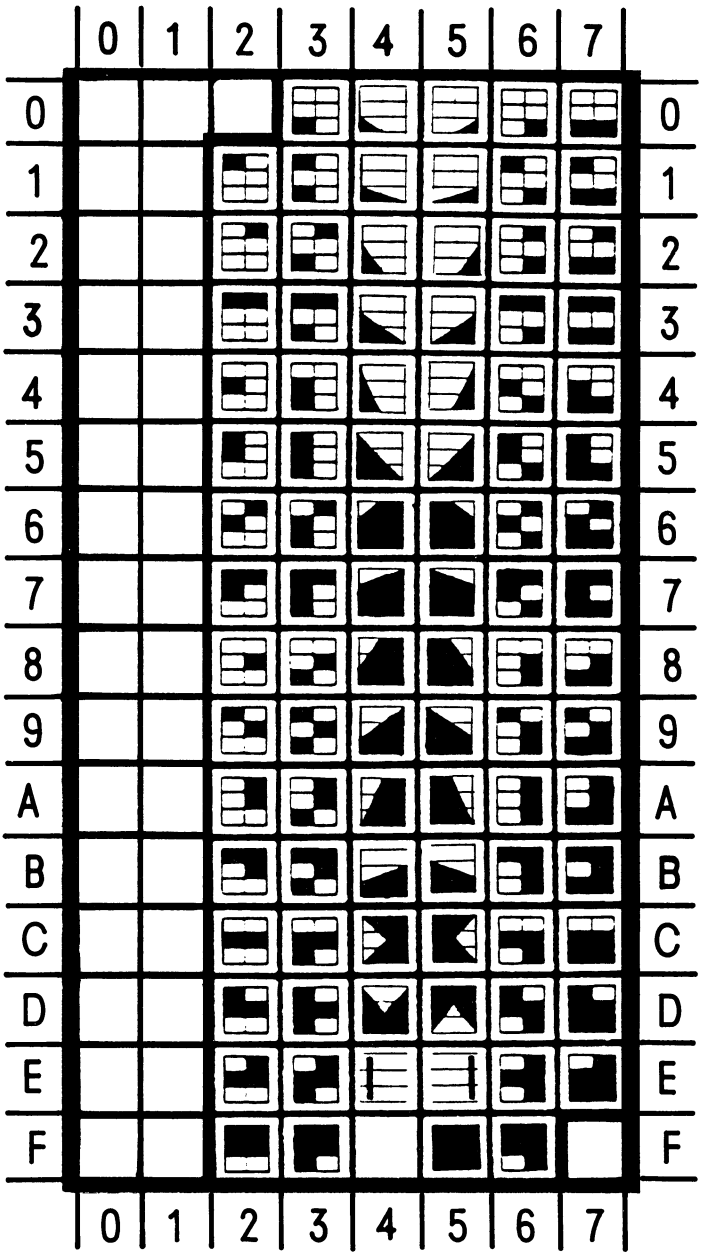


	0	1	2	3	4	5	6	7	
0				è	ň	ş	Å	©	0
1			à	é	ń	ß	Æ	Ω	1
2			á	ê	ñ	ţ	Đ	µ	2
3			â	ë	ò	ť	İ	°	3
4			ä	ě	ó	ù	IJ	ç	4
5			å	ę	ô	ú	ŧ	₧	5
6			ą	ğ	ö	û	Ø	π	6
7			ã	ı	õ	ü	Œ	˘	7
8			ă	î	ø	ů	Þ	´	8
9			æ	ì	ő	ű	Ä	˝	9
A			ç	ǐ	œ	ý	Ö	Ñ	A
B			č	ï	þ	ÿ	Ü	˩	B
C			ć	ij	ř	ž	ß	˩	C
D			ð	ł	ŕ	ź	$	~	D
E			đ	ŧ	š	ż	£	ˇ	E
F			ď	ľ	ś	É	®		F
	0	1	2	3	4	5	6	7	

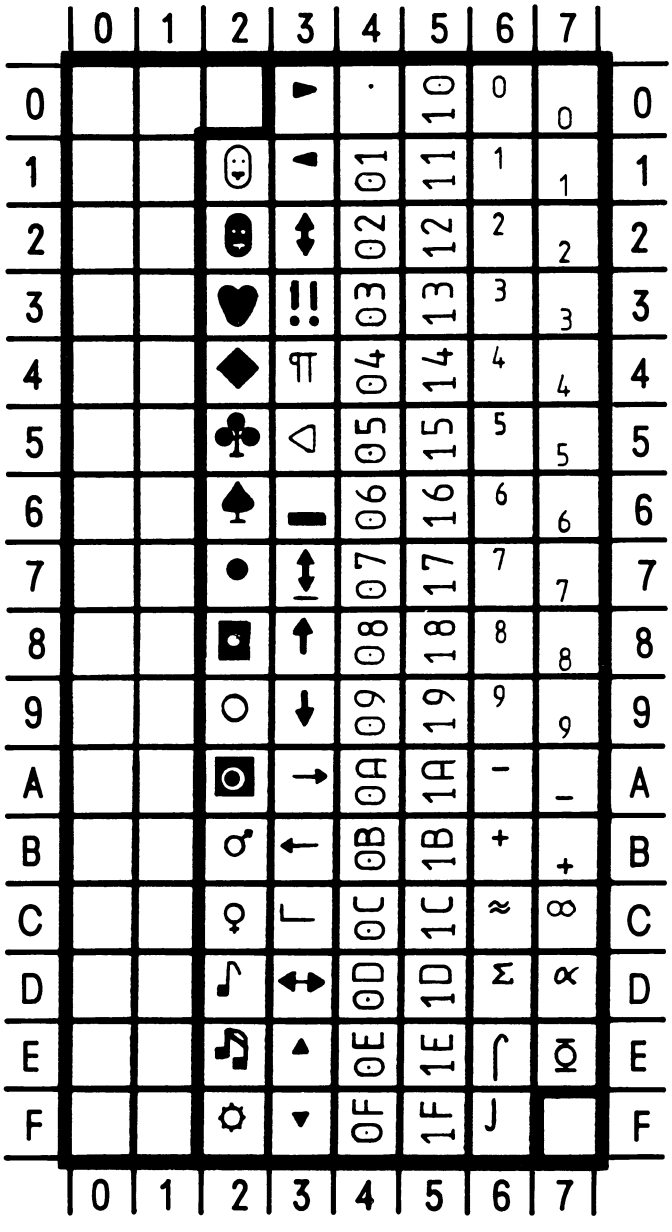
1.5.1.5 Zeichensatz: Klammern



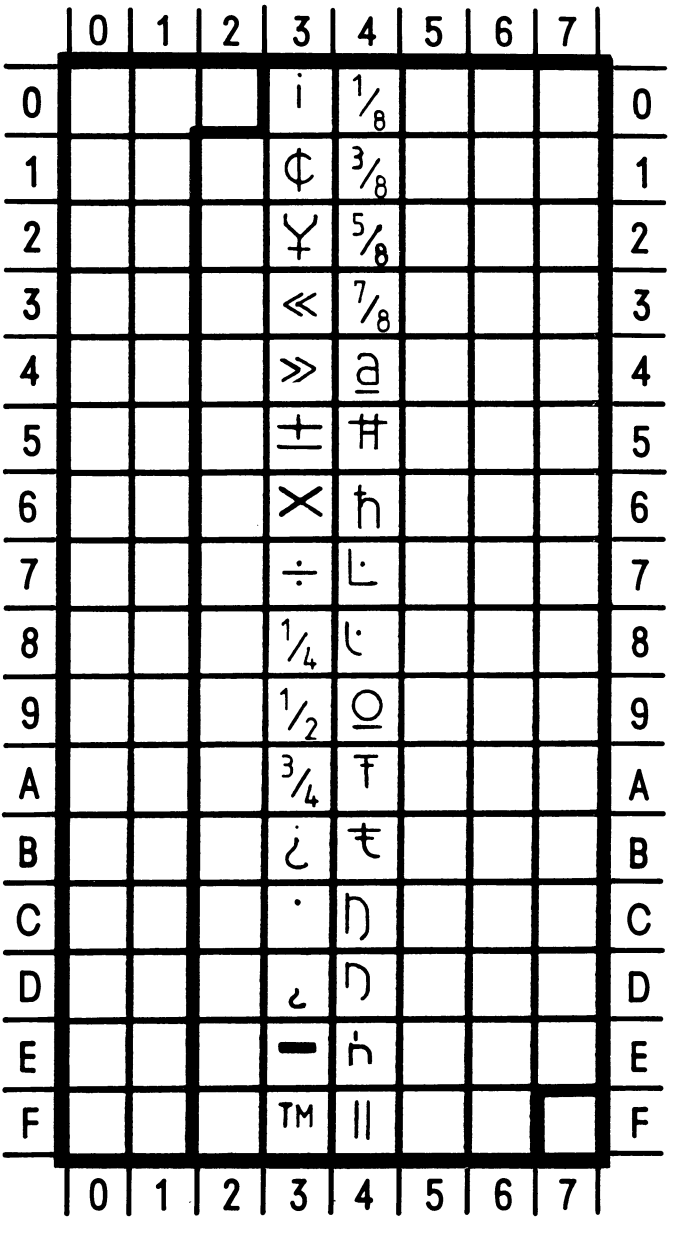
1.5.1.6 Zeichensatz: Mosaic

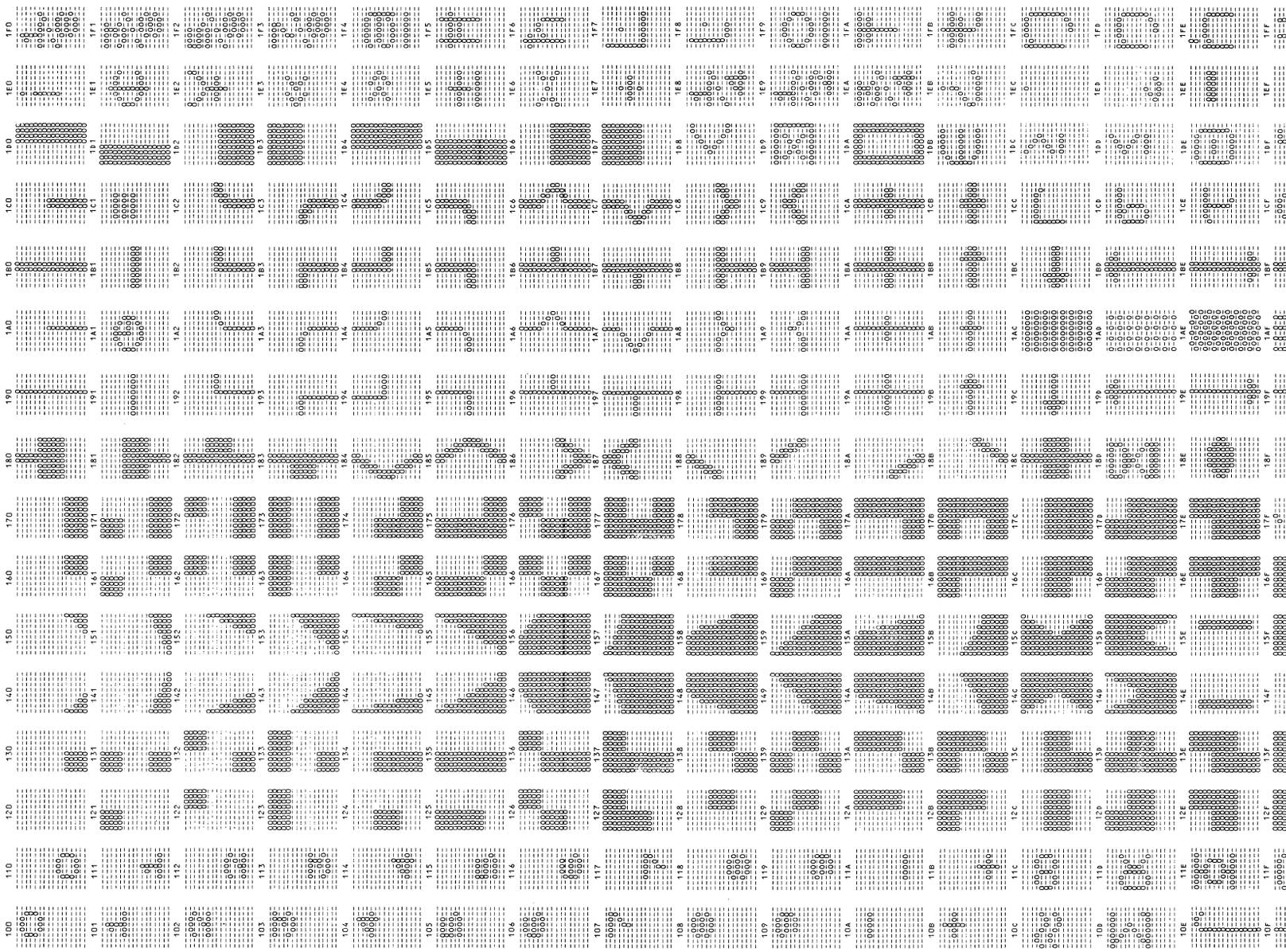


1.5.1.7 Zeichensatz: IBM



1.5.1.8 Zeichensatz: Mathematisch



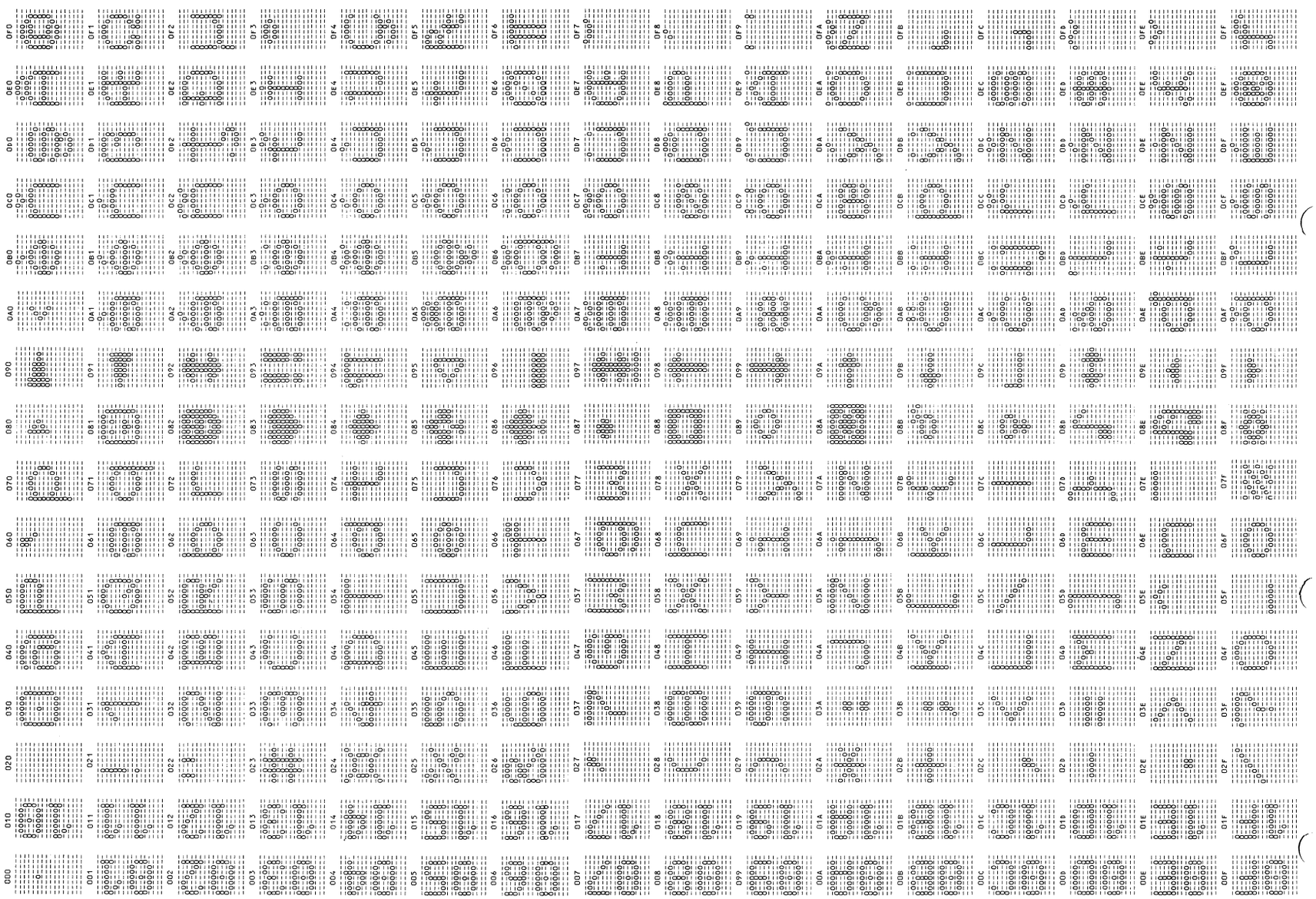


1.5.2 Unterschiedliche Zeichen bei nationalen Tastatur-Varianten

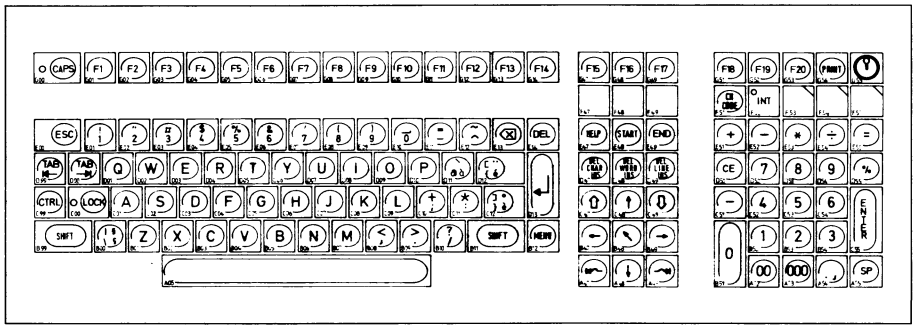
Gegenüberstellung mit interner Matrixgeneratoradresse bei PC-X und PC-MX.

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

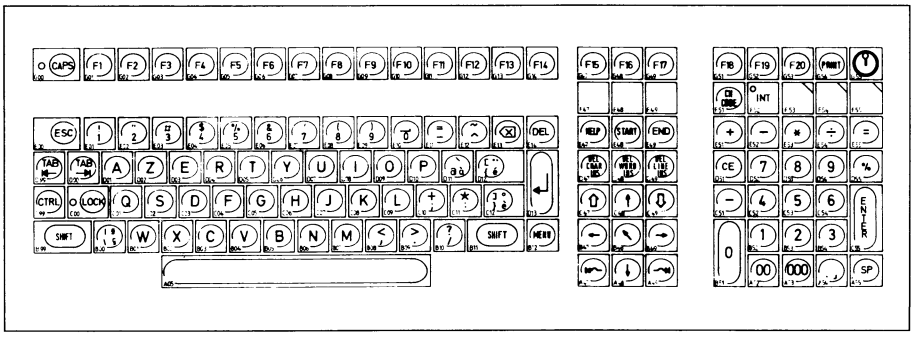
1.5.3 Zeichen des Matrixgenerators mit internen Adressen



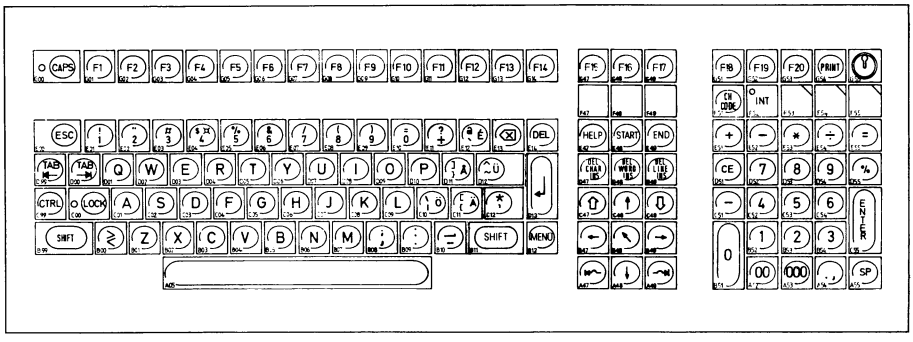
1.6.3 Belgisch-Flämisch (V3) 97801-113/97811-113



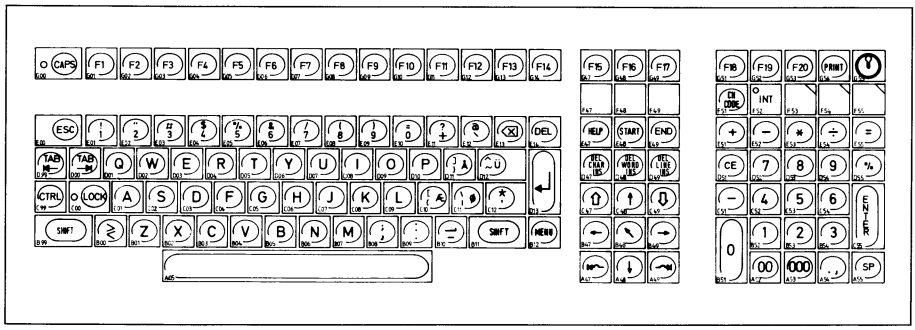
1.6.4 Belgisch-Französisch (V7) 97801-117/97811-117



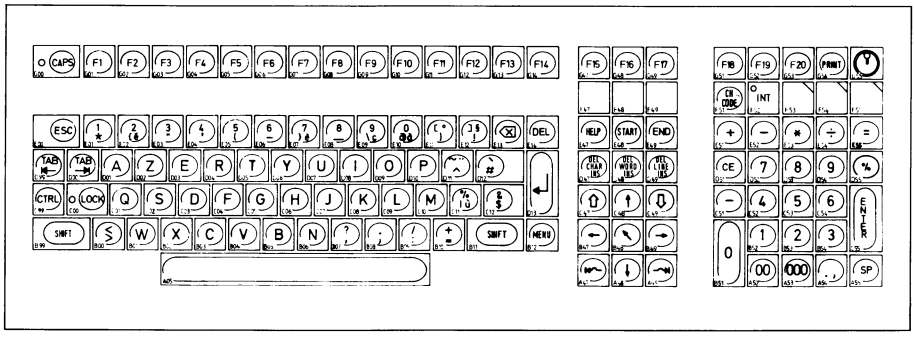
1.6.5 Schwedisch (V4) 97801-114/97811-114



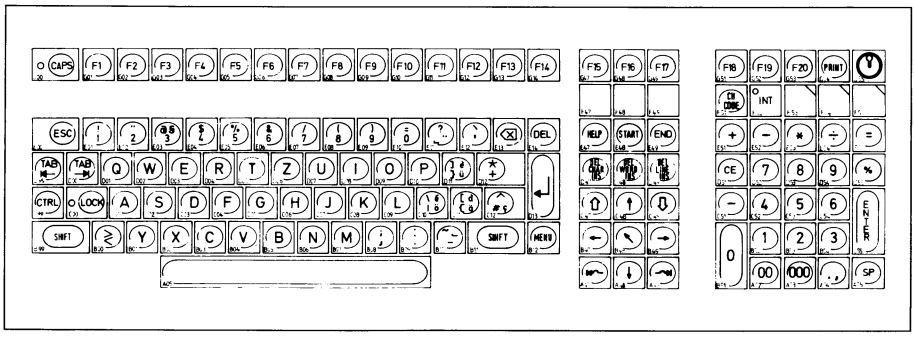
1.6.6 Dänisch (V5) 97801-115/97811-115



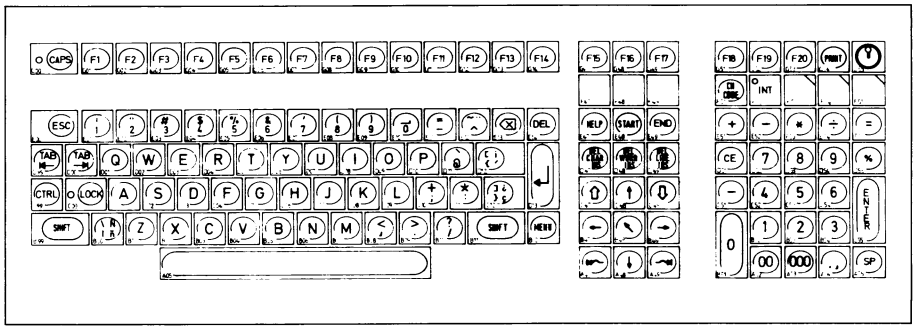
1.6.7 Französisch (V6) 97801-116/97811-116



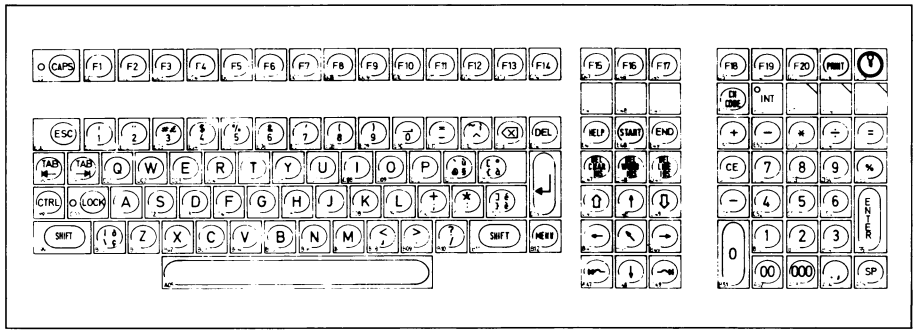
1.6.8 Schweizerisch (V8) 97801-118/97811-118



1.6.9 Spanisch (V9) 97801-119/97811-119



1.6.10 Italienisch (V10) 97801-120/97811-120



1.7 Beispiele

1.7.1 Beispiel 1: Festlegung der Zeichensätze 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-MX (4-Platz-System)

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

Die Einträge für die möglichen Tastaturvarianten sind wie folgt:

```
97801-111       International
\E(B\E[6u

97801-112       Deutsch
\E(K\E[7u

97801-117       Belgisch  AZERTY
\E*B\EQ@0A1[0F9\1340AA]0F3{0B1|0EC}0B0\E\134\E(x\E[6u

97801-113       Belgisch  QWERTY
\E*B\EQ@0A1[0F9\1340AA]0F3{0B1|0EC}0B0\E\134\E(x\E[6u

97801-115       Dänisch
\E*B\EQ[0E1\1340E6]0E0\1360EB{0A90C8}0A5~0D7\E\134\E(x\E[6u

97801-116       Französisch
\E*B\EQ@0A1[0F3\1340AA]0EC{0B1|0D4}0B0~0F9\E\134\E(x\E[6u

97801-120       Italienisch
\E*B\EQ#0EE@0EC[0F3\1340AA]0B1`0D4{0A1|0C3}0B0~0B9\E\134\E(x\E[6u

97801-114       Schwedisch
\E*B\EQ$024@0DF[0E9\1340EA]0E0\1360EB`0B1{0A4|0C6}0A5~0D7\E\134\E(x\E[6u

97801-118       Schweizerisch
\E*B\EQ#0AA@0EC[0A1\1340B1]0B0_0F9{0A4|0C6}0D7~05F\E\134\E(x\E[6u

97801-119       Spanisch
\E*B\EQ[1E0\1340FA]1EB{027|0C2}0AA~0F9\E\134\E(x\E[6u
```

→ Umschalten auf nicht deutsche Tastatur

→ G2 nach 'nationale Hälfte' von G0 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.

1.7.2 Beispiel 2: 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);
}

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

2 Weitere Schnittstellen des PC-MX

2.1 Schnittstellen im Grundausbau (SS97/RS232)

2.1.1 Allgemeines

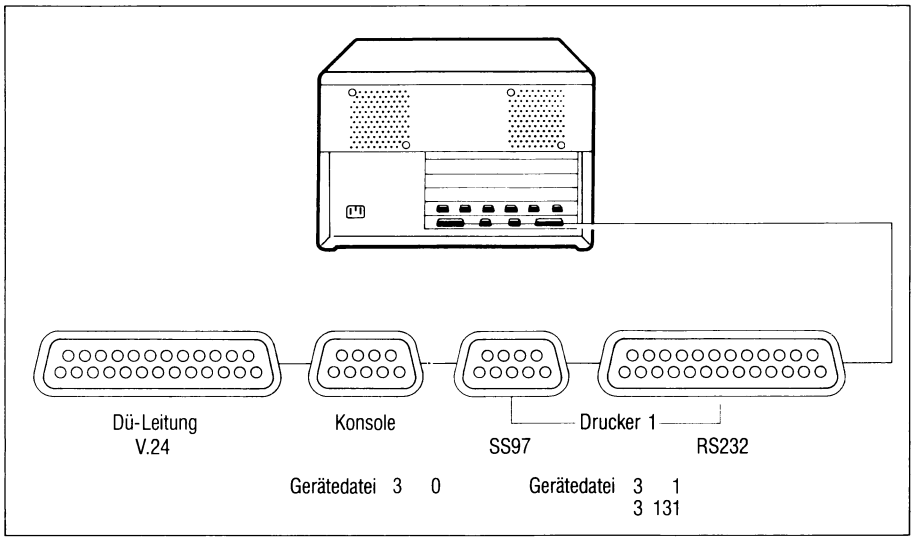


Bild 2-1 Anschlußfeld der Systemeinheit (SE)

Diese CONAC-Schnittstelle (SS97/RS232) ist im Normalfall - außer sie wird bei der Installation des SINIX-Systems frei gehalten - vom ersten Drucker belegt.

Diese Schnittstelle ist die einzige, deren elektrische Eigenschaften mittels Hardwareschalter von V.11-Pegel auf V.28-Pegel umgestellt werden können (0 V, +5 V bzw. -12 V, +12 V).

Es kann nur der elektrische Pegel der Schnittstelle umgeschaltet werden, nicht das logische Protokoll, mit dem das angeschlossene Gerät betrieben wird.

Es können nur Geräte angeschlossen werden, die die Datenflußsteuerung

- mit XON-XOFF oder
- durch Steuerung der Schnittstellenleitung M2 (Sendebereitschaft) 106

durchführen. Letzteres bedeutet, daß der Kanal zu senden aufhört, wenn der Pegel an Pin 5 (M2) auf -3 V geht.

Hinweis

SINIX Version 1.0B: Es ist nur die Datenflußsteuerung mit XON-XOFF in Ausgaberichtung realisiert, d.h. bei Datenausgabe PC → Gerät kann das Gerät mit XOFF die Datenausgabe stoppen und mit XON wieder starten. Bei Dateneingabe Gerät → PC sind durch die Anwendungen geeignete Maßnahmen zu ergreifen, daß der Datenpuffer im PC nicht überläuft.

SINIX Version 1.0C: Es ist die Datenflußsteuerung mit XON-XOFF in Ein- und Ausgaberichtung realisiert.

2.1.1.1 Diagnosehilfe

Auf der Flachbaugruppe CONAC befinden sich drei Anzeigen. Die Anzeige H2 (siehe Bild 2-3) zeigt die Sendedaten der Druckerschnittstelle SS97/RS232 an. Hiermit kann leicht überprüft werden, ob an der Schnittstelle Daten ausgegeben werden.

Fehlermöglichkeiten, wenn keine Daten ausgegeben werden:

- Beschaltung der Schnittstelle falsch
  (Schalter, Schnittstellenleitungen M5, M2, M1),
- Programm, Kommando fehlerhaft
- Stecker zum Anschlußfeld nicht mehr richtig gesteckt.

2.1.2 Umschaltung der Schnittstelle (SS97/RS232)

Die Sendedaten D1 werden immer auf beiden Schnittstellen ausgegeben (siehe Bild 2-2).
Über den Schalter S4 ist die Empfangsdatenleitung D2 von SS97 auf RS232 umzuschalten.
Die Schnittstellenleitungen M1, M2 und M5 können von außen beschaltet oder über die Schalter S5, S6 und S7 auf definierten Pegel geschaltet werden.

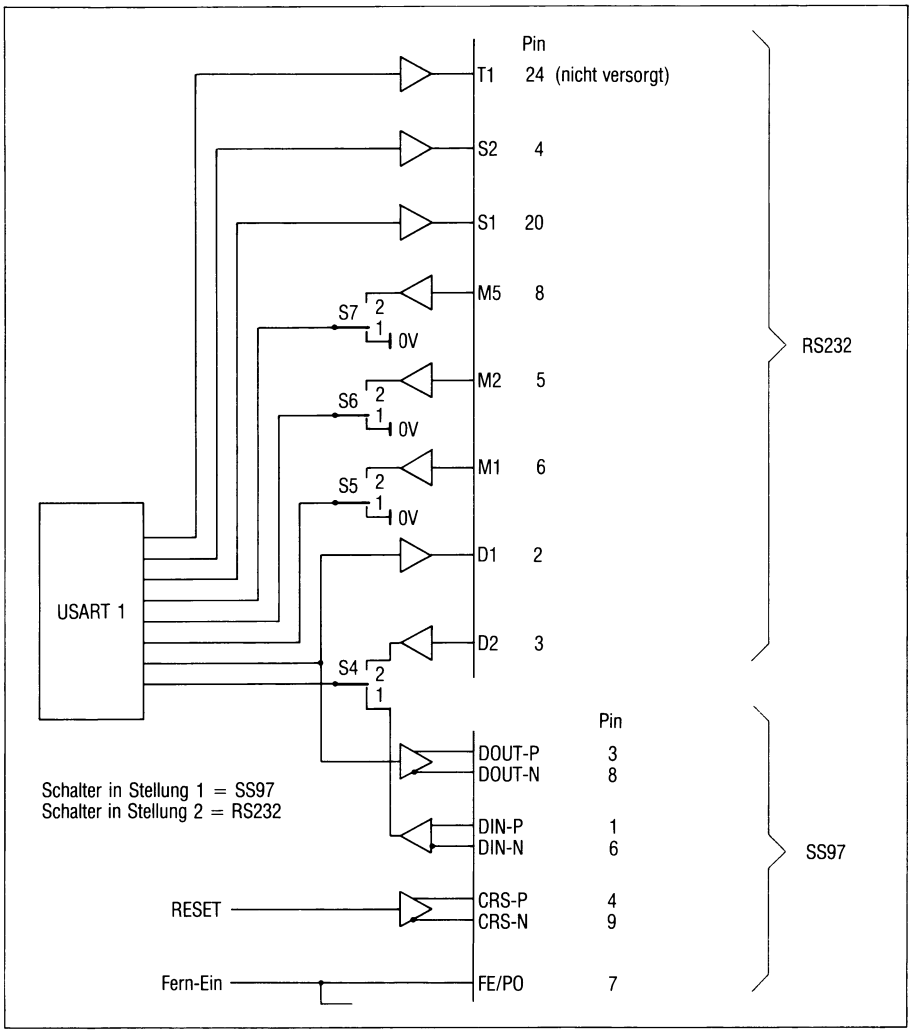


Bild 2-2 Umschaltung SS97/RS232

2.1.3 Umschaltung auf der Flachbaugruppe CONAC (SS97/RS232)

Im PC-MX sind alle Flachbaugruppen (Fbg.), auch Logikkarten oder Printed Boards etc. genannt, mit Kurzbezeichnungen (z.B. CONAC) und Nummern (z.B. S26361-D245-V1) versehen.
Über diese Bezeichnungen werden die Fbg. in diesem Buch identifiziert.

Achtung

Auf dieser Fbg. befindet sich auch die Batterie für die Systemuhr. Legen Sie die Fbg. nicht auf eine leitende Unterlage! Ein Kurzschluß zerstört die Batterie.
Überprüfen Sie nach dem Wiedereinsetzen der Flachbaugruppe die Systemzeit (setzen mit /etc/mc).

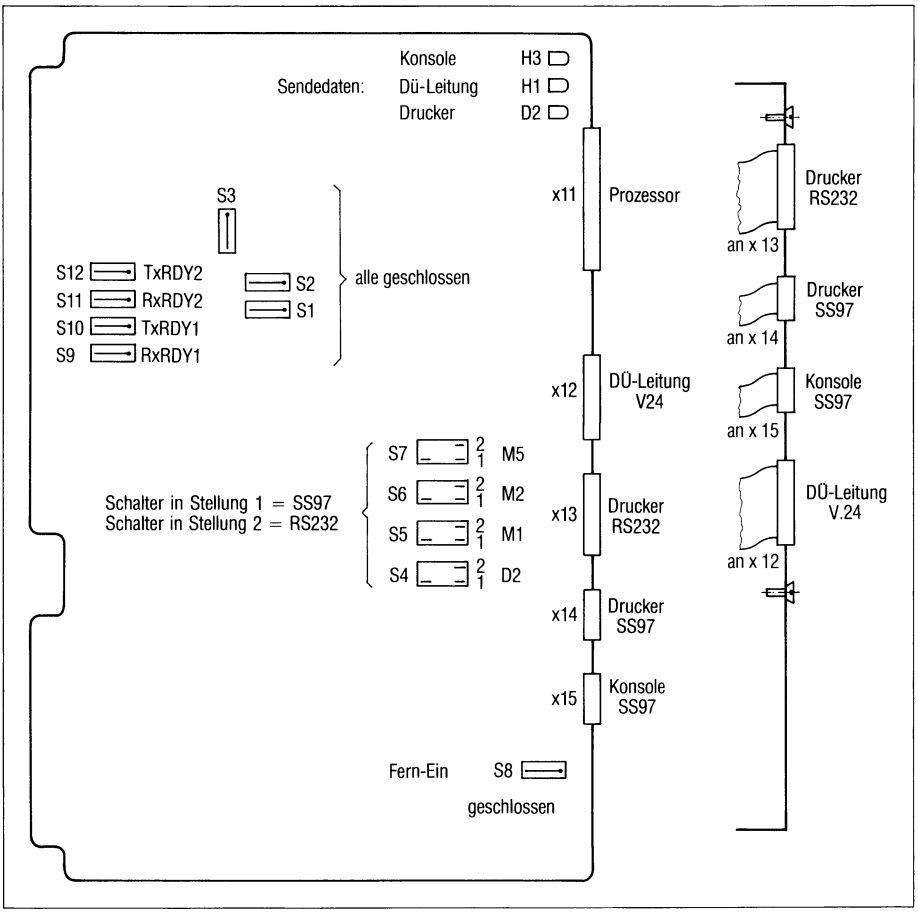


Bild 2-3 Flachbaugruppe CONAC D245-V1

2.1.4 Pin-Belegung der Schnittstelle (SS97/RS232)

2.1.4.1 Schnittstelle SS97

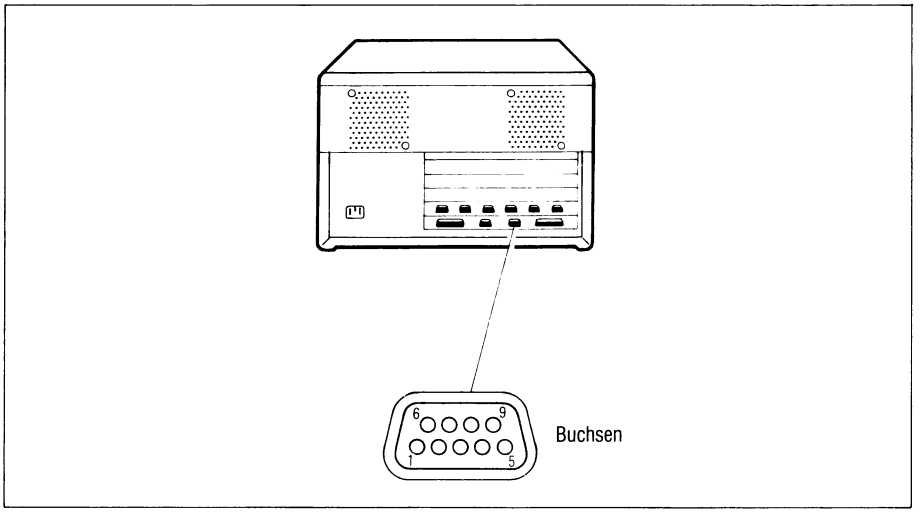


Bild 2-4 Pin-Belegung der Schnittstelle SS97

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)
5	0 V	Masseleitung
2	UH	Hilfsspannung +12 V (max. 30 mA)

2.1.4.2 Schnittstelle RS232

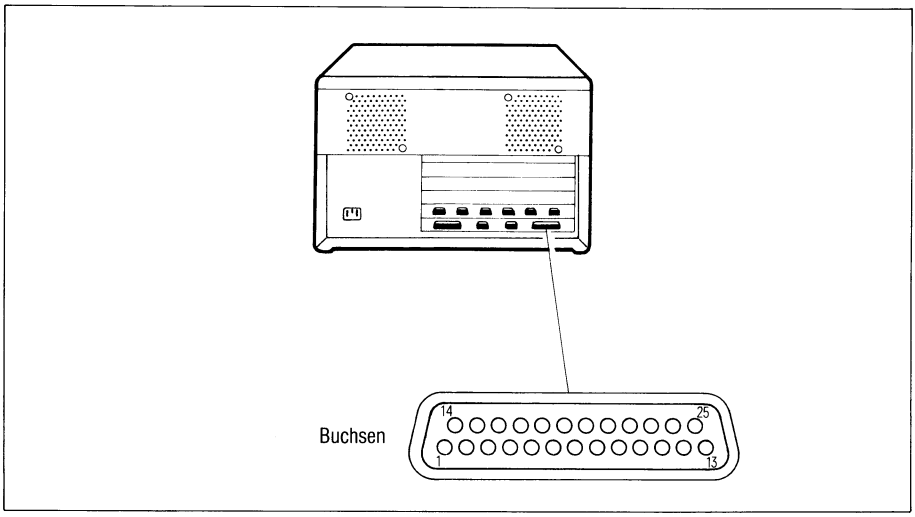


Bild 2-5 Pin-Belegung der Schnittstelle RS232

Stift	Bezeichnung DIN	Bezeichnung EIA	Erklärung
1	E1	PG	Schutzerde
2	D1	TCD	Sendedaten
3	D2	RCD	Empfangsdaten
4	S2	RTS	Sendeteil einschalten
5	M2	CTS	Sendebereitschaft
6	M1	DSR	Betriebsbereitschaft
7	E2	SG	Signalerde
8	M5	DCD	Empfangssignalpegel
20	S1	DTR	DEE betriebsbereit
24	T1	-	Sendeschrittakt, nicht versorgt

Signalabhängigkeiten

Wenn der PC-MX senden will, gibt er S1 und S2 aus.
Daten werden
— erst gesendet, wenn M2 aktiv ist und
— solange gesendet, wie das Peripheriegerät M2 aktiv hält.

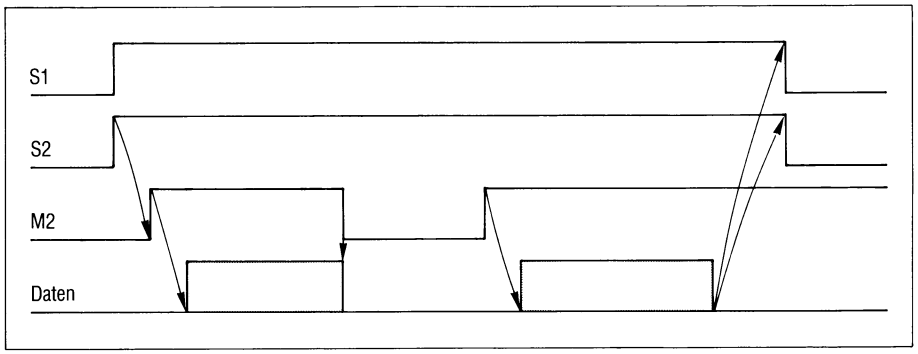


Bild 2-6 Signalabhängigkeiten

2.1.5 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      3,  1 Mar 12 17:49 lp9001-2-D1
```

Die Majornummer 3 bezeichnet die Fbg. CONAC (ohne DÜ-Anschluß).
Die Minornummer 1 legt den Ausgang und den Coderahmen fest.

1	= SS97/RS232	7 Bit + Parity ungerade ist fest vorgegeben
131	= SS97/RS232	Coderahmen frei wählbar (es ist Bit-Image-Printing möglich, wenn auf 8 Bit umgeschaltet wird) Minornummer 131 ist Bit-Image-Printing möglich)

Einrichten einer Gerätedatei

```
cd /dev
/etc/mknod lpfremd c 3 131
           |       | | |
           |       | | Minornummer
           |       | |
           |       | Majornummer
           |       |
           |       Character Device
           |
           frei gewählter Name
```

b) Einstellen der Schalter auf Fbg. CONAC

Annahme: Der Drucker liefert keine Schnittstellensignale M1, M2 und M5.

S4 in Stellung 2
S5, S6, S7 in Stellung 1

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 X'0A' (Line Feed) abgeschlossen ist, mit X'0D0A'(Carriage return und Line Feed)

ausgegeben.

Ist eine andere Ausgabegeschwindigkeit gefordert, so ist bei der Version SINIX 1.0B folgende Vorgehensweise erforderlich:

```
sleep 65000 >/dev/lpfremd&      Kanal öffnen
stty  1200 >/dev/lpfremd        Geschwindigkeit setzen
```

Der 'sleep' ist nötig, da der TTY-Treiber den Kanal bei jedem 'close' auf 'Standard' (9600 bit/s, nl, echo) setzen würde.
Dieses Kommando, im Hintergrund laufend, hält den Kanal jedoch für 65000 sek. = 18 h offen.
Diese Kommando-Sequenz kann man in die Datei /etc/rc einfügen. Damit wird bei jedem Systemstart der Kanal automatisch richtig eingestellt.

2.2 Ein-/Ausgabeprozessor

2.2.1 SERAC: 6 x SS97 (97802-201)

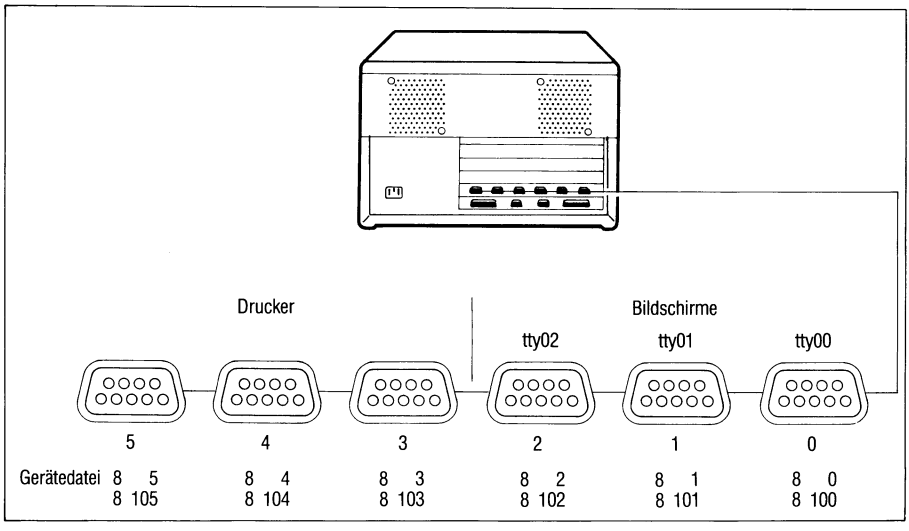


Bild 2-7 Anschlußfeld der SE mit E/A-Prozessor SERAC

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.

2.2.2 SERAD: 4 x SS97 und 2 x RS232 (97802-202)

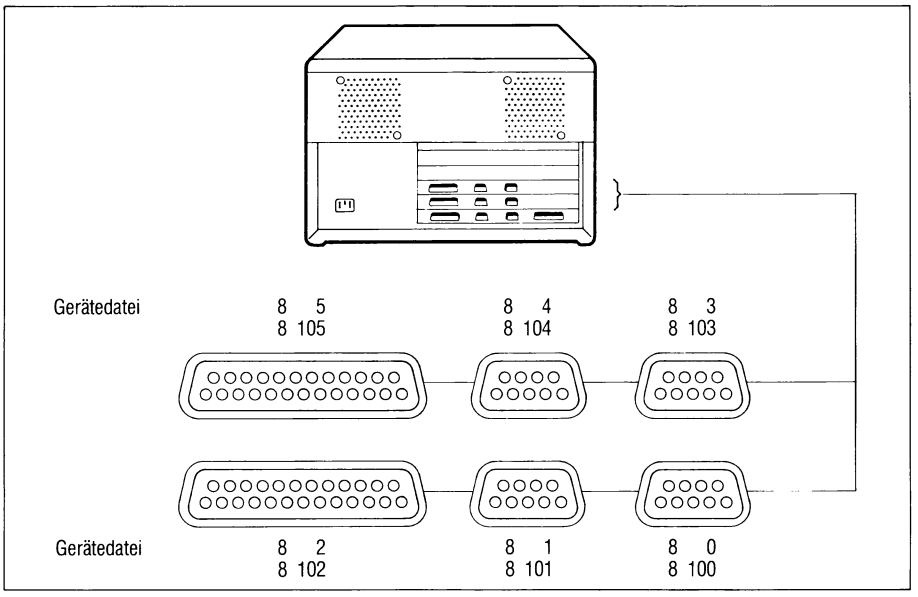


Bild 2-8 Anschlußfeld der SE mit E/A-Prozessor SERAD

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).

2.2.3 Mehrere Ein-/Ausgabeprozessoren

Von den derzeit freigegebenen SINIX-Betriebssystem-Versionen 1.0B und 1.0C wird der Betrieb mit mehr als einem E/A-Prozessor leider nicht unterstützt!

Der E/A-Prozessor SERAD mit 4 x SS97 und 2 x RS232 (97802-202) kann nur anstelle der bisherigen FBG. SERAC mit 6 x SS97 (97802-201) eingebaut werden.
Beide Baugruppen verhalten sich an der SW-Schnittstelle gleich.

Eine Umrüstung kann nur durch den Siemens-Servie erfolgen.

Hinweis

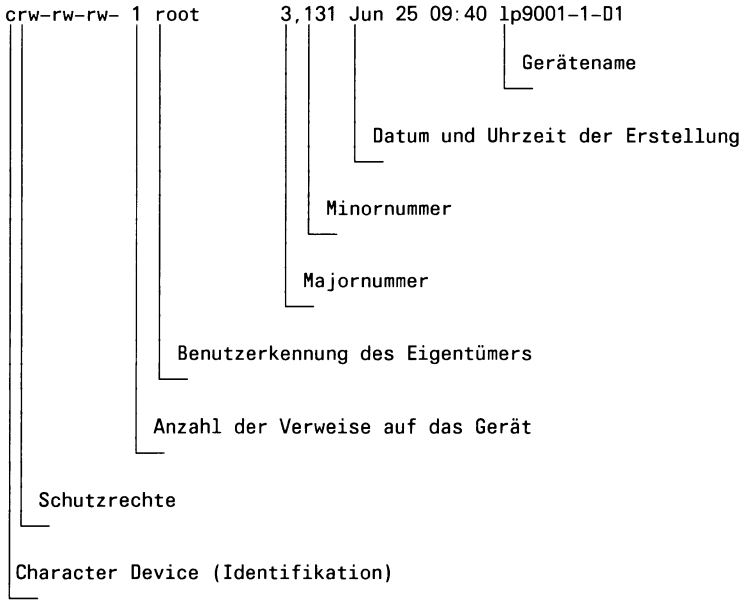
Sollte nach einer Betriebssystem-Modifikation der Betrieb des PC-MX mit zwei E/A-Prozessoren freigegeben werden, so gilt folgendes:

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 der Miltibus-Interrupt INT6 und die I/O-Adresse 1100 eingestellt werden.

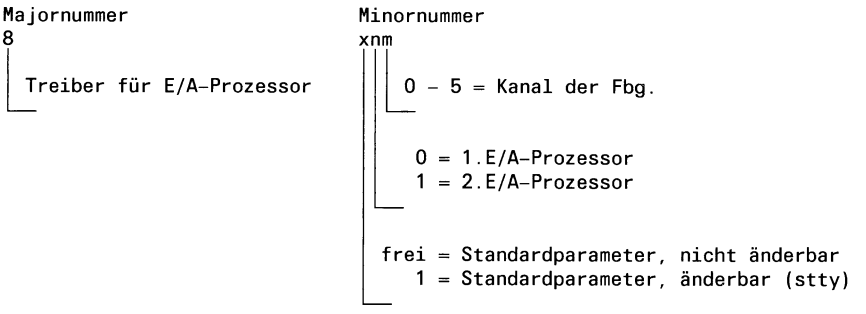
2.2.4 Allgemeines über Gerätedateien

Über die Gerätedateien werden die angeschlossenen Geräte direkt angesprochen (siehe auch SINIX Buch 1, Kapitel 5).
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 bei zwei E/A-Prozessoren



Standardparameter

- Bei 'Hunderter-Kanälen' 100, 101 und 102:

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

- Bei den übrigen '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        8,115 Jun 24 14:55 /dev/lpfremd
```

An Kanal 5 des 2.E/A-Prozessors ist ein Drucker angeschlossen, der mit 9600 bit/s betrieben wird.

2.3 Schnittstellenbeschreibung

2.3.1 Schnittstelle SS97

2.3.1.1 Pin-Belegung

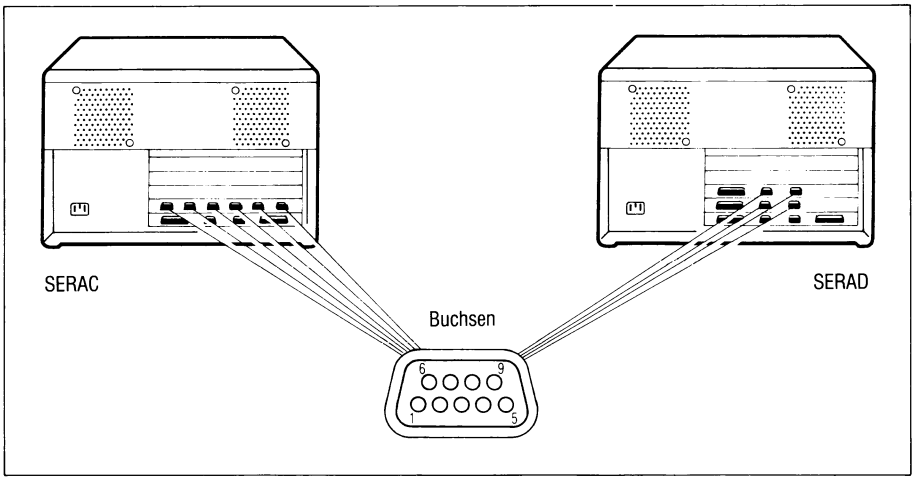


Bild 2-9 Pin-Belegung der Schnittstelle SS97

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)
5	0 V	Masseleitung
2	UH	Hilfsspannung +12 V (max. 30 mA)

2.3.1.2 Elektrische Kennwerte

Signalzustände

DIN-Leitung

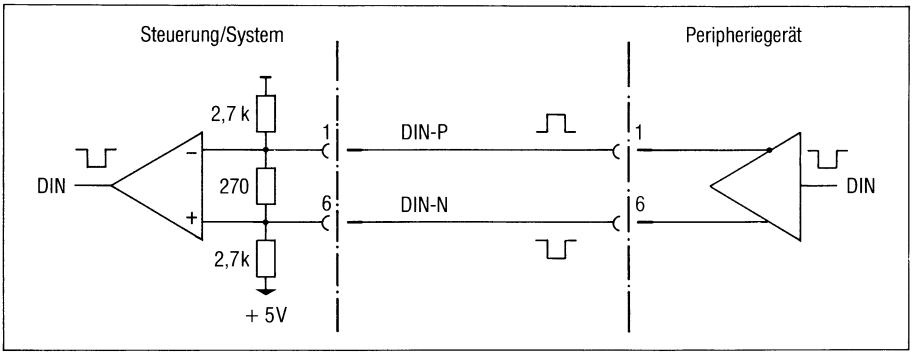


Bild 2-10 DIN-Leitung

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

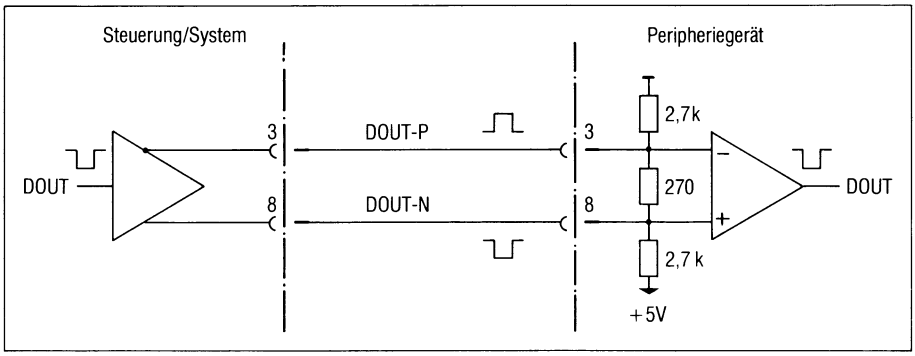


Bild 2-11 DOUT-Leitung

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

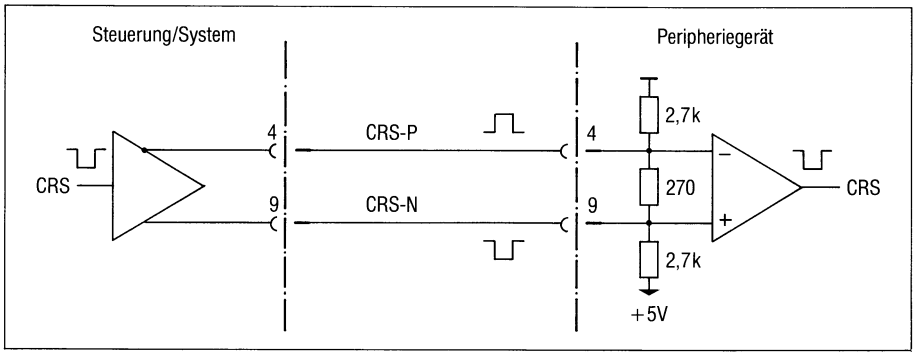


Bild 2-12 CRS-Leitung

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

a) Treibereingänge/Empfängerausgänge

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

b) Treiberausgänge/Empfängereingänge

H ≥ +3 V
L ≤ +2 V

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:

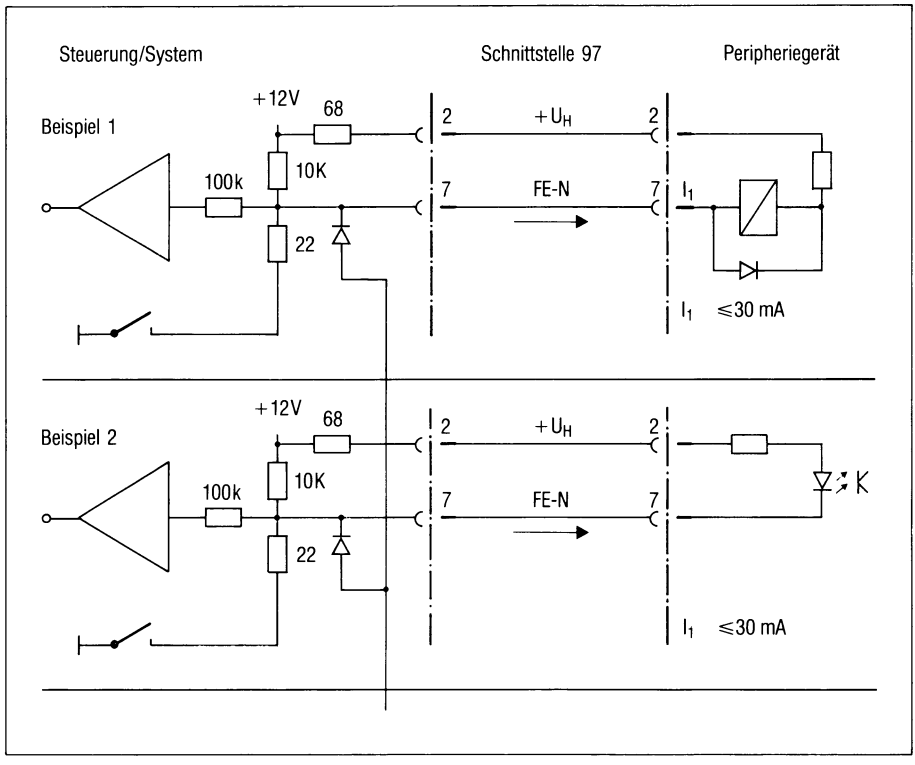


Bild 2-13 Schaltungsvorschläge für Fern-Ein

Steckerbelegung

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

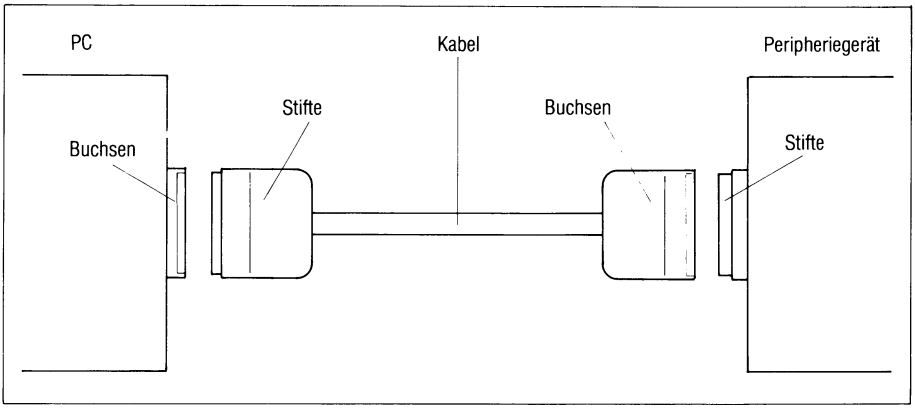


Bild 2-14 Steckerbelegung (9-poliger Stecker)

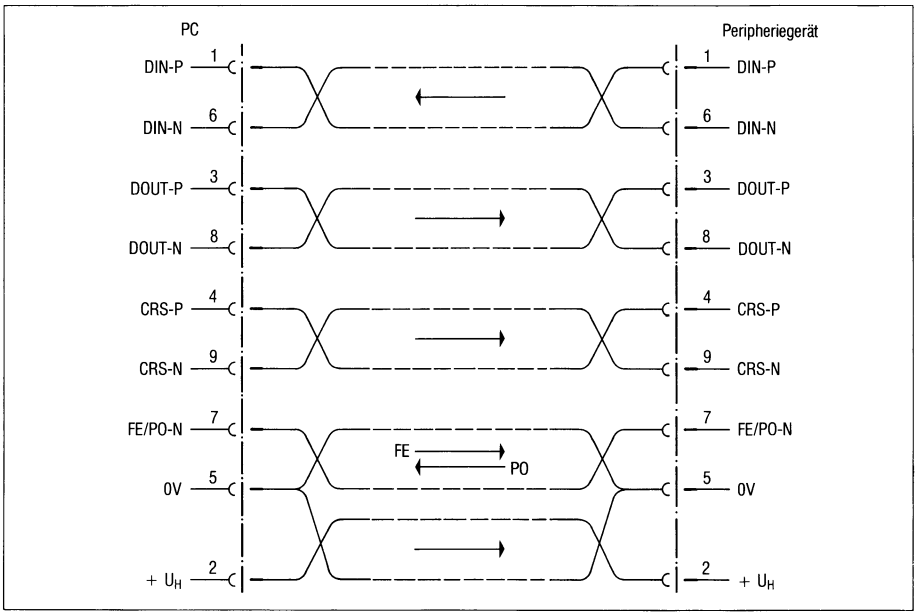


Bild 2-15 Zuordnung der Signale zum Stecker und logische Signalrichtung

Bemerkungen

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 2-16 vorgeschlagen, beschaltet werden. Der entsprechende Sender kann entfallen.

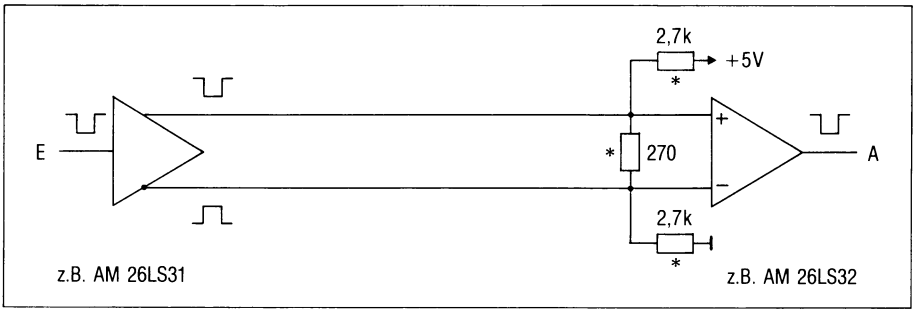


Bild 2-16 Beschaltung nicht benutzter Signalleitungen

* 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!

2.3.2 Schnittstelle RS232

2.3.2.1 Pin-Belegung

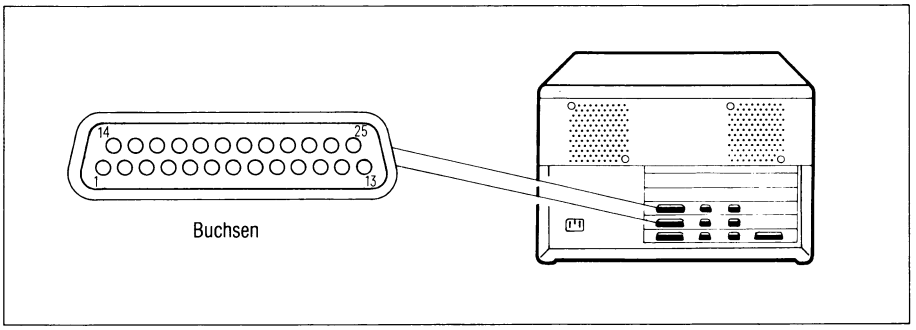


Bild 2-17 Pin-Belegung der Schnittstelle RS232

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

2.3.2.2 Elektrische Kennwerte

Schnittstellen-Ersatzschaltbild

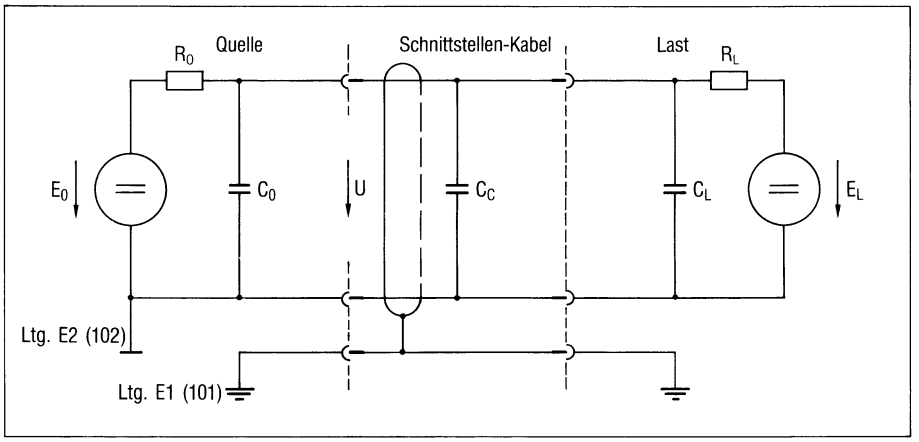


Bild 2-18 Schnittstellen-Ersatzschaltbild

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_O ≙ 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

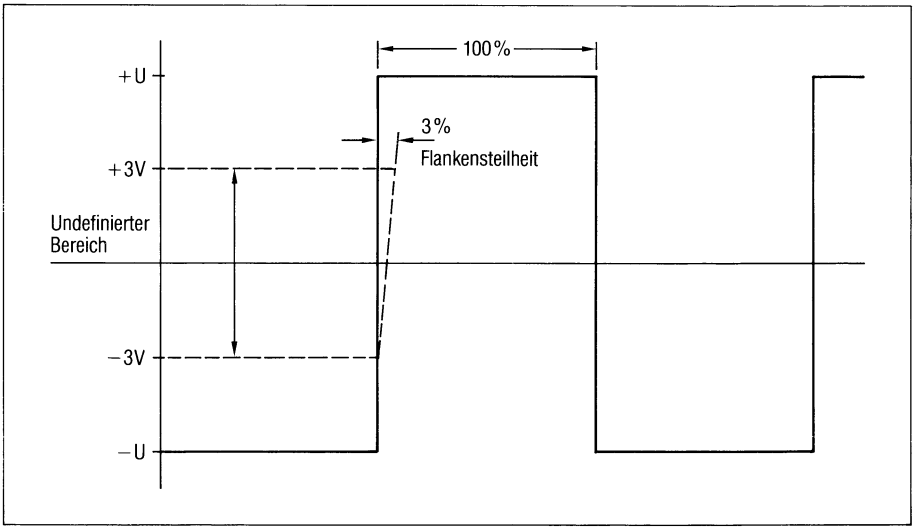


Bild 2-19 Impulsdiagramm

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

Beispiel:

Übertragungsgeschwindigkeit = 4800 bit/s

Schrittdauer (100%) = 208,3 µs

Abweichung (3%) = 6,249 µs

2.3.2.3 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.

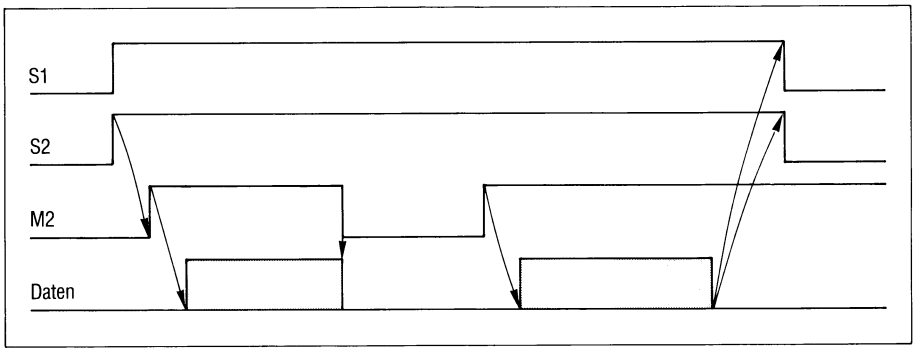


Bild 2-20 Signalabhängigkeiten

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

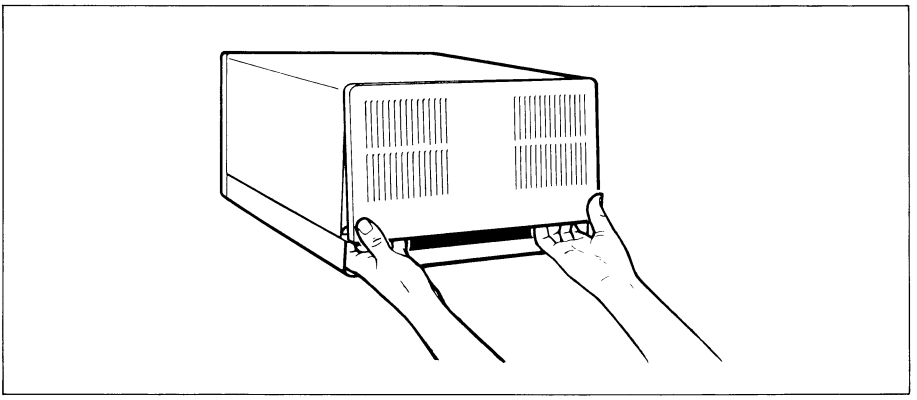
Die max. Kabellänge kann 30 m betragen. Die über diese geschirmte Schnittstelle miteinander verbundenen Peripheriegeräte müssen am selben Netzverteiler (220 V) angeschlossen sein, um Ausgleichsströme zu verhindern.

2.4 Konstruktiver Aufbau

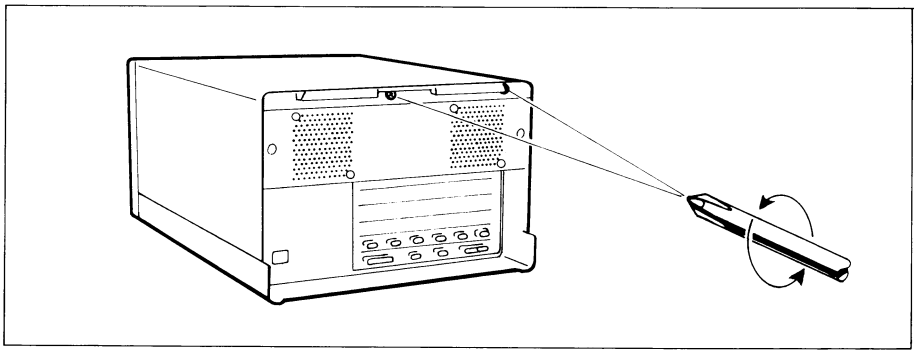
2.4.1 Öffnen des Gehäuses beim PC-MX

Abdeckung abnehmen

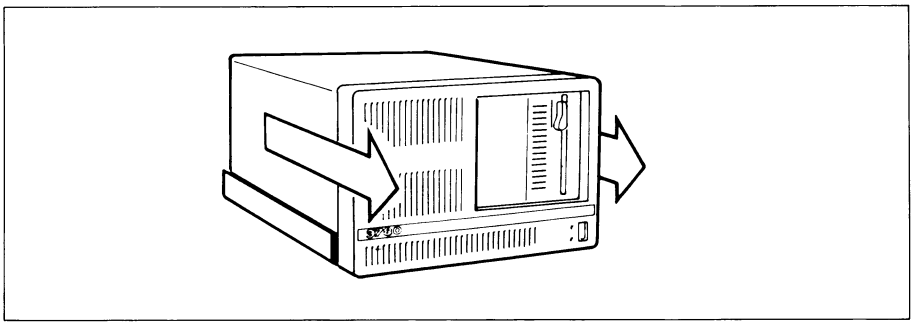
- Netzstecker ziehen
- Federbügel nach oben biegen
- Abdeckung nach oben wegklappen

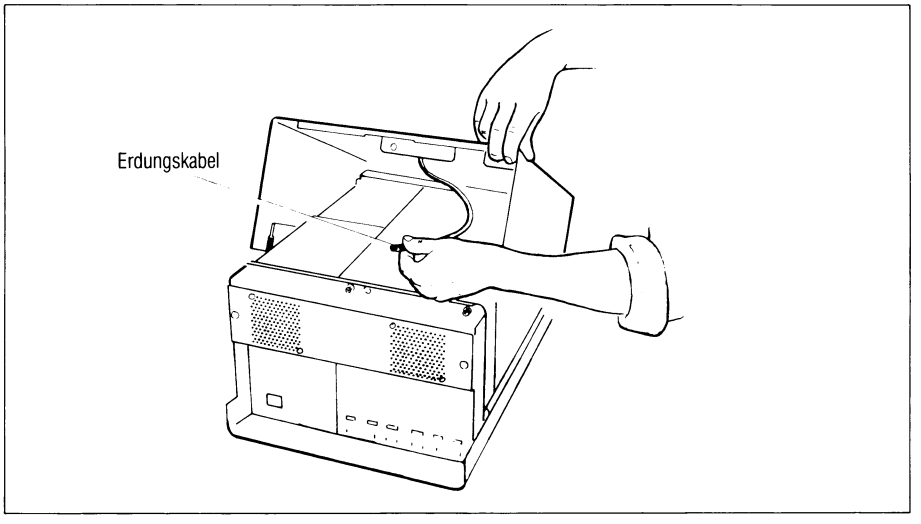


- 2 Schnellverschlußschrauben öffnen



- Gehäuse nach vorne schieben
- Gehäuse hinten anheben
- Erdungskabel abziehen
- Gehäuse abnehmen





2.4.2 Flachbaugruppenbelegung des PC-MX

Achtung

Wenn Sie an den Baugruppen Veränderungen vornehmen oder Baugruppen einsetzen, die das Betriebsverhalten des PC's verändern, erlischt die Gewährleistung.

Auf allen Baugruppen sind elektrostatisch gefährdete Bauelemente (EGB). Für den Umgang mit diesen Baugruppen gelten die EGB-Vorschriften:

- Ziehen der Baugruppen nur nach vorheriger Entladung am Gehäuse.
- Ablage nur auf leitfähiger, geerdeter Arbeitsunterlage.
  Achtung: Vorsicht bei Fbg. mit Akku, z.B. CONAC.
- Transport nur in der hierfür vorgesehenen Verpackung.
- Berühren der Bauteile oder Leiterbahnen nur nach vorheriger Erdung über das Handgelenk (also nicht berühren).

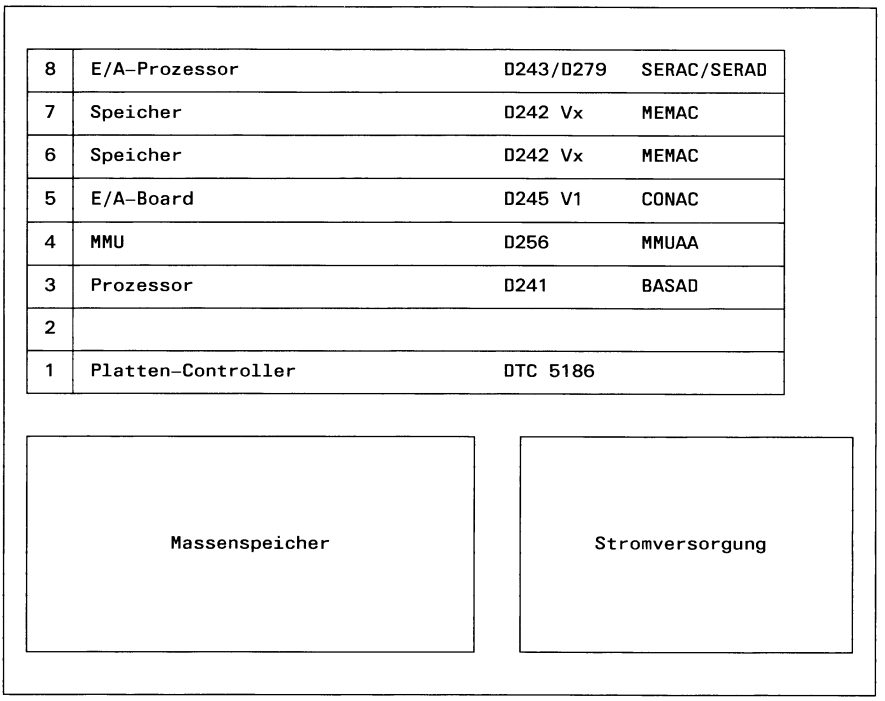


Bild 2-21 Aufbau des PC-MX mit Standardbelegung (Draufsicht)

3 Weitere Schnittstellen des PC-X

3.1 Schnittstellen im Grundausbau des PC-X

3.1.1 Allgemeines

Der PC-X 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.

3.1.2 Übersicht

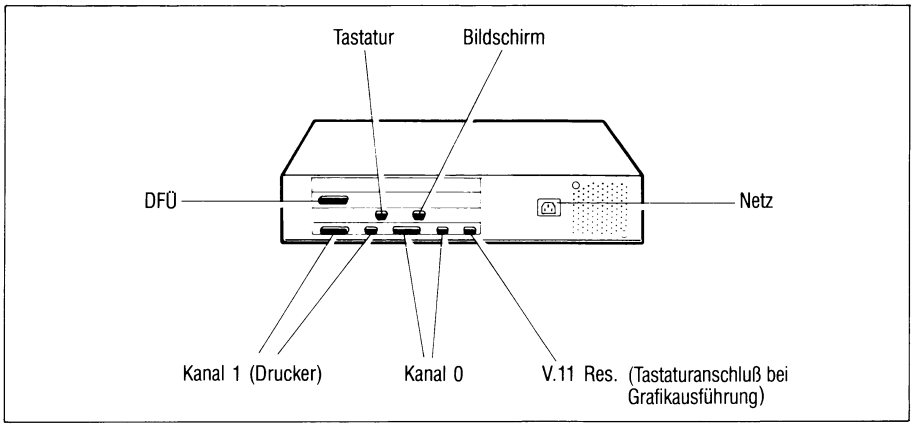


Bild 3-1 Anschlußfeld der Systemeinheit (SE)

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

- dem Systemboard,
- der Stromversorgung,
- einem Diskettenlaufwerk,
- einer Festplatte mit Controller,
- dem Frontpanel,
- den Einbauplätzen für Erweiterungsboards.

Am Systemboard sind folgende wichtigen 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 Erweiterungsboards	intern/ parallel	nein	96-poliger Erweiterungsstecker
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

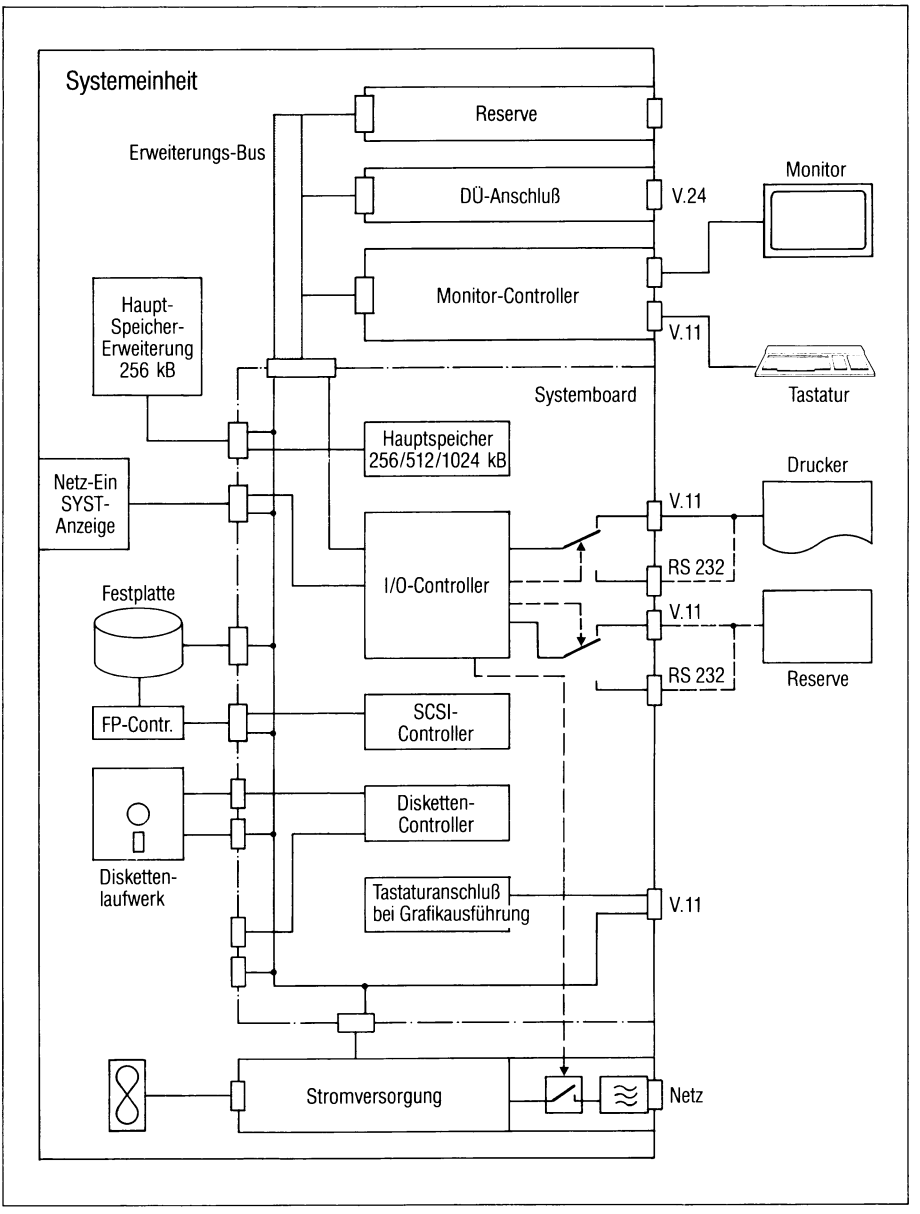


Bild 3-2 Blockschaltbild der Systemeinheit

3.1.3 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.

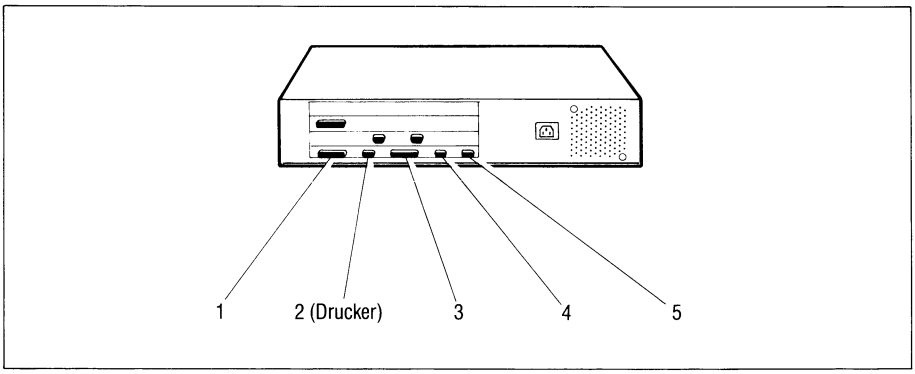


Bild 3-3 Externe Schnittstellen

3.1.4 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.

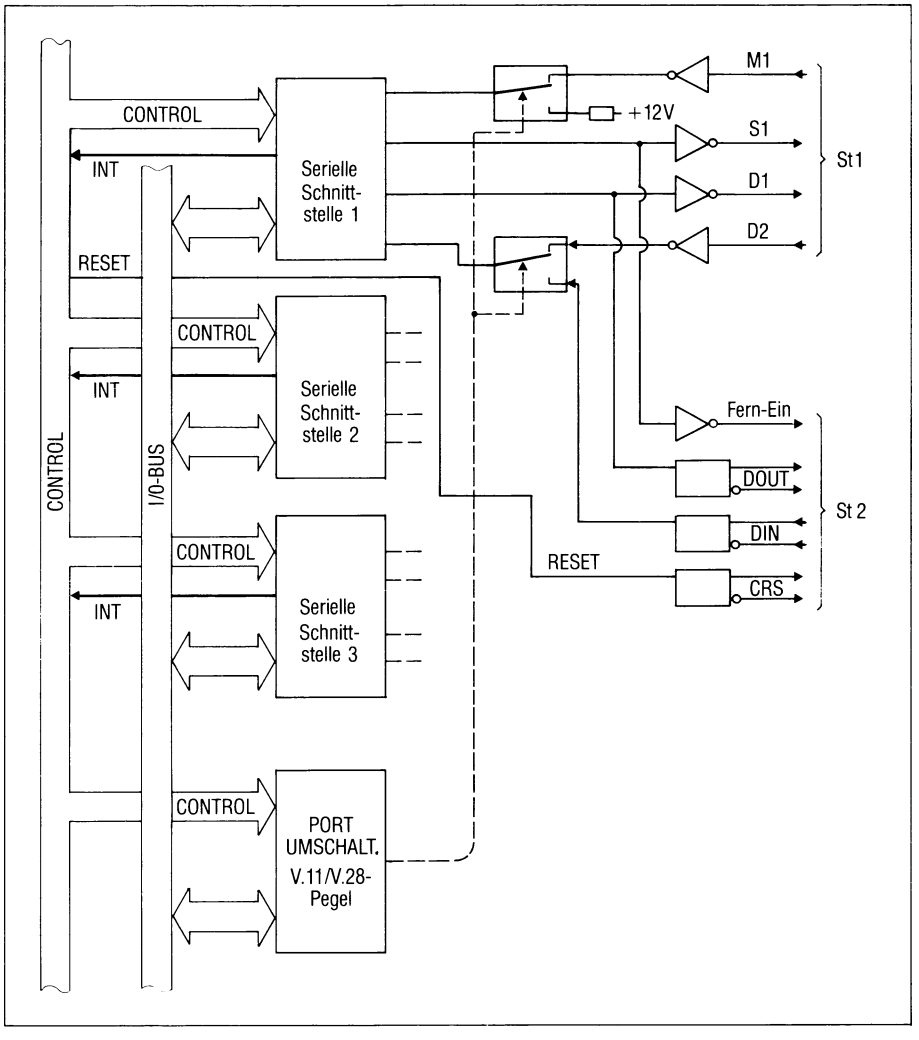


Bild 3-4 Umschaltung SS97/RS232

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.

Diese Datenflußsteuerung wird von SINIX V1.0 in beiden Richtungen (Eingabe und Ausgabe) durchgeführt.

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)
```

3.1.5 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        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 X'0A'(Line Feed) abgeschlossen ist, mit X'0D0A' (Carriage return und Line Feed)

ausgegeben.

Ist eine andere Ausgabegeschwindigkeit gefordert, so ist bei der Version SINIX 1.0 mit dem Kommando stty die Einstellung der gewünschten Übertragungsgeschwindigkeit möglich.

3.2 Pin-Belegung der Stecker am Anschlußfeld des Systemboards

3.2.1 Schnittstelle SS97 (Stecker 2 und 4)

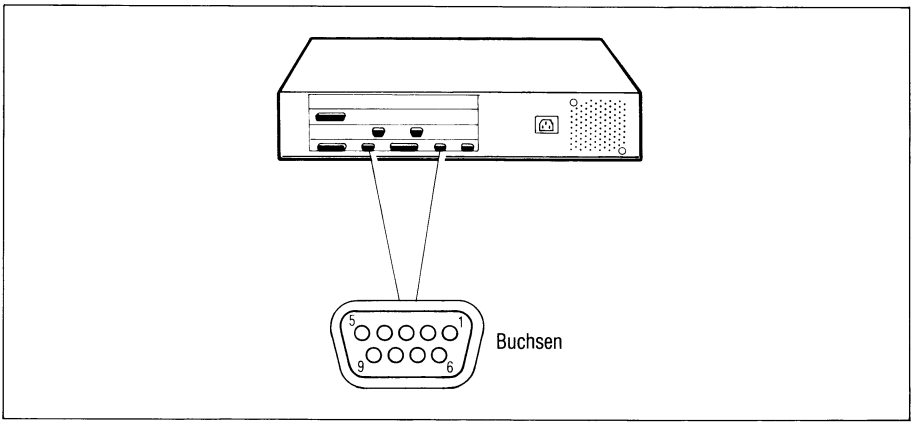


Bild 3-5 Schnittstelle SS97

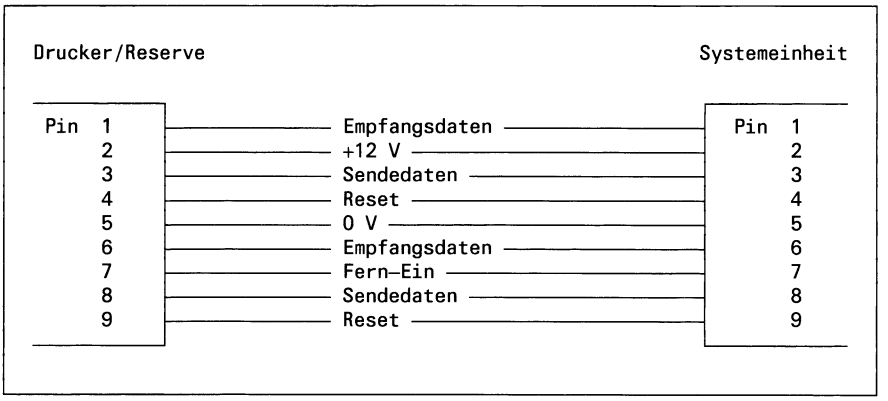


Bild 3-6 Pin-Belegung der Schnittstelle SS97 am PC-X

3.2.2 Schnittstelle RS232 (Stecker 1 und 3)

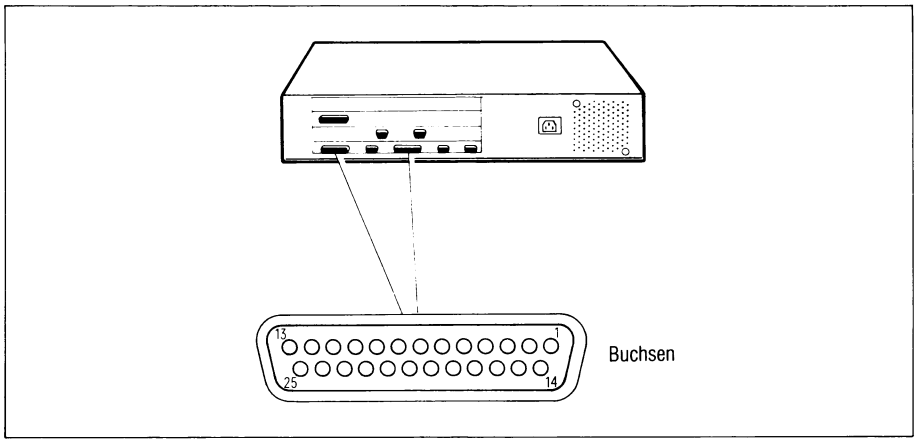


Bild 3-7 Schnittstelle RS232

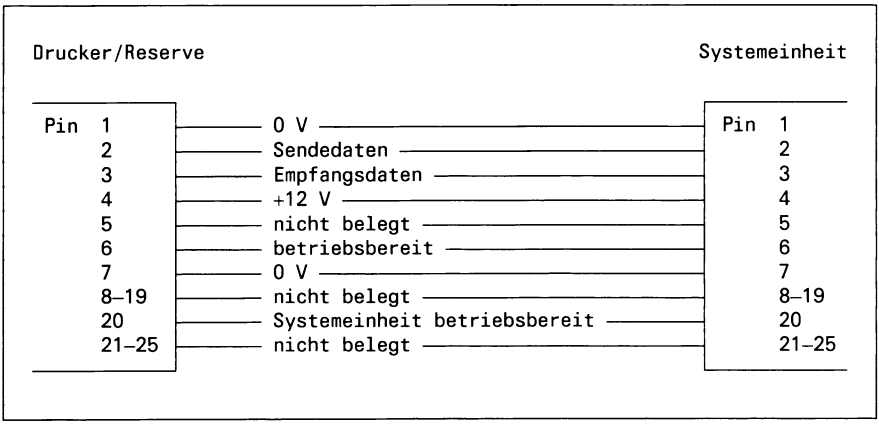


Bild 3-8 Pin-Belegung der Schnittstelle RS232 am PC-X

Stift	Bezeichnung nach EIA	DIN	CCITT	Erklärung
1	PG	E1	101	Schutzerde
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
7	SG	E2	102	Signalerde
8	DCD	M5	109	Empfangssignalpegel
20	DTR	S1.2	108.2	DEE betriebsbereit
24	-	T1	113	Sendeschrittakt

3.2.3 Schnittstelle V.11 (Tastaturanschluß, Stecker 5)

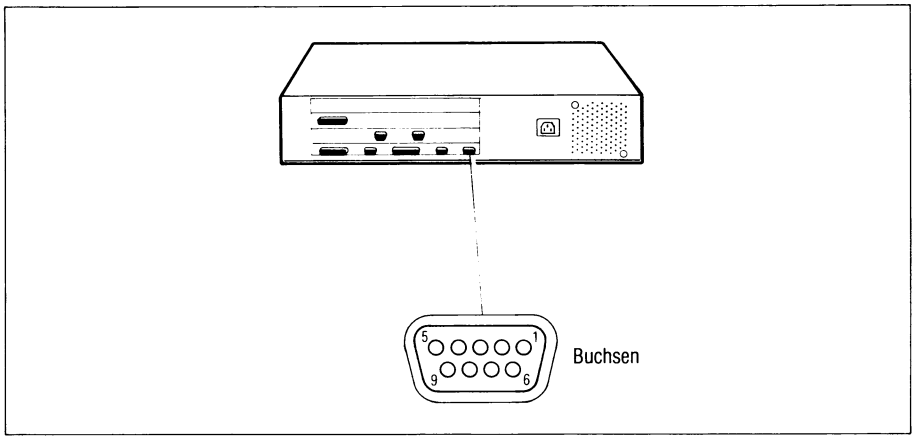


Bild 3-9 Schnittstelle V.11

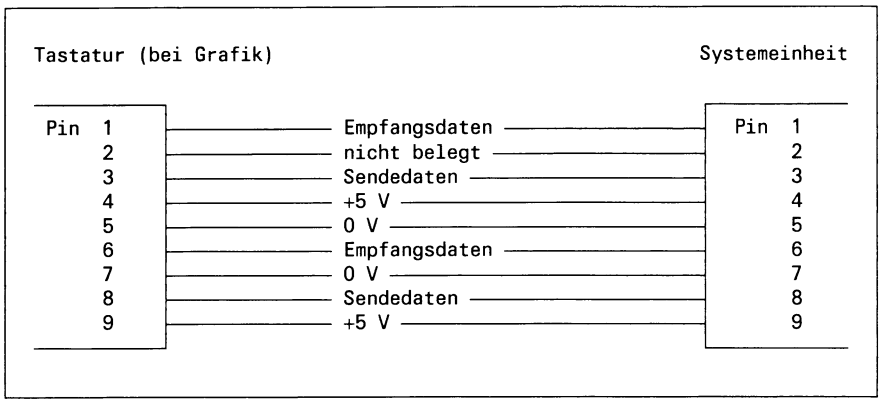


Bild 3-10 Pin-Belegung der Schnittstelle V.11 am PC-X

3.3 Elektrische Kennwerte der Schnittstellen

3.3.1 Schnittstelle SS97

3.3.1.1 Signalzustände

DIN-Leitung

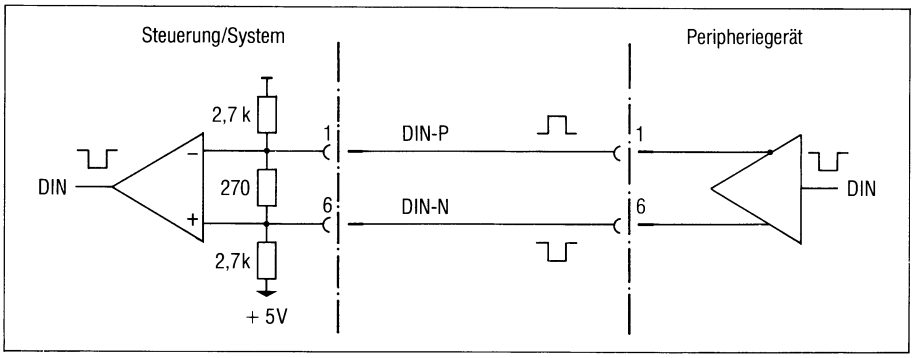


Bild 3-11 DIN-Leitung

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

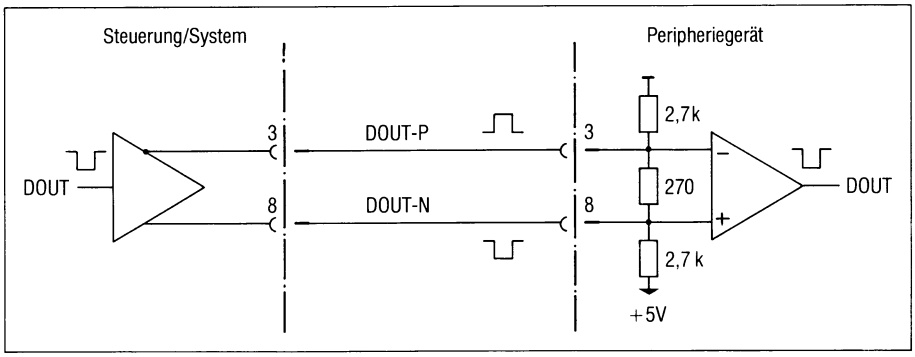


Bild 3-12 DOUT-Leitung

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

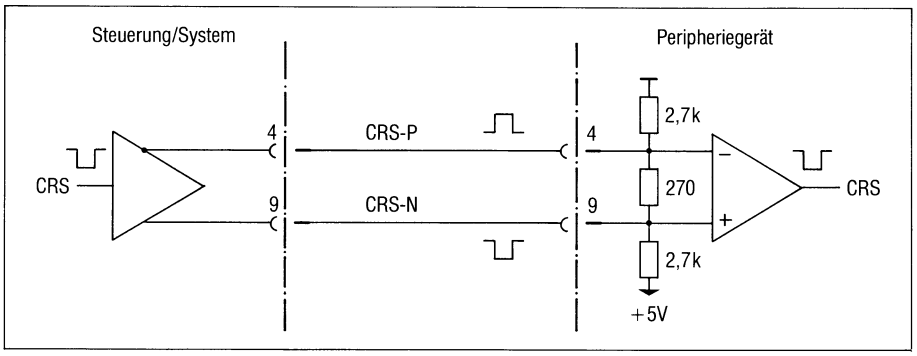


Bild 3-13 CRS-Leitung

CRS	CRS-P	CRS-N
H	L	H ≙ Ruhezustand
L	H	L ≙ Arbeitszustand (t CRS > 70 ms)

a) Treibereingänge/Empfängerausgänge

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

b) Treiberausgänge/Empfängereingänge

H ≥ +3 V
L ≤ +2 V

3.3.1.2 Beschaltung der Leitung FE/PO und UH

Kann eine Komponente nur manuell über ihren 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:

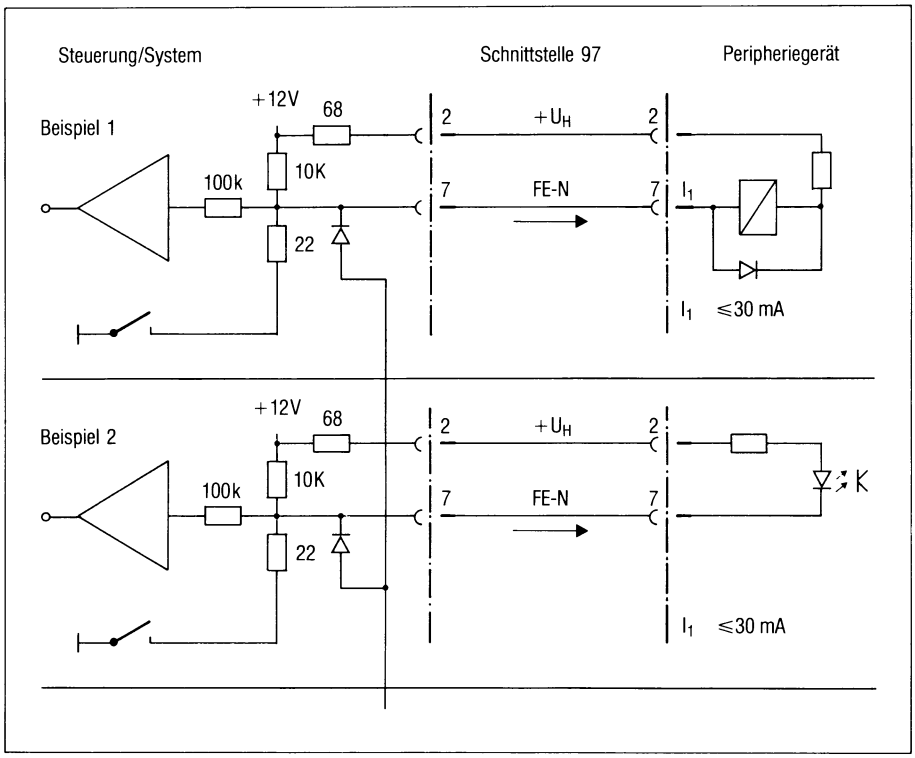


Bild 3-14 Schaltungsvorschläge für Fern-Ein

3.3.1.3 Steckerbelegung

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

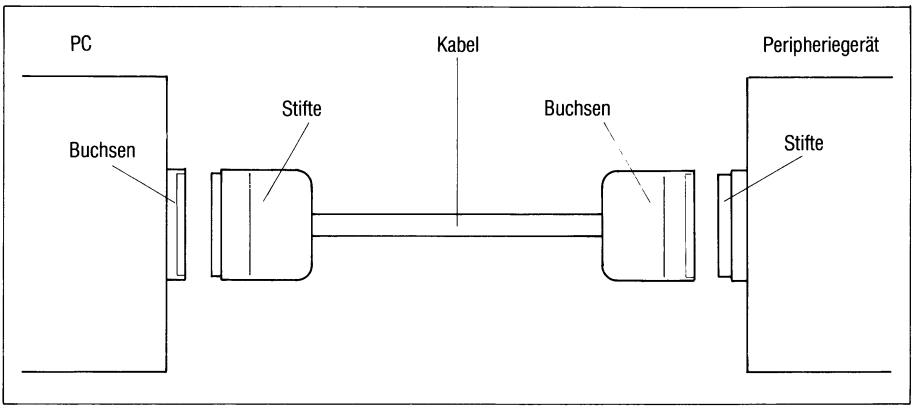


Bild 3-15 Steckerbelegung (9-poliger Stecker)

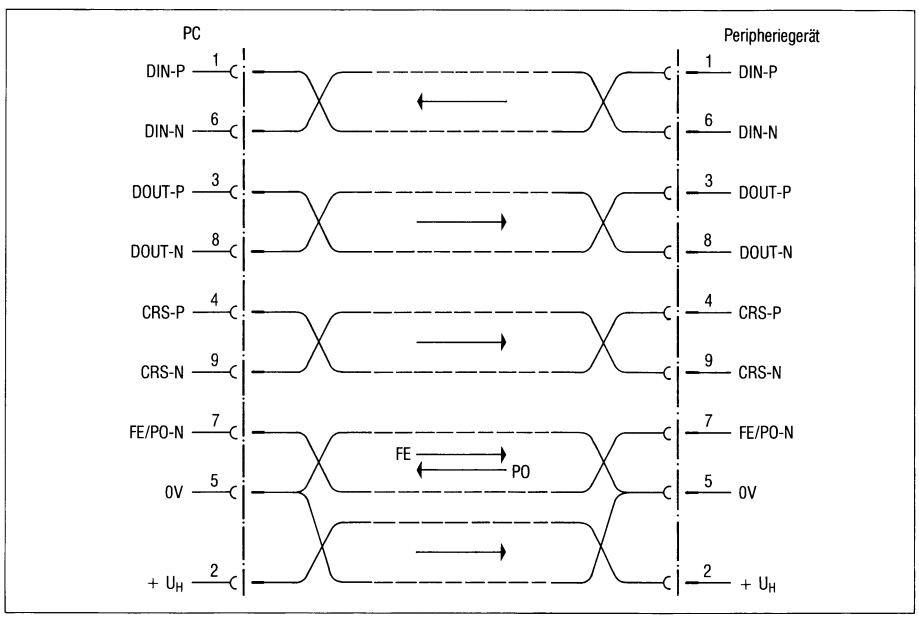


Bild 3-16 Zuordnung der Signale zum Stecker und logische Signalrichtung

3.3.1.4 Bemerkungen

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 2-16 vorgeschlagen, beschaltet werden. Der entsprechende Sender kann entfallen.

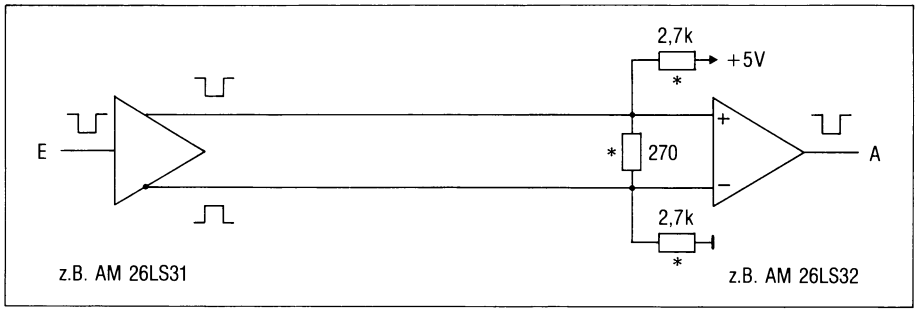


Bild 3-17 Beschaltung nicht benutzter Signalleitungen

* 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!

3.3.2 Schnittstelle RS232C

3.3.2.1 Elektrische Kennwerte

Schnittstellen-Ersatzschaltbild

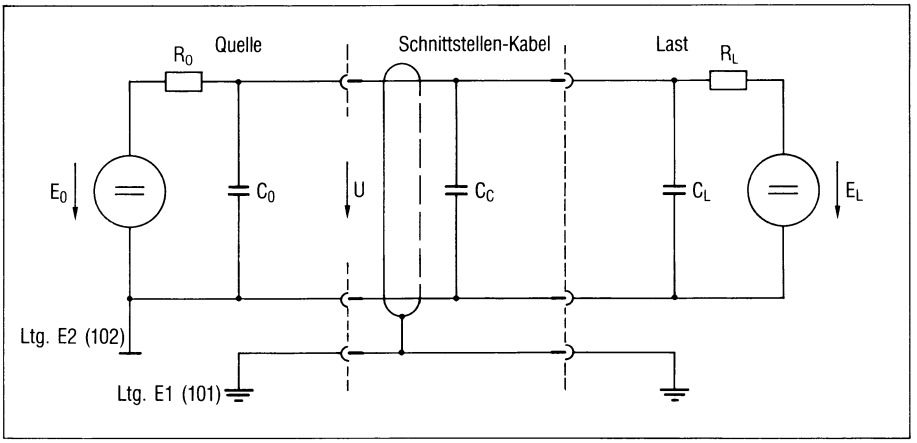


Bild 3-18 Schnittstellen-Ersatzschaltbild

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

E_0 ≙ Leerlaufspannung der Quelle
$+25\ V \geq E_0 \leq -25\ V$

R_0 ≙ Quellenwiderstand
Bei Ausfall der Stromversorgung $R_0 \geq 300$ Ohm.

C_O ≙ Quellenkapazität, nicht definiert.

C_C ≙ Kapazität des Schnittstellenkabels
≤ 2000 pF

C_L ≙ Lastkapazität
≤ 2500 pF

R_L = Gleichstromwiderstand
$3000\ \text{Ohm} \leq R_L \leq 7000\ \text{Ohm}$

E_L ≙ Leerlaufspannung der Last
$+24\ V \geq E_L \leq -24\ V$

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 sbergangsbereich gekennzeichnet. Der Zustand des Signals oder der Leitung kann nicht eindeutig definiert werden, wenn die Spannung sich in diesem Überganfsbereich 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

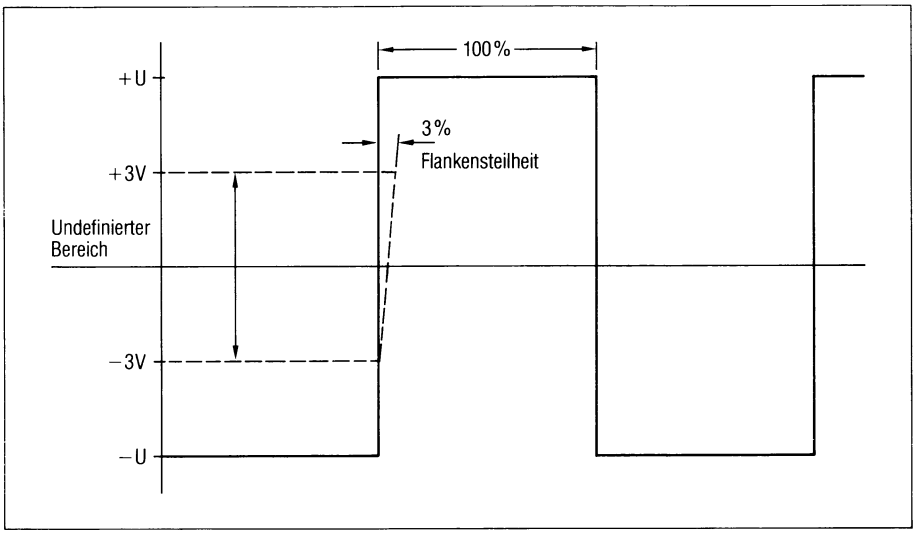


Bild 3-19 Impulsdiagramm

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

Beispiel:

Übertragungsgeschwindigkeit = 4800 bit/s

Schrittdauer (100%) = 208,3 µs

Abweichung (3%) = 6,249 µs

3.3.2.2 Bemerkungen

Normen

Die Norm RS232C 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.

4 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

4.1 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 Inhaltsverzeichnis </dev> (ls -1 /dev, gekürzt):

```
crw--w--w- 1 dptr      3,  0 Sep 29 17:45 console          ┐
crw--w--w- 1 solf      8,  0 Sep 29 17:46 tty00            │ Control-
crw--w--w- 1 grw       8,  1 Sep 29 17:47 tty01            │ Terminals
crw--w--w- 1 roat      8,  2 Sep 29 17:48 tty02            ┘

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

crw-rw-rw- 1 root      3,  1 Sep  5 14:45 lp9001-1-D1    Drucker 9001
crw-rw-rw- 1 gast      8,105 Sep 29 17:45 plotter        Plotter
crw-rw-rw- 1 root      2,  2 Sep 29 15:42 null           Biteimer

brw-rw-rw- 1 root      0,  0 Sep 18 12:11 fl0
brw-rw-rw- 1 root      0,  1 Jul 30 14:01 fl1
brw-rw-rw- 2 root      0,  2 Sep 23 14:01 fl2
                                                  Datei-
                                                    name
                                          Datum und Uhrzeit
                                            der Erstellung
                                   Minornummer  2)
                             Majornummer  1)
                    Benutzerkennung des Eigentümers
                 Anzahl der Verweise
           Datei-Attribute
     Character-/Block-Device
```

1) Die Majornummer definiert den Flachbaugruppen-Typ:

   3 CONAC / PC-X-Systemboard
   8 SERAC / SERAD

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

CONAC (PC-MX):

0 Konsol-Schnittstelle, siehe /etc/ttytype

1 Schnittstelle RS232 bzw. V.11 (Umschalten durch Schalter)
- Parameter teilweise fest vorgegeben. 3)

131 Schnittstelle RS232 bzw. V.11 (Umschalten durch Schalter)
- alle Parameter einstellbar

PC-X-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)
```

SERAC:

Die Minornummer ist dezimal-3-stellig:

SERAD:

```
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
```

*) *Achtung:* Das Kommando <stty> täuscht in diesen Fällen eine versuchte Umparametrisierung vor!

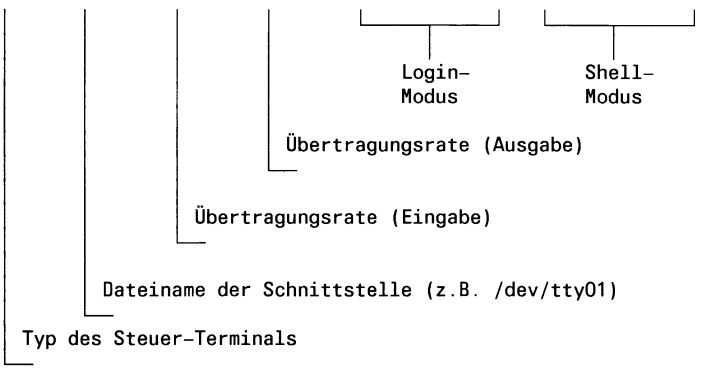
4.2 Die Anpassung der TTY-Schnittstelle

4.2.1 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   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

4.2.2 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/serase = '^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/lp9001-1-D1 (Anzeige Druckerparameter)

```
Übertragungsgeschwindigkeit 9600 bit/serase = '^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
```

4.2.3 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.
Weiters werden aus Kompatibilitätsgründen 2 Möglichkeiten des <ioctl>-Systemaufrufes zur Verfügung gestellt, die sich durch unter-

schiedliche Header-Dateien mit unterschiedlichen Strukturen unterscheiden.
Die 'alte' Schnittstelle entspricht XENIX-V7.
Die 'neue' Schnittstelle entspricht UNIX-System-III.

'Offiziell' ist in den derzeit ausgelieferten SINIX-Versionen (1.0C) nur die TTY-Schnittstelle entsprechend XENIX-V7 mit der entsprechenden Header-Datei <sgtty.h> enthalten.

Es wird jedoch auch schon 'inoffiziell' die weiterentwickelte Schnittstelle S-III unterstützt, allerdings fehlt die entsprechende Header-Datei:
< termio.h>.

Ein Beispiel für diese Header-Datei finden Sie in der Beschreibung und auf der mitgelieferten Diskette, jedoch ohne Anspruch auf Vollständigkeit und ohne jede Gewährleistung.

4.3 TTY-Schnittstelle entsprechend XENIX-V7

Die zu verwendende Header-Datei <sgtty.h> sieht folgendermaßen aus:

```
(cat /usr/include/sgtty.h)

/*
 * Structure for stty and gtty system calls.
 */

struct sgttyb {
            char    sg_ispeed;          /* input speed */
            char    sg_ospeed;          /* output speed */
            char    sg_erase;           /* erase character */
            char    sg_kill;            /* kill character */
            int     sg_flags;           /* mode flags */
            };

/*
 * List of special characters
 */
struct tchars {
           char    t_intrc;     /* interrupt */
           char    t_quitc;     /* quit */
           char    t_startc;    /* start output */
           char    t_stopc;     /* stop output */
           char    t_eofc;      /* end-of-file */
           char    t_brkc;      /* input delimiter (like nl) */
           };
/*
 * Modes
 */

#define TANDEM  01
#define CBREAK  02
#define LCASE   04
#define ECHO    010
#define CRMOD   020
#define RAW     040
#define ODDP    0100
#define EVENP   0200
#define ANYP    0300
#define NLDELAY 001400
#define TBDELAY 006000
#define XTABS   06000
#define CRDELAY 030000
#define VTDELAY 040000
#define BSDELAY 0100000
#define ALLDELAY 0177400
```

```
/*
 * Delay algorithms
 */

#define CR0     0
#define CR1     010000
#define CR2     020000
#define CR3     030000
#define NL0     0
#define NL1     000400
#define NL2     001000
#define NL3     001400
#define TAB0    0
#define TAB1    002000
#define TAB2    004000
#define FF0     0
#define FF1     040000
#define BS0     0
#define BS1     0100000

/*
 * Speeds
 */

#define B0      0     (nicht implementiert)
#define B50     1
#define B75     2
#define B110    3
#define B134    4
#define B150    5
#define B200    6     (nicht implementiert)
#define B300    7
#define B600    8
#define B1200   9
#define B1800   10
#define B2400   11
#define B4800   12
#define B9600   13
#define EXTA    14
#define EXTB    15
#define B19200  EXTA
#define B38400  EXTB
```

```
/*
 * tty ioctl commands
 */

#define TIOCGETD        (('t'<<8)|0)
#define TIOCSETD        (('t'<<8)|1)
#define TIOCHPCL        (('t'<<8)|2)
#define TIOCMODG        (('t'<<8)|3)
#define TIOCMODS        (('t'<<8)|4)
#define TIOCGETP        (('t'<<8)|8)
#define TIOCSETP        (('t'<<8)|9)
#define TIOCSETN        (('t'<<8)|10)
#define TIOCEXCL        (('t'<<8)|13)
#define TIOCNXCL        (('t'<<8)|14)
#define TIOCFLUSH       (('t'<<8)|16)
#define TIOCSETC        (('t'<<8)|17)
#define TIOCGETC        (('t'<<8)|18)
#define DIOCLSTN        (('d'<<8)|1)
#define DIOCNTRL        (('d'<<8)|2)
#define DIOCMPX         (('d'<<8)|3)
#define DIOCNMPX        (('d'<<8)|4)
#define DIOCSCALL       (('d'<<8)|5)
#define DIOCRCALL       (('d'<<8)|6)
#define DIOCPGRP        (('d'<<8)|7)
#define DIOCGETP        (('d'<<8)|8)
#define DIOCSETP        (('d'<<8)|9)
#define DIOCLOSE        (('d'<<8)|10)
#define DIOCTIME        (('d'<<8)|11)
#define DIOCRESET       (('d'<<8)|12)
#define FIOCLEX         (('f'<<8)|1)
#define FIONCLEX        (('f'<<8)|2)
#define MXLSTN          (('x'<<8)|1)
#define MXNBLK          (('x'<<8)|2)
```

4.3.1 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.

4.3.2 Allgemeines

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>.
Es stehen die File-Pointer <stdin> für die Eingabe und <stdout> und <stderr> für die Ausgabe zur Verfügung.
Für die File-Pointer <stdin> und <stdout> stehen spezielle, vereinfachte Aufrufe zur Verfügung:

```
getchar ();   entspricht  getc (stdin);
putchar (c);  entspricht  putc (c, stdout);
```

Auch die Aufrufe <printf> und <scanf> arbeiten mit <stdout> bzw. <stdin>. Für Low-Level-Ein-/Ausgabe-Operationen sind dies die File-Descriptoren. 0, 1, 2.

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.
Weiters 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 Hängezustand 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). Weiters 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.
Weiters 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 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 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_erase;     /* 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.

4.3.3 Schnittstellen-Parameter

Das Setzen der entsprechenden Schnittstellen-Parameter erfolgt durch entsprechende <#define-Symbole>, die ebenfalls 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. Weiters werden folgende Backslash-Folgen akzeptiert bzw. ausgegeben

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

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 !

4.3.3.1 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.

4.3.3.2 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.

4.3.3.3 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.
Weiters 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. EingabeWarteschlange 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 letztemal 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 &= ~RAW;                    /* 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 */
}
```

4.4 TTY-Schnittstelle entsprechend UNIX-System-III

Es wird nochmals darauf hingewiesen, daß diese Schnittstelle nicht offiziell freigegeben und dokumentiert ist.
Die folgende Beschreibung bezieht sich nur auf die als Beispiel mitgelieferte Datei <termio.h>.
Wenn damit gearbeitet werden soll, ist diese in das Inhaltsverzeichnis </usr/include> einzulesen.
Es sind jedoch die im Folgenden beschriebenen Funktionen in der derzeit ausgelieferten SINIX-Version 1.0C implementiert und wurden zum Teil getestet.

Die zu verwendende Header-Datei <termio.h> sieht folgendermaßen aus:

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

/*
 *----------------------------------------------------------------------------
 * termio.h
 *
 *                              Beispiel, keine freigegebene Version !!!!!
 *----------------------------------------------------------------------------
 */



/*
 *----------------------------------------------------------------------------
 * Everything from here to the end has to do with the fact that we are making
 * a Version 7 driver for what is essentially a System III device.
 *
 * The first set of defines is for the System III <ioctl> - commands.
 *
 * These yield extensions to the standard Version 7 capabilities, for those
 * programs which are aware of them and choose to use them.
 * The extensions are conditionally compiled, and they can be removed by
 * taking away the definition of TCGETA.
 *----------------------------------------------------------------------------
 */
```

```
/*
 *----------------------------------------------------------------------------
 * System III <ioctl> - commands.
 *----------------------------------------------------------------------------
 */

#define TCGETA   ('T'<<8|1)     /* Get termio parameters */
#define TCSETA   ('T'<<8|2)     /* Set termio parameters immediately */
#define TCSETAW  ('T'<<8|3)     /* Set after draining output */
#define TCSETAF  ('T'<<8|4)     /* Set after draining and flushing */
#define TCSBRK   ('T'<<8|5)     /* Drain output and optionally send break */
#define TCXONC   ('T'<<8|6)     /* Start/Stop control */
#define TCFLSH   ('T'<<8|7)     /* Flush input and/or output buffers */

/*
 *----------------------------------------------------------------------------
 * Now we define all of the things that have to do with the System III
 * termio structure.  We use the standard System III names, prefixed by
 * "T_" to avoid conflicts.
 *----------------------------------------------------------------------------
 *----------------------------------------------------------------------------
 * Input modes.
 *----------------------------------------------------------------------------
 */

#define T_IGNBRK    0000001    /* Ignore break condition */
#define T_BRKINT    0000002    /* Signal interrupt on break */
#define T_IGNPAR    0000004    /* Ignore characters with parity errors */
#define T_PARMRK    0000010    /* Mark parity errors */
#define T_INPCK     0000020    /* Enable input parity check */
#define T_ISTRIP    0000040    /* Strip characters */
#define T_INLCR     0000100    /* Map NL to CR on input */
#define T_IGNCR     0000200    /* Ignore CR */
#define T_ICRNL     0000400    /* Map CR to NL on input */
#define T_IUCLC     0001000    /* Map upper-case to lower-case on input */
#define T_IXON      0002000    /* Enable start/stop output control */
#define T_IXANY     0004000    /* Enable any character to restart output */
#define T_IXOFF     0010000    /* Enable start/stop input control */

/*
 *----------------------------------------------------------------------------
 * Output modes.
 *----------------------------------------------------------------------------
 */
#define T_OPOST     0000001    /* Postprocess output */
#define T_OLCUC     0000002    /* Map lower-case to upper-case on output   */
#define T_ONLCR     0000004    /* Map NL to CR-NL on output */
#define T_OCRNL     0000010    /* Map CR to NL on output */
#define T_ONOCR     0000020    /* No CR output at column 0 */
#define T_ONLRET    0000040    /* NL performs CR function */
#define T_OFILL     0000100    /* Use fill characters for delay */
#define T_OFDEL     0000200    /* Fill is DEL, else is NUL */
#define T_NLDLY     0000400    /* Select new line delays */
#define T_NL0       0
#define T_NL1       0000400
#define T_CRDLY     0003000    /* Select carriage return delays */
#define T_CR0       0
#define T_CR1       0001000
```

```
#define T_CR2       0002000
#define T_CR3       0003000
#define T_TABDLY    0014000    /* Select horizontal tab delays */
#define T_TAB0      0
#define T_TAB1      0004000
#define T_TAB2      0010000
#define T_TAB3      0014000    /* Expand tabs to spaces */
#define T_BSDLY     0020000    /* Select backspace delays */
#define T_BS0       0
#define T_BS1       0020000
#define T_VTDLY     0040000    /* Select vertical tab delays */
#define T_VT0       0
#define T_VT1       0040000
#define T_FFDLY     0100000    /* Select form feed delays */
#define T_FF0       0
#define T_FF1       0100000

/*
 *----------------------------------------------------------------------------
 * Control modes.
 *----------------------------------------------------------------------------
 */

#define T_CBAUD     0000017    /* Baud rate */
#define T_B0        0          /* Hang up line */   (nicht implementiert)
#define T_B50       0000001    /* 50 baud */
#define T_B75       0000002    /* 75 baud */
#define T_B110      0000003    /* 110 baud */
#define T_B134      0000004    /* 134.5 baud */
#define T_B150      0000005    /* 150 baud */
#define T_B200      0000006    /* 200 baud */       (nicht implementiert)
#define T_B300      0000007    /* 300 baud */
#define T_B600      0000010    /* 600 baud */
#define T_B1200     0000011    /* 1200 baud */
#define T_B1800     0000012    /* 1800 baud */
#define T_B2400     0000013    /* 2400 baud */
#define T_B4800     0000014    /* 4800 baud */
#define T_B9600     0000015    /* 9600 baud */
#define T_EXTA      0000016    /* 19200 baud */
#define T_B19200    0000016    /* 19200 baud */
#define T_EXTB      0000017    /* 38400 baud */
#define T_B38400    0000017    /* 38400 baud */
#define T_CSIZE     0000060    /* Character size (excl. parity bit) */
#define T_CS5       0          /* 5 bits */
#define T_CS6       0000020    /* 6 bits */
#define T_CS7       0000040    /* 7 bits */
#define T_CS8       0000060    /* 8 bits */
#define T_CSTOPB    0000100    /* Send two stop bits, else send one */
#define T_CREAD     0000200    /* Enable receiver */
#define T_PARENB    0000400    /* Parity enable */
#define T_PARODD    0001000    /* Odd parity, else even */
#define T_HUPCL     0002000    /* Hang up on last close */
#define T_CLOCAL    0004000    /* Local line, else dial-up */
```

```
/*
 *-------------------------------------------------------------------------
 * Local-modes, basic line-discipline
 *-------------------------------------------------------------------------
 */

#define T_ISIG      0000001    /* Enable signals */
#define T_ICANON    0000002    /* Canonical input */
#define T_XCASE     0000004    /* Canonical upper/lower presentation */
#define T_ECHO      0000010    /* Enable echo */
#define T_ECHOE     0000020    /* Echo erase character as BS-SP-BS */
#define T_ECHOK     0000040    /* Echo NL after kill character */
#define T_ECHONL    0000100    /* Echo NL */
#define T_NOFLSH    0000200    /* Disable flush after interrupt or quit */

/*
 *-------------------------------------------------------------------------
 * Indices of control characters in termio structure c_cchar array.
 *-------------------------------------------------------------------------
 */

#define T_NCC   8      /* Number of control characters in termio structure */

                         /* for canonical-mode */

#define T_VINTR         0       /* Interrupt */
#define T_VQUIT         1       /* Quit */
#define T_VERASE        2       /* Erase */
#define T_VKILL         3       /* Kill */
#define T_VEOF          4       /* End of file */
#define T_VEOL          5       /* Additional end of line */

                           /* for non-canonical-mode : */

#define T_VMIN          4      /* Min. input characters to satisfy read */
#define T_VTIME         5      /* Max. time (.1 seconds) to satisfy read */
                               /* (bei 9780/81 z.Zt. immer auf 0 setzen !! */
```

```
/*
 *----------------------------------------------------------------------------
 * Default control characters.
 *----------------------------------------------------------------------------
 */

#define T_CINTR         0177        /* DEL */
#define T_CQUIT         034         /* FS (CNTRL-\) */
#define T_CERASE        08          /* <X! Backspace */
#define T_CKILL         030         /* CNTRL-X */
#define T_CEOF          04          /* CNTRL-D */
#define T_CSTART        021         /* CNTRL-Q */
#define T_CSTOP         023         /* CNTRL-S */

/*
 *----------------------------------------------------------------------------
 * Finally, the termio structure itself.  We had to change the name
 * of "c_cc" to "c_cchar" in order to avoid a collision with a
 * Version 7 name.
 *----------------------------------------------------------------------------
 */

struct termio                           /* Structure passed by ioctl () */
    {                                   /*------------------------------*/
    unsigned short  c_iflag;            /* Input modes */
    unsigned short  c_oflag;            /* Output modes */
    unsigned short  c_cflag;            /* Control modes */
    unsigned short  c_lflag;            /* Local modes */
             char   c_line;             /* Line discipline */
             char   c_cchar[T_NCC];     /* Control characters */
    };

/* ACHTUNG : evtl. steht auf der mitgelieferten Diskette <unsigned char> */
/* bitte korrigieren */
```

4.4.1 Beschreibung des System-Aufrufes <ioctl>

```
#include <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 <termio.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.

4.4.2 Allgemeines

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>.
Es stehen die File-Pointer <stdin> für die Eingabe und <stdout> und <stderr> für die Ausgabe zur Verfügung.
Für die File-Pointer <stdin> und <stdout> stehen spezielle, vereinfachte Aufrufe zur Verfügung:

```
getchar ();   entspricht  getc (stdin);
putchar (c);  entspricht  putc (c, stdout);
```

Auch die Aufrufe <printf> und <scanf> arbeiten mit <stdout> bzw. <stdin>. Für Low level Ein/Ausgabe-Operationen sind dies die File-Descriptoren 0, 1, 2.

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.

Weiters 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 Ein gabedaten-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 Hängezustand 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). Weiters kann noch ein beliebiges Zeichen als <Eingabe-Begrenzungs-Zeichen> mit derselben Wirkung definiert werden, wenn die Eingabezeilen zu lang sind (c_cchar[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 AusgabeWarteschlange 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.

CTRL | Das FS-Zeichen erzeugt das QUIT-Signal, welches an alle 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 (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 <char c_cchar[];> an der entsprechenden Stelle:

```
c_cchar[T_VINTR]   INTR -Zeichen    Standard = 0x7f = Taste [DEL]
c_cchar[T_VQUIT]   QUIT -Zeichen    Standard = 0x1c = CTRL |
c_cchar[T_VERASE]  ERASE-Zeichen    Standard = 0x08 = Taste [<X]
c_cchar[T_VKILL]   KILL-Zeichen     Standard = 0x18 = CTRL X
c_cchar[T_VEOF]    EOF-Zeichen      Standard = 0x04 = Taste [END]
c_cchar[T_VEOL]    EOL-Zeichen      Standard = -1 = nicht verwendet
c_cchar[T_SWTCH]   SWTCH-Zeichen    nicht implementiert
```

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_cchar[T_NCC];  /* Steuer-Zeichen */
                }                          /* bzw. Non-canonical-Parameter */
```

4.4.3 Schnittstellen-Parameter

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

4.4.3.1 Daten-Eingabe-Steuerung <c_iflag>

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

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

T_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.

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

T_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 T_ISTRIP nicht gesetzt ist, wird ein gültiges Zeichen 0x7f verdoppelt.
Wenn der Parameter T_PARMRK nicht gesetzt ist, und ein fehlerhaftes Zeichen nicht ignoriert wird, wird es als NULL (0) in den Eingabepuffer geschrieben.

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

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

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

T_IGNCR Ignoriere alle gelesenen CR-Zeichen.

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

T_IUCLC — Ändere alle gelesenen Großbuchstaben in Kleinbuchstaben. Siehe auch die Parameter T_ICANON und T_XCASE in <c_lflag>.

T_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 <T_XANY> ein beliebiges weiteres Eingabe-Zeichen erkannt wird.

T_IXANY — Jedes Zeichen startet gestoppte Ausgabe.

T_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.

4.4.3.2 Daten-Ausgabe-Steuerung <o_flag>

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:

T_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.

T_OLCUC — Auszugebende Kleinbuchstaben werden in Großbuchstaben umgewandelt. Siehe auch das Flag <IUCLC>.

T_ONLCR — Ein auszugebendes NL wird als CR-NL ausgegeben.

T_OCRNL — Ein auszugebendes CR wird als NL ausgegeben.

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

T_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.

T_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 <T_OFDEL> gesetzt ist, ist das Füllzeichen DEL (0x7f).

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

T_NLDLY — Maske für das NL-Verzögerungsbit zum Löschen.

T_NL0 — Keine Verzögerung bei auszugebendem NL-Zeichen.

T_NL1 — Verzögerung von 0.10 Sekunden bei auszugebendem NL-Zeichen.

	Wenn das Flag <T_ONLRET> gesetzt ist, wird die CR-Verzögerung ausgeführt. Wenn das Flag <T_OFILL> gesetzt ist werden 2 Füllzeichen gesendet.
T_CRDLY	Maske für die CR-Verzögerungsbits zum Löschen.
T_CR0	Keine Verzögerung bei auszugebendem CR-Zeichen.
T_CR1	Spaltenabhängige Verzögerung bei auszugebendem CR-Zeichen. Wenn das Flag <T_OFILL> gesetzt ist, werden 2 Füllzeichen gesendet.
T_CR2	Verzögerung von 0.10 Sekunden bei auszugebendem CR-Zeichen. Wenn das Flag <T_OFILL> gesetzt ist, werden 4 Füllzeichen gesendet.
T_CR3	Verzögerung von 0.15 Sekunden bei auszugebendem CR-Zeichen.
T_TABDLY	Maske für die TAB-Verzögerungsbits zum Löschen.
T_TAB0	Keine Verzögerung bei auszugebendem TAB-Zeichen.
T_TAB1	Spaltenabhängige Verzögerung bei auszugebendem TAB-Zeichen. Wenn das Flag <T_OFILL> gesetzt ist, werden 2 Füllzeichen gesendet.
T_TAB2	Verzögerung von 0.10 Sekunden bei auszugebendem TAB-Zeichen. Wenn das Flag <T_OFILL> gesetzt ist, werden 2 Füllzeichen gesendet.
T_TAB3	Anstelle dem TAB-Zeichen wird eine entsprechende Anzahl von Spaces ausgegeben.
T_BSDLY	Maske für das BS-Verzögerungsbit zum Löschen.
T_BS0	Keine Verzögerung bei auszugebendem BS-Zeichen.
T_BS1	Verzögerung von 0.05 Sekunden bei auszugebendem BS-Zeichen. Wenn das Flag <T_OFILL> gesetzt ist, wird 1 Füllzeichen gesendet.
T_VTDLY	Maske für das VT-Verzögerungsbit zum Löschen.

T_VT0	Keine Verzögerung bei auszugebendem VT-Zeichen.
T_VT1	Verzögerung von 2 Sekunden bei auszugebendem VT-Zeichen.
T_FFDLY	Maske für das FF-Verzögerungsbit zum Löschen.
T_FF0	Keine Verzögerung bei auszugebendem FF-Zeichen.
T_FF1	Verzögerung von 2 Sekunden bei auszugebendem FF-Zeichen.

4.4.3.3 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:

T_CBAUD	Maske für die Baud-Raten-Bits zum Löschen.	
T_B0	Verbindung auflösen	(nicht implementiert)
T_B50	50 bit/s	
T_B75	75 bit/s	
T_B110	110 bit/s	
T_B134	134.5 bit/s	
T_B150	150 bit/s	
T_B200	200 bit/s	(nicht implementiert)
T_B300	300 bit/s	
T_B600	600 bit/s	
T_B1200	1200 bit/s	
T_B1800	1800 bit/s	
T_B2400	2400 bit/s	
T_B4800	4800 bit/s	
T_B9600	9600 bit/s	
T_B19200	19200 bit/s	
T_EXTA	19200 bit/s	
T_B38400	38400 bit/s	
T_EXTB	38400 bit/s	

T_CSIZE — Maske für T_CS-Bits zum Löschen

T_CS5 — Zeichenrahmen ist 5 Bits ohne Paritätsbit für Ein- und Ausgabe.

T_CS6 — Zeichenrahmen ist 6 Bits ohne Paritätsbit für Ein- und Ausgabe.

T_CS7 — Zeichenrahmen ist 7 Bits ohne Paritätsbit für Ein- und Ausgabe.

T_CS8 — Zeichenrahmen ist 8 Bits ohne Paritätsbit für Ein- und Ausgabe.

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

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

T_PARENB Wenn das Flag gesetzt ist, wird die Paritätsbit-Generierung und Prüfung unterstützt. Das Flag <T_PARODD> definiert die gewünschte Zeichenparität.

T_PARODD Dieses Flag ist nur bei gesetztem <T_PARENB>-Flag wirksam. Wenn das Flag gesetzt ist, ist die Zeichenparität ungerade. Wenn das Flag nicht gesetzt ist, ist die Zeichenparität gerade.

T_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).

T_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.

T_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).

4.4.3.4 Steuerung der Terminal-Funktionen <c_lflag>

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

T_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 <T_ISIG> verhindert werden, indem als auszulösender Zeichencode <-1> in das Array <c_cchar[]> eingetragen wird.

<SWTCH> ist derzeit nicht implementiert.

T_ICANON — Wenn zusätzlich das Flag T_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 <T_VMIN> Zeichen verfügbar sind oder zwischen dem Empfang von 2 Zeichen die Zeit <T_VTIME> in Zehntel-Sekunden abläuft.
Die entsprechenden Werte für <T_VMIN> und <T_VTIME> sind im Array <c_cchar[]> an der entsprechenden Stelle einzutragen; (in diesem Fall werden die anderen Einträge im Array nicht bewertet).

c_cchar [T_VMIN] = — Anzahl der blockweise zu lesenden Zeichen.

c_cchar [T_VTIME] = — Timeout in Zehntel-Sekunden.

Dadurch werden schnelle <Burst>-Leseoperationen ermöglicht. Durch geeignete Kombination von <T_VMIN> und <T_VTIME> ergeben sich 4 mögliche Verhaltensweisen:

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

T_VMIN > 0 und T_VTIME = 0:
Der Leseauftrag ist erfüllt, wenn T_VMIN Zeichen empfangen wurden.

T_VMIN = 0 und T_VTIME > 0:
Der Timer wird sofort gestartet. Der Leseauftrag ist erfüllt, wenn 1 Zeichen verfügbar ist oder der Timer vorher abläuft
(derzeit nicht implementiert).

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

T_XCASE — Wenn dieses und das <T_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.

Weiters werden folgende Backslash-Folgen akzeptiert bzw. ausgegeben:

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

Beispiele:

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

T_ECHO — Alle empfangenen gültigen Zeichen werden sofort wieder automatisch ausgegeben.

T_ECHOE — Wenn zusätzlich die Flags <T_ICANON> und <T_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 <T_ECHO> nicht gesetzt ist, wird das ERASE-Zeichen (Taste ⌫) als 2-Zeichen-Folge ge-echoed
SP-BS.

T_ECHOK — Wenn zusätzlich das Flag <T_ICANON> gesetzt ist, wird nach einem erkannten KILL-Zeichen (CTRL X) zusätzlich das Zeichen NL ge-echoed, um das Löschen der Zeile anzuzeigen.

T_ECHONL — Wenn zusätzlich das Flag <T_ICANON> gesetzt ist, wird das NL-Zeichen in jedem Fall ge-echoed, auch wenn das Flag <T_ECHO> nicht gesetzt ist.

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

<SWTCH> ist derzeit nicht implementiert.

4.4.4 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)`
`ioctl (fildes, TCGETS, argp)` nicht implementiert.

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

`ioctl (fildes, TCSETA, argp)`
`ioctl (fildes, TCSETS, argp)` nicht implementiert.

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 <termio> festgelegten Parametern parametrisiert.

`ioctl (fildes, TCSBRK, 1)`

Es wird gewartet, bis die Ausgabewarteschlange leer ist.

`ioctl (fildes, TCSBRK, 0)`

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

5 Spool-System

5.1 Allgemeiner Überblick über die Druckerverwaltung unter SINIX

5.1.1 Dateien und Dateiverzeichnisse

Folgende Dateien und Dateiverzeichnisse sind für die Druckerverwaltung notwendig:

`/usr/tmp/copies/*`

Wird der lpr über Pipe versorgt oder mit dem Schalter -cp aufgerufen, so wird die zu druckende Datei in diesem Dateiverzeichnis zwischengespeichert.

`/usr/spool/lpd/pid`

In dieser Datei wird die Prozeßnummer des /etc/qdaemon hinterlegt.

`/usr/spool/lpd/qdir`

In diesem Dateiverzeichnis werden alle Druckaufträge in Kurzform notiert.

`/usr/spool/lpd/stat`

In den Dateien s0.D?Q werden die notwendigen Informationen über die Drucker und deren Status in binärer Form gespeichert.

In den temporären Dateien p0.D?Q wird die Prozeßnummer des Bakkends /usr/lib/lp900x gespeichert, falls ein solches Programm aktiv ist.

`/usr/lib/qconfig`

Diese Datei enthält im Klartext die aktuelle Druckerkonfiguration.

`/usr/lib/qconfig.bin`

Diese Datei enthält in Binärform die aktuelle Druckerkonfiguration.

Folgende Dateien werden zusätzlich vom SINIX-Menüsystem verwaltet:

`/usr/lib/queues`

Diese Datei enthält zeilenweise alle erlaubten -ws Werte.

`/usr/lib/qdescr`

Klartextbeschreibung der aktuellen Druckerkonfiguration.

5.1.2 Programme für die Druckerverwaltung

lpr

Der lpr notiert die Aufträge in den entsprechenden Dateien.
Danach sendet er dem /etc/qdaemon ein Signal, daß ein neuer Auftrag vorliegt.
Er kann auch die Dateien in /usr/spool/lpd/stat lesen und zeigt dann den aktuellen Zustand der Drucker an (Schalter 'q').

`/etc/qdaemon`

Das Betriebssystem SINIX erlaubt es, daß mehrere Dateien zum Lesen und zum Schreiben geöffnet werden, also auch Gerätedateien. Es wäre schlecht, wenn mehrere Druckaufträge gleichzeitig auf dieselbe Gerätedatei schreiben würden. Der /etc/qdaemon sorgt dafür, daß immer nur ein Auftrag an einen Drucker vergeben wird.
Er ruft das Backend auf und interpretiert dessen Return-Code. Wenn der Return-Code des Backends gleich 64 ist, so wird der Drucker gesperrt, der Druckauftrag bleibt erhalten und der Benutzer erhält eine 'mail'.
Andernfalls wird der Auftrag in den entsprechenden Dateien gelöscht und der Drucker auf BEREIT gestellt.

`/usr/lib/lp9001` bzw.
`/usr/lib/9004`
`"Backend"`

Diese Programme sind druckerspezifisch. Sie sorgen dafür, daß die Gerätedatei für den Drucker richtig eingestellt wird, der Drucker in den Grundzustand gesetzt und überprüft wird, ob der Drucker richtig arbeitet.
Kopf- und Anhangseiten werden ebenfalls von diesen Programmen erstellt. Es sind C-Anwenderprogramme, spezielle Kenntnisse des Systemkernes zur Erstellung eines solchen Programmes sind nicht notwendig, jedoch Kenntnisse der Schnittstellen zu qdaemon und lpr.

`"treiber"`

Der Treiber ist Teil des Systemkernes. Er ist so allgemein wie möglich gehalten. Es ist wahrscheinlich möglich, alle Geräte, die irgendwie mit einer der Schnittstellen verträglich sind, mit dem vorhandenen Treiber zu betreiben.

`/usr/lib/digest`

Sollte die Datei /usr/lib/qconfig neueren Datums als die Datei /usr/lib/qconfig.bin sein, so wird die Datei qconfig übersetzt und die Daten werden in binärer Form in /usr/lib/qconfig.bin abgelegt.
Auf Wunsch können Parameter für das Backend in der Datei qconfig angegeben werden.

Beispiel:

Soll standardmäßig in Schmalschrift gedruckt werden, so kann man in der Datei qconfig eintragen:

```
treiber=/usr/lib/lp9001 -pb3
```

Bekanntmachung der Änderung durch lpr -rr

5.1.3 Die Funktionen des lpr

Soll eine Datei gedruckt werden, so wird das mit dem Kommando lpr Datei erreicht. Der lpr erzeugt dann in dem Dateiverzeichnis /usr/spool/lpd/qdir eine Datei mit eindeutigem Namen, die etwa wie folgt aussieht:

```
horst 0 0 0
/usr/horst/Dokumente/besch.ASF
8 1 1 1
horst
/usr/horst/Dokumente logfile
Fr 25.Jan.1985, 16:48:46
/usr/horst/Dokumente/besch.ASF
-pl=60
```

Die wichtigsten Einträge haben folgende Bedeutung:

In Zeile 1 steht die Benutzerkennung des Auftraggebers.

In Zeile 2 steht der Name des Dokumentes, in diesem Fall /usr/horst/Dokumente/besch.ASF (oder z.B. PIPE.345).(lpr -tl=...).

Die Zahl 8 in der Zeile 3 bedeutet, daß die Dateigröße 8 Blöcke beträgt. Die darauffolgende Zahl 1 zeigt, daß eine Kopie von dieser Datei gemacht werden soll (Schalter -nc=1 beim lpr).

In Zeile 4 steht der Adressat, für den der Auftrag bestimmt ist. (lpr -to=horst).

Zeile 5 zeigt, in welchem Dateiverzeichnis der Auftrag abgegeben wurde, und Zeile 6 zeigt Datum und Uhrzeit des Druckauftrages.

In Zeile 7 steht entweder der relative oder der absolute Pfadname der Datei, die gedruckt werden soll.

In Zeile 8 stehen alle Schalter, die der lpr an das Backend weitergibt. In diesem Fall den Schalter -pl=60.

Bei dem Aufruf pr Datei lpr ändern sich die Einträge etwa wie folgt:

```
horst 0 0 0
PIPE.401
8 1 1 1
horst
/usr/spool/lpd/qdir logfile
Di 12.Feb.1985, 13:39:11
/usr/tmp/copies/taa00401
```

In diesem Fall ist der Name der zu druckenden Datei PIPE.401 und die zu druckenden Daten stehen in der Datei /usr/tmp/copies/taa00401.

Beim Aufruf lpr -q werden die Dateien im Dateiverzeichnis /usr/spool/lpd/stat interpretiert.
Sie sind in einer binären Form abgelegt und sollten auf keinen Fall mit einem Editor bearbeitet werden, da Editoren diese Datei auf jeden Fall verändern (an das Dateiende wird ein Line Feed Zeichen gesetzt).

Die Datei s0.D1Q hat folgenden Inhalt:

```
(xd s0.D1Q)

 0  01010000 01000000  00000000 00000000  |                |
10  46722032 352e4a61  6e2e3139 38352c20  |Fr 25.Jan.1985, |
20  31363a34 383a3436  20004469 2031322e  |16:48:46  Di 12.|
30  4665622e 31393835  2c203134 3a31303a  |Feb.1985, 14:10:|
40  34320000 686f7273  74006e6e 74000000  |42  horst nnt   |
50  00000000 00000000  00000000 686f7273  |            hors|
60  74006e6e 74000000  00000000 00000000  |t nnt           |
70  00000000 7273742f  446f6b75 6d656e74  |    rst/Dokument|
80  652f6265 7363682e  41534600 556e6265  |e/besch.ASF Unbe|
90  6b616e6e 74000000  00000000 00000000  |kannt           |
a0  00000000 00                           |                |
```

Ab Byte 16 (X'10') sind ähnliche Einträge wie in den Dateien in /usr/spool/lpd/qdir, nur in anderer Form.

Außerdem wird zusätzlich eingetragen, wann mit dem Drucken des Auftrags begonnen wurde.

Die beiden ersten Bytes sind dabei am wichtigsten.

Das erste Byte zeigt den Zustand des Druckers an. Dabei bedeutet:

1 BEREIT
2 LAEUFT
3 WARTET
4 GESPERRT

Das zweite Byte zeigt, wieviele Kopien von diesem Dokument ausgegeben werden sollen.

Hat der lpr einen Auftrag angenommen, so sendet er dem qdaemon das Signal 5. Die Prozeßnummer des qdaemon ist in der Datei /usr/spool/lpd/pid gespeichert.

Wird die Druckerkonfiguration verändert, dann ist die Datei /usr/lib/qconfig neueren Datums als die Datei /usr/lib/qconfig.bin. Der Befehl lpr -rr ruft nun das Programm /usr/lib/digest auf und dieses Programm übersetzt die Datei /usr/lib/qconfig, sofern die Datei /usr/lib/qconfig neueren Datums ist.
/usr/lib/digest speichert die übersetzten Daten in der Datei /usr/lib/qconfig.bin. Danach wird dem qdaemon ein Signal gesendet. Der qdaemon sollte dann die Datei /usr/lib/qconfig.bin wieder neu lesen, leider killt das Signal den qdaemon. Er muß deshalb neu gestartet werden.

Von den Schaltern, die zusammen mit dem lpr-Befehl angegeben werden können (siehe SINIX, Buch 1, Seite 6-154 ff), werden viele direkt vom lpr erkannt und verarbeitet und alle anderen an das Backend weitergegeben.
Der lpr verarbeitet folgende Schalter:

-ws=,	-pr=,	-ap=,	-ca,	-cp,	-nc=,	-no,	-to=,
-rm,	-q,	-dd,	-dk,	-du,	-rr,	-dg,	-sy=

Der Befehl lpr -du arbeitet etwa in folgender Weise:

In der Datei /usr/spool/stat/s0.D?Q wird das erste Byte auf 1 gesetzt. Danach wird dem qdaemon das Signal 5 gesendet. Der qdaemon versucht nun im Dateiverzeichnis /usr/spool/lpd/qdir einen neuen Auftrag zu finden.
Gleichgültig, ob ein Auftrag vorliegt oder nicht, der Drucker ist wieder aktiv.

Der Befehl lpr -dk arbeitet in folgender Weise:

Ist ein Drucker aktiv, so ist die Prozeßnummer des aktiven Backends in der Datei /usr/spool/lpd/stat/p0.D?Q abgelegt. Diesem Prozeß wird das Signal 15 gesendet. Das Backend versucht nun, noch ein Form Feed Zeichen an den Drucker abzusetzen, um sich dann mit exit (64) zu beenden. Der qdaemon sperrt daraufhin den Drucker.
Je nach Zustand des Druckers ist es unter Umständen nicht einmal möglich, ein einziges Byte an den Drucker zu senden. Dies geschieht z.B., wenn der Drucker ausgeschaltet ist, oder wenn die Pause-Taste gedrückt wurde und der Puffer mit Daten voll ist. Der Prozeß schläft dann. Ein weiteres Signal 15 sorgt dafür, daß der Drucker auf jeden Fall gesperrt wird .
Deshalb muß man manchmal das Kommando lpr -dk zweimal aufrufen. Ist für diesen Drucker kein Backend aktiv, so wird Byte 1 in der Datei /usr/spool/lpd/stat/s0.D?Q verändert.

5.1.4 Die Funktionen des qdaemon

Der qdaemon wird in der Datei /etc/rc gestartet. Er ist das zentrale Programm der Druckerverwaltung. Seine Aufgabe ist es, dafür zu sorgen, daß immer nur ein Druckauftrag an einem Drucker ausgegeben wird.
Den Programmablauf kann man sich wie folgt vorstellen, wobei gleichgültig ist, ob der qdaemon in der Datei /etc/rc oder mit einem Kommando gestartet wird:

1. Zuerst muß sichergestellt werden, daß kein qdaemon bereits aktiv ist. Dies wird auf folgende Art festgestellt:

   a) Existiert die Datei /usr/spool/lpd/pid nicht, so ist kein weiterer qdaemon aktiv.

   b) Existiert diese Datei und existiert ein Prozeß mit der gleichen Prozeßnummer wie in der Datei /usr/spool/lpd/pid, so ist ein qdaemon aktiv.

   Manipulationen der Datei /usr/spool/lpd/pid sollten unterlassen werden, da sonst mehrere qdaemon aktiv sein können oder das Programm /etc/qdaemon sich nicht starten läßt.

   Glaubt der qdaemon, daß schon ein weiterer qdaemon aktiv ist, so beendet er sich. Da der qdaemon ein Hintergrundprozeß ist, muß er sofort nach Aufruf ein fork ausführen, wobei sich danach der Vaterprozeß sofort beendet.

2. Der qdaemon liest dann die Datei /usr/lib/qconfig.bin. Damit kennt er die aktuelle Druckerkonfiguration.

3. Nicht vorhandene Statusdateien in /usr/spool/lpd/stat werden vom qdaemon angelegt.

4. Ein Druckauftrag in /usr/spool/lpd/qdir, falls vorhanden, wird in Binärform umgewandelt und in der entsprechenden Statusdatei in /usr/spool/lpd/ stat/* abgelegt.

5. Sollte kein Druckauftrag vorliegen, so wartet der qdaemon, bis er das Signal 5 erhält.

6. Danach verdoppelt er sich (fork) und sein Sohnprozeß öffnet folgende Dateien:

Filedesc. 0	Die Gerätedatei des Druckers zum Lesen
Filedesc. 1	Die Gerätedatei des Druckers zum Schreiben
Filedesc. 2	Die Datei /dev/null
Filedesc. 3	Die Statusdatei des entsprechenden Druckers (die Datei /usr/spool/lpd/stat/s0.D?Q)

7. Außerdem wird noch die Datei /usr/spool/lpd/stat/p0.D?Q erzeugt und in dieser Datei wird die Prozeßnummer dieses Sohnprozesses hinterlegt. Der Status des Druckers wird auf LAEUFT gesetzt.

8. Der Sohn-qdaemon wechselt in das Dateiverzeichnis, in dem der Auftrag abgegeben wurde.

9. Es folgt der Aufruf exec ('lp900x','Schalter','Datei'). Der qdaemon weiß aus der Datei qconfig, welches Programm er aufrufen muß (siehe Datei qconfiq 'treiber=/usr/lib/lp900x').
   Damit übernimmt das Backend die weiteren Aufgaben.

10. Der Vater-qdaemon wartet nun so lange, bis das Backend sich beendet und interpretiert den Return-Code. Ist der Return-Code ungleich 64, so wird der Druckauftrag vom qdaemon gelöscht und ein neuer Drukkauftrag kann an diesem Drucker ausgegeben werden (siehe 5.1.2).

5.1.5 Die Funktionen des Backend

Die Programme /usr/lib/lp9004 bzw. /usr/lib/lp9001 werden in der Datei /usr/ lib/qconfig unglücklicherweise Treiber genannt.
Dies ist üblich, aber nicht richtig.

Alle Schalter, die der lpr nicht kennt, werden dem Backend übergeben:

Beispiel:

Schalter des Backends lp9001:

-statusfile	Interner Schalter, der vom qdaemon gesetzt wird.
-pb=	Funktion siehe SINIX, Buch 1: lpr
-int	"
-ab=	"
-bis=	"
-dt	"
-pb1	"
-pb2	"
-pb3	"

Dies sind sicher nicht alle Schalter, aber die wichtigsten.
Auch bei einem selbstgeschriebenen Backend kann man eigene Schalter definieren (siehe dazu SINIX, Buch 1, Seite 6-160).

Schließlich wird auch noch der Name der zu druckenden Datei übergeben. Das Backend liest die Datei /usr/spool/lpd/stat/s0.D?Q (FILEDESC. 3). Byte 2 dieser Datei beschreibt, wieviel Ausgaben dieser Datei gewünscht werden.
In Byte 9 kann vom Backend hinterlegt werden, wieviel Prozent des Auftrages bereits erledigt sind.
Die Bytes 11 und 12 stehen zur Ausgabe der Anzahl bereits gedruckter Seiten zur Verfügung.
Der Rest sind Informationen für Kopf und Anhang.

Das Backend setzt nun die Gerätedatei in den richtigen Zustand.
Dies kann in C mit der Funktion ioctl(0,TIOCSETP,&structur) realisiert werden (siehe CES-Manual).

Danach wird der Drucker in den Grundzustand gesetzt, Tabulatorstopps gesetzt und eventuell eine oder mehrere Escape-Folgen und ein Form Feed Zeichen gesendet. Eine Escape-Folge wird z.B. für den Schalter pb3 des Druckers 9001 gesendet. Anschließend wird mit dem ETX/ACK Protokoll

überprüft, ob der Drucker richtig arbeitet. Der Datenfluß wird vom Treiber mit dem XON-XOFF-Protokoll gesteuert.

Sollte der Drucker länger als etwa 200 Sekunden benötigen, um 512 Zeichen zu verarbeiten, oder sollte es länger als 200 Sekunden dauern, bis die Antwort 'ACK' auf ein vom Backend gesendetes 'ETX' eintrifft, so setzt das Backend den Treiber wieder in den alten Zustand, sperrt den Drucker und beendet sich (siehe dazu die Funktionen alarm() und signal() im CES-Manual).

Wird das Signal 15 vom lpr an das Backend gesendet, so versucht das Backend noch ein Form Feed Zeichen an den Drucker zu senden, bevor es sich selbst mit exit (64) beendet.
Erst ein zweites Signal 15 beendet das Backend auf jeden Fall, wobei vorher noch der Druckertreiber in den alten Zustand gesetzt und der Drukker als gesperrt gekennzeichnet wird.
So etwas geschieht beim Aufruf lpr -dk.

Das Programm /bin/cat ist in einfacher Form ebenfalls ein Backend.
Hier muß man allerdings die Werte für die Gerätedatei selbst einstellen. Eventuelle Fehlermeldungen werden dann in /dev/null (elektronischer Papierkorb) geschrieben und alle übergebenen Schalter werden ignoriert. Es ist aber eine einfache Möglichkeit zu prüfen, ob ein neuer Drucker an den PC-X oder PC-MX angeschlossen werden kann.

5.1.6 Die Funktion des Treibers

Der Treiber ist Teil des Systemkernes. Seine Aufgabe besteht darin, Daten an eine Schnittstelle weiterzugeben und die Schnittstelle entsprechend zu verändern. Die möglichen Einstellungen sind in SINIX, Buch 1 beschrieben.

Zum Betrieb eines Druckers sind die Einstellungen 'cbreak' und 'raw' interessant.
Im cbreak-Mode führt der Treiber ein XON-XOFF-Protokoll (Datenflußsteuerung). Die Parität kann wahlweise auf odd (ungerade) oder even odd (kein Paritätsbit) eingestellt werden.
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.

5.2 Betrieb von verschiedenen Druckern am PC-MX/PC-X

Im Prinzip kann nahezu jeder Drucker mit einer Schnittstelle SS97 oder RS232 verwendet werden. Standardmäßig wird jedoch nur der Anschluß der Siemens-Drucker 9001 und 9004 unterstützt.
Es ergibt sich oft das Problem, daß ein Drucker, der mit anderen Steuerzeichen (Escape-Sequenzen) arbeitet, andere Meldungen zurückliefert, eine andere Parität oder Baudrate erfordert als die beiden Standard-Drucker.
In einem solchen Fall läßt sich dieser Drucker nicht ohne weiteres über die Spool-Druckverwaltung betreiben.

5.2.1 Auswahl der Treiber

Ein Drucker, der an dem CONAC-Board angeschlossen ist, wird über eine Gerätedatei mit der Majornummer 3 angesprochen, während für das SERAC-Board (PC-MX) die Majornummer 8 verwendet werden muß. Die Majornummer legt den verwendeten Treiber fest (vgl. SINIX, Buch 1, S. 5-25).
Wenn z.B. der Drucker 9001 als Drucker D1 konfiguriert wurde, dann sollte man in dem Dateiverzeichnis '/dev' mit dem Befehl 'ls -l lp*' folgende Gerätedatei finden :

```
crw-rw-rw- 1 root        3,  1 Mar 20 13:33 lp9001-1-D1
```

Der Name dieser Datei spielt dabei keine Rolle, wesentlich sind die Angaben der Major- und Minornummer (hier 3, 1).

5.2.2 Umgehung der Standarddruckverwaltung.

Es besteht die Möglichkeit, direkt Daten zu diesem Treiber und damit zu einem angeschlossenen Fremddrucker zu senden.
Der Befehl
`cat .u DATEINAME >/dev/lp9001-1-D1`
bewirkt beispielsweise die Ausgabe einer Datei auf dem Drucker, falls der Treiber richtig 'eingestellt' ist und der Drucker ein XON-XOFF-Protokoll führt.
Die Einstellung des Treibers kann mit dem Kommando
`stty >/dev/lp9001-1-D1`
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, der Treiber im 'cbreak'-Mode arbeitet und daß **kein** 'echo' gesetzt ist.
Eine dauerhafte Veränderung der Treiber-Einstellung kann mit der folgenden Kommandofolge ereicht werden :

```
sleep 65000 < /dev/lp9001-1-D1&          (&-Zeichen nicht vergessen!)
stty SCHALTER >/dev/lp9001-1-D1          (SINIX, Buch 1, S. 6-250ff).
```

Das sleep-Kommando ist nur bei Verwendung des CONAC-Treibers beim PC-MX (SINIX Version 1.0B) erforderlich.
Als SCHALTER gibt man z.B. an:

```
-echo -nl -tabs odd cbreak
```

5.2.2.1 Spezielle Bemerkung für den PC-MX (SINIX Version 1.0B)

Falls Daten ohne Paritätsbit übertragen werden sollen (z.B. im Bit-Image-Mode eines Matrixdruckers), dann muß für den CONAC-Treiber (Majornummer 3) die Minornummer 131 gewählt werden. Nur dann kann der Kanal entsprechend umparametrisiert werden.

Beispiel:

```
/etc/mknod lpneu c 3 131
```

5.2.2.2 Spezielle Bemerkung für den PC-X

Der CONAC-Treiber läßt sich durch den Befehl
`stty cbreak -echo even odd > /dev/lp9001-1-D1`
so umschalten, daß er kein Paritätsbit überträgt.

Mit der Minornummer 1 wählt man hier den SS97-Ausgang (ST2) und mit der Minornummer 131 den RS232-Ausgang (ST1).
Die beiden Ausgänge ST3 (RS232) und ST4 (SS97) kann man über die Major- und Minornummern 3, 130 und 3, 0 ansprechen.
Hier kann z.B. ein zweiter Drucker angeschlossen werden, wenn die Gerätedateien existieren und entsprechende Einträge in der Datei /usr/lib/qconfig vorgenommen wurden.

5.2.3 Betrieb eines Druckers über die Standarddruckverwaltung

Um einen Drucker mit dem üblichen 'lpr'- Befehl ansprechen zu können, muß man ein individuelles Backend in die Druckerverwaltung integrieren. Das Backend bildet eine speziell auf den jeweiligen Drucker angepaßte Verbindung zwischen der allgemeinen Druckerverwaltung lpr bzw. /etc/qdaemon und den allgemeinen Treibern für das CONAC- oder SERAC-Board.
Es wird im Programm /etc/qdaemon aufgerufen, wobei alle Schalter, die der lpr nicht kennt, und der Name der zu druckenden Datei als Argumente übergeben werden.

Die Standard Ein/Ausgabe des Backends wird bereits vom qdaemon auf den Treiber umgelegt.
Für die Siemens-Drucker 9001 und 9004 stehen die Backends

/usr/lib/lp9001 und /usr/lib/lp9004 zur Verfügung. Das Backend lp9001 interpretiert z.B. folgende Schalter:

`-int, -ab=, -bis=, -pb=, -pl=, -dt, -pb1, -pb2, -pb3.`

Die Bedeutung der Schalter ist in SINIX, Buch 1, Seite 6-154 dokumentiert.
Der Pfadname des zu verwendenden Backends muß in der Datei /usr/lib/qconfig hinter der Angabe 'treiber =' angegeben werden.
An dieser Stelle können auch Standardschalter für das Backend gesetzt werden. (Die Bezeichnung 'treiber' ist hier mißverständlich!)

Beispiel für die Datei /usr/lib/qconfig:

```
D1Q:
        queue = D1
        gerät = D11
D11:
        file = /dev/lp9001-1-D1
        treiber = /usr/lib/lp9001 -dt
        kopf = niemals
        zugriff = lesen
        anhang = niemals
```

Der Name der Gerätedatei, über die die Ausgabe auf den Drucker geleitet werden soll (der Verweis auf den wirklichen 'Treiber'), muß hinter der Angabe 'file =' eingetragen werden.

5.2.3.1 Das primitive Backend /bin/cat

Wenn der Treiber richtig eingestellt ist, kann man das Kommando 'cat' als universelles, primitives Backend verwenden. Die oben genannten Schalter werden dann allerdings ignoriert.
Anstelle von /usr/lib/lp9001 muß dann /bin/cat .u in die Datei /usr/lib/qconfig eingetragen werden.

Nach jeder Änderung der qconfig-Datei muß der super-user den Befehl lpr -rr abgeben, um der Druckerverwaltung die Änderung mitzuteilen. Anschließend muß das Programm /etc/qdaemon neu gestartet werden: /etc/qdaemon

5.2.4 Eigenes Backend

Ein eigenes spezielles Backend für den betreffenden Drucker kann in der Programmiersprache C mit den Hilfsmitteln des C-Entwicklungssystems realisiert werden.
Es sollte folgende Aufgaben erfüllen:

- Gerätedatei (Treiber) richtig einstellen.
- Interpretation der oben angegebenen Schalter. Gegebenenfalls müssen die entsprechenden ESC-Folgen an den Drucker gesendet werden.
- Eröffnen und Lesen der Datei, deren Name als Argument übergeben wird.
- Anzahl der gewünschten Kopien feststellen.
- Ausgabe des Dateiinhalts in einer für den Drucker geeigneten Form.
- Prüfen, ob der Drucker ordnungsgemäß arbeitet (ETX – ACK – Protokoll).
  Im Fehlerfall muß sich das Backend mit exit(64) beenden, damit der Drucker vom qdaemon gesperrt wird.
- Eventuelle Meldungen des Druckers interpretieren und Fehlermeldungen in die Mail-Datei des Auftraggebers schreiben.
- Abfangen des Signals 15.
  Das Signal 15 wird durch den Befehl 'lpr -dk' an das Backend gesendet und sollte zu einem kontrollierten Abbruch mit exit-Status 64 führen.
- Die Druckerverwaltung informieren, wieviel Prozent des Auftrages bereits erledigt und wieviele Seiten bereits gedruckt worden sind.

Zur Erfüllung dieser Aufgaben steht dem Backend eine vom Programm /etc/qdaemon erzeugte und eröffnete Datei /usr/spool/lpd/stat/s0.D?Q zur Verfügung, auf die über den Filedeskriptor 3 zugegriffen werden kann. In dieser Datei sind u.a. der Name des Auftraggebers, die gewünschte Anzahl von Kopien und der Status des Druckers vermerkt.

Beispiel für die Datei `/usr/spool/lpd/stat/s0.D1Q`
(ausgegeben mit dem Befehl `xd /usr/spool/lpd/stat/s0.D1Q`)

```
0   02010000 01000000 1a000300 00000000 |                |
10  46722032 382e4465 7a2e3139 38342c20 |Fr 28.Dez.1984, |
20  30303a33 333a3036 20004672 2032322e |00:33:06  Fr 22.|
30  4d61722e 31393835 2c203039 3a34333a |Mar.1985, 09:43:|
40  35310000 6672616e 7a006e6e 74000000 |51  franz nnt   |
50  00000000 00000000 00000000 6672616e |            fran|
60  7a006e6e 74000000 00000000 00000000 |z nnt           |
70  00000000 2f757372 2f67616d 65732f77 |    /usr/games/w|
80  75726d2e 63000000 00000000 556e6265 |urm.c       Unbe|
90  6b616e6e 74000000 00000000 00000000 |kannt           |
a0  00000000 00                         |                |
```

Das erste Byte zeigt den Zustand des Druckers an.
Dabei bedeutet:

1 BEREIT
2 LAEUFT
3 WARTET
4 GESPERRT

Das zweite Byte zeigt, wieviele Kopien gedruckt werden sollen.

Im neunten Byte kann vom Backend hinterlegt werden, wieviel Prozent des Auftrages bereits erledigt sind (hier 1a = 26%).
Die Bytes elf und zwölf stehen zur Ausgabe der Anzahl bereits gedruckter Seiten zur Verfügung.
Diese Angaben können vom Benutzer mit dem Befehl lpr -q angefordert werden.

Das erste Datum gibt den Zeitpunkt der letzten Änderung der Datei, das zweite Datum den Zeitpunkt des Druckauftrages an.

Der erste Name ist offenbar der Name des Auftragsgebers.

Wenn sich das Backend durch exit(64) beendet, dann setzt der qdaemon das erste Byte dieser Datei auf 4 und sperrt damit den Drucker. Der Druckauftrag bleibt jedoch erhalten.

Die folgenden Beispiel-C-Programme erfüllen die meisten der angegebenen Backend-Aufgaben für diesen Drucker 9001 bzw. 9004.
Sie sind so geschrieben, daß man sie leicht an andere Drucker anpassen kann.

Um das Programm für den Drucker 9001 möglichst übersichtlich zu halten, wurde die Möglichkeit des Bit-Image-Printing (Grafikdruck) in diesem Beispiel nicht berücksichtigt.
Beide Backends verzichten auf die Angabe, wieviel Prozent eines Auftrages bereits gedruckt wurden.
Das Programm für den Drucker 9001 setzt den Treiber in den cbreak-Mode, während das Backend für den Drucker 9004 im raw-Mode arbeitet und selbst für die richtige Parität der Zeichen sorgt.

5.2.5 Beispiele

5.2.5.1 Backend für den Drucker 9001

Das folgende Quellprogramm finden Sie auf der Diskette 'SSHB' unter dem Dateinamen backend01.c.

```
static char SCCSID[] = "@(#)backend01.c 1.5 85/10/10";

/* Mods:
 *      m01     franz   1. Aufrufargumente von links nach rechts verarbeiten.
 *                      2. Drucker nur auf WARTET (statt GESPERRT) in reset.
 *                      3. Weiteren alarm-Aufruf eingefuegt.
 *
 *      m02     franz   1. Vor umsetzen() weiteren alarm-Aufruf eingebaut.
 *                      2. Nach Ausgabe von FF wird stdin/stdout mit
 *                         TIOCFLUSH behandelt (in Routine reset()).
 *
 *      m03     franz   Tabulatoren werden erst nach Auswertung der Schalter
 *                      gesetzt.
 */

/*                                                                      */
/* Einfaches Backend fuer den Drucker 9001, das sich leicht an einen anderen  */
/* Drucker anpassen laesst. Die Moeglichkeit des Bit-Image-Modes wird nicht   */
/* unterstuetzt.                                                        */
/* Dieses Backend setzt den Treiber in den CBREAK-Mode und ueberlaesst ihm    */
/* das XON-XOFF-Protokoll.                                              */
/* Falls der Drucker ein ETX-ACK-Protokoll unterstuetzt, kann mit Hilfe der   */
/* Funktion "pruefe()" ueberprueft werden, ob der Drucker angeschlossen ist   */
/* und ordnungsgemaess arbeitet.                                        */
/* Andernfalls muss man anstelle von "pruefe()" eine leere Funktion einsetzen.*/

# include <stdio.h>
# include <sgtty.h>
# include <signal.h>
# define LF '\012'      /* Line Feed Zeichen           */
# define FF '\014'      /* Form Feed Zeichen           */
# define CR '\015'      /* Carriage Return Zeichen     */
# define ESC '\033'     /* Escape Zeichen              */
# define ZEIT 100       /* Maximale Zeit, die der Drucker fuer 1000 Zeichen  */
                        /* benoetigen darf.                                  */

int npcol = 80 ;        /* Anzahl Spalten pro Zeile     */
int nplin = 72 ;        /* Anzahl Zeilen  pro Seite     */
int npab  =  1 ;        /* Der Text wird ab  Seite npab  gedruckt */
int npbis = 20000 ;     /* Der Text wird bis Seite npbis gedruckt */

char  *dt, *asc, *pb1, *pb2, *pb3 , *auftrag ;
int   reset(), umsetzen(), druckstr() ;

main(argc,argv)
int argc;
char *argv[];
{
        extern int npcol, nplin, npab, npbis ;
        extern char *dt, *asc, *pb1, *pb2, *pb3 , *auftrag ;
        FILE *fpein ;
        char cnc , *tabsetz ;
        int  nc, i ;                                    /* m01 */
```

```
        /* Escape Seqenzen fuer den Drucker */
        dt       = "\033(K" ;                  /* Deutscher Zeichensatz   */
        asc      = "\033(B" ;                  /* ASCII Zeichensatz       */
        pb1      = "\033[1w" ;                 /* Schreibschritt 1/10     */
        pb2      = "\033[2w" ;                 /* Schreibschritt 1/12     */
        pb3      = "\033[4w" ;                 /* Schreibschritt 1/17     */
        tabsetz  = "\033[009;017;025;033;041;049;057;065;073;081;089;097q" ;
                                               /* Tabulatoren             */

        /* ############################################################ */
        /*      Dateideskriptor 0 zeigt auf die Geraetedatei (lesend)   */
        /*                      (stdin)                                 */
        /*      Dateideskriptor 1 zeigt auf die Geraetedatei (schreibend) */
        /*                      (stdout)                                */
        /*      Dateideskriptor 2 zeigt auf die Geraetedatei            */
        /*      /dev/null       (stderr)                                */
        /*      Dateideskriptor 3 zeigt auf die Datei in                */
        /*      /usr/spool/lpd/stat/s0.D?Q                              */
        /* ############################################################ */


        signal(SIGTERM, reset);         /* Sollte Signal 14 oder 15 gesendet */
        signal(SIGALRM, reset);         /* werden, so verlasse das Backend   */
                                        /* ueber die Funktion reset()        */

        alarm(ZEIT);                                          /* m02 */

        umsetzen() ;                    /* Treiber richtig einstellen        */

        auftrag = argv[argc - 1] ;
        pruefe() ;                      /* Pruefen, ob ein Drucker ange-     */
                                        /* schlossen ist                     */

        for ( i = 1 ; i < argc ; i++ )                        /* m01 */
        {
             if ( *argv[i] == '-' )                           /* m01 */
                  schalter( argv[i] + 1 );   /* Schalter interpretieren * m01 */
             else
             {
                  auftrag = argv[i] ;                         /* m01 */
                  fpein = fopen(auftrag , "r");   /* Datei zum Lesen oeffnen */
             }
        }

        druckstr( tabsetz ) ;          /* Tabulatoren setzen */  /* m03 */

        if ( fpein == NULL )           /* Datei konnte nicht geoeffnet werden */
               exit(1);

        lseek(3 , (long)1 , 0 ) ;
        read( 3 , &cnc , 1 ) ;          /* Anzahl der gewuenschten Kopien    */
        nc = cnc & 0177 ;

        while ( nc-- > 0 )
        {
              /*##############################################*/
               druckorg(fpein);         /* Gebe den zu druckenden Text aus */
              /*##############################################*/
               rewind( fpein ) ;
        }

        exit(0);                       /* Backend ordnungsgemass verlassen    */
}
```

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

umsetzen()                              /* Treiber richtig einstellen  */
{
      static struct sgttyb modus ;      /* siehe CES-Manual Nov. 1984, Teil 2 */
                                        /* S. 1-3                             */
      /* Grundzustand lesen  */
      ioctl( 1, TIOCGETP, &modus ) ;
      modus.sg_ispeed = B9600 ;         /* input  Baudrate  */
      modus.sg_ospeed = B9600 ;         /* output Baudrate  */
                                        /* siehe CES-Manual Nov. 1984, Teil 2  */
                                        /* S. 1-3 bis 1-6 unter TTY(4)         */

      modus.sg_flags = CBREAK | ODDP; /* cbreak-Modus mit ungerader Paritaet */

      ioctl(1 , TIOCSETP, &modus ) ;  /* Einstellung setzen */
}

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

schalter( pschalt )             /* Schalter interpretieren und gegebenenfalls */
                                /* entsprechende Escape-Sequenzen senden      */
char *pschalt ;
{
        extern int npcol, nplin, nline, npab, npzaehl, npbis ;
        extern char *dt, *asc, *pb1, *pb2, *pb3 ;
        int i ;

        if ( strncmp( pschalt , "ab=" , 3 ) == 0 )
                npab = ( i = atoi( pschalt + 3 ) ) > 1 ? i : 1  ;
        else if ( strncmp( pschalt, "bis=" , 4 ) == 0 )
                npbis = (i = atoi( pschalt + 4 ) ) > npab ? i : npbis ;
        else if ( strncmp( pschalt, "pb=", 3 ) == 0 )
                npcol = (i = atoi( pschalt + 3 ) ) > 0 ? i : npcol ;
        else if ( strncmp( pschalt, "pl=", 3 ) == 0 )
                nplin = (i = atoi( pschalt + 3 ) ) > 5 ? i : nplin ;
        else if ( strcmp( pschalt, "dt" ) == 0 )
                druckstr( dt ) ;
        else if ( strcmp( pschalt, "int" ) == 0 )
                druckstr( asc ) ;
        else if ( strcmp( pschalt, "pb1" ) == 0 )
                druckstr( pb1 ) ;
        else if ( strcmp( pschalt, "pb2" ) == 0 )
        {
                npcol = ( npcol == 80 ) ? 96 : npcol ;
                druckstr( pb2 ) ;
        }
        else if ( strcmp( pschalt, "pb3" ) == 0 )
        {
                npcol = ( npcol == 80 ) ? 132 : npcol ;
                druckstr( pb3 ) ;
        }

}

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

reset()                         /* kontrollierter Abbruch des Backends */
{
        static int merk = 0 ;
        char c ;
```

```
        signal(SIGTERM, reset); /* Sollte Signal 14 oder 15 gesendet werden, */
        signal(SIGALRM, reset); /* so rufe die Funktion reset erneut auf.    */

        if ( merk++ == 0 )
        {
                alarm( 30 ) ;           /* Falls keine Daten an den Drucker */
                                        /* gesendet werden koennen, dann    */
                                        /* wird Signal 14 gesendet.         */
                lseek( 3, (long)0, 0 ) ;
                c = '\003' ;                                    /* m01 */
                write( 3, &c, 1 ) ;     /* Auftrag auf 'WARTET' stellen   */

                putchar(FF) ;    /* Versuche, ein Formfeed an den Drucker   */
                                 /* zu senden.                              */
                sleep(5);                                       /* m02 */
        }

        ioctl( 0, TIOCFLUSH, NULL );    /* Alle Zeichen, die */ /* m02 */
        ioctl( 1, TIOCFLUSH, NULL );    /* sich noch in den  */ /* m02 */
                                        /* internen Puffern befinden, werden */
                                        /* 'ausgespuelt'.                    */

        exit( 64 ) ;     /* Exitstatus, der ein Sperren des Druckers bewirkt */
}
/* ------------------------------------------------------------------------ */

druckstr( pstring )   /* Gibt einen String aus  */
char *pstring ;
{
        while ( *pstring != '\0' )
                  putchar( *pstring++ ) ;
}


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

druckorg(fpein)     /* Der zu druckende Text wird eingelesen und ausgegeben */
FILE *fpein ;
{
        extern int npcol, nplin, npab, npbis ;
        int c, i, fflag, ncol = 0,  nline = 1 , nseite = 1 ;
        int j_esc = 100, druckflag = 0 , k = 0 ;

        alarm( ZEIT ) ;                                         /* m01 */
        putchar( CR ) ;
        while ( ( c = getc(fpein) ) != EOF    )
        {
                if ( nseite >= npab )
                        druckflag = 1 ;
                fflag = 0 ;  k++ ;

                if ( j_esc > 1 )        /* das Zeichen hinter einem ESC wird */
                {                       /* nicht untersucht.                 */
                        switch(c)
                        {
                        case LF:
                                nline++ ;
                        case FF:
                                if ( druckflag && ncol > 0 )
                                        putchar( CR ) ;
                                if ( k > 999 )
                                {
```

```
                                        pruefe() ;  /* Drucker ueberpruefen   */
                                        alarm(ZEIT) ;
                                           /* Der Drucker hat ZEIT sekunden   */
                                           /* fuer die naechsten 1000 Zeichen */
                                        k = 0 ;
                                }
                        case CR:
                                ncol = 0 ;
                                break ;
                        case ESC:
                                j_esc = 0 ;
                                break ;
                        case '\t':
                                /* Bei Tabulatorzeichen muss        */
                                /* ncol richtig gezaehlt werden.    */
                                for ( i = 8 ; i <= ncol ; i += 8 )
                                                ;
                                ncol = i ;
                                break ;
                        default:
                                ncol++ ;
                        }
                }
                if ( ncol >  npcol ) continue ;

                if ( nline > nplin || c == FF )
                {
                        fflag = 1 ;
                        lseek( 3, (long)10, 0 );  /* Anzahl fertiger Seiten  */
                        write( 3, &nseite, 2 ) ;  /* bekanntgeben.           */
                        if ( druckflag )
                                putchar( FF ) ;
                        nline = 1 ; nseite++ ;
                }
                if ( nseite > npbis ) break ;

                switch(j_esc)
                {
                case 1:
                        if ( c == '[' || c == '(' )
                               ncol-- ;      /* Das naechste Zeichen gehoert */
                                             /* dann auch zu ESC-Folge       */
                        j_esc = 99 ;
                case 0:
                        j_esc++ ;
                default:
                        if ( fflag != 1 && druckflag )
                               putchar( c ) ;
                        break ;
                }
        }
        putchar( CR ) ;
        if ( fflag != 1 )
              putchar( FF ) ;
        fflush( stdout ) ;
        return ;
}
```

```
/* ************************************************************************ */


/* Die folgenden Funktionen sind nur sinnvoll, wenn der Drucker ein ETX-ACK- */
/* Protokoll unterstuetzt.                                                  */
/* Andernfalls schreibt man einfach :                                       */
/*                      pruefe()                                            */
/*                      {                                                   */
/*                      }                                                   */
/*                                                                          */

# define ETX '\003'       /* End of Text Zeichen          */
# define ACK '\006'       /* Acknowledge Zeichen         */

pruefe()                  /* Ueberprueft die Bereitschaft des Druckers mit   */
                          /* Hilfe des ETX-ACK-Protokolls                    */
{
        char c, antwort = '!' ;
        int mailsend() ;

        lseek( 3, (long)0, 0 ) ;
        c = '\003' ;
        write( 3, &c, 1 ) ;             /* Auftrag auf 'WARTET' stellen     */

        signal(SIGALRM, mailsend);  /* Sollte Signal 14 gesendet werden, */
                                    /* so rufe die Funktion mailsend auf.*/

        alarm( ZEIT );              /* Der Drucker hat ZEIT Sekunden um  */
                                    /* auf das folgende ETX mit ACK zu   */
                                    /* antworten.                        */
        putchar( ETX ) ;
        fflush( stdout ) ;          /* ETX senden                        */

        do
                if ( read( 0, &antwort, 1 ) < 1 )
                        mailsend();
        while (  antwort & 0177 != ACK ) ;

        alarm(0) ;                  /* Alarm abschalten                  */

        lseek( 3, (long)0, 0 ) ;
        c = '\002' ;
        write( 3, &c, 1);           /* Auftrag auf 'LAEUFT' stellen      */

        signal(SIGALRM, reset);     /* Sollte Signal 14 gesendet werden, */
                                    /* so verlasse das Backend in Zukunft */
                                    /* ueber die Funktion reset.         */

        return ;                    /* Drucker ist in Ordnung           */
}
```

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

mailsend()              /* Mail schreiben und Backend beenden  */
{
        extern char *auftrag ;
        char user[25], maildatei[45] ;
        FILE *faus ;
        long zeit, time() ;

        signal(SIGALRM, reset ) ;

        strcpy( maildatei, "/usr/spool/mail/" ) ;
        lseek( 3, (long)68, 0 ) ;
        read( 3, user, 25 ) ;               /* Benutzernamen lesen    */
        strcat( maildatei , user ) ;        /* Pfadname der Maildatei */

        faus = fopen( maildatei, "a");      /* Maildatei eroeffnen    */

        if ( faus != NULL )
        {
             zeit = time(&zeit) ;
             fprintf( faus,"Von %s am %s ", user, gctime( &zeit ) ) ;
             fprintf( faus,"Auftrag `%s' : Drucker sendet kein ACK\n",auftrag);
             fclose( faus ) ;
        }

        reset() ;
}


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

5.2.5.2 Backend für den Drucker 9004

Das folgende Quellprogramm finden Sie auf der Diskette 'SSHB' unter dem Dateinamen backend04.c.

```
static char SCCSID[] = "@(#)backend04.c 1.4 85/10/10";

/* Backend fuer den Drucker 9004 mit Einzelblatt- oder Endlospapiereinzug.     */
/*                                                                             */
/* Einzelblatteinzug:                                                          */
/*  Wenn der 'lpr'-Befehl ohne Filter verwendet werden soll, dann muessen die  */
/*  Schalter -pl=65 und -pb=81 gesetzt werden ( z. B. in  /usr/lib/qconfig ).  */
/*  Der Schalter -pl=65 bewirkt zusaetzlich, dass 2 Leerzeilen am Anfang jeder */
/*  Seite unterdrueckt werden (sinnvoll in Verbindung mit dem 'pr'-Kommando).  */
/*  Wenn Dateien mit 'pr' aufbereitet werden sollen, so muss der Schalter      */
/*              '-l67'          angegeben werden.                              */
/*  Beispiel :  pr -l67 DATEINAME | lpr -pl=65 -pb=81                          */
/*                                                                             */
/*  Falls das Backend in Verbindung mit anderen Software-Produkten (z.B. HIT ) */
/*  verwendet werden soll, sollte der Schalter '-pl=65' nicht gesetzt werden.  */
/*                                                                             */
/* Endlospapiereinzug (unter SINIX 1 0B ) :                                    */
/*  Um eine ueberfluessige Leerseite am Beginn zu vermeiden, sollte der        */
/*  Schalter '-nff' gesetzt werden.                                            */
/*  Unter SINIX 1.0C ist dies nicht mehr noetig !                              */
/*                                                                             */
/* Dieses Backend setzt den Treiber in den RAW-Mode, und fuehrt ein ETX-ACK-   */
/* Protokoll.                                                                  */
/* Es kann ein zusaetzlicher Schalter '-form= ' angegeben werden, der die      */
/* Formularlaenge ( Zeilen pro Seite ) beim Drucker festlegt.                  */
/* In Gegensatz zum Standardbackend wird auch der Schalter '-pb2' ausgewertet  */
/*                                                                             */
/* Version vom 06.09.85                                                        */
/*                                                                             */

# include <stdio.h>
# include <sgtty.h>
# include <signal.h>

# define STX '\002'     /* STX - Zeichen                */
# define ETX '\003'     /* End of Text Zeichen          */
# define ACK '\006'     /* Acknowledge Zeichen          */
# define LF '\012'      /* Line Feed Zeichen            */
# define FF '\014'      /* Form Feed Zeichen            */
# define CR '\015'      /* Carriage Return Zeichen      */
# define SO '\016'      /* Shift Out                    */
# define SI '\017'      /* Shift In                     */
# define ESC '\033'     /* Escape Zeichen               */
# define EM '\031'      /* End of Medium Zeichen        */

# define BLOCK  350     /* normale Blockgroesse                                */
# define MBLOCK 420     /* maximale Blockgroesse                               */
# define ZEIT   100     /* Maximale Zeit, die der Drucker fuer MBLOCK Zeichen */
                        /* benoetigen darf.                                    */

int npcol =136 ;        /* Anzahl Spalten pro Zeile              */
int nplin = 72 ;        /* Anzahl Zeilen  pro Seite              */
int npab  =  1 ;        /* Der Text wird ab  Seite npab  gedruckt */
int npbis = 20000 ;     /* Der Text wird bis Seite npbis gedruckt */
int asf = 1 ;           /* Flagge fuer ASF-Anschluss             */
```

```
char *blattaus, *gzust, *bidi_ein, *bidi_aus, *pb2, *statab, drubuff[BLOCK + 100], formul[7] ;
int  nnbuff = 0, reset(), umsetzen() ;

main(argc,argv)
int argc;
char *argv[];
{
        FILE *fpein ;
        char cnc , *tabsetz , c ;
        int  nc, i ;

        /* Escape Seqenzen fuer den Drucker */
        blattaus = "\033\031J" ;              /*  Blattauswurf fuer ASF   */
        gzust    = "\033\015P" ;              /*  Grundzustand            */
        tabsetz  = "\0331" ;                  /*  Tabulator setzen        */
        bidi_ein = "\033/" ;                  /*  Bidirektionaldruck ein  */
        bidi_aus = "\033\\" ;                 /*  Bidirektionaldruck aus  */
        pb2      = "\033H011" ;               /*  12 Zeichen pro Zoll     */
        statab   = "\033\0321" ;              /*  Druckerstatus abfragen  */

        /* ############################################################ */
        /*      Dateideskriptor 0 zeigt auf die Geraetedatei (lesend)   */
        /*                      (stdin)                                 */
        /*      Dateideskriptor 1 zeigt auf die Geraetedatei (schreibend) */
        /*                      (stdout)                                */
        /*      Dateideskriptor 2 zeigt auf die Geraetedatei            */
        /*      /dev/null       (stderr)                                */
        /*      Dateideskriptor 3 zeigt auf die Datei in                */
        /*      /usr/spool/lpd/stat/s0.D?Q                              */
        /* ############################################################ */

        signal(SIGTERM, reset);         /* Sollte Signal 14 oder 15 gesendet */
        signal(SIGALRM, reset);         /* werden, so verlasse das Backend   */
                                        /* ueber die Funktion reset()        */
        lseek( 3, (long)0, 0 ) ;
        c = '\003' ;
        write( 3, &c, 1 ) ;             /* Auftrag auf 'WARTET' stellen      */

        alarm(ZEIT) ;
        umsetzen( 1 ) ;                 /* Treiber richtig einstellen        */

        drustr( gzust ) ;
        drustr( statab ) ;              /* Status abfragen                   */

        pruefe() ;                      /* Pruefen, ob ein Drucker ange-     */
                                        /* schlossen ist                     */
        lseek( 3, (long)0, 0 ) ;
        c = '\002' ;
        write( 3, &c, 1 ) ;             /* Auftrag auf 'LAEUFT' stellen      */

        for ( i = 1 ; i < argc ; i++ )
        {
             if ( *argv[i] == '-' )
                  schalter( argv[i] + 1 ) ;       /* Schalter interpretieren */
             else
                  fpein = fopen(argv[i] , "r");   /* Datei zum Lesen oeffnen */
        }

        if ( fpein == NULL )          /* Datei konnte nicht geoeffnet werden */
               exit(1);
```

```
        /* Tabulatoren setzen   */
        for ( i = 1 ; i <= 12 ; i++ )
                drustr("        \0331") ;

        druchar( CR ) ;

        lseek(3 , (long)1 , 0 ) ;
        read( 3 , &cnc , 1 ) ;          /*   Anzahl der gewuenschten Kopien    */
        nc = cnc & 0177 ;

        while ( nc-- > 0 )
        {
              /*##############################################*/
               druckorg(fpein);         /* Gebe den zu druckenden Text aus     */
              /*##############################################*/
               rewind( fpein ) ;
        }
        drustr( gzust ) ;
        pruefe() ;
        umsetzen(0) ;
        exit(0);                        /* Backend ordnungsgemass verlassen    */
}

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

umsetzen( i )                                   /* Treiber richtig einstellen   */
int i ;
{
      static struct sgttyb modus, savmodus ; /* siehe CES-Manual Nov. 1984,  */
                                             /* Teil 2, S. 1-3               */
      if ( i == 1 )
      {
           /* Grundzustand lesen  */
           ioctl( 1, TIOCGETP, &savmodus ) ;
           ioctl( 1, TIOCGETP, &modus ) ;
           modus.sg_ispeed = B1200 ;  /* input  Baudrate  */
           modus.sg_ospeed = B1200 ;  /* output Baudrate  */
                                      /* siehe CES-Manual Nov. 1984, Teil 2  */
                                      /* S. 1-3 bis 1-6 unter TTY(4)         */

           modus.sg_flags = RAW ;     /* RAW-Modus  */

           ioctl(1 , TIOCSETP, &modus ) ;    /* Einstellung setzen      */
      }
      else if ( i == 0 )
           ioctl(1 , TIOCSETP, &savmodus ) ; /* Grundeinstellung setzen */
}
/* ------------------------------------------------------------------------ */

schalter( pschalt )                           /* Schalter interpretieren  */
char *pschalt ;
{
```

```
        int i ;

        if ( strncmp( pschalt , "ab=" , 3 ) == 0 )
                npab = ( i = atoi( pschalt + 3 ) ) > 1 ? i : 1  ;
        else if ( strncmp( pschalt, "bis=" , 4 ) == 0 )
                npbis = (i = atoi( pschalt + 4 ) ) >= npab ? i : npbis ;
        else if ( strncmp( pschalt, "pb=", 3 ) == 0 )
                npcol = (i = atoi( pschalt + 3 ) ) > 0 ? i : npcol ;
        else if ( strncmp( pschalt, "pl=", 3 ) == 0 )
                nplin = (i = atoi( pschalt + 3 ) ) > 5 ? i : nplin ;
        else if ( strcmp( pschalt, "nff" ) == 0 )
                asf = 0 ;
        else if ( strcmp( pschalt, "pb2" ) == 0 )
                drustr( pb2 ) ;
        else if ( strncmp( pschalt, "form=", 5 ) == 0 )
        {
                i = atoi( pschalt + 5 ) ;
                if ( i > 2 && i < 127 )
                {
                        nplin = i ;
                        sprintf( formul, "\033F%03d" , i  )  ;
                        drustr( formul ) ;
                }
        }
}

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

reset()                         /* kontrollierter Abbruch des Backends      */
{
        static int merk = 0 ;
        char c ;

        signal(SIGTERM, reset); /* Sollte Signal 14 oder 15 gesendet werden, */
        signal(SIGALRM, reset); /* so rufe die Funktion reset erneut auf.    */

        if ( merk++ == 0 )
        {
                alarm( 20 ) ;           /* Falls keine Daten an den Drucker */
                                        /* gesendet werden koennen, dann    */
                                        /* wird Signal 14 gesendet.         */
                lseek( 3, (long)0, 0 ) ;
                c = '\003' ;
                write( 3, &c, 1 ) ;     /* Auftrag auf 'WARTET' stellen     */

                if ( asf )
                        drustr( blattaus ) ;
                else
                        druchar( FF ) ;
                drustr( gzust ) ;
                druflush() ;
                sleep( 5 ) ;

                ioctl( 0, TIOCFLUSH, NULL ) ;  /* Alle Zeichen, die sich noch in den */
                ioctl( 1, TIOCFLUSH, NULL ) ;  /* internen Puffern befinden, werden  */
                                               /* 'ausgespuelt'.                     */
                umsetzen(0) ;
        }

        exit( 64 ) ;     /* Exitstatus, der ein Sperren des Druckers bewirkt */
}
```

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

druckorg(fpein)      /* Der zu druckende Text wird eingelesen und ausgegeben */
FILE *fpein ;
{
        int c0 = ' ', c1, c2, i, lfflag = 0, ncol = 0,  nline = 1 , nseite = 1 ;
        int j_esc = 100, druckflag = 0 ;

        pruefe() ;

        if ( asf )
                druchar( FF ) ;

        for ( c1 = getc(fpein) ; c1 != EOF ; c1 = c2  )
        {
                c2 = getc( fpein ) ;

                if ( nseite >= npab )
                        druckflag = 1 ;
                lfflag++ ;

                if ( j_esc > 10 )       /* die Zeichen einer ESC-Folge werden */
                {                       /* nicht untersucht.                  */

                if ( nnbuff > BLOCK )   /* Drucker ueberpruefen       */
                      pruefe() ;        /* Datenblock ausgeben        */

                switch(c1)
                {
                case LF:
                        if ( nplin == 65 && lfflag < 3 && ncol == 0 )
                                continue ;

                        if ( nline++ == 1 && druckflag )
                                drustr( bidi_ein ) ;
                case FF:
                        if ( druckflag )
                        {
                                if ( nline >= nplin || c1 == FF )
                                        drustr( bidi_aus ) ;
                                if ( ncol > 0 )
                                        druchar( CR ) ;
                        }
                case CR:
                        ncol = 0 ;
                        break ;
                /* -------------------------  */
                case ESC:
                        j_esc = 0 ;
                case SO:
                case SI:
                        break ;
                case '\b':
                        ncol-- ;
                        break ;
                case '\t':
                        /* Bei Tabulatorzeichen muss        */
                        /* ncol richtig gezaehlt werden.    */
                        for ( i = 8 ; i <= ncol ; i += 8 )
                                        ;
```

```
                ncol = i ;
                break ;
        default:
                ncol++ ;
        }
}
if ( ncol >  npcol ) continue ;

if ( nline > nplin || c1 == FF )
{
        if ( c2 != EOF )
                c1 = FF ;
        else
                c1 = CR ;
        lfflag = 0 ;
        lseek( 3, (long)10, 0 );   /* Anzahl fertiger Seiten  */
        write( 3, (char *)&nseite, 2 ) ;  /* bekanntgeben.    */
        nline = 1 ; nseite++ ;
}
if ( nseite > npbis ) break ;

switch(j_esc)
{
case 4:
       j_esc = 99 ;
case 3:
case 2:
       j_esc++ ;
       if ( druckflag )
              druchar(c1) ;
       switch( c0 )
       {
       case EM:
              if ( c1 != 'R' )
              {
                   lseek( 3, (long)10, 0 ); /* Anzahl fertiger Seiten */
                   write( 3, (char *)&nseite, 2 ) ; /*  bekanntgeben. */
                   nseite++ ;  nline = 1 ; ncol = 0 ;
              }
       case CR:
              j_esc = 100 ;
              break ;
       case 'F':
       case 'S':
       case 'T':
       case 'H':
       case 'V':
              if ( c1 < '0' || c1 > '9' )
                     j_esc = 100 ;
       }
       break ;
    /* ------------------------------------ */
case 1:
       if (c1==EM||c1==CR||c1=='F'||c1=='S'||c1=='T'||c1=='H'||c1=='V')
             c0 = c1 ;
       else
             j_esc = 99 ;
```

```
                case 0:
                        j_esc++ ;
                default:
                        if ( druckflag )
                                druchar( c1 ) ;
                        break ;
                }
        }
        druchar( CR ) ;
        pruefe() ;
        if ( asf )
                drustr( blattaus ) ;
        else
                druchar( FF ) ;
        return ;
}

/* ************************************************************************** */

pruefe()                /* Ueberprueft die Bereitschaft des Druckers mit   */
                        /* Hilfe des ETX-ACK-Protokolls                    */
{
        int i = 10 ;
        char c, antwort , merk = ' ' ;

        alarm( ZEIT );              /* Der Drucker hat ZEIT Sekunden um    */
                                    /* auf das folgende ETX mit ACK zu     */
                                    /* antworten.                          */
        druchar( ETX ) ;            /* ETX an das Ende des Buffers schreiben */
        druflush( ) ;               /* Buffer ausgeben                     */

        do
        {
                if ( read( 0, &c, 1 ) < 1 )
                            reset();
                antwort = c & 0177 ;
                if ( antwort == STX )
                        i = 1 ;
                else if ( antwort == ACK )
                        merk = antwort ;

                switch(i)
                {
                case 3:
                        asf = antwort & 2 ;
                        i = 10 ;
                        break ;
                case 2:
                case 1:
                        i++ ;
                }
        }
        while (  merk != ACK || rdchk( 0 ) > 0 ) ;

        alarm(0) ;                  /* Alarm abschalten                    */

        return ;                    /* Drucker ist in Ordnung              */
}
```

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

druchar( c )
char c ;
{
        drubuff[ nnbuff++ ] = paritaet( c ) ;
        if ( nnbuff > MBLOCK )
                pruefe() ;
}

druflush()
{
        if ( write( 1, drubuff, nnbuff ) < nnbuff )
                reset() ;
        nnbuff = 0 ;
}
drustr( pstr )
char *pstr ;
{
        while ( *pstr != '\0' )
                druchar( *pstr++ ) ;
}

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

paritaet( c )    /* erzwingt ungerade Paritaet der Zeichen */
char c ;
{
        static int eins = 1 ;
        int bitzaehl, i ;

        for ( bitzaehl = 0 , i = 0 ; i <  7 ; i++ )
                if ( c & ( eins << i )  )
                        bitzaehl++ ;

        if ( bitzaehl%2 == 0 )    /* Falls Paritaet gerade, dann */
                     c |= 0200 ;  /* das hoechste Bit setzen     */

        return( c & 0377 ) ;
}
```

6 DÜ-Schnittstelle PC-X, PC-MX

6.1 Allgemeines

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:

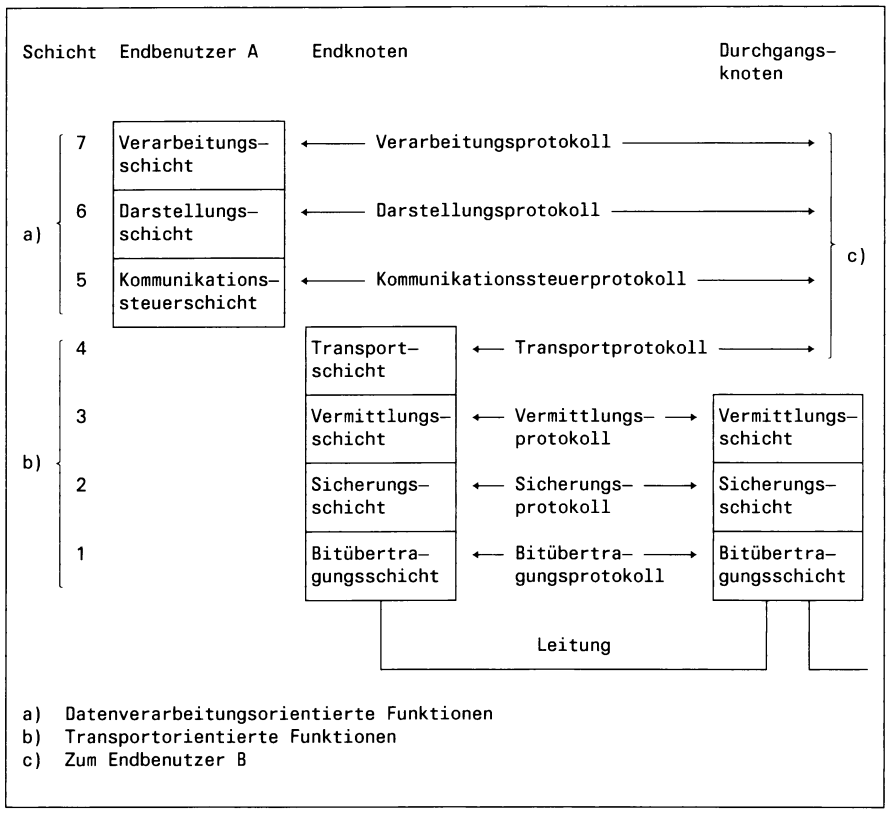


Bild 6-1 ISO-7-Schichten-Modell

6.1.1 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.

6.2 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).

6.2.1 Beschreibung der einzelnen Leitungen

6.2.1.1 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.

6.2.1.2 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.

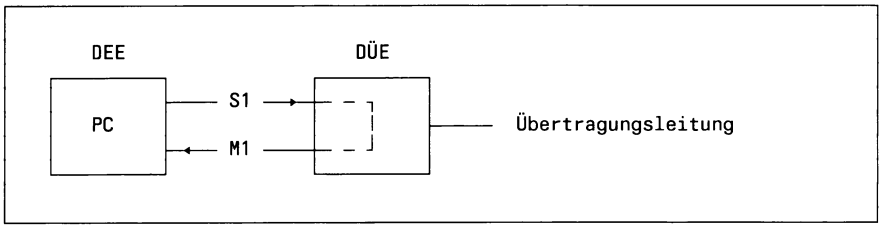


Bild 6-2 Standleitung

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.

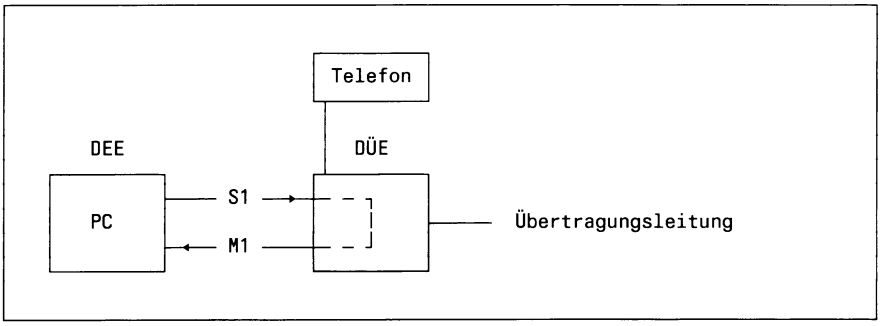


Bild 6-3 Wahlleitung

6.2.1.3 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

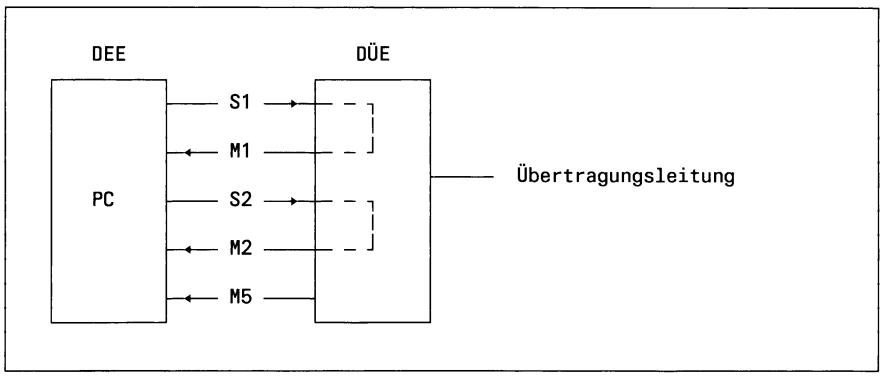


Bild 6-4 Duplex-Betrieb

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.

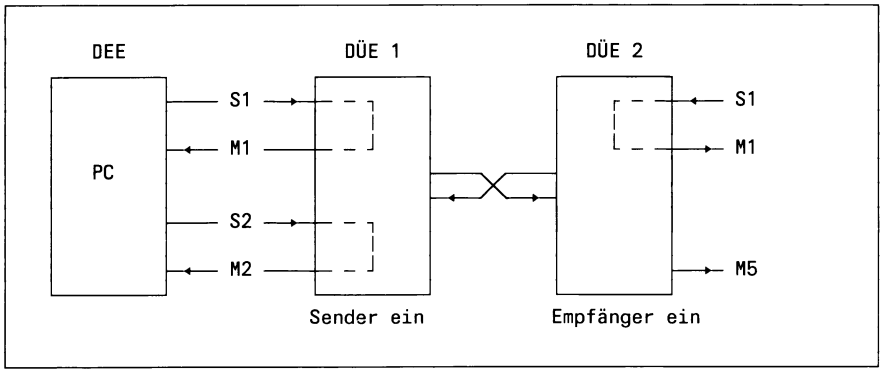


Bild 6-5 Halbduplex-Betrieb

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.

6.2.1.4 Daten D1, D2

D1 Sendedaten vom Rechner
D2 Empfangsdaten zum Rechner

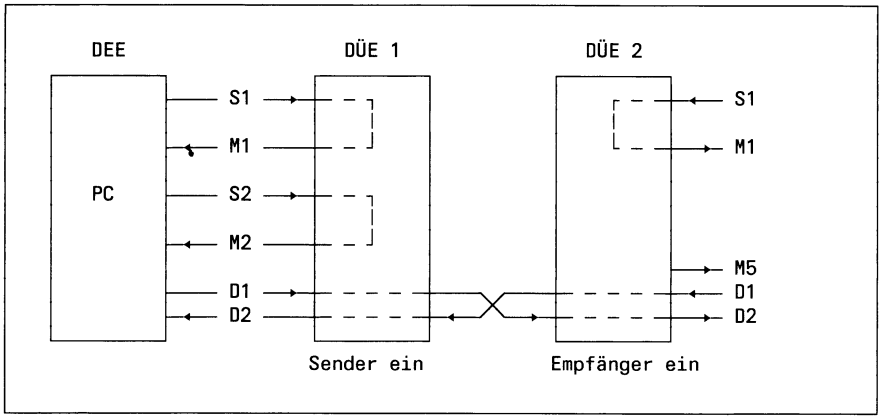


Bild 6-6 Daten D1, D2

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.

6.2.1.5 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

6.3 Die Übertragungprozedur MSV1

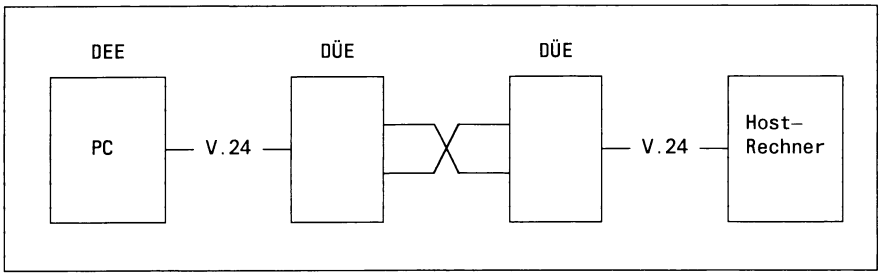


Bild 6-7 Übertragungsprozedur MSV1

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-MX 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.

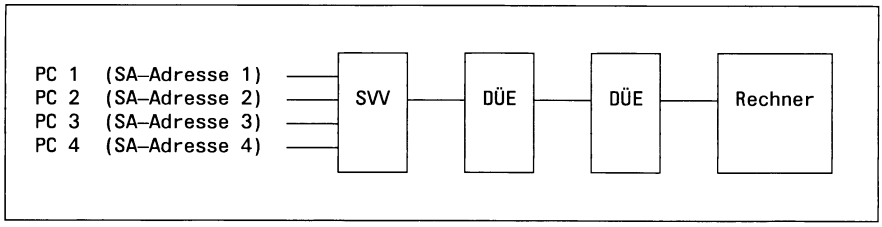


Bild 6-8 Schnittstellenvervielfacher (SVV)

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.

6.4 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.

6.5 Technische Daten

Technische Daten SINIX 1.0B (ohne Kommunikations-Prozessor)

Synchrone Übertragung, normierter Modus	: 3 oder 7 SYN
Übertragungs-geschwindigkeit	: 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)

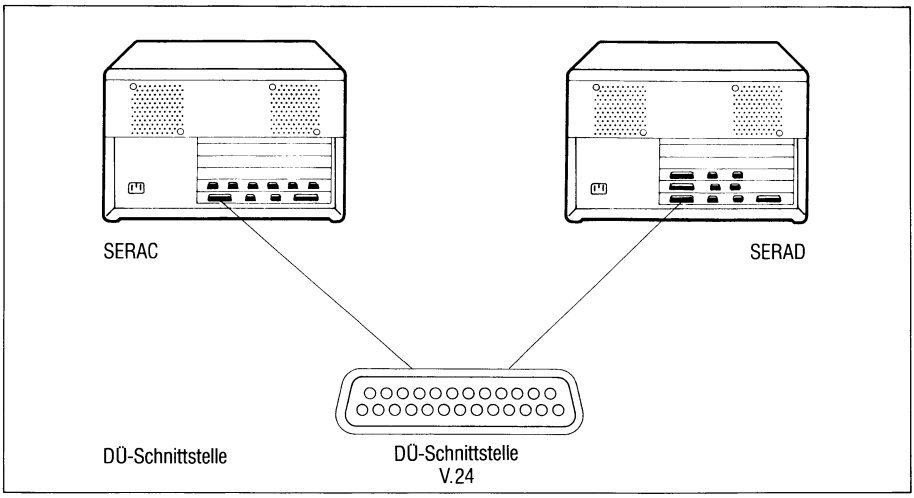


Bild 6-9 Anschlußfeld des PC-MX

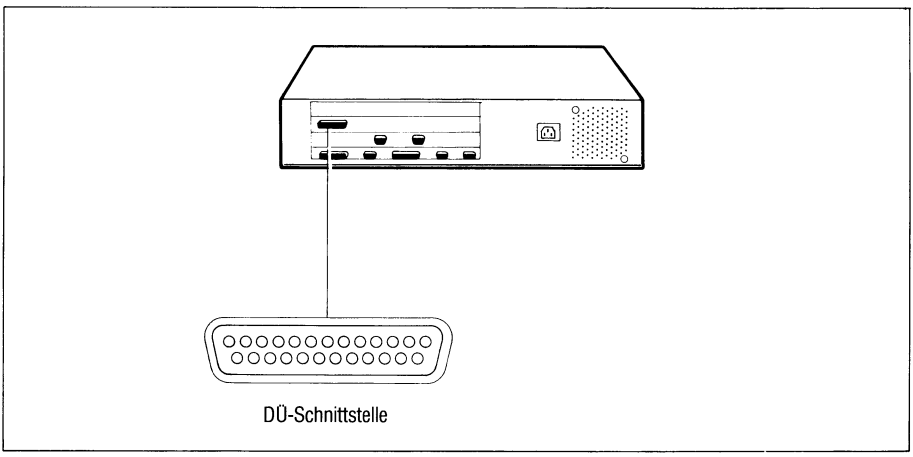


Bild 6-10 Anschlußfeld des PC-X

7 Behandlung von Disketten beim PC-X und PC-MX

7.1 Allgemeines

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.

7.2 Gerätedateien

Außer den eben genannten Standard-Geräteeinträgen, gibt es vier weitere Einträge:

`/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	2	2	2	2	1	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

Um dies realisieren zu können, müssen folgende Geräteeinträge vorhanden und mit den richtigen Schutzbits vorbelegt 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
```

Der Systemverwalter kann mit folgenden Kommandos diese Geräteeinträge 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 der vorher genannten 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 sie von einem anderen Anwender verändert werden oder das System abgestellt wird.

7.3 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).

7.3.1 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);
}
```

7.3.2 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).

7.3.3 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).

7.4 Disketten lesen

7.4.1 Allgemeines

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. Für MS-DOS-Disketten ist ein solches Programm in Bearbeitung.

Bemerkungen

Es gibt ein Programm 'flmsd', das physikalisch lesbare MS-DOS-Disketten logisch aufbereitet (siehe 7.4.2.1).

Für den Siemens PC-D gibt es das Dienstprogramm MTAR, das einen Dateiaustausch MS-DOS ⟷ SINIX ermöglicht. MTAR basiert auf dem tar-Format, und erlaubt es PC-D-Anwendern u.a., Disketten im tar-Format zu beschreiben, die dann unter SINIX mit dem tar-Kommando (bzw. far-Kommando) gelesen werden können. MTAR wird ca. Mitte des Jahres 1985 freigegeben.

Bezüglich der erwähnten Programme wenden Sie sich bitte bei Bedarf an Ihre zuständige Siemens-Zweigniederlassung.

Die Beschreibungen der Dienstprogramme beziehen sich auf den aktuellen Hard- und Softwarestand (Februar '85). Änderungen vorbehalten.

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.

7.4.2 Vorgehen bei MS-DOS-Disketten

Empfehlung für Eilige

Die folgenden Schritte sollen eine Datei auf MS-DOS-Disketten physikalisch verfügbar machen und unrelevante Daten, die der Diskettenverwaltung dienen, herausfiltern.

Unter MS-DOS

Diskette frisch formatieren, nur **eine** Datei auf der Diskette einspielen.

Unter SINIX

```
cp /dev/flx datei1
```

Diskette in SINIX-Datei kopieren.

```
tail +13b datei1 > datei2
```

Die ersten 12 Blöcke 'wegwerfen' (12 x 512 Bytes) und den Rest in datei2 zwischenspeichern. Dies ist nur sinnvoll, wenn der Datenteil bei Block 13 beginnt.

```
tr -d 'r 66' datei2 > datei3
```

Füllzeichen 0xf6 vom Formatieren löschen und den Rest in der datei3 sichern. Es können auch andere Füllzeichen auftreten, z.B 0xdb beim PC-D.

Die Kommandofolge kann auch zusammengefaßt werden:

```
tail +13b /dev/flx | tr -d 'r366' > datei3
```

Falls die Datei keine Zeilentrenner (0x0a) enthält, so kann sie z.B. nicht mit dem CED bearbeitet werden. Sie sollte daher von einem kleinen Programm aufbereitet werden, das z.B. nach je 80 Zeichen einen Zeilentrenner einfügt. Andere Systeme verwenden als Zeilenendekriterium oft CR + LF (0x0d; 0x0a) und als Dateiendekriterium SUB (0x1a).
Für deutsche Umlaute wird oft ein hex.-Code > X'0x7f' verwendet. Diese Zeichen müssen geändert oder entfernt werden.

7.4.2.1 Programm zur logischen Aufbereitung von MS-DOS-Disketten

Aufrufformat

```
flmsd -schalter quelle [datei...] [dateiverzeichnis...]
```

Beschreibung

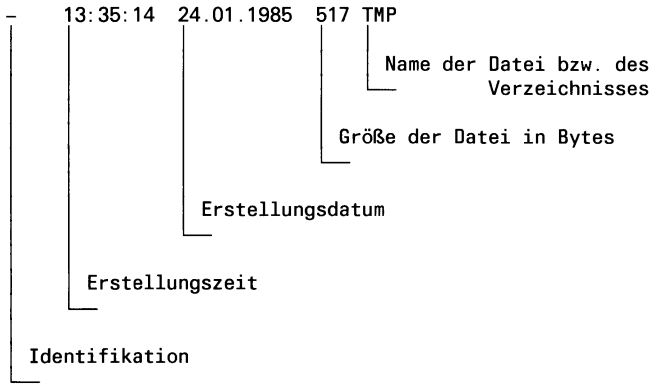
flmsd bereitet SINIX-Dateien, in die MS-DOS-Disketten kopiert wurden (cp /dev/flx quelle), entsprechend dem logischen MS-DOS-Format auf.

Wahl der Schalter:

t flmsd gibt das Inhaltsverzeichnis aus.

u In Verbindung mit Schalter t: flmsd gibt rekursiv auch Unterverzeichnisse aus.

l In Verbindung mit Schalter t oder tu: Vollständige Information ausgeben in der Form:



Bei der Identifikation bedeutet:

— normale Datei

d Unterdateiverzeichnis

f Kopieren der angegebenen Dateien oder Dateiverzeichnisse. Dabei gibt es folgende Möglichkeiten: Gibt man nur Schalter f und die Quelldatei an, so werden alle Dateien und rekursiv auch alle Unterverzeichnisse kopiert (flmsd -f quelle). Sollen einzelne Dateien oder Verzeichnisse kopiert werden, so müssen diese mit ihrem vollständigen Pfadnamen angegeben werden, wie er von 'flmsd -tu quelle' ausgegeben wird. Zu beachten: Bei der Angabe des zu kopierenden Verzeichnisses DIR/DIR/DIR/DIR/LETZTDIR wird nur das SINIX-Verzeichnis LETZTDIR erzeugt.

flmsd verarbeitet MS-DOS-Formate des Siemens PC-D, des IBM-PC (single/double sided, 8/9 Sektoren/Spur) und Formate, die mit diesen übereinstimmen. Falls das Format nicht eindeutig bestimmt werden konnte, werden Default-Werte gesetzt. Sollte ein dem Programm nicht bekanntes Format verwendet werden, das prinzipiell im MS-DOS Format aufgebaut ist, so können die erforderlichen Parameter selbst gesetzt werden (Funktion z). Es ist dabei genaue Kenntnis des Formats notwendig (eventuell mit xd quelle analysieren).

y flmsd gibt die gesetzten Parameter aus.

z flmsd fordert zur Eingabe eigener Parameter auf.

Die Parameterwerte werden nicht gespeichert, sodaß bei jedem Aufruf eine Neubesetzung notwendig ist. Ein Einlesen der Parameter aus einer Datei erleichtert die Arbeit: flmsd -zt quelle < paramfile.

Bemerkungen

Der Inhalt von Dateien wird unverändert übernommen, also auch für SINIX u.U. unbrauchbare Zeichen (z.B. schließt der edlin des Siemens PC-D jede Zeile mit CR (0x0d) und LF (0x0a) ab), CR ist für SINIX durch tr -d ' 015' d dateialt > dateineu aus der Datei entfernen).

Änderung in Vorbereitung.

8 Kopplung

8.1 Allgemein

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-MX (SINIX 1.0B)

Die am PC-MX (Fbg. CONAC, siehe 2-1) verfügbare Schnittstelle RS232 ist vom Hardware-Design nur als Ausgabeschnittstelle (für Drucker) ausgelegt. Sollen über diese Schnittstelle Daten eingelesen werden, so sind folgende Einschränkungen zu beachten:

- keine Flußsteuerung für Eingabe, daher langsame Eingabegeschwindigkeit,
- auf Anwenderebene überprüfen, ob alles angekommen ist.

Bemerkungen zum PC-MX (SINIX 1.0C)

Mit SINIX 1.0C steht ein neuer Ein-/Ausgabeprozessor zur Verfügung, der statt des vorhandenen Ein-/Ausgabeprozessors oder zusätzlich gesteckt werden kann.
Dieser E/A-Prozessor hat 4 Schnittstellen SS97 und 2 Schnittstellen RS232. Bei diesen Schnittstellen entfallen die oben genannten Einschränkungen bei Übertragungsgeschwindigkeiten bis 19200 bit/s.
Bestell-Nr. dieses E/A-Prozessors: 97802-202
Die Basis-E/A-Schnittstelle (Fbg. CONAC) kann mit Flußsteuerung bis 4800 bit/s ohne Einschränkungen betrieben 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.

8.2 Kabel

Erfolgt eine Kopplung über die Schnittstelle RS232, so ist ein sog. Nullmodem erforderlich.

Beispiel:

Kabel für Schnittstelle RS232

PC-MX Schalterstellung Teil 2 beachten! Nur S4 auf Stellung 2
Fremd-PC Vorschriften beachten!

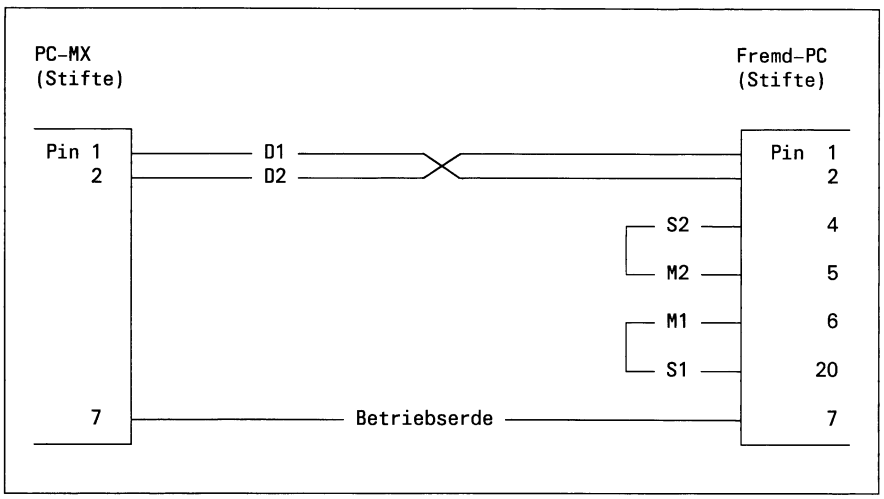


Bild 8-1 Nullmodem

Erfolgt eine Kopplung über die Schnittstelle SS97, so ist folgendes Kreuzungskabel erforderlich:

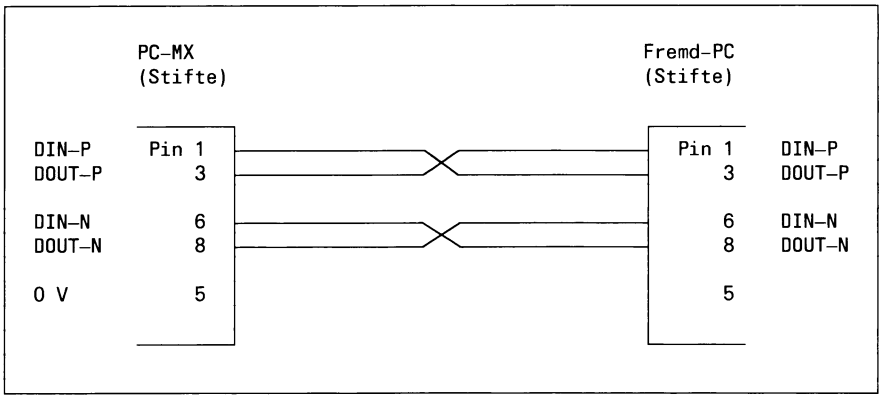


Bild 8-2 Kreuzungskabel für SS97

8.3 Beispiele

8.3.1 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
```

8.3.2 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>
#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                 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); }

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

8.3.3 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);
}
```

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!

A.1 Inhaltsverzeichnis der Diskette

Folgende Dateien sind auf der mitgelieferten Diskette 'SSHB' enthalten:

```
rwxr-xr-x  3/3     9888 Oct 10 11:19 1985 backend01
r--r--r--  3/3    13084 Oct 10 11:17 1985 backend01.c
rwxr-xr-x  3/3     8438 Oct 10 11:20 1985 backend04
r--r--r--  3/3    14109 Oct 10 11:17 1985 backend04.c
r--r--r--  3/3     4818 Jun 13 16:39 1985 c_get.c
rw-r--r--  3/3     1416 Jun 13 16:39 1985 c_get.o
r-xr-xr-x  3/3      173 May 23 14:50 1985 c_make
rwxr-xr-x  3/3     9400 Jun 13 16:39 1985 c_test
r--r--r--  3/3     2349 May 23 14:50 1985 c_test.c
rw-r--r--  3/3     2432 May 23 14:52 1985 c_test.o
rwxr-xr-x  3/3     6334 Jun 13 16:35 1985 chcode
r--r--r--  3/3     3224 Jun 13 16:34 1985 chcode.c
rwxr-xr-x  3/3     7920 May  7 18:54 1985 eing
r--r--r--  3/3     1879 May  7 18:53 1985 eing.c
rwxr-xr-x  3/3     6540 May  2 17:32 1985 fl6parget
r--r--r--  3/3     1502 May  2 17:31 1985 fl6parget.c
rwxr-xr-x  3/3     1972 May  2 17:33 1985 fl6parset
r--r--r--  3/3     1598 May  2 17:31 1985 fl6parset.c
r-xr-xr-x  3/3     6069 Jun 13 16:37 1985 getrs232
rwxr-xr-x  3/3     9500 Jun 13 16:43 1985 gettty
r--r--r--  3/3    12135 Jun 13 16:42 1985 gettty.c
r--r--r--  3/3     2601 May  7 18:34 1985 tbuild.c
rw-r--r--  3/3      984 May  7 18:35 1985 tbuild.o
r--r--r--  3/3     9232 Oct  9 14:18 1985 termio.h
r-xr-xr-x  3/3      156 May  3 11:28 1985 tmake
rwxr-xr-x  3/3     8274 May  7 18:35 1985 txample
r--r--r--  3/3     4641 May  7 18:34 1985 txample.c
rw-r--r--  3/3     2536 May  7 18:35 1985 txample.o
rwxr-xr-x  3/3     7936 Jun 13 16:44 1985 zed
r--r--r--  3/3    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	Diese Dateien entsprechen in etwa den Dateien tbuild.c,....,tmake, basieren aber auf verbesserten Routinen zur Verarbeitung von Eingabesequenzen (c_init, c_get).
c_get.o	
c_test.c	
c_test.o	
c_test	
c_make	
chcode.c	Quellprogramm zum Setzen der CH-CODE-Taste
chcode	daraus erzeugtes lauffähiges Programm
backend01.c	Quellprogramm eines einfachen Backends für den Drucker 9001
backend01	daraus erzeugtes lauffähiges Programm
backend04.c	Quellprogramm eines einfachen Backends für den Drucker 9004
backend04	daraus erzeugtes lauffähiges Programm
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
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   "l0"

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.
 */


      * 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 backend01.c

```
static char SCCSID[] = "@(#)backend01.c 1.5 85/10/10";

/* Mods:
 *      m01     franz   1. Aufrufargumente von links nach rechts verarbeiten.
 *                      2. Drucker nur auf WARTET (statt GESPERRT) in reset.
 *                      3. Weiteren alarm-Aufruf eingefuegt.
 *
 *      m02     franz   1. Vor umsetzen() weiteren alarm-Aufruf eingebaut.
 *                      2. Nach Ausgabe von FF wird stdin/stdout mit
 *                         TIOCFLUSH behandelt (in Routine reset()).
 *
 *      m03     franz   Tabulatoren werden erst nach Auswertung der Schalter
 *                      gesetzt.
 */

/*                                                                      */
/* Einfaches Backend fuer den Drucker 9001, das sich leicht an einen anderen  */
/* Drucker anpassen laesst. Die Moeglichkeit des Bit-Image-Modes wird nicht   */
/* unterstuetzt.                                                        */
/* Dieses Backend setzt den Treiber in den CBREAK-Mode und ueberlaesst ihm    */
/* das XON-XOFF-Protokoll.                                              */
/* Falls der Drucker ein ETX-ACK-Protokoll unterstuetzt, kann mit Hilfe der   */
/* Funktion "pruefe()" ueberprueft werden, ob der Drucker angeschlossen ist   */
/* und ordnungsgemaess arbeitet.                                        */
/* Andernfalls muss man anstelle von "pruefe()" eine leere Funktion einsetzen.*/

# include <stdio.h>
# include <sgtty.h>
# include <signal.h>
# define LF '\012'      /* Line Feed Zeichen           */
# define FF '\014'      /* Form Feed Zeichen           */
# define CR '\015'      /* Carriage Return Zeichen     */
# define ESC '\033'     /* Escape Zeichen              */
# define ZEIT 100       /* Maximale Zeit, die der Drucker fuer 1000 Zeichen  */
                        /* benoetigen darf.                                  */

int npcol = 80 ;        /* Anzahl Spalten pro Zeile     */
int nplin = 72 ;        /* Anzahl Zeilen  pro Seite     */
int npab  =  1 ;        /* Der Text wird ab  Seite npab  gedruckt */
int npbis = 20000 ;     /* Der Text wird bis Seite npbis gedruckt */

char  *dt, *asc, *pb1, *pb2, *pb3 , *auftrag ;
int   reset(), umsetzen(), druckstr() ;

main(argc,argv)
int argc;
char *argv[];
{
        extern int npcol, nplin, npab, npbis ;
        extern char *dt, *asc, *pb1, *pb2, *pb3 , *auftrag ;
        FILE *fpein ;
        char cnc , *tabsetz ;
        int  nc, i ;                                    /* m01 */
```

```
/* Escape Seqenzen fuer den Drucker */
dt       = "\033(K" ;                  /* Deutscher Zeichensatz   */
asc      = "\033(B" ;                  /* ASCII Zeichensatz       */
pb1      = "\033[1w" ;                 /* Schreibschritt 1/10     */
pb2      = "\033[2w" ;                 /* Schreibschritt 1/12     */
pb3      = "\033[4w" ;                 /* Schreibschritt 1/17     */
tabsetz  = "\033[009;017;025;033;041;049;057;065;073;081;089;097q" ;
                                       /* Tabulatoren             */

/* ############################################################ */
/*      Dateideskriptor 0 zeigt auf die Geraetedatei (lesend)   */
/*                      (stdin)                                 */
/*      Dateideskriptor 1 zeigt auf die Geraetedatei (schreibend) */
/*                      (stdout)                                */
/*      Dateideskriptor 2 zeigt auf die Geraetedatei            */
/*      /dev/null       (stderr)                                */
/*      Dateideskriptor 3 zeigt auf die Datei in                */
/*      /usr/spool/lpd/stat/s0.D?Q                              */
/* ############################################################ */


signal(SIGTERM, reset);          /* Sollte Signal 14 oder 15 gesendet */
signal(SIGALRM, reset);          /* werden, so verlasse das Backend   */
                                 /* ueber die Funktion reset()        */

alarm(ZEIT);                                              /* m02 */

umsetzen() ;                     /* Treiber richtig einstellen        */

auftrag = argv[argc - 1] ;
pruefe() ;                       /* Pruefen, ob ein Drucker ange-     */
                                 /* schlossen ist                     */

for ( i = 1 ; i < argc ; i++ )                            /* m01 */
{
     if ( *argv[i] == '-' )                               /* m01 */
          schalter( argv[i] + 1 );   /* Schalter interpretieren * m01 */
     else
     {
          auftrag = argv[i] ;                             /* m01 */
          fpein = fopen(auftrag , "r");   /* Datei zum Lesen oeffnen */
     }
}

druckstr( tabsetz ) ;          /* Tabulatoren setzen */  /* m03 */

if ( fpein == NULL )           /* Datei konnte nicht geoeffnet werden */
        exit(1);

lseek(3 , (long)1 , 0 ) ;
read( 3 , &cnc , 1 ) ;          /* Anzahl der gewuenschten Kopien    */
nc = cnc & 0177 ;

while ( nc-- > 0 )
{
      /*##############################################*/
       druckorg(fpein);        /* Gebe den zu druckenden Text aus */
      /*##############################################*/
       rewind( fpein ) ;
}

exit(0);                       /* Backend ordnungsgemass verlassen    */
}
```

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

umsetzen()                              /* Treiber richtig einstellen  */
{
      static struct sgttyb modus ;      /* siehe CES-Manual Nov. 1984, Teil 2 */
                                        /* S. 1-3                             */
      /* Grundzustand lesen  */
      ioctl( 1, TIOCGETP, &modus ) ;
      modus.sg_ispeed = B9600 ;         /* input  Baudrate  */
      modus.sg_ospeed = B9600 ;         /* output Baudrate  */
                                        /* siehe CES-Manual Nov. 1984, Teil 2  */
                                        /* S. 1-3 bis 1-6 unter TTY(4)         */

      modus.sg_flags = CBREAK | ODDP; /* cbreak-Modus mit ungerader Paritaet */

      ioctl(1 , TIOCSETP, &modus ) ;  /* Einstellung setzen */
}

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

schalter( pschalt )             /* Schalter interpretieren und gegebenenfalls */
                                /* entsprechende Escape-Sequenzen senden      */
char *pschalt ;
{
        extern int npcol, nplin, nline, npab, npzaehl, npbis ;
        extern char *dt, *asc, *pb1, *pb2, *pb3 ;
        int i ;

        if ( strncmp( pschalt , "ab=" , 3 ) == 0 )
                npab = ( i = atoi( pschalt + 3 ) ) > 1 ? i : 1  ;
        else if ( strncmp( pschalt, "bis=" , 4 ) == 0 )
                npbis = (i = atoi( pschalt + 4 ) ) > npab ? i : npbis ;
        else if ( strncmp( pschalt, "pb=", 3 ) == 0 )
                npcol = (i = atoi( pschalt + 3 ) ) > 0 ? i : npcol ;
        else if ( strncmp( pschalt, "pl=", 3 ) == 0 )
                nplin = (i = atoi( pschalt + 3 ) ) > 5 ? i : nplin ;
        else if ( strcmp( pschalt, "dt" ) == 0 )
                druckstr( dt ) ;
        else if ( strcmp( pschalt, "int" ) == 0 )
                druckstr( asc ) ;
        else if ( strcmp( pschalt, "pb1" ) == 0 )
                druckstr( pb1 ) ;
        else if ( strcmp( pschalt, "pb2" ) == 0 )
        {
                npcol = ( npcol == 80 ) ? 96 : npcol ;
                druckstr( pb2 ) ;
        }
        else if ( strcmp( pschalt, "pb3" ) == 0 )
        {
                npcol = ( npcol == 80 ) ? 132 : npcol ;
                druckstr( pb3 ) ;
        }
}

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

reset()                         /* kontrollierter Abbruch des Backends */
{
        static int merk = 0 ;
        char c ;
```

```
        signal(SIGTERM, reset); /* Sollte Signal 14 oder 15 gesendet werden, */
        signal(SIGALRM, reset); /* so rufe die Funktion reset erneut auf.    */

        if ( merk++ == 0 )
        {
                alarm( 30 ) ;           /* Falls keine Daten an den Drucker */
                                        /* gesendet werden koennen, dann    */
                                        /* wird Signal 14 gesendet.         */
                lseek( 3, (long)0, 0 ) ;
                c = '\003' ;                                    /* m01 */
                write( 3, &c, 1 ) ;     /* Auftrag auf 'WARTET' stellen   */

                putchar(FF) ;    /* Versuche, ein Formfeed an den Drucker   */
                                 /* zu senden.                              */
                sleep(5);                                       /* m02 */
        }

        ioctl( 0, TIOCFLUSH, NULL );    /* Alle Zeichen, die */ /* m02 */
        ioctl( 1, TIOCFLUSH, NULL );    /* sich noch in den  */ /* m02 */
                                        /* internen Puffern befinden, werden */
                                        /* 'ausgespuelt'.                    */

        exit( 64 ) ;     /* Exitstatus, der ein Sperren des Druckers bewirkt */
}
/* ------------------------------------------------------------------------ */

druckstr( pstring )   /* Gibt einen String aus  */
char *pstring ;
{
        while ( *pstring != '\0' )
                 putchar( *pstring++ ) ;
}


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

druckorg(fpein)     /* Der zu druckende Text wird eingelesen und ausgegeben */
FILE *fpein ;
{
        extern int npcol, nplin, npab, npbis ;
        int c, i, fflag, ncol = 0,  nline = 1 , nseite = 1 ;
        int j_esc = 100, druckflag = 0 , k = 0 ;

        alarm( ZEIT ) ;                                         /* m01 */
        putchar( CR ) ;
        while ( ( c = getc(fpein) ) != EOF   )
        {
                if ( nseite >= npab )
                           druckflag = 1 ;
                fflag = 0 ;  k++ ;

                if ( j_esc > 1 )        /* das Zeichen hinter einem ESC wird */
                {                       /* nicht untersucht.                 */
                        switch(c)
                        {
                        case LF:
                                nline++ ;
                        case FF:
                                if ( druckflag && ncol > 0 )
                                        putchar( CR ) ;
                                if ( k > 999 )
                                {
```

```
                                pruefe() ;  /* Drucker ueberpruefen   */
                                alarm(ZEIT) ;
                                   /* Der Drucker hat ZEIT sekunden   */
                                   /* fuer die naechsten 1000 Zeichen */
                                k = 0 ;
                        }
                case CR:
                        ncol = 0 ;
                        break ;
                case ESC:
                        j_esc = 0 ;
                        break ;
                case '\t':
                        /* Bei Tabulatorzeichen muss        */
                        /* ncol richtig gezaehlt werden.    */
                        for ( i = 8 ; i <= ncol ; i += 8 )
                                ;
                        ncol = i ;
                        break ;
                default:
                        ncol++ ;
                }
        }
        if ( ncol >  npcol ) continue ;

        if ( nline > nplin || c == FF )
        {
                fflag = 1 ;
                lseek( 3, (long)10, 0 );  /* Anzahl fertiger Seiten  */
                write( 3, &nseite, 2 ) ;  /* bekanntgeben.           */
                if ( druckflag )
                        putchar( FF ) ;
                nline = 1 ; nseite++ ;
        }
        if ( nseite > npbis ) break ;

        switch(j_esc)
        {
        case 1:
                if ( c == '[' || c == '(' )
                       ncol-- ;      /* Das naechste Zeichen gehoert */
                                     /* dann auch zu ESC-Folge       */
                j_esc = 99 ;
        case 0:
                j_esc++ ;
        default:
                if ( fflag != 1 && druckflag )
                      putchar( c ) ;
                break ;
        }
    }
    putchar( CR ) ;
    if ( fflag != 1 )
          putchar( FF ) ;
    fflush( stdout ) ;
    return ;
}
```

```
/* ************************************************************************ */


/* Die folgenden Funktionen sind nur sinnvoll, wenn der Drucker ein ETX-ACK- */
/* Protokoll unterstuetzt.                                                  */
/* Andernfalls schreibt man einfach :                                       */
/*                      pruefe()                                            */
/*                      {                                                   */
/*                      }                                                   */
/*                                                                          */

# define ETX '\003'       /* End of Text Zeichen          */
# define ACK '\006'       /* Acknowledge Zeichen         */

pruefe()                  /* Ueberprueft die Bereitschaft des Druckers mit   */
                          /* Hilfe des ETX-ACK-Protokolls                    */
{
        char c, antwort = '!' ;
        int mailsend() ;

        lseek( 3, (long)0, 0 ) ;
        c = '\003' ;
        write( 3, &c, 1 ) ;             /* Auftrag auf 'WARTET' stellen     */

        signal(SIGALRM, mailsend);  /* Sollte Signal 14 gesendet werden, */
                                    /* so rufe die Funktion mailsend auf.*/

        alarm( ZEIT );              /* Der Drucker hat ZEIT Sekunden um  */
                                    /* auf das folgende ETX mit ACK zu   */
                                    /* antworten.                        */
        putchar( ETX ) ;
        fflush( stdout ) ;          /* ETX senden                        */

        do
                if ( read( 0, &antwort, 1 ) < 1 )
                        mailsend();
        while (  antwort & 0177 != ACK ) ;

        alarm(0) ;                      /* Alarm abschalten                 */

        lseek( 3, (long)0, 0 ) ;
        c = '\002' ;
        write( 3, &c, 1);               /* Auftrag auf 'LAEUFT' stellen     */

        signal(SIGALRM, reset);         /* Sollte Signal 14 gesendet werden,  */
                                        /* so verlasse das Backend in Zukunft */
                                        /* ueber die Funktion reset.          */

        return ;                        /* Drucker ist in Ordnung           */
}
```

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

mailsend()              /* Mail schreiben und Backend beenden  */
{
        extern char *auftrag ;
        char user[25], maildatei[45] ;
        FILE *faus ;
        long zeit, time() ;

        signal(SIGALRM, reset ) ;

        strcpy( maildatei, "/usr/spool/mail/" ) ;
        lseek( 3, (long)68, 0 ) ;
        read( 3, user, 25 ) ;              /* Benutzernamen lesen    */
        strcat( maildatei , user ) ;       /* Pfadname der Maildatei */

        faus = fopen( maildatei, "a");     /* Maildatei eroeffnen    */

        if ( faus != NULL )
        {
             zeit = time(&zeit) ;
             fprintf( faus,"Von %s am %s ", user, gctime( &zeit ) ) ;
             fprintf( faus,"Auftrag `%s' : Drucker sendet kein ACK\n",auftrag);
             fclose( faus ) ;
        }

        reset() ;
}


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

A.1.2.7 backend04.c

```
static char SCCSID[] = "@(#)backend04.c 1.4 85/10/10";

/* Backend fuer den Drucker 9004 mit Einzelblatt- oder Endlospapiereinzug.     */
/*                                                                             */
/* Einzelblatteinzug:                                                          */
/*  Wenn der 'lpr'-Befehl ohne Filter verwendet werden soll, dann muessen die  */
/*  Schalter -pl=65 und -pb=81 gesetzt werden ( z. B. in  /usr/lib/qconfig ).  */
/*  Der Schalter -pl=65 bewirkt zusaetzlich, dass 2 Leerzeilen am Anfang jeder */
/*  Seite unterdrueckt werden (sinnvoll in Verbindung mit dem 'pr'-Kommando).  */
/*  Wenn Dateien mit 'pr' aufbereitet werden sollen, so muss der Schalter      */
/*              '-l67'          angegeben werden.                              */
/*  Beispiel :  pr -l67 DATEINAME | lpr -pl=65 -pb=81                          */
/*                                                                             */
/*  Falls das Backend in Verbindung mit anderen Software-Produkten (z.B. HIT ) */
/*  verwendet werden soll, sollte der Schalter '-pl=65' nicht gesetzt werden.  */
/*                                                                             */
/* Endlospapiereinzug (unter SINIX 1 0B ) :                                    */
/*  Um eine ueberfluessige Leerseite am Beginn zu vermeiden, sollte der        */
/*  Schalter '-nff' gesetzt werden.                                            */
/*  Unter SINIX 1.0C ist dies nicht mehr noetig !                              */
/*                                                                             */
/* Dieses Backend setzt den Treiber in den RAW-Mode, und fuehrt ein ETX-ACK-   */
/* Protokoll.                                                                  */
/* Es kann ein zusaetzlicher Schalter '-form= ' angegeben werden, der die      */
/* Formularlaenge ( Zeilen pro Seite ) beim Drucker festlegt.                  */
/* In Gegensatz zum Standardbackend wird auch der Schalter '-pb2' ausgewertet  */
/*                                                                             */
/* Version vom 06.09.85                                                        */
/*                                                                             */

# include <stdio.h>
# include <sgtty.h>
# include <signal.h>

# define STX '\002'     /* STX - Zeichen                */
# define ETX '\003'     /* End of Text Zeichen          */
# define ACK '\006'     /* Acknowledge Zeichen          */
# define LF '\012'      /* Line Feed Zeichen            */
# define FF '\014'      /* Form Feed Zeichen            */
# define CR '\015'      /* Carriage Return Zeichen      */
# define SO '\016'      /* Shift Out                    */
# define SI '\017'      /* Shift In                     */
# define ESC '\033'     /* Escape Zeichen               */
# define EM '\031'      /* End of Medium Zeichen        */

# define BLOCK  350     /* normale Blockgroesse                                */
# define MBLOCK 420     /* maximale Blockgroesse                               */
# define ZEIT   100     /* Maximale Zeit, die der Drucker fuer MBLOCK Zeichen */
                        /* benoetigen darf.                                    */

int npcol =136 ;        /* Anzahl Spalten pro Zeile              */
int nplin = 72 ;        /* Anzahl Zeilen  pro Seite              */
int npab  =  1 ;        /* Der Text wird ab  Seite npab  gedruckt */
int npbis = 20000 ;     /* Der Text wird bis Seite npbis gedruckt */
int asf = 1 ;           /* Flagge fuer ASF-Anschluss             */
```

```
char *blattaus, *gzust, *bidi_ein, *bidi_aus, *pb2, *statab, drubuff[BLOCK + 100], formul[7] ;
int  nnbuff = 0, reset(), umsetzen() ;

main(argc,argv)
int argc;
char *argv[];
{
        FILE *fpein ;
        char cnc , *tabsetz , c ;
        int  nc, i ;

        /* Escape Seqenzen fuer den Drucker */
        blattaus = "\033\031J" ;              /*  Blattauswurf fuer ASF   */
        gzust    = "\033\015P" ;              /*  Grundzustand            */
        tabsetz  = "\0331" ;                  /*  Tabulator setzen        */
        bidi_ein = "\033/" ;                  /*  Bidirektionaldruck ein  */
        bidi_aus = "\033\\" ;                 /*  Bidirektionaldruck aus  */
        pb2      = "\033H011" ;               /*  12 Zeichen pro Zoll     */
        statab   = "\033\0321" ;              /*  Druckerstatus abfragen  */

        /* ############################################################ */
        /*      Dateideskriptor 0 zeigt auf die Geraetedatei (lesend)   */
        /*                      (stdin)                                 */
        /*      Dateideskriptor 1 zeigt auf die Geraetedatei (schreibend) */
        /*                      (stdout)                                */
        /*      Dateideskriptor 2 zeigt auf die Geraetedatei            */
        /*      /dev/null       (stderr)                                */
        /*      Dateideskriptor 3 zeigt auf die Datei in                */
        /*      /usr/spool/lpd/stat/s0.D?Q                              */
        /* ############################################################ */

        signal(SIGTERM, reset);         /* Sollte Signal 14 oder 15 gesendet */
        signal(SIGALRM, reset);         /* werden, so verlasse das Backend   */
                                        /* ueber die Funktion reset()        */
        lseek( 3, (long)0, 0 ) ;
        c = '\003' ;
        write( 3, &c, 1 ) ;             /* Auftrag auf 'WARTET' stellen      */

        alarm(ZEIT) ;
        umsetzen( 1 ) ;                 /* Treiber richtig einstellen        */

        drustr( gzust ) ;
        drustr( statab ) ;              /* Status abfragen                   */

        pruefe() ;                      /* Pruefen, ob ein Drucker ange-     */
                                        /* schlossen ist                     */
        lseek( 3, (long)0, 0 ) ;
        c = '\002' ;
        write( 3, &c, 1 ) ;             /* Auftrag auf 'LAEUFT' stellen      */

        for ( i = 1 ; i < argc ; i++ )
        {
             if ( *argv[i] == '-' )
                  schalter( argv[i] + 1 ) ;       /* Schalter interpretieren */
             else
                  fpein = fopen(argv[i] , "r");   /* Datei zum Lesen oeffnen */
        }

        if ( fpein == NULL )          /* Datei konnte nicht geoeffnet werden */
               exit(1);
```

```
        /* Tabulatoren setzen   */
        for ( i = 1 ; i <= 12 ; i++ )
                drustr("        \0331") ;

        druchar( CR ) ;

        lseek(3 , (long)1 , 0 ) ;
        read( 3 , &cnc , 1 ) ;          /*   Anzahl der gewuenschten Kopien    */
        nc = cnc & 0177 ;

        while ( nc-- > 0 )
        {
              /*##############################################*/
               druckorg(fpein);         /* Gebe den zu druckenden Text aus     */
              /*##############################################*/
               rewind( fpein ) ;
        }
        drustr( gzust ) ;
        pruefe() ;
        umsetzen(0) ;
        exit(0);                        /* Backend ordnungsgemass verlassen    */
}

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

umsetzen( i )                                   /* Treiber richtig einstellen   */
int i ;
{
      static struct sgttyb modus, savmodus ; /* siehe CES-Manual Nov. 1984,  */
                                             /* Teil 2, S. 1-3               */
      if ( i == 1 )
      {
           /* Grundzustand lesen  */
           ioctl( 1, TIOCGETP, &savmodus ) ;
           ioctl( 1, TIOCGETP, &modus ) ;
           modus.sg_ispeed = B1200 ;  /* input  Baudrate  */
           modus.sg_ospeed = B1200 ;  /* output Baudrate  */
                                      /* siehe CES-Manual Nov. 1984, Teil 2  */
                                      /* S. 1-3 bis 1-6 unter TTY(4)         */

           modus.sg_flags = RAW ;     /* RAW-Modus  */

           ioctl(1 , TIOCSETP, &modus ) ;    /* Einstellung setzen      */
      }
      else if ( i == 0 )
           ioctl(1 , TIOCSETP, &savmodus ) ; /* Grundeinstellung setzen */
}
/* ------------------------------------------------------------------------ */

schalter( pschalt )                             /* Schalter interpretieren  */
char *pschalt ;
```

```
{
        int i ;
        if ( strncmp( pschalt , "ab=" , 3 ) == 0 )
                npab = ( i = atoi( pschalt + 3 ) ) > 1 ? i : 1  ;
        else if ( strncmp( pschalt, "bis=" , 4 ) == 0 )
                npbis = (i = atoi( pschalt + 4 ) ) >= npab ? i : npbis ;
        else if ( strncmp( pschalt, "pb=", 3 ) == 0 )
                npcol = (i = atoi( pschalt + 3 ) ) > 0 ? i : npcol ;
        else if ( strncmp( pschalt, "pl=", 3 ) == 0 )
                nplin = (i = atoi( pschalt + 3 ) ) > 5 ? i : nplin ;
        else if ( strcmp( pschalt, "nff" ) == 0 )
                asf = 0 ;
        else if ( strcmp( pschalt, "pb2" ) == 0 )
                drustr( pb2 ) ;
        else if ( strncmp( pschalt, "form=", 5 ) == 0 )
        {
                i = atoi( pschalt + 5 ) ;
                if ( i > 2 && i < 127 )
                {
                        nplin = i ;
                        sprintf( formul, "\033F%03d" , i  )  ;
                        drustr( formul ) ;
                }
        }
}

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

reset()                         /* kontrollierter Abbruch des Backends      */
{
        static int merk = 0 ;
        char c ;

        signal(SIGTERM, reset); /* Sollte Signal 14 oder 15 gesendet werden, */
        signal(SIGALRM, reset); /* so rufe die Funktion reset erneut auf.    */

        if ( merk++ == 0 )
        {
                alarm( 20 ) ;           /* Falls keine Daten an den Drucker */
                                        /* gesendet werden koennen, dann    */
                                        /* wird Signal 14 gesendet.         */
                lseek( 3, (long)0, 0 ) ;
                c = '\003' ;
                write( 3, &c, 1 ) ;     /* Auftrag auf 'WARTET' stellen     */

                if ( asf )
                        drustr( blattaus ) ;
                else
                        druchar( FF ) ;
                drustr( gzust ) ;
                druflush() ;
                sleep( 5 ) ;

                ioctl( 0, TIOCFLUSH, NULL ) ;  /* Alle Zeichen, die sich noch in den */
                ioctl( 1, TIOCFLUSH, NULL ) ;  /* internen Puffern befinden, werden  */
                                               /* 'ausgespuelt'.                     */
                umsetzen(0) ;
        }

        exit( 64 ) ;     /* Exitstatus, der ein Sperren des Druckers bewirkt */
}
```

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

druckorg(fpein)      /* Der zu druckende Text wird eingelesen und ausgegeben */
FILE *fpein ;
{
        int c0 = ' ', c1, c2, i, lfflag = 0, ncol = 0,  nline = 1 , nseite = 1 ;
        int j_esc = 100, druckflag = 0 ;

        pruefe() ;

        if ( asf )
                druchar( FF ) ;

        for ( c1 = getc(fpein) ; c1 != EOF ; c1 = c2  )
        {
                c2 = getc( fpein ) ;

                if ( nseite >= npab )
                        druckflag = 1 ;
                lfflag++ ;

                if ( j_esc > 10 )       /* die Zeichen einer ESC-Folge werden */
                {                       /* nicht untersucht.                  */
                if ( nnbuff > BLOCK )    /* Drucker ueberpruefen      */
                      pruefe() ;         /* Datenblock ausgeben       */

                switch(c1)
                {
                case LF:
                        if ( nplin == 65 && lfflag < 3 && ncol == 0 )
                                continue ;

                        if ( nline++ == 1 && druckflag )
                                drustr( bidi_ein ) ;
                case FF:
                        if ( druckflag )
                        {
                                if ( nline >= nplin || c1 == FF )
                                        drustr( bidi_aus ) ;
                                if ( ncol > 0 )
                                        druchar( CR ) ;
                        }
                case CR:
                        ncol = 0 ;
                        break ;
                /* --------------------------  */
                case ESC:
                        j_esc = 0 ;
                case SO:
                case SI:
                        break ;
                case '\b':
                        ncol-- ;
                        break ;
                case '\t':
                        /* Bei Tabulatorzeichen muss        */
                        /* ncol richtig gezaehlt werden.    */
                        for ( i = 8 ; i <= ncol ; i += 8 )
                                        ;
```

```
                ncol = i ;
                break ;
        default:
                ncol++ ;
        }
}
if ( ncol >  npcol ) continue ;

if ( nline > nplin || c1 == FF )
{
        if ( c2 != EOF )
                c1 = FF ;
        else
                c1 = CR ;
        lfflag = 0 ;
        lseek( 3, (long)10, 0 );   /* Anzahl fertiger Seiten  */
        write( 3, (char *)&nseite, 2 ) ;  /* bekanntgeben.    */
        nline = 1 ; nseite++ ;
}
if ( nseite > npbis ) break ;

switch(j_esc)
{
case 4:
       j_esc = 99 ;
case 3:
case 2:
       j_esc++ ;
       if ( druckflag )
              druchar(c1) ;
       switch( c0 )
       {
       case EM:
              if ( c1 != 'R' )
              {
                   lseek( 3, (long)10, 0 ); /* Anzahl fertiger Seiten */
                   write( 3, (char *)&nseite, 2 ) ; /*  bekanntgeben. */
                   nseite++ ;  nline = 1 ; ncol = 0 ;
              }
       case CR:
              j_esc = 100 ;
              break ;
       case 'F':
       case 'S':
       case 'T':
       case 'H':
       case 'V':
              if ( c1 < '0' || c1 > '9' )
                     j_esc = 100 ;

       }
       break ;
    /* ------------------------------------ */
case 1:
       if (c1==EM||c1==CR||c1=='F'||c1=='S'||c1=='T'||c1=='H'||c1=='V')
            c0 = c1 ;
       else
            j_esc = 99 ;
```

```
                case 0:
                        j_esc++ ;
                default:
                        if ( druckflag )
                                druchar( c1 ) ;
                        break ;
                }
        }
        druchar( CR ) ;
        pruefe() ;
        if ( asf )
                drustr( blattaus ) ;
        else
                druchar( FF ) ;
        return ;
}

/* ************************************************************************** */

pruefe()                /* Ueberprueft die Bereitschaft des Druckers mit   */
                        /* Hilfe des ETX-ACK-Protokolls                    */
{
        int i = 10 ;
        char c, antwort , merk = ' ' ;

        alarm( ZEIT );              /* Der Drucker hat ZEIT Sekunden um    */
                                    /* auf das folgende ETX mit ACK zu     */
                                    /* antworten.                          */
        druchar( ETX ) ;            /* ETX an das Ende des Buffers schreiben */
        druflush( ) ;               /* Buffer ausgeben                     */

        do
        {
                if ( read( 0, &c, 1 ) < 1 )
                        reset();
                antwort = c & 0177 ;
                if ( antwort == STX )
                        i = 1 ;
                else if ( antwort == ACK )
                        merk = antwort ;

                switch(i)
                {
                case 3:
                        asf = antwort & 2 ;
                        i = 10 ;
                        break ;
                case 2:
                case 1:
                        i++ ;
                }
        }
        while (  merk != ACK || rdchk( 0 ) > 0 ) ;

        alarm(0) ;                  /* Alarm abschalten                    */

        return ;                    /* Drucker ist in Ordnung              */
}
```

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

druchar( c )
char c ;
{
        drubuff[ nnbuff++ ] = paritaet( c ) ;
        if ( nnbuff > MBLOCK )
                pruefe() ;
}

druflush()
{
        if ( write( 1, drubuff, nnbuff ) < nnbuff )
                reset() ;
        nnbuff = 0 ;
}
drustr( pstr )
char *pstr ;
{
        while ( *pstr != '\0' )
                druchar( *pstr++ ) ;
}

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

paritaet( c )    /* erzwingt ungerade Paritaet der Zeichen */
char c ;
{
        static int eins = 1 ;
        int bitzaehl, i ;

        for ( bitzaehl = 0 , i = 0 ; i <  7 ; i++ )
                if ( c & ( eins << i )  )
                        bitzaehl++ ;

        if ( bitzaehl%2 == 0 )    /* Falls Paritaet gerade, dann */
                    c |= 0200 ;  /* das hoechste Bit setzen     */

        return( c & 0377 ) ;
}
```

A.1.2.8 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.1.2.9 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.1.2.10 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);
}
```

A.1.2.11 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); }

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

A.1.2.12 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          /* Flusssteuerung bei Ausgabe */
#define         T_IXOFF         0010000          /* Flusssteuerung 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[0K",ESC                  /* Zeilenrest loeschen */
#define         BRLOE   "%c[0J",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.1.2.13 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 _
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 _
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