Betriebssystem fuer Rechnerarbeitsplaetze

E P O S

Anwenderdokumentation

Inhalt:

Kurzbeschreibung des Betiebssystems EPOS

Aufbau und Verwendung der Dokumentation

Moduluebersicht

Modulbeschreibungen

Inhaltsverzeichnis

	Kurzbeschreibung von EPOS	4
	Aufbau der Moduldokumentation	6
	Gliederung der Moduldokumentation	6
0.	EPOS	8
1.	EPOS-Urlader	10
1.1.	EPOSUL	10
2.	EPOStyp.COM	12
2.1.	EPOSINST	14
2.2.	EPOSCCP	16
2.3.	EPOSBDOS	17
2.4.	EPOSBIOS	18
2.4.1.	EPOSCOM	21
2.4.2.	Bildschirm	22
2.4.2.1.	EPOSTTY1	22
2.4.2.2.	EPOSTTY2	24
2.4.2.3.	EPOSTTY3	26
2.4.3.	Tastatur	29
2.4.3.1.	EPOSTAS1	29
2.4.3.2.	EPOSTAS2	31
2.4.3.3.	EPOSTAS3	33
2.4.3.4.	EPOSTAS4	35
2.4.3.5.	EPOSTAS5	37
2.4.3.6.	EPOSSCAN	39
2.4.3.7.	EPOSCON1	41
2.4.3.8.	EPOSCON2	43
2.4.3.9.	EPOSCON3	45
2.4.3.10.	TACOM	48
2.4.4.	Floppy	54
2.4.4.1.	EPOSRF	54
2.4.4.2.	EPOSRF1	54
2.4.4.3.	EPOSRF2	56
2.4.4.4.	EPOSRF3	58
2.4.4.5.	EPOSRF4	60
2.4.4.6.	EPOSBLK	62
2.4.4.7.	EPOSFDC	64
2.4.5.	Drucker	69
2.4.5.1.	EPOSLST1	69
2.4.5.2.	EPOSLST2	71
2.4.6.	EPOSRAM	73
3.	Systemprogramme	74
3.1.	DISK	74
3.1.1.	EPOSFMT	74
3.1.2	PUTSYS	77
3.1.3.	FORMATE	79

3.2.	NET	81
3.2.1.	NLOAD	81
3.2.2.	NSAVE	84
3.3.	SIOS	87
3.3.1.	SLOAD	87
3.3.2.	SSAVE	90
3.3.3.	SSLAVE	93

Kurzbeschreibung des Betriebssystems EPOS

EPOS gestattet die Nutzung von Rechnerarbeitsplaetzen verschiedenster Ausstattung.
EPOS ist ein Einzelnutzer-Betriebssystem und ist kompatibel zu CP/M, SCP, CP/A usw..
Es ist fuer die Nutzung auf EPMR-RAP ausgelegt, kann aber auf andere Konfigurationen angepasst werden.
Die folgenden Baugruppen sind mindestens noetig, um EPOS nutzen zu koennen:

- ZRE mit abschaltbarem Urlader-EPROM
- 64 k Hauptspeicher
- Tastaturansteuerung
- Videoansteuerung
- RAM-Floppy oder Floppy-Disk

Folgende Komponenten sind optional:

- SCOM-LAN-Anschluss
- V.24 oder IFSS fuer Druckeranschluss
- V.24 oder IFSS fuer Datenuebertragung
- Heimkassettenanschluss
- Floppy-Disk
- RAM-Floppy-Erweiterung

Ein Urlader realisiert nach dem Einschalten bzw. nach RESET das Laden und den Start des Betriebssystems. Deshalb muss der Urlader auf einem EPROM auf Adresse 0000H nach Einschalten des RAP aktiv sein.

Fuer den Betriebsystemanlauf wird nach dem Laden die Programmsteuerung zuerst an den Kaltstartinitialisierungsteil EPOSINST uebergeben. EPOSINST schafft alle programmtechnischen Voraussetzungen fuer EPOS. Dieser Teil wird nur beim Kaltstart benoetigt und danach von anderen Komponenten ueberladen.

Waehrend des Ladens realisiert der Kaltstartteil ausserdem die sogenannte Starthilfe. Im System kann ein zusaetzliches beliebiges Programm (COM-File) mitgefuehrt werden. Dieses ist bei RAP wichtig, die nur mit RAM-Floppy ausgeruestet sind. Da EPOS ein diskettenorientiertes BS ist und systemseitig keine Medien wie LAN oder Heimkassette unterstuetzt werden, wird nach dem Einschalten ein Programm benoetigt, welches in der Lage ist, weitere fuer die Arbeit notwendige Dateien von den entsprechenden Medien zu laden.
Diese Starthilfe steht nach Anlauf des BS ab Adresse 100H zur Verfuegung und kann entweder mit der Hand oder automatisch durch ein Kaltstartkommando (Standardfall) auf die RAM-Floppy geladen werden.

Beim Laden zeigt EPOSINST zeigt die Systemmeldung und eine Konfigurationsuebersicht an:

```
EPOS  V1.3 EPMR/he 1.90
A:448/R
B:5*1024*80*2/0
C:5*1024*80*2/1
C:16*256*80*2/1
D:16*256*40*1/1
Tas:K7672.03
PRT:DC1/DC3
RAM-Floppy Loeschen ? (J/N)
```

Wenn eine RAM-Floppy installiert ist, dann ist jetzt eine Bedienerhandlung noetig. Sonst erscheint die letzt Zeile nicht.

Nach Einschalten des RAP ist Loeschen erforderlich, um die zufaellige Belegung der RAM-Floppy zu beseitigen.
Nur bei Eingabe des Buchstaben J wird geloescht, ansonsten nicht.
Danach befindet sich das System in der Grundschleife des CCP und erwartet weitere Bedienereingaben. Erkennbar ist dieser Zustand am angezeigten Prompt:

```
  A>_
```

Alles weitere ist in einem CP/M-Handbuch nachzulesen.

Aufbau und Benutzung dieser Dokumentation

Diese Dokumentation ist modulorientiert aufgebaut, dass heisst, zu jedem Modul (Programm, Quellprogramm, INCLUDE-File usw.) existiert ein Beschreibungsblock. Die Module sind dabei hierarchisch geordnet und mit einer Bezugsnummer (siehe Inhaltsverzeichnis) versehen.

Gliederung der Moduldokumentation

Uebersicht

1. Benennung
2. Kurzbeschreibung
3. Daten (benoetigter Speicher, Laufzeit usw.)
4. Programmiersprache
5. Genutzte Module

Anleitung_fuer_den_Bediener

6. Anwendung (Zweck, Voraussetzungen, usw.)
7. Bedienanleitung, Programmablauf (Eingaben, Fehlermeldungen, Aufruf, Ein- und Ausgangsparameter usw.)

Anleitung_fuer_den_Systemprogrammierer

8. Generierung (Linktabelle, Speicherplan, Generierparameter, usw.)

9. Einschraenkungen bei der Nutzung

Die Punkte 1. - 5. dienen als Uebersicht und sollen die Orientierung erleichtern. Die Punkte 6. und 7. sind fuer den Anwender bzw. fuer den Bediener des Moduls oder des Programms gedacht, wohingegen sich die Punkte 8. und 9. vor allen an Systemprogrammierer wenden.

Hier noch einige allgemeine Hinweise zu EPOS:

EPOS ist ein CP/M-kompatibles Betriebssystem (CP/M ist ein eingetragenes Warenzeichen der Firma Digital Research). Es praesentiert sich deshalb auch in gleicher Weise wie andere CP/M-Kompatible. Wenn es also um die Benutzung des Betriebssystems EPOS geht, kann man sich in jedem CP/M-Handbuch informieren.
Die Bedienung weiterer spezieller Systemprogramme muss in den ensprechenden Modulbeschreibungen nachgelesen werden. Diese Programme sind aber so bedienerfreundlich ausgelegt, dass es auch ohne die Beschreibung moeglich ist, damit Ergebnisse zu erzielen.

Falls es notwendig sein sollte, ein neues EPOS zu generieren, sind hier einmal die Schritte zusammengefasst:

- Einstellen der Parameter entspechend der Hardware in der Datei EPOSCOM.typ (2.4.1.).
  Diese Datei wird durch Umbenennen und Aendern von EPOSCOM.MAC erzeugt. Die Datei wird waehrend der Generierung in EPOSCOM.MAC umbenannt. Diese Verfahrensweise erlaubt eine gute Uebersicht ueber moegliche Generiervarianten.

- Assemblieren und Linken des Urladers (1.1.)

- Assemblieren und Linken der Komponenten EPOSINST, EPOSCCP, EPOSBDOS und EPOSBIOS (2.1. 2.2. 2.3. 2.).

- Anpassen der Systemprogramme

Ausfuehrliche Anweisungen sind jeweils im Punkt 8. der Modulbeschreibung zu finden.

Zur Unterstuetzung der Generierung sind folgende Submit-Dateien vorhanden:

EPOSGEN.SUB - Generieren eines Betriebssystems EPOStyp.COM
EPOSGES.SUB - Generieren aller Komponenten von EPOS

1. Benennung

```
**********************************************************
*                                                        *
*          0. EPOS                                       *
*                                                        *
**********************************************************
```

2. Kurzbeschreibung

CP/M-kompatibles Betriebssystem fuer Rechnerarbeitsplaetze (RAP) mit generierbarem Funktionsumfang

3. Daten

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

EPOS besteht aus 3 Komponenten

1. Urlader

2. Betriebssystem EPOStyp.COM
   (typ ist eine Kennung fuer den generierten Rechnertyp)

3. Hardware- oder systemabhaengige Programme (Systemprogramme).
   Fuer RAP mit Floppy-Disk existieren folgende Systemprogramme:

   - EPOSFMT - Formatierprogramm
   - PUTSYS - Systemspuren einer Floppy-Disk einrichten
   - FORMATE - Diskettenformate der Laufwerke aendern

   Fuer LAN-RAP existieren folgende spezielle Systemprogramme

   - NLOAD Laden einer Datei von LAN-Server auf (RAM-) Floppy

   - NSAVE Retten einer Datei von (RAM-) Floppy zum LAN-Server

6. Anwendung

EPOS gestattet die Nutzung von Rechnerarbeitsplaetzen verschiedenster Ausstattung.
EPOS ist ein Einzelnutzer-Betriebssystem und ist kompatibel zu CP/M, SCP, CP/A usw..

Es ist fuer die Nutzung auf EPMR-RAP ausgelegt, kann aber auf andere Konfigurationen angepasst werden.
Die folgenden Baugruppen sind mindestens noetig, um EPOS nutzen zu koennen:

- ZRE mit abschaltbarem Urlader-EPROM
- 64 k Hauptspeicher
- Tastaturansteuerung
- Videoansteuerung
- RAM-Floppy oder Floppy-Disk

Folgende Komponenten sind optional:

- SCOM-LAN-Anschluss
- V.24 oder IFSS fuer Druckeranschluss
- V.24 oder IFSS fuer Datenuebertragung
- Heimkassettenanschluss
- Floppy-Disk
- RAM-Floppy-Erweiterung

7. Programmablauf, Bedienanleitung

8. Generierung

Die Generierung der EPOS-Komponenten erfolgt in 3 Schritten:

- Generierung des Urladers und Brennen des EPROMs

- Generierung von EPOStyp.COM und Ablegen auf den entsprechenden Medien (Systemspuren der Floppy-Disk oder EPROM-Bank)

- Anpassen der Systemprpgramme

  Fuer die beiden ersten Punkte hat die Datei EPOSCOM.typ zentrale Bedeutung. Hier sind alle gemeinsamen und hardwareabhaengigen Vereinbarungen abgelegt. Fuer die meisten Anwendungsfaelle wird ein Modifizieren dieser Datei die Anpassung von EPOS auf die konkrete Hardware des RAP moeglich machen. (siehe 2.4.1.)

9. Einschraenkungen bei der Nutzung

keine

1. EPOS-Urlader

1. Benennung

1.1. EPOSUL

2. Kurzbeschreibung

Betriebssystemurlader fuer das Betriebssystem EPOS von Floppy-Disk oder EPROMs.

3. Daten

Speicherbedarf: ca 1000 Byte
(maximal 2 kByte = 1 x U2716)

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

-(incl) EPOSCOM.typ Generierdaten fuer EPOS
-(incl) EPOSFDC.MAC Treiberprogramm fuer FDC U8272
-(incl) EPSUMLD.MAC Umladeprozesse

6. Anwendung

EPOSUL realisiert nach dem Einschalten bzw. nach RESET das Laden und den Start des Betriebssystems. Deshalb muss EPOSUL auf einem EPROM auf Adresse 0000H nach Einschalten des RAP aktiv sein.

7. Programmablauf, Bedienanleitung

Der Urlader erfordert keine Bedienhandlungen.
Nach Programmanlauf laedt der Urlader sich in den RAM (Adresse URLAD) um und schaltet den EPROM ab.
Danach wird die Diskette im Laufwerk PKLW analysiert und mit den gewonnenen Daten (DPB) der 1. Sektor der 1. Spur gelesen. Nach Pruefen der Kennung ('EP') und Bestimmen der

Ladeparameter wird das BS geladen und angsprungen.
Der Urlader erzeugt keine Fehlermeldungen. Die Arbeit des Urladers kann an der LED am Floppy-Disk erkannt werden.

Wenn in EPOSCOM.typ das Flag fuer den EPROM-Urlader gesetzt ist (dul equ 0), verlaufen die Ladevorgaenge folgendermassen:
Der Urlader laedt den Inhalt von der EPROM-Bank in den Hauptspeicher ab Adresse TPA-200h. Anschliessend werden der Urlader und die EPROM-Bank abgeschaltet und der RAM aktiviert. Dann wird das Programm angesprungen. Damit uebernimmt das Programm (normalerweise ist dieses das EPOS-BS) die Steuerung.
Der Urlader erzeugt keine Fehlermeldung.

8. Generierung

EPOSUL (wie auch alle anderen EPOS-Komponenten) wird durch Aendern der Generierparameter im EPOSCOM.typ mittels eines Editors auf die entsprechende Hardware angepasst.
In speziellen Faellen (z.B. bei einer besonderen EPROM-Konfiguration) kann sich auch eine Aenderung im Quellcode von EPOSUML.MAC notwendig machen.

Weitere Verarbeitung:

```
Assemblieren:  ASM =EPOSUL
Linken:        LINK EPOSUL, EPOSUtyp/N/E
Programmieren eines EPROM mit EPOSUtyp.COM
```

9. Einschraenkungen bei der Nutzung

Durch den Charakter der Umladeprozesse werden folgende RAM-Bereiche veraendert:
- 2 kByte ab Adresse URLAD fuer den Urlader
- TPA-200h bis TPA fuer EPOSINST

Das bedeutet, dass nach RESET diese RAM-Zellen ihren Inhalt verloren haben.

1. Benennung

****
*
* 2. EPOStyp.COM
*
****

2. Kurzbeschreibung

Betriebssystemkern von EPOS

3. Daten

- EPOSINST (2.1.) 0,5 kByte
- EPOSCCP (2.2.) 2 kByte
- EPOSBDOS (2.3.) 3,5 kByte
- EPOSBIOS (2.4.) 2..4 kByte

insgesamt =8..10 kByte

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

EPOSxx.COM besteht aus 4 Modulen

-(link) EPOSINST	Kaltstartinstallation	(2.1)
-(link) EPOSCCP	Consol-Command-Processor	(2.2)
-(link) EPOSBDOS	Basic-Disk-Operating System	(2.3)
-(link) EPOSBIOS	Basic-I/O-Operating System	(2.4)

6. Anwendung

EPOStyp.COM muss durch Urlader oder gleichgelagerte Ladeprogramme (hier ist auch der CCP moeglich) auf eine beliebige Adresse in den Hauptspeicher geladen und angesprungen werden. Dann uebernimmt der Kaltstartinitialisierungsteil von EPOS EPOSINST die Steuerung (siehe 2.1).

7. Programmablauf, Bedienanleitung

Es sind keine besonderen Bedienhandlungen notwendig.
Die Urlader laden EPOStyp.COM automatisch von entsprechenden Medien (DISK, EPROM). EPOStyp.COM kann auch als normales COM-File von Floppy-Disk geladen werden.

8. Generierung

Nach Modifizieren von EPOSCOM.typ (2.4.1.) werden die Komponenten assembliert.

Bedienablauf:

```
A>ASM
*=EPOSINST
*=EPOSCCP
*=EPOSBDOS
*=EPOSBIOS
*^C
A>
```

Waehrend des Assemblerlaufes werden jeweils die fuer die Komponenten wichtigen Daten (Beginn, Laenge im Speicher, Generiervarianten, Hardwareadressen) ausgegeben. Durch Protokollieren dieser Angaben auf dem Drucker (Druecken von ^P), liegt ein Systemreport vor, der die wichtigsten Systemdaten enthaelt.

Danach wird EPOStyp.COM gelinkt (typ ist eine Bezeichner des Rechnertyps und kann frei gewaehlt werden).

```
A>LINK
*EPOSINST, EPOSCCP, EPOSBDOS, EPOSBIOS
*EPOStyp/N/E
A>
```

Das Einspielen der Starthilfe (wenn noetig, siehe 2.1) geschieht folgendermassen:

```
A>DU  EPOStyp.COM
 #Istname.COM
 #Rnnnn
 #^C
A>SAVE  zz  EPOStyp.COM
```

stname - ist der Name des Starthilfsprogrammes

nnnn - ist die Adresse des fuer die Starthilfe reservierten Speicherbereiches und ist dem Systemreport von EPOSBIOS zu entnehmen

zz - ist die Laenge von EPOStyp.COM in 256-Byte-Einheiten und der Auschrift des Linkers zu entnehmen.

Zur Unterstuetzung der Generierung sind folgende Submit-Dateien vorhanden:

EPOSGEN.SUB - Generieren eines Betriebssystems EPOStyp.COM
EPOSGES.SUB - Generieren aller Komponenten von EPOS

1. Benennung

****
*
* 2.1. EPOSINST
*
****

2. Kurzbeschreibung

Kaltstartinstallierung von EPOS

3. Daten

Speicherlaenge: ca. 290 Byte
(max. 512 Byte (200H))

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

-(incl) EPOSCOM.MAC Generierdaten EPOS (2.4.1)
EPOSINST benutzt BIOS-Rufe

6. Anwendung

EPOSINST schafft alle programmtechnischen Voraussetzungen fuer EPOS. Dieser Teil wird beim Kaltstart benoetigt und danach von anderen Komponenten ueberladen.

Waehrend des Ladens realisiert der Kaltstartteil ausserdem die sogenannte Starthilfe. Im System kann ein zusaetzliches beliebiges Programm (COM-File) mitgefuehrt werden. Dieses ist bei RAP wichtig, die nur mit RAM-Floppy ausgeruestet sind. Da EPOS ein diskettenorientiertes BS ist und systemseitig keine Medien wie LAN oder Heimkassette unterstuetzt, wird nach dem Einschalten ein Programm benoetigt, welches in der Lage ist, weitere fuer die Arbeit notwendigen Dateien von den entsprechenden Medien zu laden.
Diese Starthilfe steht nach Anlauf des BS ab Adresse 100H zur Verfuegung und kann entweder mit der Hand oder automatisch durch das Kaltstartkommando (Standardfall) auf die RAM-Floppy geladen werden.

7. Programmablauf, Bedienanleitung

EPOSINST fuehrt folgende Aktivitaeten aus:

- Umladen der Komponenten
  EPOSCCP (2.2)
  EPOSBDOS (2.3)
  EPOSBIOS (2.4)
  Starthilfe

- Interruptmode des Prozessors einstellen

- Starten des Systemtaktes

- wenn RAM-Floppy vorhanden ist und der CCP nicht resident sein soll, wird der CCP auf die RAM-Floppy geladen

- Anzeige der Ueberschrift

- Anzeige der Konfigurationsuebersicht

- Bedienerhandlung: (wenn RAM-Floppy vorhanden)
  RAM-Floppy loeschen J/N: _
  Nach Einschalten des RAP ist Loeschen erforderlich, um die zufaellige Belegung der RAM-Floppy zu beseitigen.

- wenn RAM-Floppy vorhanden wird der CCP auf die Systemspur der RAM-Floppy geschrieben

- Ansprung der Marke KBOOT im BIOS

8. Generierung

EPOSINST wird durch Aendern der Generierparameter im EPOSCOM.typ (2.4.1) mittels eines Editors auf die entsprechende Hardware angepasst.
In speziellen Faellen kann auch eine Aenderung im Quellcode von EPOSINST.MAC notwendig sein.

Weitere Verarbeitung:

Assemblieren: ASM =EPOSINST

9. Einschraenkungen bei der Nutzung

Durch den Charakter der Umladeprozesse wird EPOSINST vor den CCP geladenn (TPA-200h).
Das bedeutet, dass nach RESET diese RAM-Zellen ihren Inhalt verloren haben.

1. Benennung

```
**********************************************************
*                                                        *
*          2.2. EPOSCCP                                  *
*                                                        *
**********************************************************
```

2. Kurzbeschreibung

Der Consol-Command-Processor von EPOS steuert den Bedienerdialog in der Systemgrundschleife.

3. Daten

Laenge: 2 kByte

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

-(incl) EPOSCOM.typ Generierdaten (2.4.1.)
EPOSCCP benutzt BDOS-Rufe

6. Anwendung

EPOSCCP ist ein Standardbestandteil von EPOS.

7. Programmablauf, Bedienanleitung

Dieses Programm wickelt im Systemgrundzustand die Kommunikation mit dem Bediener ab. (siehe auch CP/M-Handbuch Teil 2.2 Teil II Pkt. 1. 2.)

8. Generierung

Es ist keine Generierung des Funktionsumfanges moeglich.

Weitere Verarbeitung:

Assemblieren: ASM =EPOSCCP

9. Einschraenkungen bei der Nutzung

Keine Einschraenkungen.

1. Benennung

****
*
* 2.3. EPOSBDOS
*
****

2. Kurzbeschreibung

Das Basic-Disk-Operation-System von EPOS stellt die logischen und den hardwarunabhaengigen Teil des Betriebssystems dar. Hier werden alle komplexen Datei- und Konsolfunktionen realisiert.

3. Daten

Laenge: 3,5 kByte

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

-(incl) EPOSCOM.typ Generierdaten (2.4.1.)
EPOSBDOS benutzt BIOS-Rufe

6. Anwendung

EPOSBDOS ist ein Standardbestandteil von EPOS.

7. Programmablauf. Bedienanleitung

EPOSBDOS stellt eine Unterprogrammsammlung dar. Alle Unterprogramme werden ueber einen Sprungverteiler, der ueber die Adresse 0005h (call 5) erreicht wird angesprungen.
Die Unterprogramme realisieren alle logischen (hardwareunabhaengigen) programmtechnischen Systemfunktionen. Die Kenntnis dieser Funktionen ist fuer den Programmierer notwendig (siehe CP/M-Handbuch Teil IV Pkt. 1. und 2.)

8. Generierung

Es ist keine Generierung des Funktionsumfanges moeglich.

Weitere Verarbeitung:

Assemblieren: ASM =EPOSBDOS

9. Einschraenkungen bei der Nutzung

keine Einschraenkungen

1. Benennung

****
*
* 2.4. EPOSBIOS
*
****

2. Kurzbeschreibung

EPOSBIOS stellt den hardwareabhaengigen Teil des Betriebssystems dar.

3. Daten

Laenge: 2..4 kByte
BIOS-Rumpf 200..500 Byte

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

Je nach Generierung koennen per Include-Anweisung alle unter Punkt 2.4. (siehe Inhaltsverzeichnis) aufgefuehrten Module benutzt werden.

6. Anwendung

EPOSBIOS ist ein Standardbestandteil von EPOS.

7. Programmablauf, Bedienanleitung

EPOSBIOS stellt eine Unterprogrammsammlung dar. Alle Unterprogramme werden ueber eine Sprungleiste am Anfang von BIOS erreicht. Die Unterprogramme realisieren alle physischen (hardwareabhaengigen) programmtechnischen Systemfunktionen. Die Kenntnis dieser Funktionen ist fuer den Programmierer notwendig (siehe CP/M-Handbuch Teil IV Pkt. 3.).
Der Modul EPOSBIOS.MAC stellt den Rumpf des BIOS dar. Hier befinden sich alle Standardbestandteile von BIOS wie Warmstart, Diskettenparameterkoepfe usw. Ausserdem werden hier je nach Generierung die jeweiligen Geraetetreiber durch INCLUDE-Anweisungen eingeordnet.

An der Adresse 0000h steht ein Sprung (jmp wboot) zum BIOS-Warmstart. Diese Routine steht an zweiter Stelle in der BIOS-Sprungleiste. Aus dem Inhalt der Speicherzelle 0001h laesst sich also der Anfang vom BIOS ermitteln. Der Anfang von BIOS hat folgenden Aufbau:

```
;Sprungleiste
;-----------------------------------------------
        jp      boot    ;+0
        jp      wboot   ;+3
        jp      const   ;+6
        jp      conin   ;+9
        jp      conout  ;+12
        jp      lst     ;+15
        jp      punch   ;+18
        jp      reader  ;+21
        jp      home    ;+24
        jp      seldsk  ;+27
        jp      settrk  ;+30
        jp      setsec  ;+33
        jp      setdma  ;+36
        jp      read    ;+39
        jp      write   ;+42
        jp      lstst   ;+45
        jp      tran    ;+48

;Parameterteil
;-----------------------------------------------
        db      'EP 12' ;+51    Version

cposxx: dw      scbeg   ;+56    Kursor-Position

        dw      floppy  ;+58    Physische Floppyroutine
        dw      hstbuf  ;+60    Floppy-Block-Puffer

        dw      dph0    ;+62    Kopf der Koepfe
        db      maxdsk  ;+64    Anzahl der logischen und
        db      diskanz ;+65    der physischen Laufwerke
        dw      rfkap   ;+66    RAM-Floppy-Kapazitaet

        dw      ktab    ;+68    Adresse der Code-Tabelle
        dw      tbuf    ;+70    und des Puffer der Tastatur

        dw      a_debug ;+72    BIOS-Monitor-Ansprung
```

Die Sprungleiste ist CP/M-typisch, und im CP/M-Handbuch beschrieben. Der Parameterblock hinter der Sprungleiste ist EPOS-speziell und fuer Sonderanwendungen gedacht. Dieses trifft auch fuer die Belegung der Adresse 0040h zu. Hier ist der Beginn der Interruptvektor-Tabelle abgelegt. Die ersten 16 Byte der Tabelle sind fuer das System reserviert.

8. Generierung

BIOS laesst im Funktionsumfang sowie in Bezug auf Hardwareanpassung einen grossen Spielraum zu. EPOSBIOS (wie auch alle anderen EPOS-Komponenten) wird durch Aendern der Generierparameter im EPOSCOM.typ (2.4.1.) mittels eines Editors auf die entsprechende Hardware angepasst. In speziellen Faellen kann sich auch eine Aenderung im Quellcode der jeweiligen Komponenten notwendig machen.

Weitere Verarbeitung:

Assemblieren: ASM =EPOSBDOS
Waehrend des Assemblerlaufes werden die wichtigen Daten (Beginn, Laenge im Speicher, Generiervarianten, Hardware-adressen) ausgegeben. Durch Protokollieren dieser Angaben auf dem Drucker (Druecken von ^P) liegt ein Systemreport vor, der die wichtigsten Systemdaten enthaelt.

9. Einschraenkungen bei der Nutzung

keine Einschraenkungen

1. Benennung

```
**********************************************************
*                                                        *
*          2.4.1. EPOSCOM                                *
*                                                        *
**********************************************************
```

2. Kurzbeschreibung

Dieser Programmteil enthaelt die gemeinsamen Vereinbarungen und Generierdaten fuer alle EPOS-Komponenten.

3. Daten

Dieser Modul dient nur der Steuerung der Assemblerlaeufe und erzeugt selbst keinen Code.

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

6. Anwendung

Die Anpassung von EPOS auf andere Hardware oder Veraenderung des Funktionsumfanges wird bei EPOS auf Quellcode-Niveau realisiert. Dadurch wird groesste Variabilitaet erreicht. Um Aenderungen in allen Modulen wirksam werden zu lassen, sind in diesem Programmteil alle Vereinbarungen und Assemblersteuervariablen zusammengefasst.

7. Programmablauf, Bedienanleitung

8. Generierung

Mittels eines Editors sind die entspechenden Variablen zu modifizieren. Die Datei ist ausfuehrlich kommentiert.

9. EinschraenKungen bei der Nutzung

Es sind nicht alle Moeglichkeiten der Hardwarekonfiguration eines Rechners durch Aendern dieser Datei nutzbar. Die Standard-MRBS- und NANOS-Baugruppen sind aber ohne weitere Aenderungen ansteuerbar.

2.4.2. Bildschirm

1. Benennung

2.4.2.1. EPOSTTY1

2. Kurzbeschreibung

Treiberroutine fuer Bildschirm

3. Daten

Laenge: ca. 320 Byte + 55 Byte Software-Kursor

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

6. Anwendung

EPOSTTY dient der Ansteuerung eines Monitors. Dabei wird davon ausgegangen, dass der Bildwiederholspeicher ein Teil des Hauptspeichers ist. Ausserdem werden Hardwareunterstuetzung fuer inverse und intensive Zeichendarstellung und die Kursordarstellung durch Setzen von Bit 7 im Zeichencode erwartet (z.B.ROBOTRON ABS K7023, ABS K7024).

7. Programmablauf, Bedienanleitung

Der Bildschirmtreiber ist ein Unterprogramm und hat folgende Parameter:

Eingangsparameter:
Reg. C: anzuzeigendes Zeichen im ASCII-Code

Ausgangsparameter:
keine

Aufrufbedingungen:
UP-Name: CONOUT
Die Register AF,BC,DE und HL werden veraendert.
Neben den darstellbaren ASCII-Zeichen (20h...7fh) sind fogende **Steuerzeichen** moeglich:

01h	HOME	Kursor an den Anfang des Bildschirms
08h	BS	Kursor ein Zeichen nach links
0ah	LF	Kursor eine Zeile nach unten
0ch	CLEAR	Bildschirm loeschen und den Kursor an den Anfang des Bildschirms
0dh	CR	Kursor an den Zeilenanfang
14h	EOS	Loeschen des Bildschirms ab aktueller Kursorposition bis Ende
15h	RIGHT	Kursor eine Position nach rechts
16h	del.EOL	Loeschen der Zeile ab aktueller Kursorposition bis Ende
18h	DEL	Loeschen der gesamten Zeile und Setzen des Kursors an den Zeilenanfang
1ah	UP	Kursor eine Zeile nach oben
1bh	YX	Einleiten einer Kursorpositionierung. die anschliessenden beiden Bytes geben an: Y= Zeilennummer (0...linmb-1) X= Zeichenposition (0...zlen-1) (Y+80h, X+80h)
7fh	DEL	Kursor ein Zeichen nach links und an die Kursorposition Space ausgeben
82h	CursorOn	Kursor einschalten
83h	CursorOff	Kursor ausschalten
84h	Normal	Normale Zeichendarstellung einstellen
85h	Inverse	Inverse Zeichendarstellung einstellen
86h	Intensive	Intensive Zeichendarstellung einstellen
87h	I+I	Intensive und inverse Zeichendarstellung einstellen

Die intensive und inverse Darstellung muss von der Hardware unterstuetzt werden. Umschaltzeichen sind dabei die Steuerzeichen-80h. Dieses ist bei ROBOTRON ABS K7023 und ABS K7024 der Fall.

Programmablauf
- Ansprung des UP DEFTA (Stringtastendefinition, siehe EPOSCON2.MAC oder EPOSCON3.MAC).
- Zeilennr. oder Zeichenpos. fuer Kursorpositionierung ausfiltern.
- Steuerzeichen mittels **contab** und **conadr** ausfiltern und Sonderbehandlung ausfuehren.
- Sonst Zeichen an aktuelle Kursorposition schreiben.

8. Generierung

Die Generierung erfolgt durch Aendern der Variablen **linmb, zlen, scbeg, softcu** in EPOSCOM.typ (2.4.1.).

9. Einschraenkungen bei der Nutzung

keine

1. Benennung

****
*
* 2.4.2.2. EPOSTTY2
*
****

2. Kurzbeschreibung

Treiberroutine fuer Bildschirm

3. Daten

Laenge: ca. 340 Byte + 55 Byte Softwar-Kursor

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

6. Anwendung

EPOSTTY2 dient der Ansteuerung eines Monitors. Dabei wird davon ausgegangen, dass der Bildwiederholspeicher ein Teil des Hauptspeichers ist. Das Hervorheben von Zeichen wird hardwareseitig durch Setzen von Bit 7 im Zeichencode realisiert (z.B. NANOS-Video-Baugruppen).

7. Programmablauf, Bedienanleitung

Der Bildschirmtreiber ist ein Unterprogramm und hat folgende Parameter:

Eingangsparameter:
Reg. C: anzuzeigendes Zeichen im ASCII-Code

Ausgangsparameter:
keine

Aufrufbedingungen:
UP-Name: CONOUT
Die Register AF,BC,DE und HL werden veraendert.
Neben den darstellbaren ASCII-Zeichen (20h...7fh) sind fogende **Steuerzeichen** moeglich:

01h	HOME	Kursor an den Anfang des Bildschirms
08h	BS	Kursor ein Zeichen nach links
0ah	LF	Kursor eine Zeile nach unten
0ch	CLEAR	Bildschirm loeschen und den Kursor an den

		Anfang des Bildschirms
0dh	CR	Kursor an den Zeilenanfang
14h	EOS	Loeschen des Bildschirms ab aktueller Kursorposition bis Ende
15h	RIGHT	Kursor eine Position nach rechts
16h	del.EOL	Loeschen der Zeile ab aktueller Kursorposition bis Ende
18h	DEL	Loeschen der gesamten Zeile und Setzen des Kursors an den Zeilenanfang
1ah	UP	Kursor eine Zeile nach oben
1bh	XY	Einleiten einer Kursorpositionierung. die anschliessenden beiden Bytes geben an: Y= Zeilennummer (0...linmb-1) X= Zeichenposition (0...zlen-1) (Y+escoff, X+escoff) (escoff im allg. 80h)
7fh	DEL	Kursor ein Zeichen nach links und an die Kursorposition Space ausgeben
82h	CursorOn	Kursor einschalten
83h	CursorOff	Kursor ausschalten
84h	Normal	Normale Zeichendarstellung einstellen
85h	Inverse	Hervorgehobene Zeichendarstellung einstellen
86h	Intensive	Hervorgehobene Zeichendarstellung einstellen
87h	I+I	Hervorgehobene Zeichendarstellung einstellen

Die hervorgehobene (unterstrichene oder inverse) Darstellung wird durch Setzen von BIT 7 im Zeichencode (gleiche Darstellung wie der Kursor) realisiert.

Programmablauf

- Ansprung des UP DEFTA (Stringtastendefinition, siehe EPOSCON2.MAC oder EPOSCON3.MAC).
- Zeilennr. oder Zeichenpos. fuer Kursorpositionierung ausfiltern.
- Steuerzeichen mittels **contab** und **conadr** ausfiltern und Sonderbehandlung ausfuehren.
- Sonst Zeichen an aktuelle Kursorposition schreiben.

8. Generierung

Die Generierung erfolgt durch Aendern der Variablen **linmb, zlen, scbeg, softcu** in EPOSCOM.typ (2.4.1.).
Ausserdem kann **escoff** veraendert werden (siehe Steuerzeichen 1bh).

9. Einschraenkungen bei der Nutzung

keine

1. Benennung

****
*
* 2.4.2.3. EPOSTTY3
*
****

2. Kurzbeschreibung

Treiberroutine fuer Bildschirm

3. Daten

Laenge: ca. 400 Byte + 55 Byte Softwar-Kursor

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

6. Anwendung

EPOSTTY3 dient der Ansteuerung eines Monitors. Dabei wird davon ausgegangen, dass der Bildwiederholspeicher ein Teil des Hauptspeichers ist. Das Hervorheben von Zeichen wird hardwareseitig durch Setzen von Bit 7 im Zeichencode realisiert (z.B. NANOS-Video-Baugruppen).
Ausserdem koennen, durch Steuerzeichen umschaltbar, besondere Sonderzeichenbloecke angezeigt werden.

7. Programmablauf, Bedienanleitung

Der Bildschirmtreiber ist ein Unterprogramm und hat folgende Parameter:

Eingangsparameter:
Reg. C: anzuzeigendes Zeichen im ASCII-Code

Ausgangsparameter:
keine

Aufrufbedingungen:
UP-Name: CONOUT
Die Register AF,BC,DE und HL werden veraendert.
Neben den darstellbaren ASCII-Zeichen (20h...7fh) sind fogende **Steuerzeichen** moeglich:

01h	HOME	Kursor an den Anfang des Bildschirms
08h	BS	Kursor ein Zeichen nach links
0ah	LF	Kursor eine Zeile nach unten

0ch	CLEAR	Bildschirm loeschen und den Kursor an den Anfang des Bildschirms
0dh	CR	Kursor an den Zeilenanfang
14h	EOS	Loeschen des Bildschirms ab aktueller Kursorposition bis Ende
15h	RIGHT	Kursor eine Position nach rechts
16h	del.EOL	Loeschen der Zeile ab aktueller Kursorposition bis Ende
18h	DEL	Loeschen der gesamten Zeile und Setzen des Kursors an den Zeilenanfang
1ah	UP	Kursor eine Zeile nach oben
1bh	XY	Einleiten einer Kursorpositionierung. die anschliessenden beiden Bytes geben an: Y= Zeilennummer (0...linmb-1) X= Zeichenposition (0...zlen-1) (Y+escoff, X+escoff) (escoff im allg. 80h)
7fh	DEL	Kursor ein Zeichen nach links und an die Kursorposition Space ausgeben
82h	CursorOn	Kursor einschalten
83h	CursorOff	Kursor ausschalten
84h	Normal	Normale Zeichendarstellung einstellen, Sonderzeichenbloecke aus
85h	Inverse	Hervorgehobene Zeichendarstellung einstellen
86h	Intensive	Hervorgehobene Zeichendarstellung einstellen
87h	I+I	Hervorgehobene Zeichendarstellung einstellen
88h	bst	Sonderzeichenblock 1 einschalten
89h	bsa	Sonderzeichenblock 2 einschalten

Die hervorgehobene (unterstrichene oder inverse) Darstellung wird durch Setzen von BIT 7 im Zeichencode (gleiche Darstellung wie der Kursor) realisiert.

Der Sondertastenblock 1 bewirkt das Umsetzen folgender Zeichen:

```
5bh     01h     ;Ae
5ch     02h     ;Oe
5dh     03h     ;Ue
7bh     04h     ;ae
7ch     05h     ;oe
7dh     06h     ;ue
7eh     07h     ;~
```

Der Sondertastenblock 2 bewirkt das Umsetzen folgender Zeichen:

```
22h     08h     ;"
23h     09h     ;#
27h     0ah     ;'
2ah     0bh     ;*
3ch     0ch     ;<
3eh     0dh     ;>
40h     0eh     ;@
5eh     0fh     ;^
5fh     10h     ;_
60h     11h     ;`
```

Der Zeichengenerator auf der Video-Ansteuerkarte muss an den Positionen 01h..11h die entsprechenden Sonderzeichen enthalten.

Programmablauf

- Ansprung des UP DEFTA (Stringtastendefinition, siehe EPOSCON2.MAC oder EPOSCON3.MAC).
- Zeilennr. oder Zeichenpos. fuer Kursorpositionierung ausfiltern.
- Steuerzeichen mittels **contab** und **conadr** ausfiltern und Sonderbehandlung ausfuehren.
- Wenn Sondertastenblock 1 oder 2 ein, dann werden die entsprechenden Tastencodes umgesetzt.
- Sonst Zeichen an aktuelle Kursorposition schreiben.

8. Generierung

Die Generierung erfolgt durch Aendern der Variablen **linmb, zlen, scbeg, softcu** in EPOSCOM.typ (2.4.1.).
Ausserdem kann **escoff** veraendert werden (siehe Steuerzeichen 1bh).

9. Einschraenkungen bei der Nutzung

Es ist ein besonderer Zeichengenerator notwendig.
Dieser Treiber gehoert nicht zum Nachnutzungsumfang von EPOS.

2.4.3. Tastatur

1. Benennung

2.4.3.1. EPOSTAS1

2. Kurzbeschreibung

Geraetetreiber fuer eine ueber ein serielles Port angeschlossene Tastatur

3. Daten

Laenge: ca. 50 Byte

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

6. Anwendung

Dieser Treiber bedient eine ueber eine serielle Schnittstelle (IFSS) angeschlossene Tastatur (K76xx).

7. Programmablauf, Bedienanleitung

Der Treiber besteht aus zwei Unterprogrammen:
Initialisierung und Abfrage.

Die Initialisierungsroutine hat folgende Parameter:

Eingangsparameter:
keine

Ausgangsparameter:
keine

Aufrufbedingungen:
UP-Name: TINIT
Die Register AF,BC und HL werden veraendert.

Programmablauf:
Diese Routine initialisiert den CTC fuer die Taktversorgung und die SIO fuer die Schnittstellenbedienung. Danach wird das Steuerzeichen 44h (Piep) an die Tastatur ausgegeben und als Antwort das Zeichen TYP von der Tastatur erwartet.
Die Abfrageroutine hat folgende Parameter:

Eingangsparameter:
keine

Ausgangsparameter:
Reg. A: Tastencode, wenn CY-Flag= 1
unbestimmt, wenn CY-Flag= 0

Aufrufbedingungen:
UP-Name: TASIP

Programmablauf:
Diese Routine prueft, ob im Empfangspuffer der SIO ein Zeichen vorhanden ist. Wenn ja, wird dieses Zeichen eingelesen.

8. Generierung

Im Modul EPOSCOM.typ (2.4.1.) koennen die I/O-Adressen der SIO (tport) und der CTC (tctc) verandert werden.

9. Einschraenkungen bei der Nutzung

Der Treiber unterstuezt (ausser beim Initialisierungs-Piep) keine Ausgabe von Steuerzeichen zur Tastatur.

1. Benennung

****
*
* 2.4.3.2. EPOSTAS2
*
****

2. Kurzbeschreibung

Geraetetreiber fuer eine ueber ein paralleles Port angeschlossene Tastatur

3. Daten

Laenge: ca. 40 Byte

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

6. Anwendung

Dieser Treiber bedient eine ueber eine parallele Schnittstelle angeschlossene Tastatur. Die von der Tastatur uebertragenen 8 Bit werden folgendermassen interpretiert:
Bit 0..6: ASCII-Code der Taste
Bit 7: =1 Taste gedrueckt (Code gueltig)
=0 keine Taste gedrueckt

7. Programmablauf, Bedienanleitung

Der Treiber besteht aus zwei Unterprogrammen: Initialisierung und Abfrage.

Die Initialisierungsroutine hat folgende Parameter:

Eingangsparameter:
keine

Ausgangsparameter:
keine

Aufrufbedingungen:
UP-Name: TINIT
Die Register AF,BC und HL werden veraendert.

Programmablauf:
Diese Routine initialisiert die PIO fuer die Schnittstellenbedienung.

Die Abfrageroutine hat folgende Parameter:

Eingangsparameter:
keine

Ausgangsparameter:
Reg. A: Tastencode, wenn CY-Flag= 1
unbestimmt, wenn CY-Flag= 0

Aufrufbedingungen:
UP-Name: TASIP
Das Register HL wird veraendert.

Programmablauf:
Diese Routine prueft mittels Bit 7 des Tastaturcodes und des bei der letzten Tastaturabfrage gelesenen Codes, ob eine Tastenbetaetigung vorliegt.

8. Generierung

Im Modul EPOSCOM.typ (2.4.1.) kann die I/O-Adresse der PIO (tport) veraendert werden.

9. Einschraenkungen bei der Nutzung

keine

1. Benennung

****
* *
* 2.4.3.3. EPOSTAS3 *
* *
****

2. Kurzbeschreibung

Geraetetreiber fuer eine ueber eine ATS K7028 angeschlossene parallele Tastatur K763x.

3. Daten

Laenge: ca. 50 Byte

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

6. Anwendung

Dieser Treiber realisiert den physischen Teil des Tastaturhandlings einer ueber ATS K7028 angeschlossene parallele Tastatur.

7. Programmablauf, Bedienanleitung

Der Treiber besteht aus zwei Unterprogrammen:
Initialisierung und Abfrage.

Die Initialisierungsroutine hat folgende Parameter:

Eingangsparameter:
keine

Ausgangsparameter:
keine

Aufrufbedingungen:
UP-Name: TINIT
Die Register AF,BC und HL werden veraendert.

Programmablauf:
Diese Routine setzt die Tastatur durch Ausgabe des entsprechenen Steuercodes zurueck und erwartet das Senden des TYP-Zeichens von der Tastatur.

Die Abfrageroutine hat folgende Parameter:

Eingangsparameter:
keine

Ausgangsparameter:
Reg. A: Tastencode, wenn CY-Flag= 1
unbestimmt, wenn CY-Flag= 0

Aufrufbedingungen:
UP-Name: TASIP

Programmablauf:

Diese Routine prueft an Hand eines Steuerbits, ob von der Tastatur ein Code bereitgestellt wurde. Wenn ja, wird es uebergeben. Gleichzeitig wird der CRTL-Status gesetzt.

8. Generierung

Im Modul EPOSCOM.typ (2.4.1.) kann die I/O-Adresse der ATS (tport) veraendert werden.

9. Einschraenkungen bei der Nutzung

Der Treiber unterstuetzt (ausser beim Initialisierungs-Reset) keine Ausgabe von Steuerzeichen zur Tastatur.

1. Benennung

****
*
* 2.4.3.4. EPOSTAS4
*
****

2. Kurzbeschreibung

Geraetetreiber fuer eine ueber ein serielles Port angeschlossene Tastatur K7672.03

3. Daten

Laenge: ca. 150 Byte

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

6. Anwendung

Dieser Treiber bedient eine ueber eine serielle Schnittstelle (IFSS) angeschlossene Tastatur (K7672.03).

7. Programmablauf, Bedienanleitung

Der Treiber besteht aus zwei Unterprogrammen:
Initialisierung und Abfrage.

Die Initialisierungsroutine hat folgende Parameter:

Eingangsparameter:
keine

Ausgangsparameter:
keine

Aufrufbedingungen:
UP-Name: TINIT
Die Register AF,BC,DE und HL werden veraendert.

Programmablauf:
Diese Routine initialisiert den CTC fuer die Taktversorgung und die SIO fuer die Schnittstellenbedienung. Danach werden die Steuerzeichen 1bh 'c'(Reset) an die Tastatur ausgegeben und als Antwort das Zeichen 11h (Ende Selbsttest) von der Tastatur erwartet. Dannach wird durch Ausgabe der entsprechenden Steuercodefolge der Scan-Mode eingeschaltet.
Die Abfrageroutine hat folgende Parameter:

Eingangsparameter:
keine

Ausgangsparameter:
Reg. A: Tastencode, wenn CY-Flag= 1
unbestimmt, wenn CY-Flag= 0

Aufrufbedingungen:
UP-Name: TASIP

Programmablauf:
Diese Routine prueft, ob im Empfangspuffer der SIO ein Zeichen vorhanden ist. Wenn ja, wird dieses Zeichen eingelesen. Dabei werden jeweils die beiden letzten Tastencodes zwischengespeichert. Wenn beim Einlesen ein SIO-Empfaenger-ueberlauf festgestellt wird, kann auf Grund der Spezifik der Tastatur K7672.02 als den vorletzten Code der durch den Ueberlauf vorlorengegangene Code rekonstruiért werden.

8. Generierung

Im Modul EPOSCOM.typ (2.4.1.) koennen die I/O-Adressen der SIO (tport) und der CTC (tctc) verandert werden.

9. Einschraenkungen bei der Nutzung

Der Treiber unterstuezt keine Ausgabe von Steuerzeichen zur Tastatur.
Der Treiber wird zusammen mit dem Modul EPOSSCAN.MAC benutzt.

1. Benennung

****
*
* 2.4.3.5. EPOSTAS5
*
****

2. Kurzbeschreibung

Geraetetreiber fuer eine ueber ein serielles Port angeschlossene Tastatur K7672.03 im Interruptbetrieb

3. Daten

Laenge: ca. 150 Byte

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

6. Anwendung

Dieser Treiber bedient eine ueber eine serielle Schnittstelle (IFSS) angeschlossene Tastatur (K7672.03). Die SIO wird dabei im Interrupt-Betrieb abgefragt.

7. Programmablauf, Bedienanleitung

Der Treiber besteht aus zwei Unterprogrammen, Initialisierung und Abfrage, und einer Interrupt-Service-Routine.

Die Initialisierungsroutine hat folgende Parameter:

Eingangsparameter:
keine

Ausgangsparameter:
keine

Aufrufbedingungen:
UP-Name: TINIT
Die Register AF,BC,DE und HL werden veraendert.

Programmablauf:
Diese Routine initialisiert den CTC fuer die Taktversorgung und die SIO fuer die Schnittstellenbedienung in der Interruptberiebsart. Danach werden die Steuerzeichen 1bh 'c'(Reset) an die Tastatur ausgegeben und als Antwort das Zeichen 11h (Ende Selbsttest) von der Tastatur erwartet. Dannach wird durch Ausgabe der entsprechenden. Steuercode-folge der Scan-Mode eingeschaltet.

Die Abfrageroutine hat folgende Parameter:

Eingangsparameter:
keine

Ausgangsparameter:
Reg. A: Tastencode, wenn CY-Flag= 1
unbestimmt, wenn CY-Flag= 0

Aufrufbedingungen:
UP-Name: TASIP

Programmablauf:
Diese Routine prueft, ob im Zwischenpuffer (Code-Puffer) ein Zeichen vorhanden ist. Wenn ja, wird dieses Zeichen eingelesen.

Die Interrupt-Service-Routine wird aktiviert, wenn im SIO-Empfangspuffer ein Code zur Abholung bereitsteht. Nach Registerretten, wird der Code von der SIO geholt und in einem Zwischenpuffer (Code-Puffer) abgelegt.

8. Generierung

Im Modul EPOSCOM.typ (2.4.1.) koennen die I/O-Adressen der SIO (tport) und der CTC (tctc) verandert werden.

9. Einschraenkungen bei der Nutzung

Der Treiber unterstuezt keine Ausgabe von Steuerzeichen zur Tastatur.

1. Benennung

****
* *
* 2.4.3.6. EPOSSCAN *
* *
****

2. Kurzbeschreibung

Verarbeitung der von der Tastatur K7672.03 gelieferten Scan-Codes

3. Daten

Laenge: ca. 200 Byte

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

6. Anwendung

Dieses Programm uebersetzt die Verarbeitung der von der Tastatur gelieferten Scan-Codes in ASCII-Codes. Dabei werden gedrueckte Sondertasten (Shift, Ctrl, Caps-Lock und Alt) beruecksichtigt.

7. Programmablauf, Bedienanleitung

Diese Routine schaltet sich zwischen die physische Tastaturroutine TASIP und die logische Tastenverarbeitung z.B. in EPOSCON2.
Zuerst versucht diese Routine einen Code von der Tastatur (Aufruf von TASIP) zu erlangen. Ist keiner vorhanden, wird die Routine mit rueckgesetzem CY-Flag verlassen. Ansonsten wird ueberprueft, ob es sich um ein Make-Code (Taste wurde gedrueckt) oder ein Break-Code (Taste wurde losgelassen) von einer der Sondertasten (Shift-right, Shift-left, Ctrl oder Alt) handelt. in diesem Falle werden entsprechende Merkzellen gesetzt bzw rueckgesetzt. Wenn es sich um keine Sondertaste handelt, wird der Tastencode in einen ASCII-Code gewandelt. Dabei werden, je nach Zustand von Shift und Caps-Lock verschiedene Codierungstabellen benutzt. Der gewonnene Code wird mit gesetztem CY-Flag an die uebergeordnete Routine uebergeben.

8. Generierung

Im Modul EPOSCOM.typ (2.4.1.) koennen die I/O-Adressen der SIO (tport) und der CTC (tctc) verandert werden.

9. Einschraenkungen bei der Nutzung

Der Treiber unterstuezt keine Ausgabe von Steuerzeichen zur Tastatur.

1. Benennung

****
* *
* 2.4.3.7. EPOSCON1 *
* *
****

2. Kurzbeschreibung

Modul zur logischen Verarbeitung der Tastencodes.

3. Daten

Laenge: ca. 85 Byte + Tastaturpuffer + Kodetabelle

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

- EPOSTASx.MAC Tastaturtreiber (2.4.3.1. - 2.4.3.5.)

6. Anwendung

EPOSCON1 verarbeitet die von der Routine **TASIP** gelieferten Tastaturcodes. Dabei wird eine wahlweise Gross- Klein-Wandlung (CAPS-LOCK) und eine Umcodierung ausgewaehlter Tasten durchgefuehrt.

7. Programmablauf, Bedienanleitung

Der Modul besteht aus zwei Unterprogrammen, DEFTA und TASIN. DEFTA ist hier nur eine Dummy-Routine (Kompatibilitaet zu EPOSCON2). Die Routine TASIN hat folgende Parameter:

Eingangsparameter:
keine

Ausgangsparameter:
keine

Aufrufbedingungen:
UP-Name: TASIN
Die Register AF,BC und HL werden veraendert.

Programmablauf:
Die Routine holt sich mittels TASIP den aktuellen Tastencode. Wenn zum Zeitpunkt des Aufrufs keine Taste gedrueckt war, wird TASIN verlassen. Sonst erfolgt eine Pruefung, ob die Mode-Umschaltung (CAPS-LOCK) gedrueckt war. Dann er-

folgt die Mode-Umschaltung (**tmode**). Fuer alle andern Tastencodes wird eine Gross- <=> Klein-Wandlung durchgefuehrt. Anschliessend wird noch eine Umcodierung bestimmter Tasten mittels der Tabelle **ktab** durchgefuehrt. Wird der Code in der Tabelle gefunden, so wird er durch den entsprechenden ersetzt.
Die Tastencodes der entsprechenden Tastaturen sind in EPOSTCOD.MAC abgelegt. Dadurch wird die Variabilitaet bei der Generierung verbessert.
Danach wird der Tastencode in den Tastaturpuffer geschrieben, wo er von den BIOS-Routine **CONST** und **CONIN** geprueft bzw. geholt werden kann. Es ist zweckmaessig, diese Routine zyklisch von einem Interrupt aufrufen zu lassen. So koennen unabhaengig vom Programmlauf Tastencodes im Puffer abgelegt werden.

8. Generierung

Folgende Variablen koennen veraendert werden:
ktlen: Laenge der Umcodierungstabelle
exmod: Tastencode zur Mode-Umschaltung (CAPS-LOCK)

9. Einschraenkungen bei der Nutzung

keine

1. Benennung

****
*
* 2.4.3.8. EPOSCON2
*
****

2. Kurzbeschreibung

Modul zur logischen Verarbeitung der Tastencodes.

3. Daten

Laenge: ca. 200 Byte + Tastenpuffer + Kodetabelle

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

- EPOSTASx.MAC Tastaturtreiber (2.4.3.1. - 2.4.3.5.)

6. Anwendung

EPOSCON2 verarbeitet die von der Routine **TASIP** gelieferten Tastaturcodes. Dabei wird eine wahlweise Gross- Klein-Wandlung (CAPS-LOCK) und eine Umcodierung ausgewaehlter Tasten durchgefuehrt. Diese Umcodierung schliesst eine Zuordnung von Zeichenketten (STRINGS) auf einen Tastencode ein.

7. Programmablauf, Bedienanleitung

Der Modul besteht aus zwei Unterprogrammen, DEFTA und TASIN. DEFTA dient der Definition von STRING-Tasten waehrend der Laufzeit des Programmes. Diese Routine ist ueber BIOS-Ruf CONOUT zu erreichen und hat folgende Parameter:

Eingangsparameter:
Reg. C: Steuer-Zeichen

Ausgangsparameter:
Z-Flag=0: Das Zeichen in Reg. C ist kein Zeichen fuer eine Stringtastendefinition und muss an die Bildschirmroutine weitergegeben werden.
Z-Flag=1 Das Zeichen ist verarbeitet worden.

Aufrufbedingungen:
UP-Name: DEFTA
Die Register AF,BC, DE und HL werden veraendert.

Programmablauf:
Dem Programm muessen fuer eine neue Tastendefinition folgende Zeichen nacheinander im Reg. C uebergeben werden:
- Steuerzeichen zur Einleitung einer Tastendefinition (11h)
- Tastencode
- String (1..n Zeichen, auch Steuerzeichen)
- 00h (Ende Umcodierungs-Sequenz).

Wird anstelle des Tastencodes 00h (Ruecksetzen) uebergeben, dann werden alle nutzerdefinierten Tastenzuordnungen gestrichen und es wird wieder die Standard-Tabelle (Systemdefinitionen) benutzt.

Die Routine TASIN hat folgende Parameter:

Eingangsparameter:
keine

Ausgangsparameter:
keine

Aufrufbedingungen:
UP-Name: TASIN
Die Register AF,BC,DE und HL werden veraendert.

Programmablauf:
Die Routine prueft ob eine String-Tastenausgabe laueft. Wenn ja, dann wird das naechste Zeichen in den Tastenpuffer geschrieben und der String-Ausgabezeiger weitergerueckt.
Im anderen Fall holt sich die Routine mittels TASIP den aktuellen Tastencode. Wenn zum Zeitpunkt des Aufrufs keine Taste gedrueckt war, wird TASIN verlassen. Sonst erfolgt eine Pruefung, ob die Mode-Umschaltung (CAPS-LOCK) gedrueckt war. Dann erfolgt die Mode-Umschaltung (**tmode**). Fuer alle andern Tastencodes wird eine Gross- <=> Klein-Wandlung durchgefuehrt. Anschliessend wird eine Umcodierung bestimmter Tasten mittels der Tabelle **ktab** durchgefuehrt. Dabei wird die Tabelle von hinten durchsucht, um einen Vorrang der Nutzerdefinitionen vor den Systendefinitionen zu gewaehrleisten. Wird der Code in der Tabelle gefunden, so wird der String-Ausgabezeiger auf die zugehoerige Tastenkombination eingestellt.
Die Tastencodes der entsprechenden Tastaturen sind in EPOSTCOD.MAC abgelegt. Dadurch wird die Variabilitaet bei der Generierung verbessert.
Es ist zweckmaessig, diese Routine zyklisch von einem Interrupt aufrufen zu lassen. So koennen unabhaengig vom Programmlauf Tastencodes im Puffer abgelegt werden.

8. Generierung

Folgende Variablen koennen veraendert werden:
cdeft: Steuerzeichen zu Einleitung einer Definition
ktlen: Laenge der Umcodierungstabelle
exmod: Tastencode zur Mode-Umschaltung (CAPS-LOCK)

9. Einschraenkungen bei der Nutzung

keine

1. Benennung

****
*
* 2.4.3.9. EPOSCON3
*
****

2. Kurzbeschreibung

Modul zur logischen Verarbeitung der Tastencodes.

3. Daten

Laenge: ca. 350 Byte + Tastenpuffer + Kodetabelle

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

- EPOSTASx.MAC Tastaturtreiber (2.4.3.1. - 2.4.3.5.)

6. Anwendung

EPOSCON3 verarbeitet die von der Routine **TASIP** gelieferten Tastaturcodes. Dabei wird eine Umcodierung der von der Tastatur gelieferten Scan-Codes durchgefuehrt. Jede Taste kann unabhaengig in vier Ebenen (UnShift, Shift, Ctrl, Alt) belegt werden. Diese Umcodierung schliesst eine Zuordnung von Zeichenketten (STRINGS) auf einen Tastencode ein. Im Gegensatz zu EPOSCON2 ist hier die Umdefinition einzelner Tasten nicht der Regelfall, vielmehr wird mittels eines Dressing-Files die gesamte Codetabelle erneuert. Das Dressing-File wird mittels des Tastatur-Dressing-Compiler TACOM aus einem Text-File erzeugt. So ist ein nutzerfreundliches Tastatur-Dressing moeglich.

7. Programmablauf, Bedienanleitung

Der Modul besteht aus zwei Unterprogrammen, DEFTA und TASIN. DEFTA dient der Definition von STRING-Tasten waehrend der Laufzeit des Programmes. Diese Routine ist ueber BIOS-Ruf CONOUT zu erreichen und hat folgende Parameter:

Eingangsparameter:
Reg. C: Steuer-Zeichen

Ausgangsparameter:
Z-Flag=0: Das Zeichen in Reg. C ist kein Zeichen fuer eine Stringtastendefinition und muss an die Bildschirmroutine weitergegeben werden.
Z-Flag=1 Das Zeichen ist verarbeitet worden.

Aufrufbedingungen:
UP-Name: DEFTA
Die Register AF,BC, DE und HL werden veraendert.

Programmablauf:
Dem Programm muessen fuer eine neue Tastendefinition folgende Zeichen nacheinander im Reg. C uebergeben werden:
- Steuerzeichen zur Einleitung einer Tastendefinition (11h)
- Ruecksetzen der internen Zeiger (0f0h)

- Kennbyte der nachfolgenden Tastendefinition
- Tastencode
- String (1..n Zeichen, auch Steuerzeichen)
  (die letzten drei Anstriche koennen beliebig oft wiederholt werden).

- 0f0h (Ende Umcodierungs-Sequenz).

Die Routine TASIN hat folgende Parameter:

Eingangsparameter:
keine

Ausgangsparameter:
keine

Aufrufbedingungen:
UP-Name: TASIN
Die Register AF,BC,DE und HL werden veraendert.

Programmablauf:
Die Routine prueft, ob eine String-Tastenausgabe laueft. Wenn ja, dann wird das naechste Zeichen in den Tastenpuffer geschrieben und der String-Ausgabezeiger weitergerueckt.
Im anderen Fall holt sich die Routine mittels TASIP den aktuellen Tastencode. Wenn zum Zeitpunkt des Aufrufs keine Taste gedrueckt war, wird TASIN verlassen.
Sonst erfolgt ein Ausfiltern und Umsetzen besimmter Mehrfachtastencodes (die Tastatur liefert bei einigen Tasten eine Codefolge, diese werden auf einen Code reduziert). Anschliessend wird die Codetabelle durchsucht. Dabei werden die Ebenenbits (Kennzeichen fuer aktuelle Ebene) und die Back-Scan-Codes (diese liefert die Tastatur beim Loslassen der Tasten) bestimmter Tasten beruecksichtigt. Danach werden Sondertasten, die die Ebenenbits beeinflussen (Shift, Ctrl, CapsLock und Alt) bearbeitet. Andere in der Codetabelle gefundene Codes werden zur Abholung durch die BIOS-Rufe CONST und CONIN im Tastenpuffer abgespeichert.
Es ist zweckmaessig, diese Routine zyklisch von einem Interrupt aufrufen zu lassen. So koennen unabhaengig vom Programmlauf Tastencodes im Puffer abgelegt werden.

8. Generierung

Folgende Variablen koennen veraendert werden:
cdeft: Steuerzeichen zu Einleitung einer Definition
ktlen: Laenge der Umcodierungstabelle

9. Einschraenkungen bei der Nutzung

Bei direkter Angabe des Zeichencodes duerfen keine Werte >=0F0h angegeben werden. Diese Einschraenkung verkuerzt die Code-Tabelle und damit das Betriebssytem um bis zu 400 Byte.
Diser Treiber ist nicht im Nachnutzungsumfang von EPOS enthalten.

1. Benennung

****
*
* 2.4.3.10. TACOM
*
****

2. Kurzbeschreibung

TACOM realisiert im Zusammenhang mit dem logischem Tastaturtreiber EPOSCON3 ein komfortables, flexibles Tastaturdressing. TACOM erzeugt aus einem Textfile ein ausfuehrbares Programm.

3. Daten

Laenge: ca 13,5 kByte
Laufzeit: < 20 sec

4. Programmiersprache

Turbo-Pascal V3.01 oder kompatible

5. Genutzte Module

- (incl) HEX.INS Hexadezimal-ASCII-Konvertierungen

6. Anwendung

Das Programm TACOM (Tastaturdressing-Compiler) dient zur Aufbereitung von Programmen zur Modifizierung der Tastaturbelegung. Im Betreibssystem EPOS (speziell EPOSCON3) gibt es dazu eine Routine zur Modifizierung der Tastaturzuordnungs-Tabelle. Diese Routine wird ueber die ConOut-Schnittstelle des BIOS ereicht.
TACOM erzeugt aus einen Text-File (Tastaturdressing-File) ein COM-File, welches zu jedem beliebigem Zeitpunkt aufgerufen werden kann und die aktuelle Tastaturbelegung ueberschreibt.

7. Programmablauf, Bedienanleitung

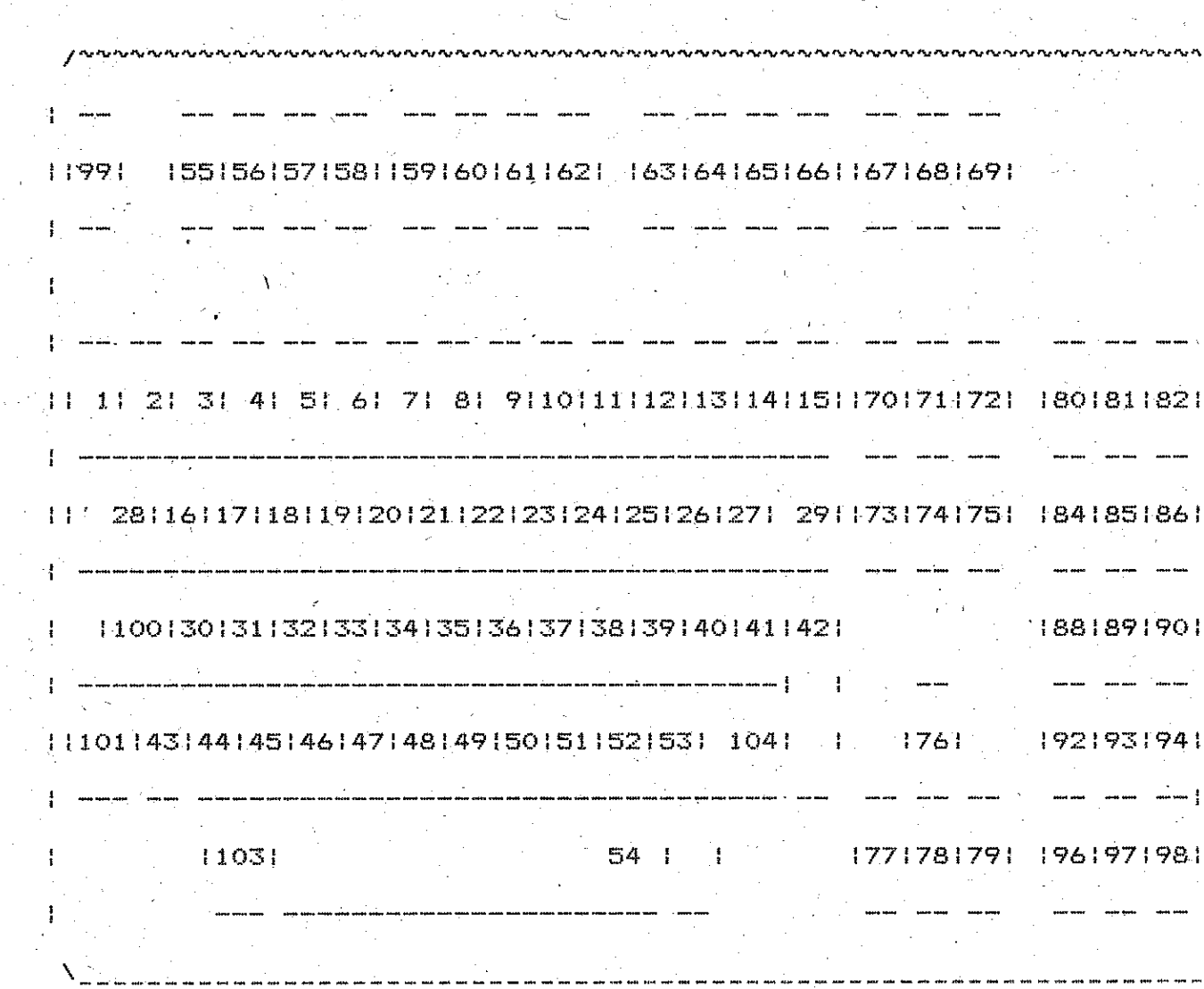
Aufruf von TACOM

TACOM benoetigt als Parameter den Namen des Tastaturdressing-Files. Er kann als Parameter in der Befehlszeile oder nach der Aufforderung "Name Text-Datei: " in der Form "name.ext" angegeben werden.
Als Produkt der Verarbeitung entsteht ein File "name.COM".

Aufbau eines Tastaturdressing-Files

In jeder Zeile wird eine Taste definiert. Die Tastenbelegungen werden in aufsteigender Reihenfolge entsprechend der Tastennummern definiert. Im unten folgenden Beispiel sind

die Tastennummern am Ende der Zeilen lediglich Kommentar. Den Tasten der Tastatur sind folgende Tastennummern zugewiesen:



Jede Taste kann in 4 Ebenen (Unshift, Shift, ALT, CTRL) belegt werden. Jede Ebene kann mit beliebig vielen Zeichen im 8-Bit-Code bzw. einer Sonderfunktion belegt werden. Die Laenge der Code-Tabelle insgesamt wird durch den im Treiber EPOSCON3.MAC reservierten Platz begrenzt. Die Belegung einer Ebene wird mit mind. einem Leerzeichen bzw. Tabulator abgeschlossen. Nicht belegte Ebenen erhalten keine Codierung. Nicht erfassbare Zeichen (z.B. CTRL-Zeichen), Zeichen mit Sonderbedeutung (Leerzeichen, Tabulator und NZ sowie #) und Tastensonderfunktionen werden wie folgt umschrieben:

Zeichenumschreibungen:

#(n)	- Direktangabe Zeichencode (n = 0...239) hexadezimal #(hh) oktal #(0ooo)
#@,#A...#Z,#[,#\,#],#^,#_	- Controlzeichen #(0) ... #(1f)
##	- #
#lz, #sp	- Leerzeichen (LZ)
#nz	- Neue Zeile (NZ)
#esc	- #(033) (=Escape)

#backtab — Rueckwaertstabulator

Sonderfunktionen:

#nc	- Kein Code
#base	- Belegung wie Unshift (sollte immer, wenn moeglich verwendet werden, optimiert die Codetabelle)

#shift_r, #shift_l	- rechte/linke Schifttaste
#alt, #ctrl	- ALT-Taste/Control-Taste
#caps	- Caps-Lock-Taste
#scroll, #pause	- Anhalten/Fortsetzen der BS-Ausgabe
#print_screen	- Ausdrucken des Bildschirminhaltes bzw. Paralellschalten des Drueckers zur Konsol-Ausgabe (^P)
#NZ	- Zeilenverlaengerung (Ignorieren des NZ (CR))
#	- Rest der Zeile ist Kommentar. Steht der Kommentar am Zeilenbeginn (1. Zeichen), so wird auch das NZ ignoriert.
#bsa,#bst,#bsn	- Schalten bei Verwendung der Bildschirmtreibers EPOSTTY3 die Sonderzeichenbloecke ein oder aus (sind keine Tastencodes, Angabe optional).

Fehlermeldungen von TACOM

Bei Fehlern im Dressing-Textfile wird der Compilerlauf mit einer Fehlermeldung abgebrochen.

Folgende Fehlermeldungen sind moeglich:

1: mehr als vier Ebenen
Es sind mehr als vier Ebenen fuer eine Taste angegeben.

2: #... kein verwertbarer Code
Hinter dem # steht ein Code, der nicht dedeutet werden kann.

3: falsche Oktalangabe

4: falsche Hexaangabe

5: Code groesser 0f0h
Bei direkter Codeangabe ist ein Wert >=0F0h gefunden worden.

Abhilfe: In EPOSCON3.MAC ist die Laenge der Tabelle zu vergroessern (KTLEN). In TACOM.PAS sind ELen und BufLen entsprechen zu aendern.

7: Sondertaste nicht in UnShift-Ebene

Bestimmte Sondertasten wirken in allen Ebenen und sind deshalb in der UnShift-Ebene anzugeben (betrifft Caps, CTRL, Shift, ALT).

8: base in UnShift-Ebene

#base kann nicht in der UnShift-Ebene stehen.

Beispiel_fuer_ein_Tastaturdressing-Textfile

```
#                                                          Tasten-
# Unshift       Shift           Supershift      Ctrl       nummer
#
 #bst
 #esc           #base           #base           #base      # 1
 1              !                                          # 2
 2              "                                          # 3
 3              @                                          # 4
 4              ¤                                          # 5
 5              %                                          # 6
 6              &                                          # 7
 7              /                                          # 8
 8              (                                          # 9
 9              )                                          # 10
 0              =                                          # 11
 ß              ?                                          # 12
 ´              `                                          # 13
 #H             #base           #base           #base      # 14
 #(7f)          #base           #base           #base      # 15
 q              Q               #nc             #Q         # 16
 w              W               #nc             #W         # 17
 e              E               #nc             #E         # 18
 r              R               #nc             #R         # 19
 t              T               #nc             #T         # 20
 z              Z               #nc             #Z         # 21
 u              U               #nc             #U         # 22
 i              I               #nc             #I         # 23
 o              O               #nc             #O         # 24
 p              P               #nc             #P         # 25
 ü              }               #nc             #]         # 26
 +              *               #nc             #nc        # 27
 #ctrl                                                     # 28
 #I             #base           #base           #base      # 29
 a              A               #nc             #A         # 30
 s              S               #nc             #S         # 31
 d              D               #nc             #D         # 32
 f              F               #nc             #F         # 33
 g              G               #nc             #G         # 34
 h              H               #nc             #H         # 35
 j              J               #nc             #J         # 36
 k              K               #nc             #K         # 37
 l              L               #nc             #L         # 38
 ö              |               #nc             #          # 39
```

```
#nz             #base           #base           #base     # 42
<               >               #nc             #nc       # 43
y               Y               #nc             #Y        # 44
x               X               #nc             #X        # 45
c               C               #nc             #C        # 46
v               V               #nc             #V        # 47
b               B               {               #B        # 48
n               N               }               #N        # 49
```



```
m               M               #nc             #M        # 50
,               ;                                         # 51
.               :                                         # 52
-               _                                         # 53
#sp             #sp             #sp             #nc       # 54
#nc                                                       # 55
#nc                                                       # 56
#nc                                                       # 57
pw#nz       #dienst         #power                        # 58
ws                                                        # 59
subm                                                      # 60
programm;#nz#Ibegin#nz#nz#Iend;                           # 61
#nc                                                       # 62
#nc                                                       # 63
#nc                                                       # 64
#nc                                                       # 65
#nc                                                       # 66
#print_screen                                             # 67
#nc                                                       # 68
#pause                                                    # 69
#nc                                                       # 70
#R              #base           #Q#R            #Q#R      # 71
#C              #base           #Q#C            #Q#C      # 72
#Q#X            #base           #base           #base     # 73
#Q#E            #base           #base           #base     # 74
#nc                                                       # 75
#E              #base           #Q#E            #Q#E      # 76
#H              #base           #A              #A        # 77
#X              #base           #Q#X            #Q#X      # 78
#D              #base           #F              #F        # 79
#nc                                                       # 80
/               #base           #C              #(0177)   # 81
*               #base                                     # 82
-               #base                                     # 83
7               #base                                     # 84
8               #base                                     # 85
9               #base                                     # 86
+               #base                                     # 87
4               #base                                     # 88
5               #base                                     # 89
6               #base                                     # 90
=               #base                                     # 91
1               #base                                     # 92
2               #base                                     # 93
3               #base                                     # 94
#nz             #base           #base           #base     # 95
0               #base                                     # 96
,               #base                                     # 97
.               #base                                     # 98
```

```
#caps                                   # 100
#shift_l                                # 101
#shift_r                                # 102
#alt                                    # 103
```


EPOS.DOK VEB DVZ Rostock/BT EPMR/Abt. REM 26.01.90

8. Generierung

In TACOM.PAS koennen im Bedarfsfall die Konstanten ELen und BufLen geaendert werden (siehe auch Fehler 6).

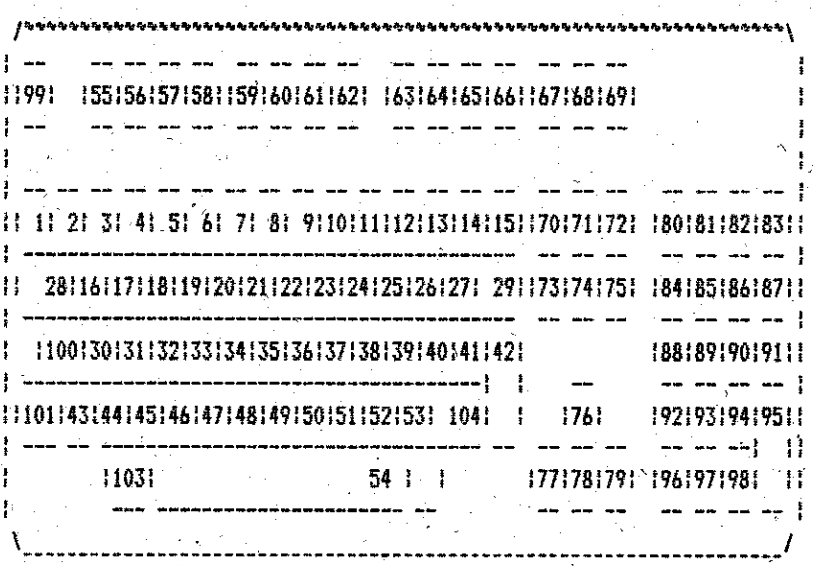
Das Programm ist mittels TurboPascal zu compilieren.
z.B.

```
A>turbo          :TurboPascal aufrufen
  y              :Fehlertexte laden
  o              :Options aendern
  c              :Ziel Diskette
  e a000         :Endadresse auf A000h herabsetzen
  q              :Dieses Menue verlassen
  c              :Compilieren
  q              :TurboPascal verlassen
```

9. Einschraenkungen bei der Nutzung

Bei direkter Angabe des Zeichencodes duerfen keine Werte >=0F0h angegeben werden. Diese Einschraenkung verkuertze die Code-Tabelle und damit das Betriebssytem um bis zu 400 Byte.
Dises Programm ist nicht im Nachnutzungsumfang von EPOS enthalten.

die Tastennummern am Ende der Zeilen lediglich Kommentar. Den Tasten der Tastatur sind folgende Tastennummern zugewiesen:



Jede Taste kann in 4 Ebenen (Unshift, Shift, ALT, CTRL) belegt werden. Jede Ebene kann mit beliebig vielen Zeichen im 8-Bit-Code bzw. einer Sonderfunktion belegt werden. Die Laenge der Code-Tabelle insgesamt wird durch den im Treiber EPOSCON3.MAC reservierten Platz begrenzt. Die Belegung einer Ebene wird mit mind. einem Leerzeichen bzw. Tabulator abgeschlossen. Nicht belegte Ebenen erhalten keine Codierung. Nicht erfassbare Zeichen (z.B. CTRL-Zeichen), Zeichen mit Sonderbedeutung (Leerzeichen, Tabulator und NZ sowie #) und Tastensonderfunktionen werden wie folgt umschrieben:

Zeichenumschreibungen:

#(n)	- Direktangabe Zeichencode (n = 0...239) hexadezimal #(hh) oktal #(Oooo)
#@,#A...#Z,#[,#\,#],#^,#_	- Controlzeichen #(0) ... #(1f)
##	- #
#lz, #sp	- Leerzeichen (LZ)
#nz	- Neue Zeile (NZ)
#esc	- #(033) (=Escape)
#eof	- EOF (End Of File) #(1a)
#backtab	- Rueckwaertstabulator

Sonderfunktionen:

#nc	- Kein Code
#base	- Belegung wie Unshift (sollte immer, wenn moeglich verwendet werden, optimiert die Codetabelle)

#shift_r, #shift_l	- rechte/linke Schifttaste
#alt, #ctrl	- ALT-Taste/Control-Taste
#caps	- Caps-Lock-Taste
#scroll, #pause	- Anhalten/Fortsetzen der BS-Ausgabe
#print_screen	- Ausdrucken des Bildschirminhaltes bzw. Paralellschalten des Drueckers zur Konsol-Ausgabe (^P)
#NZ	- Zeilenverlaengerung (Ignorieren des NZ (CR))
#	- Rest der Zeile ist Kommentar. Steht der Kommentar am Zeilenbeginn (1. Zeichen), so wird auch das NZ ignoriert.
#bsa,#bst,#bsn	- Schalten bei Verwendung der Bildschirmtreibers EPOSTTY3 die Sonderzeichenbloecke ein oder aus (sind keine Tastencodes, Angabe optional).

Fehlermeldungen_von_TACOM

Bei Fehlern im Dressing-Textfile wird der Compilerlauf mit einer Fehlermeldung abgebrochen.

Folgende Fehlermeldungen sind moeglich:

1: mehr als vier Ebenen
Es sind mehr als vier Ebenen fuer eine Taste angegeben.

2: #... kein verwertbarer Code
Hinter dem # steht ein Code, der nicht dedeutet werden kann.

3: falsche Oktalangabe

4: falsche Hexaangabe

5: Code groesser 0f0h
Bei direkter Codeangabe ist ein Wert >=0F0h gefunden worden.

6: Tabelle zu lang
Der Gesamtumfang der entstehenden Tabelle ist zu lang. Abhilfe: In EPOSCON3.MAC ist die Laenge der Tabelle zu vergroessern (KTLEN). In TACOM.PAS sind ELen und BufLen entsprechen zu aendern.

7: Sondertaste nicht in UnShift-Ebene
Bestimmte Sondertasten wirken in allen Ebenen und sind deshalb in der UnShift-Ebene anzugeben (betrifft Caps, CTRL, Shift, ALT).

8: base in UnShift-Ebene

#base kann nicht in der UnShift-Ebene stehen.

Beispiel_fuer_ein_Tastaturdressing-Textfile

# Unshift	Shift	Supershift	Ctrl	Tasten-nummer
#bst				
#esc	#base	#base	#base	# 1
1	!			# 2
2	"			# 3
3	@			# 4
4	$			# 5
5	%			# 6
6	&			# 7
7	/			# 8
8	(			# 9
9	)			# 10
0	=			# 11
ß	?			# 12
´	`			# 13
#H	#base	#base	#base	# 14
#(7f)	#base	#base	#base	# 15
q	Q	#nc	#Q	# 16
w	W	#nc	#W	# 17
e	E	#nc	#E	# 18
r	R	#nc	#R	# 19
t	T	#nc	#T	# 20
z	Z	#nc	#Z	# 21
u	U	#nc	#U	# 22
i	I	#nc	#I	# 23
o	O	#nc	#O	# 24
p	P	#nc	#P	# 25
ü	}	#nc	#]	# 26
+	*	#nc	#nc	# 27
#ctrl				# 28
#I	#base	#base	#base	# 29
a	A	#nc	#A	# 30
s	S	#nc	#S	# 31
d	D	#nc	#D	# 32
f	F	#nc	#F	# 33
g	G	#nc	#G	# 34
h	H	#nc	#H	# 35
j	J	#nc	#J	# 36
k	K	#nc	#K	# 37
l	L	#nc	#L	# 38
\	;	#nc	#_	# 39
[	{	#nc	#^	# 40
##	~	#nc	#nc	# 41
#nz	#base	#base	#base	# 42
<	>	#nc	#nc	# 43
y	Y	#nc	#Y	# 44
x	X	#nc	#X	# 45
c	C	#nc	#C	# 46
v	V	#nc	#V	# 47
b	B	{	#B	# 48
n	N	}	#N	# 49

m	M	#nc	#M	# 50
,	;			# 51
.	:			# 52
-	_			# 53
#sp	#sp	#sp	#nc	# 54
#nc				# 55
#nc				# 56
#nc				# 57
pw#nz	#dienst	#power		# 58
ws				# 59
subm				# 60
programm;#nz#Ibegin#nz#nz#Iend;				# 61
#nc				# 62
#nc				# 63
#nc				# 64
#nc				# 65
#nc				# 66
#print_screen				# 67
#nc				# 68
#pause				# 69
#nc				# 70
#R	#base	#Q#R	#Q#R	# 71
#C	#base	#Q#C	#Q#C	# 72
#Q#X	#base	#base	#base	# 73
#Q#E	#base	#base	#base	# 74
#nc				# 75
#E	#base	#Q#E	#Q#E	# 76
#H	#base	#A	#A	# 77
#X	#base	#Q#X	#Q#X	# 78
#D	#base	#F	#F	# 79
#nc				# 80
/	#base	#C	#(0177)	# 81
*	#base			# 82
-	#base			# 83
7	#base			# 84
8	#base			# 85
9	#base			# 86
+	#base			# 87
4	#base			# 88
5	#base			# 89
6	#base			# 90
=	#base			# 91
1	#base			# 92
2	#base			# 93
3	#base			# 94
#nz	#base	#base	#base	# 95
0	#base			# 96
,	#base			# 97
.	#base			# 98
#ctrl				# 99
#caps				# 100
#shift_l				# 101
#shift_r				# 102
#alt				# 103

8. Generierung

In TACOM.PAS koennen im Bedarfsfall die Konstanten ELen und BufLen geaendert werden (siehe auch Fehler 6).

Das Programm ist mittels TurboPascal zu compilieren.
z.B.

```
A>turbo        :TurboPascal aufrufen
  y            :Fehlertexte laden
  o            :Options aendern
  c            :Ziel Diskette
  e a000       :Endadresse auf A000h herabsetzen
  q            :Dieses Menue verlassen
  c            :Compilieren
  q            :TurboPascal verlassen
```

9. Einschraenkungen bei der Nutzung

Bei direkter Angabe des Zeichencodes duerfen keine Werte >=0F0h angegeben werden. Diese Einschraenkung verkuertze die Code-Tabelle und damit das Betriebssytem um bis zu 400 Byte.
Dises Programm ist nicht im Nachnutzungsumfang von EPOS enthalten.

2.4.4. Floppy

2.4.4.1. EPOSRF

1. Benennung

2.4.4.2. EPOSRF1

2. Kurzbeschreibung

Treiber fuer ein 192-kByte-RAM-Floppy auf einer 256-Kbyte-DRAM-Karte (NANOS).

3. Daten

Laenge: ca. 60 Byte + DPB (24) + Allocationsmap (33)

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

6. Anwendung

Der Treiber wird benutzt, um auf einer 256-kByte-DRAM-Karte aus dem NANOS-Baugruppensystem eine 192K-RAM-Floppy-Simulation zu realisieren. Die verbleibenden 64 kByte dieser Karte werden als Hauptspeicher genutzt. Die RAM-Floppy hat folgendes Format:

Kapazitaet:	190 kByte		
Spuren:	96	1 System	
Sektoren/Spur:	16		
Sektoren/System:	16	16 fuer DIR	
DIR-Eintraege:	64	2 kByte	
Sectoren/Block:	8	1 kByte	188 Bloecke
kByte/Eintrag:	16 kByte		

7. Programmablauf, Bedienanleitung

Das Programm RFRWOPER hat folgende Parameter:

Eingangsparameter:
Reg. B: 0: Lesen
1: Schreiben
(sektrk): Spur
(seksek): Sektor
(dmaadr): DMA-Adresse

Ausgangsparameter:
Reg. A: 0 (kein Fehler)

Aufrufbedingungen:
UP-Name: RFRWOPER
Die Register BC,DE, und HL werden veraendert.

Programmablauf:
Nach Umrechnen von Spur- und Sectornummer in Fenster- und Bankadresse wird der Interrupt gesperrt. Dies ist notwendig, damit waehrend der Umschaltung des hinter dem Fenster liegenden Speicherbereiches kein Interrupt seinen evtl. in diesem Bereich liegenden Code benoetigt. Nach dem Uebertagen des Sektors (128 Byte) wird der urspruengliche Fensterinhalt wiederhergestellt und der Interrupt wieder zugelassen.

8. Generierung

Die I/O-Adresse (**IOA**) und die Fensteradresse (**WIND**) werden in EPOSCOM.typ (2.4.1.) festgelegt.

9. Einschraenkungen bei der Nutzung

Der Code von RFRWOPER darf nicht innerhalb des Fensterbereiches liegen.

1. Benennung

****
*
* 2.4.4.3. EPOSRF2
*
****

2. Kurzbeschreibung

Treiber fuer ein 256-kByte-RAM-Floppy auf einer 256-Kbyte-DRAM-Karte (NANOS).

3. Daten

Laenge: ca. 60 Byte + DPB (24) + Allocationsmap (33)

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

6. Anwendung

Der Treiber wird benutzt, um auf einer 256-kByte-DRAM-Karte aus dem NANOS-Baugruppensystem eine RAM-Floppy-Simulation zu realisieren. Die RAM-Floppy hat folgendes Format:

Kapazitaet:	254 kByte		
Spuren:	128	1 System	
Sektoren/Spur:	16		
Sektoren/System:	16	16 fuer DIR	
DIR-Eintraege:	64	2 kByte	
Sectoren/Block:	8	1 kByte	252 Bloecke
kByte/Eintrag:	16 kByte		

7. Programmablauf, Bedienanleitung

Das Programm RFRWOPER hat folgende Parameter:

Eingangsparameter:
Reg. B: 0: Lesen
1: Schreiben
(sektrk): Spur
(seksek): Sektor
(dmaadr): DMA-Adresse

Ausgangsparameter:
Reg. A: 0 (kein Fehler)

Aufrufbedingungen:
UP-Name: RFRWOPER
Die Register BC,DE, und HL werden veraendert.

Programmablauf:
Nach Umrechnen von Spur- und Sektornummer in Fenster- und Bankadresse wird der Interrupt gesperrt. Dies ist notwendig, damit nach der Aktivierung der Baugruppe (MEMDI) kein Interrupt seinen evtl. hinter dem Fenster liegenden Code benoetigt. Nach dem Uebertragen des Sektors (128 Byte) wird der urspruengliche Fensterinhalt wiederhergestellt (Deaktivieren der Baugruppe) und der Interrupt wieder zugelassen.

8. Generierung

Die I/O-Adresse (IOA) und die Fensteradresse (WIND) werden in EPOSCOM.typ festgelegt.

9. Einschraenkungen bei der Nutzung

Der Code von RFRWOPER darf nicht innerhalb des Fensterbereiches liegen. Der Hauptspeicher muss durch eine weitere Speicherbaugruppe realisiert werden.

1. Benennung

****
*
* 2.4.4.4. EPOSRF3
*
****

2. Kurzbeschreibung

Treiber fuer ein 512-kByte-RAM-Floppy mittels zweier 256-Kbyte-DRAM-Karten (NANOS).

3. Daten

Laenge: ca. 130 Byte + DPB (24) + Allocationsmap (33)

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

6. Anwendung

Der Treiber wird benutzt, um mittels zwei 256-kByte-DRAM-Karten aus dem NANOS-Baugruppensystem eine RAM-Floppy-Simulation zu realisieren. Die RAM-Floppy hat folgendes Format:

Kapazitaet:	510 kByte		
Spuren:	256	1 System	
Sektoren/Spur:	16		
Sektoren/System:	16	16 fuer DIR	
DIR-Eintraege:	64	2 kByte	
Sectoren/Block:	16	2 kByte	254 Bloecke
kByte/Eintrag:	16 kByte		

7. Programmablauf, Bedienanleitung

Das Programm RFRWOPER hat folgende Parameter:

Eingangsparameter:
Reg. B: 0: Lesen
1: Schreiben
(sektrk): Spur
(seksek): Sektor
(dmaadr): DMA-Adresse

Ausgangsparameter:
Reg. A: 0 (kein Fehler)

Aufrufbedingungen:
UP-Name: RFRWOPER
Die Register BC,DE, und HL werden veraendert.

Programmablauf:
Zuerst erfolgt das Umrechnen von Spur- und Sectornummer in Fenster- und Bankadresse. Dabei werden die unteren Spuren in der ersten und die oberen in der zweiten Baugruppe gelesen bzw. gschrieben. Dann wird der Interrupt gesperrt. Dies ist notwendig, damit nach der Aktivierung der Baugruppe (MEMDI) kein Interrupt seinen evtl. hinter dem Fenster liegenden Code benoetigt. Nach dem Uebertragen des Sektors (128 Byte) wird der urspruengliche Fensterinhalt wiederhergestellt (Deaktivieren der Baugruppe) und der Interrupt wieder zugelassen.

8. Generierung

Die I/O-Adressen (**IOA** und **IOA2**) und die Fensteradresse (**WIND**) werden in EPOSCOM.typ (2.4.1.) festgelegt.

9. Einschraenkungen bei der Nutzung

Der Code von RFRWOPER darf nicht innerhalb des Fensterbereiches liegen. Der Hauptspeicher muss durch eine weitere

Speicherbaugruppe realisiert werden.

1. Benennung

****
*
* **2.4.4.5. EPOSRF4**
*
****

2. Kurzbeschreibung

Treiber fuer ein 448-kByte-RAM-Floppy mittels zweier 256-Kbyte-DRAM-Karten (NANOS).

3. Daten

Laenge: ca. 130 Byte + DPB (24) + Allocationsmap (33)

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

6. Anwendung

Der Treiber wird benutzt, um mittels zwei 256-kByte-DRAM-Karten aus dem NANOS-Baugruppensystem eine RAM-Floppy-Simulation zu realisieren. Die RAM-Floppy hat folgendes Format:

Kapazitaet:	446 kByte		
Spuren:	224	1 System	
Sektoren/Spur:	16		
Sektoren/System:	16	16 fuer DIR	
DIR-Eintraege:	64	2 kByte	
Sectoren/Block:	16	2 kByte	220 Bloecke
kByte/Eintrag:	16 kByte		

7. Programmablauf, Bedienanleitung

Das Programm RFRWOPER hat folgende Parameter:

Eingangsparameter:
Reg. B: 0: Lesen
1: Schreiben
(sektrk): Spur
(seksek): Sektor
(dmaadr): DMA-Adresse

Ausgangsparameter:
Reg. A: 0 (kein Fehler)

Aufrufbedingungen:
UP-Name: RFRWOPER
Die Register BC,DE, und HL werden veraendert.

Programmablauf:
Zuerst erfolgt das Umrechnen von Spur- und Sectornummer in Fenster- und Bankadresse. Dabei werden die unteren Spuren in der ersten und die oberen in der zweiten Baugruppe gelesen bzw. gschrieben. Dann wird der Interrupt gesperrt. Dies ist notwendig, damit nach der Aktivierung der Baugruppe (MEMDI) kein Interrupt seinen evtl. hinter dem Fenster liegenden Code benoetigt. Nach dem Uebertragen des Sektors (128 Byte) wird der urspruengliche Fensterinhalt wiederhergestellt (Deaktivieren der Baugruppe) und der Interrupt wieder zugelassen.

8. Generierung

Die I/O-Adressen (**IOA** und **IOA2**) und die Fensteradresse (**WIND**) werden in EPOSCOM.typ (2.4.1.) festgelegt.

9. Einschraenkungen bei der Nutzung

Der Code von RFRWOPER darf nicht innerhalb des Fensterbereiches liegen. Eine Realisierung einer 448 kByte-RAM-Floppy und des Hauptspeichers mittels zwei 256-kByte-

Baugruppen ist nur mittels Schaltungsaenderungen moeglich.

1. Benennung

****
* *
* **2.4.4.6. EPOSBLK** *
* *
****

2. Kurzbeschreibung

Programm zum geblockten Zuriff auf Disketten-Sektoren

3. Daten

Laenge: ca. 390 Byte

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

- EPOSFDC.MAC Floppy-Treiber (2.4.13.)

6. Anwendung

Dieser Modul dient zur Effektivitaetssteigerung bei Diskettenzugriffen. Dabei werden von und zur Diskette jeweils eine groessere Datenmenge (im allg. 1024 Byte) bewegt. Alle anderen Schreib- und Leseoperationen laufen dann im Speicher ab.
Bei jedem Aufruf des BIOS-WRITE-Eintrittspunktes gibt das BDOS die noetigen Informationen fuer eine effektive Sektor-Blockbildung fuer Systeme, bei denen die Sektorgroesse ein Vielfaches der Grundeinheit von 128 Byte darstellt. EPOSBLK.MAC ist ein allgemeiner Algorithmus fuer die Auswertung der BDOS-Informationen zur automatischen Blockbildung.
Bei jedem Aufruf von WRITE legt das BDOS im Register C folgende Information ab:

0 - normales Schreiben eines Sektors
1 - Schreiben eines Inhaltsverzeichnis-Sektors
2 - Schreiben des ersten Sektors eines neuen Datenblockes

Die Bedingung 0 tritt beim Schreiben in ein vorher beschriebenes Feld auf, wie z.B. beim Erneuern der Information im wahlfreien Modus, beim Schreiben in einen Sektor, der nicht der erste in einem Block ist und nicht in das Inhaltsverzeichnis geschrieben wird. Bedingung 1 tritt auf, wenn ein Sektor des Inhaltsverzeichnisses geschrieben werden soll. Bedingung 2 tritt beim Schreiben des ersten Sektors eines noch nicht plazierten Datenblockes auf. In

den meisten Faellen werden Sektoren mit mehrmals 128 Byte in einer Folge gelesen oder geschrieben, und damit ist nur geringer hoeherer Aufwand mit der Blockbildung der Datensaetze verbunden, da Leseoperationen vor dem Schreiben vermieden werden koennen.
Allgemein werden alle Leseoperationen ueber 1024 Byte grossen einen Zwischenpuffer ausgefuehrt. Innerhalb des Programms sind Werte und Variablen, welche sich auf den CP/M-Sektor beziehen, mit dem Praefix "SEK," versehen und solche, die sich auf die Diskette beziehen, enthalten den den Praefix "HST".
An dieser Stelle folgen die Beschreibungen der Disk-orientierten BIOS-Rufe (diese befinden sich in der Datei EPOSBIOS.MAC):
Obwohl das SELDSK-Programm die Disketten-Parameter (DPB) ermittelt und uebergibt, wird an diesem Punkt der Hintergrundspeicher nicht selektiert (das geschieht spaeter bei READHST oder WRITEHST). Weiterhin speichern SETTRK, SETSEC und SETDMA die Parameter nur in Systemzellen ab und loesen keine weitere Aktion aus. SECTRAN erfuellt die triviale Funktion der Uebergabe der physischen Sektornummer.
Die prinzipiellen Eintrittspunkte sind READBLK und WRITEBLK. Diese Unterprogramme ersetzen die normalen READ- bzw. WRITE-Routinen ohne Blockbildung.
Das physische Schreiben oder Lesen findet bei WRITEHST oder READHST statt, wo alle noetigen Parameter gesetzt sind: HSTDSK ist die Laufwerksnummer des Systems, HSTTRK ist die Spurnummer und HSTSEC ist die Sektornummer, welche aber noch eine Konvertierung (UP PARAM) in eine physische Sektornummer erfordert. Alle anderen Funktionen der Blockbildung werden vom gegebenen Algorithmus ausgefuehrt.

7. Programmablauf, Bedienanleitung

Die Routinen WRITEBLK und READBLK setzen Parameter in den Zellen SEKDSK, SEKTRK, SEKSEC und DMAAD voraus. Bei der Routine WRITEBLK muss das Register C wie o.g. gesetzt sein.

8. Generierung

Folgende Vereinbarungen koennen veraendert werden:
HSTSIZ: Puffergroesse (im allg. 1024)

SECSHF: =ld(HSTSIZ/128)
(ld = 2er Logarithmus)

9. Einschraenkungen bei der Nutzung

keine

1. Benennung

****
*
* **2.4.4.7. EPOSFDC**
*
****

2. Kurzbeschreibung

EPOSFDC.MAC ist ein Treiber zur Ansteuerung von Floppy-Disk-Laufwerken mittels einer NANOS-FDC (U8272).

3. Daten

Laenge: ca. 580 Byte + DPB (n*24)

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

6. Anwendung

Der Treiber EPOSFDC dient zur Ansteuerung von 8"- und 5+1/4"-Laufwerken. Hardwaremaessig ist die Floppy-Disk-Controler-Karte aus dem NANOS-Baugruppensystem notwendig. Kernstueck dieser Baugruppe ist der FDC U8272 (siehe Anwender-Dokumentation NANOS-FDC und Technische Beschreibung U8272).
Softwareseitig werden alle vom U8272 unterstuetzten Formate realisiert. Der Treiber ist fuer den Einsatz in CP/M-Systemen entwickelt worden, kann aber auch in beliebigen anderen Systemen genutzt werden.

7. Programmablauf, Bedienanleitung

Der Treiber beinhaltet folgende Komplexe:
- **Disk-Parameter-Bloecke (DPB)**, bestehend aus dem CP/M-typischem und dem hardwareabhaengigen Teil.
- **Motor-Aus-Routine:** Diese muss zyklisch angesprungen werden, zum Beispiel von einer CTC-Iinterruptservice-Routinen, um ca. 4 sec nach dem letzten Zugriff auf die Disk den Motor des Laufwerkes auszuschalten.
- **Floppy-Treiber** mit den Funktionen RESET, STATUS-LESEN, LESEN, SCHREIBEN, INDEX-LESEN und FORMATIEREN.

Disk-Parameter-Block
Fuer den Floppy Treiber ist im Register IX der DPB zu uebergeben. Die Byte haben folgende Bedeutung:

```
db      41h-t1w         ;(0) initbyte:
                        ;bit0 =1: mini-floppy
                        ;       0: 8''-floppy
                        ;bit4 =0:1 step/track 1w dd, disk sd
                        ;bit4 =1:2 step/track 1w = disk
                        ;bit5    (nicht benutzt )
                        ;bit6 =0:Praekomp. ein ab prae (1.2)
                        ;bit6 =1:keine Praekomp.
                        ;bit7 =0:1w nicht initialis.
db      040h            ;(1) mt, mfm-(fm)-cmd
db      3               ;(2) n (voreinstellung 1kbyte
db      6               ;(3) eot (voreinstellung 1kbyte
db      0eh             ;(4) gpl (voreinstellung 1kbyte
db      30h             ;(5) gpl3
db      0e5h            ;(6) Daten beim Formatieren
db      25              ;(7) prae Nr. der Spur fuer
                            Praekompensation
db      0efh            ;(8) SRT= 4 ms, HUT= 480 ms
db      003h            ;(9) HLT= 4 ms, non DMA
```

Die **Motor-Aus-Routine** wird in eine 25ms-CTC-Interrupt-Service-Routine eingebaut und schaltet nach (ausv)-maligem Aufruf den Laufwerk-Motor aus. Wenn SYSINT equ 0 gesetzt ist, wird der CTC fuer die Motorabschaltung im Floppytreiber selbst initialisiert.
Eingangsparameter: keine
Ausgangsparameter: keine
Aufrufbedingungen:
UP-Name: MOTOUT
Das UP veraendert den Inhalt vom Reg. A
Durch Veraendern des Wertes von **AUSV** kann die Ausschalt-Verzoegerung variiert werden.

Der **Floppy-Treiber** hat folgende Parameter:

Eingangsparameter:

Funktion: RESET
Reg A: 0

Funktion: STATUS-LESEN
Reg A: 1

Funktion: READ
Reg A: 2
Reg B: Sectoren-Counter
Reg C: Drive/Head (bit 0, 1: Drive bit 2: Head)
Reg D: Track
Reg E: Sector
Reg HL:Buffer
Reg IX:DPB

Funktion: WRITE
Reg A: 3
Reg B: Sectoren-Counter
Reg C: Drive/Head (bit 0, 1: Drive bit 2: Head)
Reg D: Track
Reg E: Sector
Reg HL:Buffer
Reg IX:DPB

Funktion: FORMAT_A_TRACK
Reg A: 5
Reg C: Drive/Head (bit 0, 1: Drive bit 2: Head)
Reg D: Track
Reg HL:Buffer (enthaelt pro Sector C,H,R und N)
Reg IX:DPB

Funktion: READ_ID
Reg A: 6
Reg C: Drive/Head (bit 0, 1: Drive bit 2: Head)
Reg D: Track
Reg IX:DPB

Die Parameter haben folgende Bedeutung:
- Sector-Counter: Anzahl der einzulesenden Sektoren (maximal EOT-1 = eine Spur)
- Drive/Head: Bit 0 und 1 sind binaer verschlusselt die physische Laufwerksnummer (Kanal), mit Bit 2 wird Head 0 oder 1 ausgewaehlt.
- Track: Spurnummer wie in ID-Feld (Identifikationsfeld) der Diskette (z.B. 0..79).
- Sector: Sektornummer (1..EOT-1)
- Buffer: Adresse des Speicherbereiches von/zu dem der Datentransfer stattfinden soll.
- DPB: Zeiger auf den Disketten-Parameter-Block
- C, H, R, N: Beim Formatieren muessen im Puffer fuer jeden zu formatierenden Sektor folgende Daten fuer das ID-Felder zu Verfuegung gestellt werden: C- Spur, H- Kopf, R- Sektor, N- Sektorleange. Damit ist eine beliebige Sectorreihenfolge moeglich.

Ausgangsparameter:
Reg A: Fehlercode
0: o.k.
1: nicht bereit
2: ID nicht gefunden
3: Schreib-/Lesefehler
4: Falsche Funktion aufgerufen
Reg HL:
Fkt 1: Zeiger auf Statusinformation des FDC 8272
Fkt 2, 3, 5: Naechstes zu schreibendes/lesendes Byte (weitergerueckter Pufferzeiger)

Aufrufbedingungen:
UP-Name: FLOPPY
Die Register BC, DE und HL veraendern ihren Inhalt.

Programmablauf

Das Treiberprogramm wird im folgenden in einer PASCAL-aehnlichen Pseudo-Code-Notation skizziert.

```
floppy:
   IF fkt=reset THEN DReset;          /FDC zuruecksetzen
   If fkt=status THEN hl:=result;     /Status des FDC
   Spezify-Parameter enistellen
   IF not init THEN initlw;           /Laufwerk initial.
   cmd-tab einrichten;                /Kommando-Tabelle f. FDC
   seek;                              /Spur einstellen
   CASE fkt OF
     READ_ID: read_id;                /ID Lesen
     READ:    rwoper(6);              /RW/WR-Operation
     WRITE:   IF track > dpb[prael
              THEN preacomp on        /Praekompensation
              rwoper(5);              /FDC-Cmd WRITE
     FORMAT:  cmd-tab einrichten
              moto(13);               /Laufwerk einschalten
                                      /mit FDC-Cmd FORMAT
              daten ausgeben;
              result;                 /Status vom FDC
   END_CASE;

UP initlw:                            /Laufwerk initial.
   SET init;
   moto(7)                            /FDC-Cmd RECALIBRATE
   interrupt-status holen;            /SENSE-INTERRUPT-STATUS

UP seek:                              /Spur einstellen
   spur umrechnen;
   moto(15)                           /FDC-Cmd SEEK
   interrupt-statur holen;

UP moto(cmd):                         /Laufwerk einschalten
   abschaltuhr aufziehen              /fuer MOTOUT-Routine
   laufwerk einschalten;
   laufwerk-status holen;             /SENSE-DRIVE-STATUS
   IF not ready THEN LEAVE;           /Fehlerabbruch
   fdc:=cmd;                          /Cmd zum FDC

UP result:                            /Resultat vom FDC
   FOR n:=1 to 7
    . resltb[n]:=fdc;                 /Status vom FDC holen

UP rwoper(cmd):                       /RW/WR-Operation
   transfer-programm vorbereiten
 next:
   FOR n:=1 to reprw                  /Wiederholungen
     moto(cmd);                       /Cmd zum FDC
     IF not ready THEN next;          /naechster Versuch
     transfer;                        /Daten vom/zum FDC
     result;                          /Resultat vom FDC
     IF err THEN next;
   END;
   praekomp off;                      /Praekompensation

UP raead_id                           /ID lesen
```

```
next:
 FOR n:=1 to repid                    /Wiederholungen
    moto(10)                          /Cmd zum FDC
    result;                           /Resultat vom FDC
    IF err THEN next;
 END;
```

8. Generierung

Neben der Aenderung der I/O-Adresse **UPDPRT** in EPOSCOM.typ (2.4.1) koennen folgende Vereinbarungen variiert werden:

```
;
;Vereinbarungen
;-----------------------------------------------
repid    equ     5        ;Wiederholungen readid
reprw    equ     10       ;      -"-      read/write
maxsec   equ     1024     ;max. Sectorlaenge
max_cmd  equ     7        ;Max. CMD-Nr+1
;
```

Ausserdem kann mit der Vereinbarung **FKON** kann der Funktionsumfang der Floppy-Treibers variiert werden. Die Bits dieser Vereindarung haben dabei folgende Bedeutung:

```
;fkon    equ 02h ; Konfigurationsbits Floppy-Treiber
                 ; (muss ausserhalb vereinbart werden)
                 ; Bit 0: Read ID
                 ; Bit 1: Format
                 ; Bit 2: DPBs
```

9. Einschraenkungen bei der Nutzung

Auf Grund des zeitkritischen Daten-Transfers und der deshalb notwendigen Modifizierung des Transfer-Programms fuer unterschiedliche Sektorlaengen und Lesen/Schreiben ist der Floppy-Treiber nur im RAM lauffaehig.

2.4.5. Drucker

1. Benennung

2.4.5.1. EPOSLST1

2. Kurzbeschreibung

Treiberprogramm fuer einen Drucker im DC1/DC3-Protokoll

3. Daten

Laenge: ca. 175 Byte

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

6. Anwendung

Dieses Programm dient zur Bedienung eines Druckers mit DC1/DC3-Protokoll. Er kann ueber eine serielle Standard-schnittstelle (IFSS oder V.24) angeschlossen werden. Der Treiber unterstuetzt Schnittstellen-Realisierungen mittels SIO und CTC. Der Protokollablauf ist dabei folgender: Ueber seinen Sende-Kanal sendet der Drucker folgende Steuer-signale.
DC1: Drucker ist bereit zum Empfang weiterer Druck-Zeichen
DC3: Druckerpuffer ist voll, warten bis DC1
DC4: Drucker will Status senden (z.B. Fehlermeldungen)

7. Programmablauf, Bedienanleitung

Der Druckertreiber besteht aus zwei Unterprogrammen **LST** und **LSTST** (entspechend BIOS). Die Routine LSTST erlaubt die Abfrage des Druckerstatus und hat folgende Parameter:

Eingangsparameter:
keine

Ausgangsparameter:
Reg A: Status A=0: Drucker nicht bereit
A=1: 1 Drucker bereit

Aufrufbedingungen:
UP-Name: LSTST
Die Register AF,BC,DE und HL werden veraendert.

Programmablauf:
Mittels der vom Drucker gesendeten Steuersignale wird der Drucker-Status bestimmt. Wenn der Drucker Fehlermeldungen senden will (DC4), wird dieser eingelesen, aber nicht weiter ausgewertet. Vor dem ersten Aufruf von LST ist der Drucker immer bereit.

Die Routine LST gestattet die Ausgabe eines Zeichens zum Drucker und hat folgende Parameter:

Eingangsparameter:
Reg C: auszugebendes Zeichen (es bestehen keine Einschraenkungen bezueglich des Inhaltes von C).

Ausgangsparameter:
keine

Aufrufbedingungen:
UP-Name: LST
Die Register AF,BC,DE und HL werden veraendert.

Programmablauf:
Zuerst wird der Drucker-Status bestimmt (LSTST). Wenn der Drucker bereit ist, wird das Zeichen zum Drucker uebertragen und die Routine beendet. Wenn der Drucker nicht bereit ist, wird zyklisch weiter abgefragt. Nach 0ffffh-maliger Status-Abfrage kann durch Tastenbetaetigung ein Warmstart ausgeloest werden.

8. Generierung

Die I/O-Adresse der SIO **PPRT** und der CTC **PCTC** werden in EPOSCOM.typ (2.4.1.) eingestellt.

9. Einschraenkungen bei der Nutzung

keine

1. Benennung

****
*
* 2.4.5.2. EPOSLST2
*
****

2. Kurzbeschreibung

Treiberprogramm fuer einen Drucker im DTR-Protokoll

3. Daten

Laenge: ca. 110 Byte

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

6. Anwendung

Dieses Programm dient zur Bedienung eines Druckers mit DTR-Protokoll. Er kann ueber eine serielle Standardschnittstelle V.24 angeschlossen werden. Der Treiber unterstuetzt Schnittstellen-Realisierungen mittels SIO und CTC. Der Protokollablauf ist dabei folgender: Ueber seine (Rueck-) Meldeleitung DTR signalisiert der Drucker seinen Status. Diese Leitung wird an den CTS-Eingang der SIO angeschlossen. Die Pegel haben folgende Bedeutung:

inact (high): Drucker ist bereit zum Empfang weiterer Druck-Zeichen
act (low): Druckerpuffer ist voll, warten bis Bereitschaft gemeldet wird

7. Programmablauf, Bedienanleitung

Der Druckertreiber besteht aus zwei Unterprogrammen **LST** und **LSTST** (entspechend BIOS). Die Routine LSTST erlaubt die Abfrage des Druckerstatus und hat folgende Parameter:

Eingangsparameter:
keine

Ausgangsparameter:
Reg A: Status A=0: Drucker nicht bereit
A=1: 1 Drucker bereit

Aufrufbedingungen:
UP-Name: LSTST

Die Register AF,BC,DE und HL werden nicht veraendert.

Programmablauf:
Mittels der vom Drucker gesendeten Steuersignale wird der Drucker-Status bestimmt. Vor dem ersten Aufruf von LST ist der Drucker immer bereit.

Die Routine LST gestattet die Ausgabe eines Zeichens zum Drucker und hat folgende Parameter:

Eingangsparameter:
Reg C: auszugebendes Zeichen (es bestehen keine Einschraenkungen bezueglich des Inhaltes von C).

Ausgangsparameter:
keine

Aufrufbedingungen:
UP-Name: LST
Die Register AF,BC,DE und HL werden veraendert.

Programmablauf:
Zuerst wird der Drucker-Status bestimmt (LSTST). Wenn der Drucker bereit ist, wird das Zeichen zum Drucker uebertragen und die Routine beendet. Wenn der Drucker nicht bereit ist, wird zyklisch weiter abgefragt. Nach Offffh-maliger Status-Abfrage kann durch Tastenbetaetigung ein Warmstart ausgeloest werden.

8. Generierung

Die I/O-Adresse der SIO **PPRT** und der CTC **PCTC** werden in EPOSCOM.typ (2.4.1.) eingestellt.

9. Einschraenkungen bei der Nutzung

keine

1. Benennung

****
* *
* 2.4.6. EPOSRAM *
* *
****

2. Kurzbeschreibung

Nichtinitialisierte RAM-Bereiche

3. Daten

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

6. Anwendung

Dieser Modul enthaelt alle nichtinitialisierten RAM-Bereiche fuer das BIOS. Es sind vor allem RAM-Zellen fuer die Floppy-Arbeit

7. Programmablauf, Bedienanleitung

Der Modul legt nur Adressen fest und liefert keinen Code.

8. Generierung

Es ist keine Generierung notwendig. Der Umfang von EPOSRAM.MAC wird automatisch von den eingestellten Parametern in EPOSCOM.typ.

9. Einschraenkungen bei der Nutzung

keine

3. Systemprogramme

3.1. DISK

1. Benennung

3.1.1. EPOSFMT

2. Kurzbeschreibung

Formatierprogramm fuer softsektorierte Disketten

3. Daten

Laenge: 1.5 kByte

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

- EPOSCOM.typ Generierdaten
- EPOSFDC.MAC Floppy-Treiber

6. Anwendung

Dieses Programm schafft die Moeglichkeit, Disketten zu formatieren, um sie fuer die Arbeit mit EPOS benutzbar zu machen. Die Formate sind kompatibel zu aenderen CP/M-Implementationen. Es sind folgende Formate moeglich:

1: 5 x 1024 x 80 x 2
2: 16 x 256 x 80 x 2
3: 10 x 512 x 80 x 2
4: 9 x 512 x 80 x 2
5: 16 x 256 x 40 x 1 (1.6)
6: 10 x 256 x 40 x 1 (1.2)

Das Format 5 x 1024 x 80 x 2 ermoeglicht die beste Diskettenauslastung und wird als EPOS-Standard erklaert.

Andere Formate koennen problemlos generiert werden. Dazu muessen das MACRO **FORMAT** in EPOSCOM und das Programm EPOSFMT entsprechend geaendert werden.

7. Programmablauf, Bedienanleitung

Das Programm laeuft menuegesteuert ab.

Aufruf:
A>eposfmt
Nach Aufruf des Programmes erfolgt die Anzeige der Ueberschrift
Disk-Format EPOS V1.3 EPMR/he 1/90
Kanal (0..3): _ Hier muss die Kanal-Nummer (physische Laufwerksnummer) des Laufwerks mit der zu formatierenden Diskette angegeben werden.

Anschlissend erfolgt eine Uebersicht der moeglichen Formate.
Auswahl: _ Hier ist die dem gewuenschten Format zugeordnete Nummer einzugeben.

Da es auch moeglich ist, nur Teile der Diskette neu zu formatieren, efolgt nun eine Abfrage der ersten und letzten zu formatierenden Spur. ENTER bestaetigt die vorgeschlagenen Werte (gesamte Diskette).
Welche Spuren sollen formatiert werden ?
1. Spur : 00
letzte Spur: 79

Jetzt wird die Aufforderung, die Diskette einzulegen und eine Warnung, dass alle auf der Diskette stehenden Informationen verloren gehen, ausgegeben.
Diskette einlegen
Alle Dateien werden zerstoert !! Weiter(J/N): _
Weiter geht es nur mit "J".

Nun wird die Diskette formatiert. Dabei wird die gerade bearbeitete Spur angezeigt. Nach dem Formatieren wird ein Prueflesen durchgefuehrt. Defekte Spuren werden angezeigt.
Nach erfolgreicher Formatierung steht die folgende Meldung auf dem Bildschirm:
Defekte Spuren: keine

Wenn noch weitere Disketten zu formatieren sind, ist die folgende Frage mit "J" zu beantworten.
Widerholung (J/N):

8. Generierung

In der Datei EPOSCOM.typ (2.4.1.) werden die Floppy-Parameter eingestellt. Die Laufwerk-Format-Zuordnung ist aber fuer EPOSFMT nicht relevant.

Weitere Verarbeitung:

Assemblieren: ASM =EPOSFMT
Linken : LINK EPOSFMT,EPOSFMT/N/E

9. Einschraenkungen bei der Nutzung

keine

1. Benennung

****
*
* 3.1.2. PUTSYS
*
****

2. Kurzbeschreibung

Systemspuren einer Diskette einrichten

3. Daten

Laenge: 1.4 kbyte

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

- EPOSPSYS benutzt BIOS- und BDOS-Rufe

6. Anwendung

Das Programm gestattet:

- eine Datei (i.A.EPOStyp.COM) auf den Systemspuren einer Diskette abzulegen. Dort wird sie vom Urlader EPOSUL beim Systemstart gelesen und gestartet.

- den Inhalt der Systemspuren in eine Datei abzulegen

- den Inhalt der Systemspuren einer Diskette auf die Systemspuren einer anderen Diskette zu uebertragen

Zwischen den einzelnen Operationen ist jeweils ein Diskettenwechsel moeglich. Entfaellt die Laufwerksangabe, so wird als Standard das aktuelle Laufwerk genommen.

7. Programmablauf, Bedienanleitung

Die Parameteruebergabe findet ueber die Aufrufzeile statt. Folgende Aufrufe sind moeglich:

- A>PUTSYS [d1:]datei[.ext] d2:
  d1: ist der Laufwerksname des Quellaufwerks der Datei datei.ext. Diese Laufwerksangabe ist optional.
  d2: ist der Name des Laufwerks mit der einzurichtenden Diskette.

  Wird .ext nicht angegeben, so wird .com angenommen.

Beispiel:
A>PUTSYS EPOSO2.COM B:
EPOSO2.COM wird vom aktuellen Laufwerk gelesen und auf die Systemspuren von Laufwerk B abgelegt.

- A>PUTSYS d1: [d2:]datei.ext
  d1: ist der Laufwerksname des Quellaufwerks
  d2: ist der Laufwerksname des Ziellaufwerks der Datei datei.ext. Diese Laufwerksangabe ist optional.

  Beispiel:
  A>PUTSYS A: B:EPOSO2.COM
  Der Inhalt der Systemspuren von Laufwerk A wird gelesen und in die Datei EPOSO2.COM auf Laufwerk B abgelegt. Die Datei wird vorher neu angelegt.

- A>PUTSYS d1: d2:
  d1: ist der Laufwerksname des Quellaufwerks
  d2: ist der Laufwerksname des Ziellaufwerks

  Beispiel:
  A>PUTSYS A: B:
  Der Inhalt der Systemspuren von Laufwerk A wird gelesen und auf die Systemspuren von Laufwerk B abgelegt.

Fehlermeldungen:

'File not found':	Datei nicht gefunden (falscher Name, falsches Laufwerk)
'Select Error':	Laufwerksangabe nicht moeglich
'R/W-Error':	Schreib/Lesefehler
'No System-Tracks':	Laufwerksformat laesst keine Systemspuren zu
'File exsists':	Datei existiert schon
'Dir full':	Directory ist voll
'i.O.':	Funktion fehlerfrei ausgefuehrt

8. Generierung

Es ist keine Veraenderung moeglich.

Weitere Verarbeitung:

Assemblieren: ASM =EPOSPSYS
Linken: LINK EPOSPSYS,PUTSYS/N/E

9. Einschraenkungen bei der Nutzung

keine

1. Benennung

****
*
* 3.1.3. FORMATE
*
****

2. Kurzbeschreibung

- Einstellen von Disk-Formaten
- Umlegung der RAM-Floppy
- Aenderung Zuordnung physisches LW - logisches LW
- Informationen

3. Daten

Laenge: 16 kbyte (davon 8 kbyte Laufzeitbibliothek)

4. Programmiersprache

TURBO - Pascal 3.0

5. Genutzte Module

- TURBO-Pascal
- BIOS- und BDOS-Rufe

6. Anwendung

Das Programm gestattet das logische Einstellen bzw. Aendern von Disk-Formaten fuer die einzelnen logischen Laufwerke.
Des Weiteren kann die RAM-Floppy - sofern vorhanden - umgelegt werden, d.h. sie kann einem anderen logischen Laufwerk zugeordnet werden.
Im Zusammenhang damit kann auch eine Aenderung der Zuordnung logisches Laufwerk (Laufwerke A B ...) - physische Laufwerksnummer (Kanaele 0 1 ...) vorgenommen werden.
Ausserdem koennen auch noch spezielle Informationen ueber die einzelnen logischen Laufwerke und die dort eingestellten Diskformate eingeholt werden.
Alle Aenderungen, die durch das Programm "FORMATE" vorgenommen werden, bleiben jeweils bis zum naechsten Kaltstart erhalten.

7. Programmablauf, Bedienanleitung

1. **Einfacher Aufruf:**
   A>FORMATE [1] <ET>
   1 Angabe nur bei 8''-Laufwerken

   Die Auswahl der einzelnen Funktionen erfolgt ueber das Funktionsmenue.

   Beispiel:
   A>FORMATE 1
   Die folgenden Operationen beziehen sich auf 8''-Laufwerke.

2. Aufruf mit Parameteruebergabe:

A>FORMATE <LW> <Form.Nr.> [1] <ET>

lw:	ist der Laufwerksname
Form.Nr.	ist die Nr.des auszuwaehlenden Formats
	0 - DD - DS 5 * 1024
	1 - DD - DS 16 * 256
	2 - DD - DS 10 * 512
	3 - DD - DS 9 * 512
	4 - SD - SS 16 * 256 (1.6)
	5 - SD - SS 16 * 256 (1.2)
1	Angabe nur bei 8''-Laufwerken

Beispiel:
A>FORMATE b 1
Es erfolgt die Einstellung des Formats DD-DS 16*256 fuer Laufwerk B.

Fehlermeldungen:

'Falsche Laufwerksangabe': Laufwerksangabe nicht moeglich

8. Generierung

Es ist keine Veraenderung moeglich.

Weitere Verarbeitung:

Compilieren in TURBO - Pascal 3.0

9. Einschraenkungen bei der Nutzung

Formate mit 128 Byte/Sektor sind auf Grund der Spezifik des FDC 8272 bei MFM (d.h. 5 1/4"-Disketten) nicht moeglich.

3.2. NET

1. Benennung

3.2.1. NLOAD

2. Kurzbeschreibung

Laden einer Datei von einem Dateiserver im SCOM-LAN

3. Daten

Laenge: 3 kbyte

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

- NLOAD.MAC Quellprogramm NLOAD
- (incl) NETCOM.MAC Allgemeine Vereinbarungen NET
- (incl) NSCCU1.MAC Protokoll Dateiuebertragung NET
- (link) NIOS.REL SCOM-LAN-Teiber

NLOAD benutzt BDOS-Rufe

6. Anwendung

Dieses Programm gestattet, eine Datei von einem Laufwerk des Dateiservers in einem SCOM-LAN auf ein Laufwerk des eigenen Rechners zu transferieren. Die Netz-Daten (eigene Stationsadresse, Stationsadresse des Servers und Passwort) sind fest eingestellt und koennen nur durch Quelltextaenderungen oder Patches veraendert werden.

7. Programmablauf, Bedienanleitung

Die Parameteruebergabe findet ueber die Aufrufzeile statt.
Aufruf
A>NLOAD [d1:]datei.ext
d1: ist der Laufwerksname des Quellaufwerkes beim Datei-Server. Diese Laufwerksangabe ist optional. datei.ext ist der vollstaendige Dateiname.

Beispiel:
A>NLOAD C:TEST.MAC
TEST.MAC wird vom Laufwerk C des Dateiservers gelesen und auf Laufwerk A des eigenen Rechners abgelegt.

Fehlermeldungen:

'uebertragen'	fehlerfreie Ausfuehrung
'Diskette voll'	Diskette im Ziellaufwerk ist voll
'Protokollfehler'	Bei der Uebertragung ist ein Protokollfehler aufgetreten (Stoerung, Defekt)
'Passwortfehler'	Das Passwort ist vom Server nicht akzeptiert worden
'Funktion nicht moeglich'	Server kennt die gewuenschte Funktion nicht
'Teilnehmer nicht vorhanden'	Teilnehmer ist nicht am Netz
'Datei nicht vorhanden'	Datei existiert nicht beim Server/ auf dem Laufwerk
'allgemeiner Fehler'	Fehler kann nicht spezifiziert werden

8. Generierung

In der Datei NETCOM.MAC koennen die eigene Adresse **GEADR**, die Serveradresse **DBADR** und die Modul- (Hardware-) Adresse veraendert werden. In NLOAD.MAC kann das Passwort **PASSWO** eingestellt werden.

Weitere Verarbeitung:

Assemblieren: ASM =NLOAD
Linken: LINK NLOAD,NIOS,NLOAD/N/E

Aenderung der Parameter durch Patches:
Der Anfang von NLOAD hat folgendes Aussehen:

```
        JP START

SIODA:  DEFB MODUL      ;SIO DATA A
SIOCA:  DEFB MODUL+2    ;SIO CTRL A
CTC0:   DEFB MODUL+4    ;CTC 0
CTC3:   DEFB MODUL+7    ;CTC 3
TNA:    DEFB GEADR      ;eigene Adresse
DADR:   DEFB DBADR      ;Datenbankadresse
PASSWO:DEFB '       '   ;Passwort

START:
        LD SP,STACK
              :
              :
```

Mit DU (ZSID) koennen diese Parameter geaendert werden. Die Parameter haben dabei folgende Lage:

```
100h +3   1 Byte    SIODA     ;SIO DATA A
     +4   1 Byte    SIOCA     ;SIO CTRL A
     +5   1 Byte    CTC0      ;CTC 0
     +6   1 Byte    CTC3      ;CTC 3
```

```
      +7    1 Byte     TNA         ;eigene Adresse
      +8    1 Byte     DADR        ;Datenbankadresse
      +9    8 Byte     PASSWO      ;Passwort
```

Beispiel fuer ein Patch der eigenen Adresse:

```
A>du nload.com<et>
#s107<et>
0107 12 10<et>
0108 01 .<et>
^c
A>save 11 nload.com<et>
```

Die eigene Adresse wurde von 18 (12h) auf 16(10h) geaendert.

9. Einschraenkungen bei der Nutzung

keine

1. Benennung

****
*
* 3.2.2. NSAVE
*
****

2. Kurzbeschreibung

Speichern einer Datei auf einem Dateiserver im SCOM-LAN

3. Daten

Laenge: 3 kbyte

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

- NSAVE.MAC Quellprogramm NLOAD
- (incl) NETCOM.MAC Allgemeine Vereinbarungen NET
- (incl) NSCCU1.MAC Protokoll Dateiuebertragung NET
- (link) NIOS.REL SCOM-LAN-Teiber

NSAVE benutzt BDOS-Rufe

6. Anwendung

Dieses Programm gestattet, von einem Laufwerk des eigenen Rechners eine Datei auf ein Lauwerk des Dateiservers in einem SCOM-LAN zu transferieren. Die Netz-Daten (eigene Stationsadresse, Stationsadresse des Servers und Passwort) sind fest eingestellt und koennen nur durch Quelltextaenderungen oder Patches veraendert werden.

7. Programmablauf, Bedienanleitung

Die Parameteruebergabe findet ueber die Aufrufzeile statt.
Aufruf
A>NSAVE [d1:]datei.ext
d1: ist der Laufwerksname des Ziellaufwerks beim Datei-Server. Diese Laufwerksangabe ist optional. datei.ext ist der vollstaendige Dateiname.

Beispiel:
A>NSAVE C:TEST.MAC
TEST.MAC wird vom Laufwerk A des eigenen Rechners gelesen und auf das Laufwerk C des Dateiservers geschrieben.
Fehlermeldungen:

'uebertragen'	fehlerfreie Ausfuehrung
'Diskette voll'	Diskette im Ziellaufwerk ist voll

'Protokollfehler'	Bei der Uebertragung ist ein Protokollfehler aufgetreten (Stoerung, Defekt)
'Passwortfehler'	Das Passwort ist vom Server nicht akzeptiert worden
'Funktion nicht moeglich'	Server kennt die gewuenschte Funktion nicht
'Teilnehmer nicht vorhanden'	Teilnehmer ist nicht am Netz
'Datei nicht vorhanden'	Datei existiert nicht beim Server/ auf dem Laufwerk
'allgemeimer Fehler'	Fehler kann nicht spezifiziert werden

8. Generierung

In der Datei NETCOM.MAC koennen die eigene Adresse **GEADR**, die Serveradresse **DBADR** und die Modul- (Hardware-) Adresse veraendert werden. In NLOAD.MAC kann das Passwort **PASSWO** eingestellt werden.

Weitere Verarbeitung:

Assemblieren: ASM =NSAVE
Linken: LINK NSAVE,NIOS,NSAVE/N/E

Aenderung der Parameter durch Patches:
Der Anfang von NSAVE hat folgendes Aussehen:

```
        JP START

SIODA:  DEFB MODUL      ;SIO DATA A
SIOCA:  DEFB MODUL+2    ;SIO CTRL A
CTC0:   DEFB MODUL+4    ;CTC 0
CTC3:   DEFB MODUL+7    ;CTC 3
TNA:    DEFB GEADR      ;eigene Adresse
DADR:   DEFB DBADR      ;Datenbankadresse
PASSWO:DEFB '           ;Passwort

START:
        LD SP,STACK
              :
              :
```

Mit DU (ZSID) koennen diese Parameter geaendert werden. Die Parameter haben dabei folgende Lage:

```
100h +3   1 Byte    SIODA     ;SIO DATA A
     +4   1 Byte    SIOCA     ;SIO CTRL A
     +5   1 Byte    CTC0      ;CTC 0
     +6   1 Byte    CTC3      ;CTC 3
     +7   1 Byte    TNA       ;eigene Adresse
     +8   1 Byte    DADR      ;Datenbankadresse
     +9   8 Byte    PASSWO    ;Passwort
```

Beispiel fuer ein Patch der eigenen Adresse:

```
A>du nsave.com<et>
#s107<et>
0107 12 10<et>
0108 01 .<et>
^c
A>save 11 nsave.com<et>
```

Die eigene Adresse wurde von 18 / (12h) auf 16(10h) geaendert.

3.3. SIOS

1. Benennung

3.3.1. SLOAD

2. Kurzbeschreibung

Laden einer Datei von einem Dateiserver, der ueber eine Standardschnittstelle mit dem eigenen Rechner verbunden ist.

3. Daten

Laenge: 2 kbyte

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

- SLOAD.MAC Quellprogramm SLOAD
- (incl) SIOSCOM.MAC Allgemeine Vereinbarungen SIOS
- (incl) NSCCU1.MAC Protokoll Dateiuebertragung SIOS
- (incl) SIOS.REL SIOS-Teiber

SLOAD benutzt BDOS-Rufe

6. Anwendung

Mittels der Komponenten SIOS ist es moeglich, einen Datei-Transfer zwischen zwei Rechnern durchzufuehren. Dabei werden gleiche Prinzipien wie in Netz angewandt (Server und Konsument). Die SIOS-Komponenten sind weitgehend software- und bedienungskompatibel zu den NET-Komponenten. Die SIOS-Komponenten sind an eine serielle Standardschnittstelle (V.24 oder IFSS) gebunden und setzen keine Netz-Installation voraus. Es ist allerdings nur der Anschluss eines Rechners pro Schnittstelle meoglich.
SSAVE gestattet, eine Datei von einem Laufwerk des Dateiservers auf ein Laufwerk des eigenen Rechners zu transferieren. Die Hardware-Adressen und das Passwort sind fest eingestellt und koennen nur durch Quelltextaenderungen oder Patches veraendert werden.

7. Programmablauf, Bedienanleitung

Die Parameteruebergabe findet ueber die Aufrufzeile statt.
Aufruf
A>SLOAD [d1:]datei.ext
d1: ist der Laufwerksname des Quellaufwerkes beim Datei-Server. Diese Laufwerksangabe ist optional. datei.ext ist der vollstaendige Dateiname.
Beispiel:
A>SLOAD C:TEST.MAC
TEST.MAC wird vom Laufwerk C des Dateiservers gelesen und auf Laufwerk A des eigenen Rechners abgelegt.
Fehlermeldungen:

'uebertragen'	fehlerfreie Ausfuehrung
'Diskette voll'	Diskette im Ziellaufwerk ist voll
'Protokollfehler'	Bei der Uebertragung ist ein Protokollfehler aufgetreten (Stoerung, Defekt)
'Passwortfehler'	Das Passwort ist vom Server nicht akzeptiert worden
'Funktion nicht moeglich'	Server kennt die gewuenschte Funktion nicht
'Teilnehmer nicht vorhanden'	Teilnehmer ist nicht am Netz
'Datei nicht vorhanden'	Datei existiert nicht beim Server/ auf dem Laufwerk
'allgemeimer Fehler'	Fehler kann nicht naeher spezifiziert werden

8. Generierung

In der Datei SIOSCOM.MAC kann die Modul- (Hardware-) Adresse veraendert werden. In SLOAD.MAC kann das Passwort **PASSWO** eingestellt werden.

Weitere Verarbeitung:

Assemblieren: ASM =SLOAD
Linken: LINK SLOAD,SLOAD/N/E

Aenderung der Parameter durch Patches:
Der Anfang von NLOAD hat folgendes Aussehen:

```
        JP START
        .
SIOD:   DEFB MODUL      ;SIO DATA A
SIOC:   DEFB MODUL+2    ;SIO CTRL A
CTCO    DEFB MODUL+4    ;CTC 0
        DEFB 0
        DEFB 0
        DEFB 0
PASSWO: DEFB '        ' ;Passwort

START:
        LD SP,STACK
             :
             :
```

Mit DU (ZSID) koennen diese Parameter geaendert werden. Die Parameter haben dabei folgende Lage:

```
100h +3   1 Byte    SIOD      ;SIO DATA
     +4   1 Byte    SIOC      ;SIO CTRL
     +5   1 Byte    CTC       ;CTC
     +6   1 Byte
     +7   1 Byte
     +8   1 Byte
     +9   8 Byte    PASSWO    ;Passwort
```

Beispiel fuer ein Patch der CTC-Adresse:

```
A>du sload.com<et>
#s105<et>
0105 12 10<et>
0108 00 .<et>
^C
A>save 11 sload.com<et>
```

Die CTC- Adresse wurde von 18 (12h) auf 16(10h) geaendert.

9. Einschraenkungen bei der Nutzung

keine

1. Benennung

****
*
* 3.3.2. SSAVE
*
****

2. Kurzbeschreibung

Speichern einer Datei auf einem SIOS-Dateiserver

3. Daten

Laenge: 2 kbyte

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

- SSAVE.MAC Quellprogramm SSAVE
- (incl) SIOSCOM.MAC Allgemeine Vereinbarungen NET
- (incl) S_CCU1.MAC Protokoll Dateiuebertragung NET
- (incl) SIOS.REL SIOS-Teiber

SSAVE benutzt BDOS-Rufe

6. Anwendung

Mittels der Komponenten SIOS ist es moeglich, einen Datei-Transfer zwischen zwei Rechnern durchzufuehren. Dabei werden gleiche Prinzipien wie in Netz angewandt (Server und Konsument). Die SIOS-Komponenten sind weitgehend software- und bedienungskompatibel zu der NET-Komponenten. Die SIOS-Komponenten sind an eine serielle Standardschnittstelle (V.24 oder IFSS) gebunden und setzen keine Netz-Installation voraus. Es ist allerdings nur der Anschluss eines Rechners pro Schnittstelle meoglich.
SSAVE gestattet, von einem Laufwerk des eigenen Rechners eine Datei auf ein Laufwerk des Dateiservers in einem SCOM-LAN zu transferieren. Die Hardware-Adressen und das Passwort sind fest eingestellt und koennen nur durch Quelltext-aenderungen oder Patches veraendert werden.

7. Programmablauf, Bedienanleitung

Die Parameteruebergabe findet ueber die Aufrufzeile statt.
Aufruf
A>SSAVE [d1:]datei.ext
d1: ist der Laufwerksname des Ziellaufwerks beim Datei-Server. Diese Laufwerksangabe ist optional. datei.ext ist der vollstaendige Dateiname.

Beispiel:
A>SSAVE C:TEST.MAC
TEST.MAC wird vom Laufwerk A des eigenen Rechners gelesen und auf das Laufwerk C des Dateiservers geschrieben.

Fehlermeldungen:

'uebertragen'	fehlerfreie Ausfuehrung
'Diskette voll'	Diskette im Ziellaufwerk ist voll
'Protokollfehler'	Bei der Uebertragung ist ein Protokollfehler aufgetreten (Stoerung, Defekt)
'Passwortfehler'	Das Passwort ist vom Server nicht akzeptiert worden
'Funktion nicht moeglich'	Server kennt die gewuenschte Funktion nicht
'Teilnehmer nicht vorhanden'	Teilnehmer ist nicht am Netz
'Datei nicht vorhanden'	Datei existiert nicht beim Server/ auf dem Laufwerk
'allgemeimer Fehler'	Fehler kann nicht spezifiziert werden

8. Generierung

In der Datei SIOSCOM.MAC kann die Modul- (Hardware-) Adresse veraendert werden. In NLOAD.MAC kann das Passwort **PASSWO** eingestellt werden.

Weitere Verarbeitung:

Assemblieren: ASM =SSAVE
Linken: LINK SSAVE,SSAVE/N/E

Aenderung der Parameter durch Patches:
Der Anfang von NLOAD hat folgendes Aussehen:

```
        JP START

SIOD:   DEFB MODUL      ;SIO DATA A
SIOC:   DEFB MODUL+2    ;SIO CTRL A
CTCO    DEFB MODUL+4    ;CTC 0
        DEFB 0
        DEFB 0
        DEFB 0
PASSWO: DEFB '        ' ;Passwort

START:
        LD SP,STACK
             :
             :
```

Mit DU (ZSID) koennen diese Parameter geaendert werden. Die Parameter haben dabei folgende Lage:

```
100h +3   1 Byte    SIOD      ;SIO DATA
     +4   1 Byte    SIOC      ;SIO CTRL
     +5   1 Byte    CTC       ;CTC
     +6   1 Byte
     +7   1 Byte
```

```
      +8    1 Byte
      +9    8 Byte     PASSWO     ;Passwort
```

Beispiel fuer ein Patch der CTC-Adresse:

```
A>du ssave.com<et>
#s105<et>
0105 12 10<et>
0108 00 .<et>
^c
A>save 11 ssave.com<et>
```

Die CTC- Adresse wurde von 18 (12h) auf 16(10h) geaendert.

1. Benennung

****
*
* 3.3.3. SSLAVE
*
****

2. Kurzbeschreibung

Dieses Programm realisiert SIOS-SERVER-Funktionen

3. Daten

Laenge: 3 kbyte

4. Programmiersprache

U880 - Assemblersprache
[Assembler ASM 1520 (SCPX) V1.0]

5. Genutzte Module

- SSLAVE.MAC Quellprogramm SSLAVE
- (incl) SIOSCOM.MAC Allgemeine Vereinbarungen SIOS
- (incl) S_CC.MAC Protokoll Dateiuebertragung SIOS
- (incl) SIOS.REL SIOS-Teiber

SSLAVE benutzt BDOS-Rufe

6. Anwendung

Mittels der Komponenten SIOS ist es moeglich einen Datei-Transfer zwischen zwei Rechnern durchzufuehren. Dabei werden gleiche Prinzipien wie in Netz angewandt (Server und Konsument). Die SIOS-Komponenten sind weitgehend software- und bedienungskompatibel zu den NET-Komponenten. Die SIOS-Komponenten sind an eine serielle Standardschnittstelle (V.24 oder IFSS) gebunden und setzen keine Netz-Installation voraus. Es ist allerdings nur der Anschluss eines Rechners pro Schnittstelle meoglich.
SSLAVE realisiert die Server-Funktionen im SIOS-Verbund. Es sind die Slave-Funktionen
- Datei lesen
- Datei schreiben
- Datei pruefen (Vorhandensein)
- Verzeichnis lesen
- Datei loeschen

implementiert. (Von den Programmen SSAVE (3.3.1.) und SLOAD (3.3.2.) werden nicht alle Funktionen ausgenutzt.)
Die Hardware-Adressen und das Passwort sind fest eingestellt und koennen nur durch Quelltextaenderungen oder Patches veraendert werden.

7. Programmablauf, Bedienanleitung

Aufruf
A>SSLAVE
Nach Anzeigen der Ueberschrift mit einigen Systemdaten wird die Eingabe des Passwortes erwartet. Waehrend der Arbeit des Servers werden die Zugriffe durch Ausgabe von **access** protokolliert. Die Arbeit des Servers kann durch Tastenbetaetigung abgebrochen werden.

8. Generierung

In der Datei SIOSCOM.MAC kann die Modul- (Hardware-) Adresse veraendert werden. In SSLAVE.MAC kann das Passwort **PASSWO** eingestellt werden.

Weitere Verarbeitung:

Assemblieren: ASM =SSLAVE
Linken: LINK SSLAVE,SSLAVE/N/E

Aenderung der Parameter durch Patches:
Der Anfang von SSLAVE hat folgendes Aussehen:

```
        JP START

SIOD:   DEFB MODUL      ;SIO DATA A
SIOC:   DEFB MODUL+2    ;SIO CTRL A
CTCO    DEFB MODUL+4    ;CTC 0
        DEFB 0
        DEFB 0
        DEFB 0
PASSWO: DEFB '          ' ;Passwort

START:
        LD SP,STACK
             :
             :
```

Mit DU (ZSID) koennen diese Parameter geaendert werden. Die Parameter haben dabei folgende Lage:

```
100h +3   1 Byte    SIOD        ;SIO DATA
     +4   1 Byte    SIOC        ;SIO CTRL
     +5   1 Byte    CTC         ;CTC
     +6   1 Byte
     +7   1 Byte
     +8   1 Byte
     +9   8 Byte    PASSWO      ;Passwort
```

Beispiel fuer ein Patch der CTC-Adresse:

```
A>du sslave.com<et>
#s105<et>
0105 12 10<et>
0108 00 .<et>
^c
A>save 11 sslave.com<et>
```

Die CTC- Adresse wurde von 18 (12h) auf 16(10h) geaendert.

9. Einschraenkungen bei der Nutzung

keine