Interner Bericht
DESY F58-69/2
Januar 1969

DESY - PDP5/PDP8/PDP8I - Programmiersystem

SYSTEMANWEISUNGEN

F. Akolk, G. Hochweller

Inhaltsverzeichnis

1 Einführung und Definitionen

Um die beim DESY bei allen PDP-Anlagen vorhandene relativ große Ausbaustufe (8k-Speicher, 4 DECtapes, Display, Fernschreiber, Schreibmaschine) auch für die Programmierung voll ausnutzen zu können, wurde ein eigenes Programmiersystem geschrieben. Es besteht im wesentlichen aus dem Editorprogramm SUPEDT, dem Assembler PAL4 und einem Organisationsprogramm, im folgenden kurz als SYSTEM bezeichnet, das die Verarbeitung der einzelnen Systemanweisungen steuert.

Das System wird auf folgende Weise in Kontrolle gebracht:
Die auszuführende Systemprozedur wird mit SUPEDT als normaler Text in den Textspeicher geschrieben. Jede Systemanweisung muß dabei genau eine Zeile bilden; zwischen den einzelnen Systemanweisungen dürfen sich beliebig viele Leerzeilen befinden. Das Kommando "CTRL+S" (im Kommando-mode) bewirkt dann, daß das System vom Systemband in den Speicher geladen und gestartet wird. Die einzelnen Systemanweisungen werden nacheinander ausgeführt.

Es werden folgende Bänder benötigt:

a. Systemband auf Einheit 8 (WRITE LOCK)
b. Hilfsband auf Einheit 6 (WRITE ENABLED)
c. Editorband auf Einheit 5 (WRITE LOCK), falls ein Programm übersetzt werden soll.

Nach Abarbeitung der Systemprozedur (oder bei Fehlern, die die weitere Ausführung verhindern) kehrt das System zum SUPEDT zurück.
Die Systemprozedur selbst befindet sich dann auf dem Editorblock 0 des Hilfsbandes auf Einheit 6.

Definitionen

ACS = ABSOLUTE CONTROL SECTION

Zusammenhängendes Programmstück, dessen Anfangsadresse durch n festgelegt ist (n = absolute Adresse oder globaler Name).

RCS = RELOCATABLE CONTROL SECTION

Zusammenhängendes Programmstück, das mit 'BEGREL name(,1)' beginnt und mit 'ENDREL' endet.

PROGRAMMSEGMENT

Eine Menge von ACS und/oder RCS, die in einem Stück übersetzt und auf die Bibliothek gebracht wurden.

NAMEN

Jegliche Art von Namen (RCS-Namen, Segment-Namen, globale Symbole, lokale Symbole, etc.) dürfen aus max. 6 alphanumerischen Zeichen bestehen, von denen das erste alphabetisch sein muß. Enthält ein Name mehr als 6 Zeichen, werden nur die ersten 6 Zeichen als signifikant angenommen.
Sonderzeichen sind in Namen jeglicher Art nicht zulässig.

! Als Programmnamen sind folgende Kombinationen verboten: !
!
! EMPTY, BOOT, SUPER, !

2 Programm-Segmentierung und Speicherplatz-Verteilung

Bei größeren Experimente-Programmen wird häufig der Fall eintreten, daß die Gesamtlänge des Programms die verfügbare Kernspeicher-Kapazität (im allgemeinen 8 k) überschreitet. Andererseits ist die Struktur solcher Programme meist derart, daß zu einer bestimmten Zeit nur ein gewisser Teil des gesamten Programms benötigt wird, d.h., daß zu der Zeit auch nur dieser Teil im Speicher resident sein muß. Als Beispiel kann der Initialisierungsteil eines Experimente-Programms dienen, in dem alle die Parameter festgelegt werden, die während des runs bzw. während einer längeren Zeit nicht mehr verändert werden. Ist dies geschehen, wird dieser Programmteil zunächst nicht mehr benötigt und der Speicherplatz steht den Programmteilen zur Verfügung, die nun das eigentliche Experiment steuern, Daten nehmen, Kontrollfunktionen ausüben, etc.

Um von dieser natürlichen Programmaufteilung sinnvoll Gebrauch machen zu können, muß eine Möglichkeit bestehen, einzelne Programmsegmente auf einfache Weise 'at execution time' gegeneinander auswechseln zu können.
Unter einem "Programmsegment" soll ein Programmstück verstanden werden, das in einem Stück übersetzt und auf die Bibliothek gebracht wurde, und das in jedem der drei primär voneinander unabhängigen Speicherteile (bank 1, bank 0, page 0) nur jeweils ein zusammenhängendes Stück belegt.

Um den vorhandenen Speicherplatz rationell auszunutzen, muß für eine möglichst 'dichte Packung' der einzelnen RCSs bzw. ACSs innerhalb eines Programmsegmentes gesorgt werden.
Besteht ein Programmsegment ausschließlich aus RCSs, so stellt die Speicherplatz-Verteilung, die der Assembler automatisch liefert, eine sehr gute Näherung an die optimale Verteilung dar. Um diesen Vorteil voll auszunutzen, sollten die einzelnen Programmsegmente - von ganz wenigen speziellen Ausnahmen abgesehen - nur aus RCSs bestehen und keine ACSs enthalten.

Durch den Assembler wird der Speicher von unten nach oben (d.h. von höheren Speicheradressen zu niedrigeren hin) so dicht wie möglich gefüllt. Dadurch ist automatisch gewährleistet, daß ein Programmsegment nur ein zusammenhängendes Stück im Speicher einnimmt. Während des assembly-Vorgangs werden folgende Tabellen bzw. Listen geführt:

a.) Liste der globalen Symbole, die in dem Programmsegment definiert werden
b.) Speicherkarte, aus der hervorgeht, welche Teile des Speichers noch zur Verfügung stehen
c.) Liste der Konstanten auf page 0, durch die erreicht wird, daß für gleiche Konstanten, die an verschiedenen Stellen des Programmsegmentes vorkommen, nur ein Hilfsplatz eingerichtet wird.

Die Gesamtheit dieser Listen bzw. Tabellen soll im folgenden als die LISTE des Programmsegments bezeichnet werden.

Diese Art der Speicherplatz-Verteilung impliziert, daß das niedrigste Programmsegment, dem (unabhängig von seiner Länge) der

gesamte noch vorhandene freie Speicherplatz zur Verfügung gestellt wird, nach oben hin bis zur Adresse 200 reicht. Unter dem Namen dieses Segmentes ist dann das ganze Programm vom Loader her rufbar; sämtliche übergeordneten Segmente werden bei Aufruf des niedrigsten Segmentes mitgeladen.

Jedes Programm, das den Supervisor (SUPER) benutzt, wird automatisch nach dem Laden (bzw. dem Auswechseln von Segmenten) dadurch gestartet, daß die TASK 'attached' wird, deren TCB ab Adresse 200 steht.
Hierdurch ergibt sich eine Aufteilung der Programmsegmente in zwei Gruppen:

a.) "übergeordnete" Segmente
Sie reichen nicht bis zur Adresse 200 und sind, da sie nicht startbar sind (kein TCB auf Adresse 200), auch nicht unter ihrem Namen vom Loader her rufbar.
b.) "niedrigste" Segmente
Sie enthalten immer einen TCB ab Adresse 200 und sind somit rufbar.

Da an die nicht-rufbaren Segmente immer noch mindestens ein rufbares Segment angebunden werden muß, muß auch die Liste des übergeordneten Segments, die ja die gesamte dazu notwendige Information enthält, mit auf die Bibliothek gebracht werden. Dies geschieht automatisch durch die Benutzung der Systemanweisung 'INSERT' (siehe dort).
Bei rufbaren Segmenten ist die Liste natürlich überflüssig, da keine noch niedrigeren Segmente angebunden werden können. Sämtliche rufbaren Segmente werden mit der Systemanweisung 'UPDATE' auf die Bibliothek gebracht; sie unterscheidet sich von der Systemanweisung 'INSERT' im wesentlichen dadurch, daß durch sie nur das Programmsegment selbst, nicht aber die Liste auf die Bibliothek gebracht wird.

Die Abb.1 zeigt als Beispiel die möglichen Speicherbelegungen eines 5-fach segmentierten Programms (bank 0-Teil). Bei einem derart segmentierten Programm lassen sich folgende Speicherbelegungen erreichen:

1. S1, S5
2. S1, S2, S4
3. S1, S2, S3

Das Auswechseln von Programmsegmenten erfolgt durch einen Anruf an den Supervisor.

Aufrufformat:

```
        SVC
        TRANS
        LIST
        (code)
        .
        .
BEGREL LIST
   LIST;LIBLIS(neue Segmentnamen)
ENDREL
```

Ein solcher Anruf an den Supervisor darf nur in den niedrigsten

Segmenten stehen, in unserem Beispiel also in den Segmenten S3, S4, S5. LIST ist die symbolische Adresse einer Liste, die die Ladeparameter der (des) gewünschten neuen Programmsegmente(s) enthält. Bei der Verwendung der Pseudo-Operation LIBLIS wird diese Liste durch den Assembler zusammengestellt, der sich die hierfür benötigte Information aus dem Index des betreffenden Bibliotheksbandes holt (siehe PAL4-Beschreibung). Das bedeutet jedoch, daß 'at assembly time' der Segmente S3, S4, S5 die Ladeparameter jeweils aller übrigen Segmente bekannt sein müssen. Um dies zu erreichen, wurde die Systemanweisung 'ALLOCATE name' eingeführt. Sie bewirkt, daß für das Programmsegment 'name' ausreichend Platz auf dem Bibliotheksband reserviert wird und die (dadurch ja bekannten) Ladeparameter unter dem Namen des Programmsegmentes im Index eingetragen werden. Das Programm selbst wird dann erst später nach dem assembly in den so reservierten Platz auf dem Bibliotheksband mit Hilfe der Systemanweisung 'UPDATE' eingesetzt.

Abb.2 gibt eine Zusammenstellung aller möglichen LIBLIS-Referenzen für ein gemäß Abb.1 segmentiertes Programm.

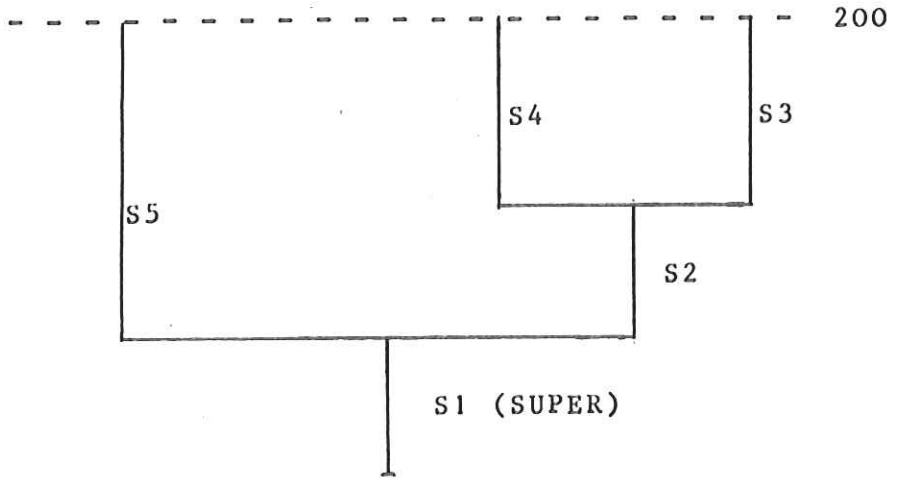

Abb.1: Mögliche Speicherbelegung eines 5-fach segmentierten Programms (bank 0-Teil)

Aufruf in	Übergang von	nach 1	2	3
S5	1	---	LIBLIS(S2,S4)	LIBLIS(S2,S3)
S4	2	LIBLIS(S5)	---	LIBLIS(S3)
S3	3	LIBLIS(S5)	LIBLIS(S4)	---

Abb.2: LIBLIS-Referenzen

Belegung:
1 S1, S5
2 S1, S2, S4
3 S1, S2, S3

3 Beschreibung der Systemanweisungen

3.1 Assembler-bezogene Systemanweisungen

3.1.1 CONNECT

Format: CONNECT n,tname(,mname)

Die Systemanweisung CONNECT veranlaßt das Laden der Liste des Programmsegmentes 'mname' von dem Bibliotheksband 'tname' auf Einheit 'n'. Das Programm 'mname' muß vorher mit der Systemanweisung INSERT auf das Bibliotheksband gebracht worden sein, da nur an solche Programme 'connected' werden darf.

Die geladene Liste besteht aus folgenden Teilen:

a.) Liste der globalen Symbole
b.) Speicherkarte
c.) Liste der Konstanten auf page 0

Sie wird in genau der gleichen Form geladen, in der sie nach dem assembly des Programmsegmentes 'mname' im Speicher stand. Das hat zur Folge, daß der Assembler ein neu zu übersetzendes Programmsegment als Fortsetzung des Segmentes 'mname' betrachtet, da er die im Segment 'mname' definierten globalen Symbole (a), den durch das Segment 'mname' belegten und somit auch den noch zur Verfügung stehenden Speicherbereich (b), und die im Segment 'mname' auf page 0 erzeugten Konstanten (c) kennt.

Sonderfälle

A. 'mname' fehlt

"leere Liste", das im folgenden zu übersetzende Programmsegment ist an kein höheres Segment angebunden; der ganze Speicher außer der letzten Seite auf bank 0 steht zur Verfügung.

B. 'mname' = BOOT

wie A., nur sind zusätzlich die globalen Symbole des "bootstrap-loaders" bekannt.

C. 'mname' = EMPTY

wie A., auch die letzte Seite auf bank 0 steht zur Verfügung.

Die Sonderfälle B und C werden ausschließlich bei der Systembanderzeugung benutzt. Insbesondere Fall C ist unter allen Umständen verboten, da bei seiner Anwendung der "bootstrap-loader" zerstört wird!

Fehler-Meldungen

'ILLEGAL SYSTEM STATEMENT'

Formatfehler in der Systemanweisung

'REPLACE TAPE 'tnam1' BY 'tnam2' AND PRESS CONTINUE'

Gefordert wurde in der Systemanweisung das Bibliotheksband mit dem Namen 'tnam2', auf der entsprechenden Einheit liegt jedoch ein Band mit dem Namen 'tnam1'.

'MNAME UNKNOWN'

Ein Programmsegment mit dem Namen 'mname' existiert auf dem Bibliotheksband nicht.

3.1.2 SET

Format: SET symbol=(symbol=...=symbol=)immediate expression

Durch die Systemanweisung SET wird 'symbol' mit dem Wert 'immediate expression' in die Liste der globalen Symbole eingetragen. Ist 'symbol' schon in der Liste enthalten, so wird es auf den neuen Wert gesetzt.

SET darf nur unmittelbar auf ein CONNECT oder ein weiteres SET folgen, da zu dieser Zeit eine globale Symbolliste im Speicher sein muß. Ein 'immediate expression' darf folgende Elemente enthalten:

a. Operanden:
   Oktalzahlen, schon bekannte globale Symbole, Codes von Speicherbezugsbefehlen

B. Operatoren:
   Addition (+), Subtraktion (-), inklusives Oder ('SPACE')

Speicherbezugsbefehle in 'immediate expressions' dürfen keine literals enthalten, d.h., die Zeichen X, Y, :, = sind verboten. Indirekte Adressierung dagegen ist erlaubt.

Beispiele:
```
CONNECT .....
SET A=17
SET B=C=DCA I 1o
SET D=B+A-3
SET E=SZA CLA
SET F=JMS I A+2
```

Eine wichtige Anwendung der Systemanweisung SET ist die Steuerung eines assembly 'at assembly time'. Näheres dazu siehe unter den PAL4-Pseudo-Operationen 'IF' und 'DUP'.

Fehler-Meldungen

'ILLEGAL SYSTEM STATEMENT'

Formatfehler in der Systemanweisung

'PA ERR'

Rechts vom Gleichheitszeichen stehen nicht erlaubte Elemente.

'GL. OV.'

Die globale Liste ist voll und kann das neue Symbol nicht mehr aufnehmen.

3.1.3 ASSEMBLE

Format: ASSEMBLE bg,bg,...,bg

Die Systemanweisung ASSEMBLE bringt den Assembler in Kontrolle und führt zur Übersetzung der Programmteile auf den Editorblöcken bg des Editorbandes auf Einheit 5.
bg bedeutet dabei entweder einen einzelnen Editorblock oder eine Reihe aufeinanderfolgender Editorblöcke.

Beispiel: ASSEMBLE 3,7-10,15

Die Programmteile auf den Editorblöcken 3,7,8,9,10,15 werden übersetzt.

Einem jeden ASSEMBLE muß ein CONNECT vorausgehen. Zwischen CONNECT und ASSEMBLE darf nur die Systemanweisung SET stehen (außer natürlich den Systemanweisungen der Gruppe 3.4, die hier aber nicht sinnvoll sind).

! Das Editorband muß auf Einheit 5 liegen. !

Assembler - Meldungen

Die meisten Meldungen beginnen mit:

Px BL y LN z

Dabei bedeuten:
Px = Assembler-Phase x
BL y = Editorblock y
LN z = Nummer der Zeile im Editorblock: z

Die Zeilennummer z gibt dabei entweder genau die Zeile des betreffenden Befehls oder die letzte Zeile der RCS an, in der der betreffende Befehl steht.

A. Meldungen, die zum Abbruch des assembly führen

a. Meldungen, die durch Programmfehler verursacht werden

P3 BL y LN z LOC USE OF A GLOB SYM

Der für das lokale Symbol gewählte Name wurde in diesem oder einem übergeordneten Programmsegment schon für ein globales Symbol benutzt oder ist identisch mit einem Maschinenbefehl.

P3 BL y LN z DUP LOC SYM

Der für das lokale Symbol gewählte Name wurde in dieser CS schon für ein anderes lokales Symbol benutzt.

P1 BL y LN z symbol DUP GLOB SYM

Der für das globale Symbol gewählte Name 'symbol' wurde in diesem oder einem übergeordneten Programmsegment schon für ein anderes globales Symbol benutzt oder ist identisch mit einem Maschinenbefehl.

P3 BL y LN z symbol UNKN GLOB SYM

Das angesprochene globale Symbol 'symbol' wurde weder in diesem noch in einem übergeordneten Programmsegment definiert.

P3 BL y LN z symbol UNDEFINED

Das angesprochene Symbol 'symbol' wurde in dieser CS nicht definiert und ist auch nicht in der Liste der globalen Symbole enthalten.

GL. OV. (bei parameter assignement) oder

P1 BL y LN z GLOB LIST OVFL (sonst)

Die Liste der globalen Symbole enthält bereits 383 Elemente und ist damit voll. Sie kann das neue Symbol nicht mehr aufnehmen.

P3 BL y LN z LOC LIST OVFL

Die Liste der lokalen Symbole enthält bereits 31 Elemente und ist damit voll. Sie kann das neue Symbol nicht mehr aufnehmen.

P1 BL y LN z LVAL OVFL

Die Liste der numerischen literals enthält bereits 31 Elemente und ist damit voll. Sie kann das neue literal nicht mehr aufnehmen.

P1 BL y LN z LSYM OVFL

Die Liste der Adreß-literals enthält bereits 31 Elemente und ist damit voll. Sie kann das neue literal nicht mehr aufnehmen.

P1 BL y LN z RCS LIST OVFL

Die Liste der RCS-Namen enthält bereits 143 Elemente und ist damit voll. Sie kann den neuen RCS-Namen nicht mehr aufnehmen.

P1 BL y LN z name DUP RCS NAME

Der für die RCS gewählte Name wurde in diesem Programmsegment schon für eine andere RCS benutzt.

P1 BL y LN z NO CS OPENED

Nach dem Ende einer CS folgt weiterer Programmtext ohne daß eine neue CS eröffnet wurde.

P1 BL y LN z RCS NAME MISSING

Eine RCS wurde durch 'BEGREL' geöffnet, es fehlt jedoch der RCS-Name hinter dem BEGREL.

P2 BL y LN z name RCS OF ZERO LENGTH

Die RCS mit dem Namen 'name' enthält keinen Befehl.

P1 BL y LN z LIBLIS ERROR

In der Pseudo-Operation 'LIBLIS' fehlen die Namen der gewünschten Programmsegmente.

name UNKN NAME IN LIBLIS

Das in der Pseudo-Operation 'LIBLIS' gewünschte Programmsegment 'name' ist in dem Index des betreffenden Bibliotheksbandes nicht vorhanden.

Px BL y LN z LIT ERROR

Nicht zulässiges Konversionszeichen in einem literal.
Literal-Referenz von einem Befehl, der kein Speicherbezugs-Befehl ist.
Formatfehler in einem Adreß-literal; erstes Element kein Symbol oder zweites Element keine Zahl .LT. 4000 (OCT).

P1 BL y LN z ADDR EXPR ERROR

Der Adreß-Teil eines Befehls enthält einen nicht erlaubten Operator.

PA ERR

In einem parameter assignment stehen rechts vom Gleichheitszeichen nicht erlaubte Operatoren.

P1 BL y LN z ILL TXT FORMAT

Ende eines Editorblocks innerhalb einer TXT-Pseudo-Operation.

P5 BL y LN z ILL REF

Nicht zulässige Adressierung über Seitengrenzen hinweg; die Ursache ist häufig eine CS, die länger als eine Seite ist.

P1 BL y LN z TEXT END WITHIN A DUP

In einer Pseudo-Operation 'DUP' werden mehr Zeilen angegeben als in dem Editorblock noch vorhanden sind.

P1 BL y LN z ILL BANK

In einer Pseudo-Operation 'BANK n' ist n .NE. 0 oder 1.

P1 BL y LN z PAGE0 OVFL

Die page 0 ist voll; die Pseudo-Operation 'PAGE 0' bzw. die Öffnung einer ACS auf PAGE 0 kann nicht mehr ausgeführt werden.

P1 BL y LN z PAGE OVERLAY

Eine ACS soll in einem Speicherbereich plaziert werden, der nicht mehr zur Verfügung steht.

PROGRAM TOO LONG IN BANK n

Der noch vorhandene freie Speicherbereich in bank n ist zu klein für das zu übersetzende Programmsegment.

ILLEGAL SYSTEM STATEMENT

Formatfehler in den Systemanweisungen 'TYPE NAMES' oder 'LISTING'.

b. Meldungen, die durch Maschinenfehler verursacht werden

P5 BL y LN z UNKN RCS NAME

Ein RCS-Name ist in Phase 5 nicht mehr bekannt.

P3 BL y LN z name ILL RCS NAME

Angeblicher Fehler im RCS-Name 'name' in Phase 3.

P5 BL y LN z TXT ERROR

Fehler in einer 'TXT'-Pseudo-Operation in Phase 5.

B. Meldungen, die nicht zum Abbruch des assembly führen

P1 BL y LN z ACS LTP

Eine ACS ist länger als eine Seite.

P1 BL y LN z RCS LTP

Eine RCS ist länger als eine Seite.

P1 BL y LN z PAGE 0 STOP

Die page 0 ist voll; von nun an werden keine Konstanten-Plätze mehr auf page 0 eingerichtet, sondern immer innerhalb der eigenen CS.

Px BL y LN z SSTOP

Die Pseudo-Operation 'SSTOP' wird ausgeführt. Die Maschine hält an und kann durch Drücken der Taste 'CONT' erneut gestartet werden.

3.1.3.1 TYPE NAMES

Format: TYPE NAMES

Die Systemanweisung TYPE NAMES bewirkt die Ausgabe der RCS-Namen mit ihren zugehörigen Anfangsadressen auf dem jeweiligen Ausgabegerät. Sie darf nur unmittelbar auf ein ASSEMBLE folgen. Ihre Ausführung wird durch die Systemanweisung NOLIST unterdrückt.

3.1.3.2 LISTING

Format: LISTING

Die Systemanweisung LISTING liefert einen vollständigen Programmtext-Abdruck in folgendem Format:

Spalte 1: Zeilennummer im Editorblock (dezimal)
Spalte 2: Speicheradresse (oktal)
Spalte 3: Speicherinhalt (oktal)
Spalte 4: Programmtext und evtl. Kommentar

Die Systemanweisung LISTING darf nur unmittelbar auf eine Systemanweisung der Gruppe 3.1.3 folgen; ihre Ausführung wird durch NOLIST unterdrückt. LISTING sollte nur mit der Schreibmaschine und nicht mit dem Teletype ausgeführt werden, obwohl dies grundsätzlich möglich ist.

Empfohlene Tabulator-Einstellungen:

linker Rand: 10
Tabulator-Stops: 16,23,30,40,60,65,70

Weitere Anmerkungen siehe unter 3.1.3.3 und 3.1.3.4.

3.1.3.3 DATE

Format: DATE: xxx

xxx können bis zu 10 beliebige Zeichen sein, die das Datum angeben.

Die Systemanweisung DATE sorgt dafür, daß in der Überschrift einer jeden Seite eines LISTING-Abdruckes das richtige Datum erscheint. Sie ist also nur dann von Interesse, wenn LISTING ausgeführt wird.
Die Systemanweisung DATE darf nur zwischen ASSEMBLE und LISTING stehen. Werden innerhalb einer Systemprozedur mehrere Anrufe an LISTING ausgeführt, so braucht die Anweisung DATE nur beim erstenmal zu erscheinen. Bei allen folgenden Anrufen an LISTING wird dann das gleiche Datum eingesetzt.

Beispiel siehe unter 3.1.3.4 (TAPE).

3.1.3.4 TAPE

Format: TAPE: xxx

xxx können bis zu 20 beliebige Zeichen sein, die den Namen des Editorbandes angeben.

Die Systemanweisung TAPE sorgt dafür, daß in der Überschrift einer jeden Seite eines LISTING-Abdruckes der richtige Editorband-Name erscheint. Sie ist also nur dann von Interesse, wenn LISTING ausgeführt wird.
Die Systemanweisung TAPE darf nur zwischen ASSEMBLE und LISTING stehen. Werden innerhalb einer Systemprozedur mehrere Anrufe an LISTING ausgeführt, so braucht die Anweisung DATE nur beim erstenmal zu erscheinen. Bei allen folgenden Anrufen an LISTING wird dann derselbe Editorband-Name eingesetzt.

Beispiel:

```
USE TYPWR
CONNECT ...
(SET ...)
ASSEMBLE ...
(TYPE NAMES)
DATE: 11-11-1967
TAPE: EDITOR SCHULZ 2
LISTING
```

3.2 Bibliotheks-bezogene Systemanweisungen

3.2.1 ALLOCATE

Format: ALLOCATE name

Die Systemanweisung ALLOCATE darf nur hinter einem CONNECT (bzw. hinter EXCEPT, siehe 3.2.2) stehen. Durch das vorausgehende CONNECT wird die Speicherkarte des Programmsegmentes 'mname' geladen. Hierdurch erfährt das System, welche Speicherbereiche durch die übergeordneten Programmsegmente noch nicht belegt sind. Dieser gesamte noch freie Speicherplatz wird durch die Systemanweisung ALLOCATE für das (erst später zu übersetzende) Programmsegment 'name' reserviert, und 'name' wird als Name eines rufbaren Programms mit vollständiger Ladeinformation in den Index des Bibliotheksbandes eingesetzt. Als entrypoint wird immer 200 angenommen; das später unter dem Namen 'name' auf das Bibliotheksband zu bringende Programmsegment muß also bei 200 beginnen.

Sollen mehrere Programmsegmente an das gleiche übergeordnete Segment angebunden werden, braucht man nicht jedesmal CONNECT zu schreiben, sondern kann sämtliche ALLOCATE-Anrufe direkt hintereinander ausführen.

Beispiel:
```
CONNECT n,tname(,mname)
(EXCEPT m)
ALLOCATE name1
ALLOCATE name2
.
ALLOCATE namei
```

Siehe auch unter 3.2.2 EXCEPT.

Fehler-Meldungen

'ILLEGAL SYSTEM STATEMENT'

Formatfehler in der Systemanweisung

'name DUPLICATE NAME'

Der Programmname 'name' ist schon im Index des betreffenden Bibliotheksbandes enthalten.

'mname ILLEGAL STORAGE LAYOUT'

Das übergeordnete Programmsegment 'mname', an das das neue Programmsegment angebunden werden soll, benutzt schon die Seite 200.

'LIBR INDEX OVFL'

Das betreffende Bibliotheksband ist voll und kann das neue Programm nicht mehr aufnehmen.

3.1.2 EXCEPT

Format: EXCEPT m

Die Systemanweisung EXCEPT darf nur unmittelbar vor einem ALLOCATE stehen. 'm' ist eine Zahl zwischen 0 und 40 (oktal). Sie gibt an, wieviel Seiten auf bank 1 von Seite 0 an beim folgenden ALLOCATE nicht mit reserviert werden sollen. Ist m .GT. 40, so wird m = 40 gesetzt.

Beispiel:
```
CONNECT ...
EXCEPT 40
ALLOCATE ...
```

Nur der freie Platz auf bank 0 wird reserviert, die gesamte bank 1 bleibt gesperrt.

3.2.3 INSERT

Format: INSERT name

Durch die Systemanweisung INSERT wird ein nicht rufbares Programmsegment (d.h. ein übergeordnetes Segment, an das später weitere Segmente angehängt werden sollen) unter dem Namen 'name' in die Bibliothek eingesetzt. Neben dem eigentlichen Programmsegment wird durch INSERT auch die dem Programmsegment zugehörige Liste (siehe unter CONNECT) auf das Bibliotheksband gebracht. Dies ist die Voraussetzung dafür, daß später weitere Segmente an das Segment 'name' angehängt werden können.

Übergeordnete Programmsegmente sollen im Normalfall keine ACSs enthalten, sondern ausschließlich aus RCSs bestehen.

Die Systemanweisung INSERT läßt sich beliebig oft mit demselben Namen 'name' ausführen. Die auf dem Bibliotheksband stehende alte Version des Programmsegments 'name' wird dann jedesmal durch die neue Version überschrieben.

==========
ACHTUNG!
Wenn die veränderte Version eines übergeordneten Segmentes durch einen erneuten Anruf an 'INSERT' auf das Bibliotheksband gebracht wurde, sind anschliessend alle an dieses Segment angebundenen Programmsegmente neu zu übersetzen.
==========

Fehler-Meldungen

'ILLEGAL SYSTEM STATEMENT'

Formatfehler in der Systemanweisung

'ILLEGAL STORAGE LAYOUT'

Das Programmsegment benutzt die Seite 200 auf bank 0. Dies ist verboten, da dann kein weiteres Segment mehr angehängt werden kann.

'LIBR INDEX OVFL'

Das betreffende Bibliotheksband ist voll und kann das neue Programmsegment nicht mehr aufnehmen.

'NEW VERSION TOO LONG'

Die neue Version des Programmsegmentes 'name' beansprucht mehr Platz auf dem Bibliotheksband als ihm zur Verfügung steht, da sie länger als die frühere Version ist.

3.2.4 UPDATE

Format: UPDATE name(,entry)

Durch die Systemanweisung UPDATE wird ein rufbares Programmsegment unter dem Namen 'name' auf das Bibliotheksband gebracht. UPDATE setzt kein vorheriges ALLOCATE voraus, d.h., der Name 'name' muß im Index des betreffenden Bibliotheksbandes noch nicht enthalten sein.

'entry' ist entweder eine in dem Programmsegment enthaltene globale Adresse oder eine Oktalzahl und gibt die Adresse an, an der das Programm nach dem Laden gestartet werden soll. Fehlt 'entry', so hält die Maschine nach dem Laden des Programms an. Dies gilt nicht für Programme, die unter SUPERVISOR-Kontrolle laufen, da diese vom SUPERVISOR gestartet werden.

Die Systemanweisung UPDATE läßt sich beliebig oft mit demselben Namen 'name' ausführen. Die auf dem Bibliotheksband stehende alte Version des Programmsegments 'name' wird dann jedesmal durch die neue Version überschrieben.

Fehler-Meldungen

'ILLEGAL SYSTEM STATEMENT'

Formatfehler in der Systemanweisung.

'LIBR INDEX OVFL'

Das Bibliotheksband ist voll und kann das neue Programmsegment nicht mehr aufnehmen.

'ENTRY UNKNOWN'

Die als entrypoint gewünschte globale Adresse wurde in den Programmsegment 'name' nicht als globale Adresse definiert.

'NEW VERSION LONGER THAN ALLOCATED SPACE'

Die neue Version des Programmsegments 'name' ist länger als die alte Version und paßt deshalb nicht auf das durch die alte Version belegte Stück auf dem Bibliotheksband. Die alte Version muß erst entfernt werden (evtl. durch DELETE).

3.2.5 PUNCH

Format: PUNCH

Durch die Systemanweisung PUNCH wird ein übersetztes Programmsegment in Binärform auf dem Lochstreifenstanzer am Teletype ausgegeben. Das Ausgabeformat entspricht dem DEC-BIN-Format, sodaß der erzeugte Lochstreifen mit dem DEC-BIN-Loader eingelesen werden kann.

Nach Ausführung der Systemanweisung PUNCH kehrt das System zum SUPEDT zurück.

Meldungen

'START PUNCH AND PRESS CONTINUE'

Taste 'ON' am Teletype-Locher und anschliessend 'CONT' drücken. Beim nächsten Halt der Maschine (keine Meldung) die Taste 'OFF' am Teletype-Locher und anschließend wieder 'CONT' drücken.

'ILLEGAL SYSTEM STATEMENT'

Formatfehler in der Systemanweisung.

3.3 Systemanweisungen zur Indexbehandlung

3.3.1 NAME

Format: NAME n,tname

Das Bibliotheksband auf Bandeinheit 'n' bekommt den Namen 'tname', d.h., der Name 'tname' wird in die ersten drei Plätze des Index eingesetzt (in getrimmtem ASCII-Code).

Fehler-Meldungen

'ILLEGAL SYSTEM STATEMENT'

Formatfehler in der Systemanweisung.

3.3.2 INDEXL

Format: INDEXL n

Sämtliche im Index des Systembandes auf Bandeinheit n aufgeführten Namen werden auf dem jeweiligen Ausgabegerät ausgeschrieben. Dabei wird unterschieden zwischen rufbaren Programmsegmenten, nicht-rufbaren Programmsegmenten und Listen von nicht-rufbaren Programmsegmenten. Die drei Typen von Namen erscheinen im Ausdruck wie folgt:

```
  name        nicht-rufbares Programmsegment
L name        Liste eines nicht-rufbaren Programmsegments
  name+       rufbares Programmsegment
```

Nach der Ausführung kehrt das System zum SUPEDT zurück.

Fehler-Meldungen

'ILLEGAL SYSTEM STATEMENT'

Formatfehler in der Systemanweisung.

3.3.3 INDEX

Format: INDEX n

Sämtliche im Index des Bibliotheksbandes auf Bandeinheit 'n' aufgeführten Namen werden mit vollständiger Ladeinformation im Editorformat in den Textspeicher des Editors gebracht. Die Ausgabe erfolgt also nicht direkt über das jeweilige Ausgabegerät; das System kehrt nach Ausführung der Systemanweisung INDEX zum SUPEDT zurück. Die im Textspeicher befindliche Information über das Bibliotheksband läßt sich nun wie normaler Editor-Text behandeln, d.h. auf dem Display anzeigen, auf Teletype oder Schreibmaschine ausgeben, etc.

Die verschiedenen Spalten im Index-Ausdruck (bzw. -Display) haben folgende Bedeutung:

a. NAME

Name des Programmsegments; ein L vor dem Namen zeigt an, daß es sich um eine Liste handelt.

B. ENTR

Entrypoint eines rufbaren Programmsegments.

C. POI

Der Inhalt dieser Spalte gibt an, an welches übergeordnete Segment dieses Programmsegment angebunden wurde. '-m' bedeutet dabei, daß das entsprechende übergeordnete Programmsegment m Plätze vorher im Index steht.

D. BLOK, SADDR, NBL

In diesen Spalten steht die Ladeinformation der einzelnen Segmentteile. Jedes Programmsegment kann bis zu drei Ladeinformationen besitzen: bank 1-Teil, page 0-Teil, bank 0-Teil.
BLOK gibt die Nummer des ersten Blockes (oktal) des Segmentteils auf dem Bibliotheksband an.
SADDR ist die Adresse des niedrigsten benutzten Speicherplatzes des Segmentteils (oktal); die Adressen des bank 0-Teils laufen dabei von 200 bis 7600, die des bank 1-Teils von 10000 bis 17600.
NBL ist die Anzahl (oktal) der für diesen Segmentteil benötigten Bandblöcke (= Speicherseiten).

Fehler-Meldungen

'ILLEGAL SYSTEM STATEMENT'

Formatfehler in der Systemanweisung.

3.3.4 DELETE

Format: DELETE n,tname,name

Sämtliche Programmsegmente und Listen auf dem Bibliotheksband 'tname' auf Bandeinheit 'n' werden von dem Programmsegment 'name' an (einschliesslich) gelöscht. Auf dem jeweiligen Ausgabegerät erscheint folgender Ausdruck:

DELETED PROGRAMMS

Namen sämtlicher Programmsegmente ab 'name'

PRESS "D" TO DELETE

Das eigentliche Löschen erfolgt erst nach Drücken der Taste "D". Wird irgendeine andere Taste gedrückt, so unterbleibt das Löschen und das System kehrt sofort zum SUPEDT zurück.

Nach dem Löschen führt das System automatisch noch die Systemanweisung INDEXL aus und liefert somit einen Ausdruck der Namen aller noch auf dem Bibliotheksband befindlichen Programmsegmente und Listen.

Fehler-Meldungen

'ILLEGAL SYSTEM STATEMENT'

Formatfehler in der Systemanweisung.

'REPLACE TAPE 'tnam1' ON UNIT 'n' BY 'tnam2' AND PRESS CONTINUE'

Gefordert wurde in der Systemanweisung das Bibliotheksband mit dem Namen 'tnam2', auf der entsprechenden Bandeinheit liegt jedoch ein Band mit dem Namen 'tnam1'.

3.4 Systemanweisungen zur 'listing'-Steuerung

3.4.1 USE

Format: USE code

Durch die Systemanweisung USE wird das Ausgabegerät bestimmt, auf dem nachfolgende listings, Fehler-Meldungen etc. erscheinen sollen.

code = TPRINT Ausgabegerät: Teletype
code = TYPWR Ausgabegerät: Schreibmaschine

Mit der Systemanweisung USE kann beliebig oft innerhalb einer Systemprozedur von einem Ausgabegerät auf ein anderes umgeschaltet werden. Steht am Anfang einer Systemprozedur keine Systemanweisung USE, so wird automatisch der Teletype als Ausgabegerät genommen.

Fehler-Meldungen

'ILLEGAL SYSTEM STATEMENT'

Formatfehler in der Systemanweisung.

3.4.2 LIST

Format: LIST

Durch die Systemanweisung LIST wird bewirkt, daß die nachfolgenden Systemanweisungen vor ihrer Ausführung auf dem gewählten Ausgabegerät ausgedruckt werden.

LIST darf nicht zwischen zwei Systemanweisungen der Gruppe 3.1.3 stehen, sonst aber beliebig oft innerhalb einer Systemprozedur vorkommen.

Fehler-Meldungen

'ILLEGAL SYSTEM STATEMENT'

Formatfehler in der Systemanweisung.

3.4.3 NOLIST

Format: NOLIST

Die Systemanweisung NOLIST unterdrückt das Ausdrucken der nachfolgenden Systemanweisungen. Durch NOLIST wird ebenfalls die Ausführung der nachfolgenden Systemanweisungen 'TYPE NAMES', 'LISTING' und 'EJECT' unterdrückt. Keinen Einfluß hat NOLIST auf die Ausgabe von Fehler-Meldungen und Assembler-Meldungen.
Am Anfang einer Systemprozedur wird automatisch die Systemanweisung NOLIST ausgeführt.

NOLIST darf nicht zwischen zwei Systemanweisungen der Gruppe 3.1.3 stehen, sonst aber beliebig oft innerhalb einer Systemprozedur vorkommen.

Fehler-Meldungen

'ILLEGAL SYSTEM STATEMENT'

Formatfehler in der Systemanweisung.

3.4.4 EJECT

Format: EJECT

Die Systemanweisung EJECT bewirkt, daß auf dem jeweiligen Ausgabegerät eine neue Seite angefangen wird. Die Ausführung der Systemanweisung EJECT wird durch NOLIST unterdrückt.

Fehler-Meldungen

'ILLEGAL SYSTEM STATEMENT'

Formatfehler in der Systemanweisung.

3.4.5 PAUSE

Format: PAUSE (beliebiger Text)

Die Systemanweisung PAUSE stoppt die Maschine bis zum Drücken der Taste 'CONT'. Auf diese Weise können innerhalb einer Systemprozedur Bänder gewechselt werden etc. Das Ausdrucken des Textes hinter PAUSE wird durch NOLIST unterdrückt.

Fehler-Meldungen

'ILLEGAL SYSTEM STATEMENT'

Formatfehler in der Systemanweisung.

3.4.6 /

Format: / (beliebiger Text)

Durch die Systemanweisung '/' wird der hinter dem Schrägstrich stehende Text auf dem jeweiligen Ausgabegerät ausgeschrieben. Man kann die Systemanweisung '/' dazu benutzen, beliebige Kommentare in das listing einer Systemprozedur einzuschieben. Die Ausführung der Systemanweisung '/' wird durch NOLIST unterdrückt.

4 Anhang

A. getrimmter ASCII-Code

Code	Zeichen	Sonder-bedeutung	Code	Zeichen	Sonder-bedeutung
00	@	'CR'	40	'SPACE'	
01	A		41	!	
02	B		42	"	
03	C		43	#	ß
04	D		44	$	'SPACE'
05	E		45	%	
06	F		46	&	'COMM'
07	G		47	'	
10	H		50	(	
11	I		51	)	
12	J		52	*	'BACK'
13	K		53	+	
14	L		54	,	
15	M		55	-	
16	N		56	.	
17	O		57	/	
20	P		60	0	
21	Q		61	1	
22	R		62	2	
23	S		63	3	
24	T		64	4	
25	U		65	5	
26	V		66	6	
27	W		67	7	
30	X		70	8	
31	Y		71	9	
32	Z		72	:	
33	[	Ä	73	;	
34	\	Ö	74	<	'NEW PAGE'
35	]	Ü	75	=	
36	↑	'UC'	76	>	'STOP'
37	←	'LC'	77	?	

ASCII = American Standard Code for Information Interchange

Die Sonderbedeutung der ASCII-Zeichen gilt für Ausgabe auf der Schreibmaschine und Benutzung des Typesetting-Programms.
Es bedeuten:

'CR' = Wagenrücklauf, Zeilenvorschub
'UC' = Umschaltung auf große Buchstaben
'LC' = Umschaltung auf kleine Buchstaben
'COMM' = Kommandozeichen (Typesetting)
'NEW PAGE' = Beginn einer neuen Seite
'STOP' = Stoppen der Ausgabe bis "CTRL+G"

B. Beispiel einer Systemprozedur

```
/SYSTEM PROCEDURE PART B
/(GENERAL PART)

/ASSEMBLY SYSTEM AND ASSEMBLER MAIN LINK
CONNECT 4,SYSTEM,BOOT
ASSEMBLE 3-9
TYPE NAMES
INSERT SSMAIN
EJECT

/ALLOCATE SYSTEM BOTH PHASES
CONNECT 4,SYSTEM,SSMAIN
EXCEPT 35
ALLOCATE SYSTEM
ALLOCATE SYSTP2

/ASSEMBLY PAL 4 MAIN SEGMENT "PPMAIN"
CONNECT 4,SYSTEM,SSMAIN
ASSEMBLE 11-13
TYPE NAMES
INSERT PPMAIN

/ALLOCATE ASSEMBLER ALL PHASES
CONNECT 4,SYSTEM,PPMAIN
EXCEPT 35
ALLOCATE PAL4
ALLOCATE PAL4P2
ALLOCATE PAL4P3
ALLOCATE PAL4P4
ALLOCATE PAL4P6
ALLOCATE PAL4P5
EJECT

/ASSEMBLY SYSTEM PART 1
CONNECT 4,SYSTEM,SSMAIN
ASSEMBLE 15-23
TYPE NAMES
UPDATE SYSTEM,200
EJECT

/ASSEMBLY SYSTEM PART 2
CONNECT 4,SYSTEM,SSMAIN
ASSEMBLE 27-29
TYPE NAMES
UPDATE SYSTP2,200
EJECT

/ASSEMBLY ASSEMBLER PHASE 1
CONNECT 4,SYSTEM,PPMAIN
SET PHASE1=1
SET PHASE3=PHASE4=PHASE5=PHASE6=0
ASSEMBLE 40-47
TYPE NAMES
UPDATE PAL4,200
EJECT
```

```
/ASSEMBLY ASSEMBLER PHASE 2
CONNECT 4,SYSTEM,PPMAIN
ASSEMBLE 35-37
TYPE NAMES
UPDATE PAL4P2,200
EJECT

/ASSEMBLY ASSEMBLER PHASE 3
CONNECT 4,SYSTEM,PPMAIN
SET PHASE3=1
SET PHASE1=PHASE4=PHASE5=PHASE6=0
ASSEMBLE 38-39,41-47
TYPE NAMES
UPDATE PAL4P3,200
EJECT

/ASSEMBLY ASSEMBLER PHASE 4
CONNECT 4,SYSTEM,PPMAIN
SET PHASE4=1
SET PHASE1=PHASE3=PHASE5=PHASE6=0
ASSEMBLE 55-56
TYPE NAMES
UPDATE PAL4P4,200
EJECT

/ASSEMBLY ASSEMBLER PHASE 5
CONNECT 4,SYSTEM,PPMAIN
SET PHASE5=1
SET PHASE1=PHASE3=PHASE4=PHASE6=PHASE7=0
ASSEMBLE 47-54
TYPE NAMES
UPDATE PAL4P5,200
EJECT

/ASSEMBLY ASSEMBLER TYPEOUT PHASE
CONNECT 4,SYSTEM,PPMAIN
SET PHASE6=1
SET PHASE1=PHASE3=PHASE4=PHASE5=PHASE7=0
ASSEMBLE 47-54
TYPE NAMES
UPDATE PAL4P6,200
EJECT

/ASSEMBLY COPY PROGRAM
CONNECT 4,SYSTEM,SUPER
ASSEMBLE 60-63
TYPE NAMES
UPDATE COPY
EJECT

LIST
/LEGAL END OF SYSTEM PROCEDURE
```

C. Beispiel eines INDEX n - Aufrufes

AUFRUF: INDEX n

Ergebnis:

INDEX OF TAPE: SYSTEM

NUMBER OF PROGR: 24
NEXT FREE BLOCK: 412

	NAME	ENTR	POI	BLOK	SADDR	NBL
L	EMPTY			10		
L	BOOT			27		
	SUPER			46	17200	3
				51	1	1
				52	5400	12
L	SUPER			64		
	GERPAT	7776		103	17200	1
	AMPAT	7776		104	17200	1
	SUPEDT	7776	- 4	105	107	1
				106	200	25
	SSMAIN			133	1	1
				134	3400	21
L	SSMAIN			155		
	SYSTEM	200	- 2	174	17200	2
				176	104	1
				177	200	15
	SYSTP2	200	- 3	214	17300	2
				216	104	1
				217	200	15
	PPMAIN		- 4	234	14000	15
				251	104	1
				252	600	3
L	PPMAIN			255		
	PAL4	200	- 2	274	17200	2
				276	142	1
				277	200	12
	PAL4P2	200	- 3	311	17200	2
				313	142	1
				314	200	12
	PAL4P3	200	- 4	326	17200	2
				330	142	1
				331	200	12
	PAL4P4	200	- 5	343	17200	2
				345	142	1
				346	200	12
	PAL4P6	200	- 6	360	17200	2
				362	142	1
				363	200	12
	PAL4P5	200	- 7	375	17200	2
				377	142	1
				400	200	12

D. Beispiel für einen INDEXL n - Aufruf

Aufruf: INDEXL n

Ergebnis:

```
PROGRAMS ON TAPE: SYSTEM

L EMPTY
L BOOT
  SUPER
L SUPER
  GERPAT +
  AMPAT  +
  SUPEDT +
  SSMAIN
L SSMAIN
  SYSTEM +
  SYSTP2 +
  PPMAIN
L PPMAIN
  PAL4   +
  PAL4P2 +
  PAL4P3 +
  PAL4P4 +
  PAL4P6 +
  PAL4P5 +
  COPY   +
```

E. Bootstrap-loader

PDP-8/PDP-8I			PDP-5		
Adr.	okt. Inhalt	Bedeutung	Adr.	okt. Inhalt	Bedeutung
7600	6224	RIF	7600	1223	TAD RV
7601	6774	DTLB	7601	6757	MMMM
7602	1220	TAD RV	7602	4216	JMS FD
7603	4212	JMS FD	7603	4216	JMS FD
7604	1221	TAD BA	7604	1224	TAD RD
7605	6201	CDF0	7605	6756	MMMF
7606	3623	DCA I AM	7606	4216	JMS FD
7607	1222	TAD RD	7607	1225	TAD BA
7610	4212	JMS FD	7610	6766	MMML
7611	7402	HLT	7611	4216	JMS FD
7612	0000	FD,0	7612	7402	HLT
7613	6766	DTCA DTXA	7613	0000	
7614	3354	DCA 7754	7614	0000	
7615	6771	DTSF	7615	0000	
7616	5215	JMP .-1	7616	0000	FD,0
7617	5612	JMP I FD	7617	6761	MMSF
7620	0600	RV,600	7620	5217	JMP .-1
7621	7577	BA,7577	7621	6772	MMCF
7622	0220	RD,220	7622	5216	JMP I FD
7623	7755	AM,7755	7623	1030	RV,1030
			7624	0022	RD,22
			7625	7600	BA,7600