TSOSLNK V21.0D (BS2000)

Benutzerhandbuch

Zielgruppe

Software-Entwickler

Inhalt

– Anweisungen und Makroaufrufe des Binders TSOSLNK zum Binden von Lade- und Großmodulen

– Kommandos des statischen Laders ELDE

Ausgabe: Mai 1991

Inhalt

Einleitung

Ein übersetztes Programm hat bei allen Sprachübersetzern (Assembler, COBOL, PL/I, FORTRAN usw.) ein einheitliches Format und wird als **Bindemodul** (Objekt Module, **OM)** bezeichnet. Dieser besteht zwar bereits aus Maschinencode, kann im Rechner aber erst ablaufen, nachdem er gebunden und geladen wurde.

Durch das **Binden** wird das übersetzte Programm mit anderen Bindemodulen zu einer ladefähigen Einheit zusammengefügt. Das ist erforderlich, denn besonders bei höheren Programmiersprachen können die meisten Programme erst dann ablaufen, wenn andere Bindemodule sie ergänzen. Die einzelnen Module können dabei zu unterschiedlichen Zeiten und von verschiedenen Übersetzern erzeugt worden sein. Die Hauptfunktion eines **Binders** besteht also darin, die zum Programmlauf erforderlichen Bindemodule aufzusuchen und miteinander zu verknüpfen, indem er jeden Modul um die Adressen ergänzt, die sich auf Felder außerhalb des betreffenden Moduls beziehen (Externverweise) und deshalb vom Sprachübersetzer noch nicht eingetragen werden konnten.

Damit die beim Binden erzeugte Einheit ablaufen kann, muß ein **Lader** sie in den Speicher bringen, so daß der Rechner zum Code zugreifen und ihn ausführen kann.

Beschreibung des Produkts

Im System **Binder-Lader-Starter (BLS)** stehen dem Benutzer folgende Systembausteine zur Verfügung:

- der Binder BINDER,
- der dynamische Bindelader DBL,
- der Binder TSOSLNK,
- der statische Lader ELDE.

Binder BINDER

Der Binder BINDER bindet Module zu einer logisch und physikalisch strukturierten ladbaren Einheit zusammen. Diese Einheit bezeichnet man als **Bindelademodul** (Link and Load Module, **LLM**). Abgespeichert wird ein LLM vom BINDER als Bibliothekselement vom Typ L in einer Programmbibliothek.

Module, aus denen der BINDER einen LLM bindet, können sein:

- Bindemodule (OMs) aus einer Bindemodulbibliothek (OML), aus einer Programmbibliothek (Typ R) oder aus der temporären EAM-Bindemoduldatei,
- bereits gebundene LLMs aus einer Programmbibliothek (Typ L).

Dynamischer Bindelader DBL

Der dynamische Bindelader DBL hat die Aufgabe, Module zu einer Ladeeinheit zu binden und diese zu laden.

Module, aus denen der DBL eine Ladeeinheit bindet, können sein:

- Bindelademodule (LLMs), die vom BINDER gebunden und in einer Programmbibliothek (Typ L) gespeichert wurden,
- Bindemodule (OMs), die von Compilern erzeugt und in einer Bindemodulbibliothek (OML), in einer Programmbibliothek (Typ R) oder in der temporären EAM-Bindemoduldatei gespeichert wurden,
- vorgebundene Bindemodule (Großmodule), die vom Binder TSOSLNK gebunden wurden.

Binder TSOSLNK

Der Binder TSOSLNK bindet:

- entweder einen oder mehrere Bindemodule (OMs) zu einem ablauffähigen Programm (Lademodul) und speichert dieses in einer katalogisierten Programmdatei oder als Bibliothekselement vom Typ C in einer Programmbibliothek,
- mehrere Bindemodule (OMs) zu einem einzigen vorgebundenen Modul (Großmodul) und hinterlegt diesen als Bibliothekselement vom Typ R in einer Programmbibliothek oder in der EAM-Bindemoduldatei.

Statischer Lader ELDE

Der statische Lader ELDE hat die Aufgabe, ein ablauffähiges Programm zu laden, das vom Binder TSOSLNK gebunden und in einer Programmdatei oder als Bibliothekselement vom Typ C in einer Programmbibliothek gespeichert wurde.

Konzept des Handbuchs

In diesem Handbuch sind der Binder TSOSLNK und der statische Lader ELDE beschrieben.

Der neue Binder BINDER und der dynamische Bindelader DBL sind in einem getrennten Handbuch "BLS" beschrieben (siehe Handbuch "Binder-Lader-Starter (BLS)" [1]).

Im einzelnen ist das Handbuch in folgende Kapitel gegliedert:

Einleitung
Dieses Kapitel enthält eine Kurzbeschreibung aller Systembausteine für Binden und Laden, sowie das Konzept dieses Handbuchs.

Aufgaben von Binder und Lader
Dieses Kapitel zeigt das Zusammenwirken der Systembausteine.

Binder TSOSLNK
Dieses Kapitel enthält den bisherigen Binder TSOSLNK. Es beschreibt den Binderlauf, den Bindevorgang, die Ausgaben, die Makroaufrufe und die Anweisungen für den Binder TSOSLNK.

Lader ELDE
Dieses Kapitel enthält den Lader ELDE. Es beschreibt die Kommandos und die Makroaufrufe für den Aufruf des Laders ELDE.

XS-Unterstützung des DBL
Dieses Kapitel wendet sich an Anwender, die auf einer XS-Anlage den Adreßraum oberhalb 16Mbyte nutzen wollen.

Literaturverweise werden im Text in Kurztiteln angegeben. Der vollständige Titel jedes Handbuchs, auf das verwiesen wird, ist im **Literaturverzeichnis** aufgeführt.

Aufgaben von Binder und Lader

Der dynamische Bindelader DBL (siehe Handbuch "BLS" [1]) bindet mehrere Bindemodule zu einem ablauffähigen Programm und lädt dieses sofort in den Hauptspeicher. Das Programm wird nicht gespeichert.

Der Lader ELDE und der dynamische Bindelader DBL sind keine Dienstprogramme, die mit dem Systemkommando START-PROGRAM eigens gestartet werden. Beide fassen vielmehr unter der Bezeichnung Lader ELDE oder dynamischer Bindelader DBL all die Funktionen des Organisationsprogramms zusammen, die ein Programm laden und zum Ablauf bringen, z.B. die Systemkommandos LOAD-PROGRAM und START-PROGRAM.

Bild 1 Verarbeitungsmöglichkeiten für einen Bindemodul

Die folgende Tabelle zeigt die Unterschiede zwischen TSOSLNK und DBL. Sie soll dem Anwender entscheiden helfen, welches Bindeverfahren für sein Programm geeignet ist.

Der Binder TSOSLNK ...	Der dynamische Bindelader DBL ...
... bindet Bindemodule.	... bindet *und* lädt Bindemodule.
... speichert die Bindemodule als ablauffähiges Programm (Lademodul) in einer katalogisierten Datei oder in einer Programmbibliothek vom Typ C. Zum Laden des Programms muß der Lader ELDE aufgerufen werden.	... speichert das Programm nicht, sondern löscht es automatisch nach dem Programmlauf.
... erzeugt seitenwechselbare Programme (Klasse-II-Programme).	... erzeugt seitenwechselbare Programme (Klasse-II-Programme).
... legt für das Programm das endgültige Adreßgefüge fest.	... kann für jeden Lauf ein anderes Adreßgefüge erstellen, da die Bausteine, d.h. die Bindemodule, jedesmal neu gesucht und zusammengefügt werden.
... liest die benötigten Bindemodule aus: - der EAM-Bindemoduldatei - Programmbibliotheken (Typ R) - Bindemodulbibliotheken (OML) - der Systemdatei SYSDTA oder - der TASKLIB.	... liest die benötigten Bindemodule aus: - der EAM-Bindemoduldatei - Programmbibliotheken (Typ R) - Bindemodulbibliotheken (OML) - der TASKLIB - einer Datei mit dem Dateiketttungsnamen BLSLIBnn oder benutzt Module, die bereits im Klasse-4-Speicher stehen (SHARE-Module).
... kann Programme erstellen, in denen beim Programmablauf einzelne Teile (Segmente) eines bereits teilweise geladenen Programms durch den Lader ELDE nachgeladen werden. Beim Binden wird festgelegt, ob und wie diese Segmente einander adreßmäßig überlagern. Die Segmente stehen nach dem Laden im Klasse-6-Speicher der Benutzertask.	... kann ein Nachladen von Bindemodulen in den Klasse-6-Speicher der Benutzertask oder den Klasse-4-Speicher des Systems veranlassen. Die Ladeadresse des nachgeladenen Programmteils wird erst beim Laden ermittelt. Adreßmäßige Überlagerung kann durch Entladen anderer Programmteile erreicht werden.

Der Binder TSOSLNK ...	Der dynamische Bindelader DBL ...
———	... bindet und lädt Bindemodule. Die maximale Grösse einzelner Bindemodule darf 16Mbyte (begrenzt durch REQM) nicht überschreiten.
... kann mehrere Bindemodule zu einem einzigen Bindemodul binden und in die temporäre EAM-Bindemoduldatei bringen (MODULE-Anweisung)	———
... kann Namen in Bindemodulen, ENTRYs, CSECTS, Externverweisen umbenennen (RENAME-Anweisung).	———
... kann Programmabschnitte auf das Attribut READ-ONLY beschränken oder kann sie auf Seitengrenze ausrichten.	... kann das Laden von Programmabschnitten auf das Attribut READ-ONLY beschränken oder kann sie auf Seitengrenze ausrichten.
... gibt Listen und Protokolle aus.	... gibt Listen und Protokolle aus.

Der Anwender kann also auf zwei verschiedene Wege sein übersetztes Programm zum Laufen bringen:

– Mit dem Binder TSOSLNK erhält er ein jederzeit lad- und ausführbares Programm in einer katalogisierten Datei oder in einem Element einer Programmbibliothek. Zum Laden dieses Programms benötigt er den Lader ELDE. Dieser Weg eignet sich für Programme, für die keine Änderungen mehr zu erwarten sind, die als adreßmäßig festgelegte Einheit stets lad- und ausführbar bereitstehen sollen.

– Der dynamische Bindelader DBL hingegen bindet und lädt ein ablauffähiges Programm, das nicht gespeichert wird.
  Er ist daher besonders für die Testphase geeignet; außerdem für Programme, die selten laufen und für solche, bei denen einzelne Programmteile austauschbar bleiben sollen. Die Bausteine des Programms, die einzelnen Bindemodule, sind bis zur Ablaufzeit platzsparend in Bibliotheken gespeichert.

Wie Bild 2 zeigt, kann in einem vom Binder TSOSLNK erstellten Programm zum Nachladen der dynamische Bindelader DBL aufgerufen werden. Eine Mischung aus beiden Methoden ist also ebenfalls möglich.

Bild 2 Möglichkeiten zum Nachladen von Programmteilen

Der Binder TSOSLNK

Der Binder TSOSLNK bindet übersetzte Programmteile, sog. Bindemodule zusammen. Ergebnis des Bindevorgangs ist

- ein ablauffähiges **Programm** (Lademodul), das aus einem oder mehreren Segmenten besteht, und als katalogisierte Datei oder als Element einer Programmbibliothek (Elementtyp C) gespeichert wird.
- ein **vorgebundener** Bindemodul **(Großmodul)**, der in der temporären EAM-Bindemoduldatei oder als Element einer Programmbibliothek (Elementtyp R) gespeichert wird.

Der Lader ELDE lädt die Segmente des Programms, die TSOSLNK gebunden hat, in den Speicher.
Der dynamische Bindelader DBL lädt den von TSOSLNK gebundenen Großmodul oder bindet ihn mit anderen Bindemodulen zusammen und lädt das Ergebnis dieses Bindevorgangs.

Binderlauf

Aufruf von TSOSLNK und Bindevorgang

TSOSLNK wird mit folgendem Kommando geladen und gestartet:

```
START-PROGRAM FROM-FILE=$TSOSLNK
```

TSOSLNK erwartet nach der Programmlademeldung die Eingabe von Anweisungen aus der Systemdatei SYSDTA.

Eine korrekte Anweisungsfolge an TSOSLNK besteht immer aus 3 Teilen:

1. Einlesen von Bindemodulen (z.B. mit INCLUDE). INCLUDE benennt die Bindemodule, die gebunden werden sollen. Diese Bindemodule stehen in der temporären EAM-Bindemoduldatei (wenn sie unmittelbar zuvor von einem Sprachübersetzer erzeugt wurden) und/oder in einer Modulbibliothek, die von dem Bibliotheksverwaltungsprogramm LMS erstellt wurde (siehe Handbuch "LMS" [3]).

   Die INCLUDE-Anweisung muß nur dann nicht eingegeben werden, wenn die zu bindenden Bindemodule aus SYSDTA gelesen werden sollen.
2. Festlegen, was das Bindeergebnis sein soll: Ein Programm (PROGRAM-Anweisung) oder ein vorgebundener Bindemodul (MODULE-Anweisung).
3. Beenden der Eingabe von Anweisungen (END-, CONTINUE-, oder BIND-Anweisung).

Dieses Anweisungsgrundgerüst kann ergänzt werden durch Anweisungen

- zur Behandlung von Externverweisen,
- zur Behandlung des Programms oder des Bindemoduls,
- zur Steuerung der Ausgabelisten und Protokolle,
- zum Aufbau von Überlagerungsstrukturen,
- zum Verfahren in Fehlerfällen.

Nachdem die Eingabe von Anweisungen abgeschlossen ist, beginnt der Bindevorgang.

1. **Befriedigen von Externverweisen**

   Externverweise in einem Bindemodul verweisen auf Programmabschnitte oder Einsprungstellen in einem anderen Bindemodul.
   TSOSLNK versucht alle Externverweise aufzulösen, d.h. die passenden Programmabschnittsnamen und Einsprungstellen zu finden und die Programmteile einzubinden.

   In der 1. Stufe durchsucht TSOSLNK alle angegebenen Bindemodule nach den passenden Programmabschnittsnamen und Einsprungstellen. Konnten dadurch nicht alle Externverweise aufgelöst werden, untersucht TSOSLNK in der 2. Stufe, ob Externverweise in einer RESOLVE-Anweisung angegeben wurden.

   Sind auch jetzt noch nicht alle Externverweise befriedigt, durchsucht TSOSLNK in Stufe 3 die Modulbibliothek TASKLIB des Anwenders und anschließend die System-Modulbibliothek TASKLIB nach passenden Programmabschnittsnamen und Einsprungstellen.

   Externverweise, die dann noch nicht aufgelöst sind, werden mit einem festen Wert besetzt, wenn der Anwender nicht ausdrücklich etwas anderes angegeben hat. Nicht aufgelöste Externverweise führen dazu, daß der Programmlauf abgebrochen wird, es sei denn, der Anwender hat ausdrücklich bestimmt, daß der Bindelauf dennoch fortgesetzt werden soll.

2. **Behandeln gemeinsamer Speicherbereiche**

   Gemeinsame Speicherbereiche, sog. COMMON-Bereiche, sind Abschnitte, die zum Zeitpunkt des Bindens keine Daten oder Befehle enthalten, sondern nur Platz dafür reservieren, z.B. als Verständigungsbereich zwischen verschiedenen Programmteilen. TSOSLNK durchsucht alle angegebenen Bindemodule daraufhin, ob gemeinsame Speicherbereiche definiert wurden und reserviert für diese Bereiche freien Speicherplatz am Ende des Programms oder des gebundenen Bindemoduls.

3. **Relativierung der Adressen**

   Alle Adressen in den eingelesenen Modulen sind bezogen auf den Beginn des Bindemoduls. Diese Adressen müssen so umgesetzt werden, daß sich im gebundenen Programm alle Adressen auf den Beginn des Programms und im Bindemodul alle Adressen auf den Beginn dieses Moduls beziehen.

Nach Abschluß des Bindens und sofern der Bindevorgang erfolgreich verlaufen ist, hinterlegt TSOSLNK

- den gebundenen Bindemodul (nach einer MODULE-Anweisung) in der temporären EAM-Bindemoduldatei oder als Element vom Typ R in einer Programmbibliothek,
- das gebundene Programm (nach einer PROGRAM-Anweisung) in einer katalogisierten Datei oder als Element vom Typ C in einer Programmbibliothek.

Neben dem Protokoll des Programmlaufs gibt TSOSLNK auf Anforderung aus:

- eine Querverweisliste,
- eine Programmübersicht in ausführlicher Form oder in Kurzform,
- eine alphabetisch sortierte Liste mit den Namen aller Programmabschnitte und Einsprungstellen.

Bild 3 Programmablauf von TSOSLNK

Überwachung des Binderlaufs mit Jobvariablen

Mit dem Aufruf

```
/START-PROGRAM FROM-FILE=$TSOSLNK,MONJV=jvname
```

kann der Binderlauf mit Jobvariablen überwacht werden (siehe Handbuch "Jobvariablen" [6]).

"jvname" bezeichnet den Namen einer programmüberwachenden Jobvariablen, in die das Betriebssystem während des Binderlaufs die Zustandswerte $T (Programm wurde normal beendet) oder $A (Programm wurde nicht normal beendet) setzt. Die Zustandswerte beinhalten eine Zustandsanzeige und einen Beendigungscode.

Die folgende Tabelle zeigt den Zusammenhang der verschiedenen Werte der Jobvariablen:

Zustandswert	Beendigungscode	Zustandsanzeige
$T	0	000
$T	1	001
$T	1	002
$T	1	003
$A	2	004
$A	3	005

Zustandsanzeige:

000 der Binderlauf wurde ohne Fehler beendet.

001 der Binderlauf wurde ohne Fehler beendet, Warnungsmeldungen wurden ausgegeben.

002 der Binderlauf wurde ohne Fehler beendet, einige unbefriedigte Externverweise wurden nicht aufgelöst.

003 der Binderlauf wurde normal beendet, einige Fehler wurden erkannt.

004 der Binderlauf wurde mit Fehlern beendet, Fehler in der Eingabe oder Anwenderfehler wurden erkannt. Kein Modul wurde erzeugt bzw. der erzeugte Modul kann nicht verwendet werden.

005 der Binderlauf wurde mit Fehlern beendet (interner Binderfehler). Kein Modul wurde erzeugt bzw. der erzeugte Modul kann nicht verwendet werden.

Beendigungscode:

0	der Binderlauf wurde ohne Fehler beendet.
1	der Binderlauf wurde ohne Fehler beendet. Warnungsmeldungen wurden ausgegeben.
2	der Binderlauf wurde mit Fehlern beendet. Die Module enthalten Fehler oder sind unvollständig.
3	der Binderlauf wurde wegen eines schwerwiegenden Fehlers beendet.

Bindevorgang

Befriedigen von Externverweisen

Gesteuert von INCLUDE-Anweisungen oder direkt von SYSDTA liest der Binder TSOSLNK Bindemodule ein. Anschließend hat er die Aufgabe, aus den einzelnen Modulen, bei denen Externverweise anzeigen, daß sie zum Ablauf andere Bindemodule benötigen, eine Einheit zu bilden. Meist reichen dazu die bereits eingelesenen Module (0. Stufe) nicht aus. Der Binder muß weitere Bindemodule aufsuchen, um für sämtliche Externverweise Programmabschnitte oder Einsprungstellen gleichen Namens zu finden. Dieser Suchvorgang verläuft beim TSOSLNK in Stufen:

0. Stufe

Der Binder versucht, Externverweise mit Adressen aus den Bindemodulen zu verknüpfen, die er aufgrund von INCLUDE-Anweisungen oder direkt von SYSDTA eingelesen hat.
Findet der Binder dabei eine Einsprungstelle (ENTRY), die den gleichen Namen wie ein Programmabschnitt (CSECT) hat, befriedigt er den Externverweis mit der Adresse des Programmabschnitts

1. Stufe

Zuerst prüft der Binder, ob RESOLVE-Anweisungen eingegeben wurden. Mit ihrer Hilfe versucht er, die noch offenen Externverweise zu befriedigen.

Zuerst bearbeitet er dabei die RESOLVE-Anweisungen, in denen Namen von Einsprungstellen bzw. Programmabschnitten explizit aufgeführt sind, und zwar in der Reihenfolge, in der die Anweisungen eingegeben wurden. Erst dann sucht er die noch ungelösten Externverweise in Bindemodulbibliotheken, deren Namen die übrigen evtl. vorhandenen RESOLVE-Anweisungen nennen. Dabei bearbeitet der Binder die Bibliotheken in der umgekehrten Reihenfolge, wie sie mit RESOLVE eingegeben wurden, also von hinten nach vorne (siehe auch RESOLVE-Anweisung, Seite 132ff).

Innerhalb einer Bibliothek beginnt der Binder die Suche beim lexikalisch *höchstwertigen* Element. Z.B. wird der Modul ABC vor dem Modul AB durchsucht.

2. Stufe

Schließlich durchsucht der Binder die Datei TASKLIB nach passenden Einsprungstellen oder Programmabschnitten. Gibt es für die Benutzerkennung der laufenden Task keine Datei namens TASKLIB, so nimmt er die Bindemodulbibliothek des Systems $TSOS.TASKLIB. Diese Stufe wird nicht durchlaufen, wenn die NCAL-Anweisung gegeben wurde.

Die 1. und 2. Stufe dieses Suchvorgangs bezeichnet man auch als **Autolink-Funktion** des Binders. Sie ermöglicht das automatische Einbinden der Bindemodule, die nicht aufgrund von INCLUDE-Anweisungen oder von SYSDTA eingelesen wurden. Die Autolink-Funktion erspart vor allem den Benutzern höherer Programmiersprachen, die oft recht zahlreich benötigten Module des Laufzeitsystems mit INCLUDE-Anweisungen angeben zu müssen.

Mit EXCLUDE-Anweisungen kann man die Autolink-Funktion teilweise oder auch vollständig unterdrücken. Für zwei Gruppen von Externverweisen wird kein Autolink durchgeführt:

- Externverweise, die mit den Zeichen I$... anfangen,
- bedingte Externverweise (WXTRN-Anweisungen).

Der Binder versucht, diese Verweise zwar aus all den Bindemodulen zu befriedigen, aus denen er das Programm aufbaut; er fügt ihretwegen aber keine neuen Bindemodule an. Bei ihnen entfällt also die Autolink-Funktion.
Der Binder ignoriert RESOLVE-Anweisungen und durchsucht für sie nicht die TASKLIB. Kann er sie nicht befriedigen, so bricht er ihretwegen den Binderlauf nicht ab. Die Namen der WXTRNs und der I$-Externverweise gibt er in die Liste der unbefriedigten Externverweise aus. Die Liste wird ausgegeben, wenn der Operand WUNSAT=Y angegeben wurde.

Findet der Binder einen Programmabschnitt oder eine Einsprungstelle, die den Externverweis befriedigt, so fügt er den zugehörigen Bindemodul an die übrigen Module des Programms an und ersetzt den Externverweis durch die sich nun ergebende Adresse. Gibt es einen Namen mehrfach, benutzt der Binder nur den ersten, den er findet.

Jedem Externverweis, den der Binder nicht befriedigen kann, gibt er die Adresse X'FFFFFFFF', sofern der Benutzer nicht mit einer ERREXIT-Anweisung einen anderen Wert vereinbart hat.

Bleiben Externverweise unbefriedigt, so wird der Binderlauf abgebrochen, sofern nicht eingegeben wurde:

- der Operand LET=Y in der PROGRAM- bzw. MODULE-Anweisung,
- die LET-Anweisung,
- die BIND-Anweisung oder
- die CONTINUE-Anweisung.

Beispiel für die Befriedigung von Externverweisen

```
/START-PROGRAM FROM-FILE=$TSOSLNK
*PROGRAM ...
*INCLUDE MOD1,BIB1
*INCLUDE *
*RESOLVE MOD2,BIB2
*RESOLVE ,BIB3
*END
```

0. Stufe zur Befriedigung der Externverweise: Nach Abschluß der Eingabe wurden aufgrund der beiden INCLUDE-Anweisungen die Bindemodule MOD1, MX und N eingelesen. An die Stelle der Externverweise N und MX trägt der Binder die Adressen dieser Programmabschnittsnamen ein.

1. Stufe zur Befriedigung der Externverweise: Aufgrund der RESOLVE-Anweisungen liest der Binder aus der Bibliothek BIB2 den Modul MOD2 und kann aus der Bibliothek BIB3 die Externverweise X und Y befriedigen, indem er die betreffenden Externverweise durch die sich ergebenden Adressen ersetzt.

2. Stufe zur Befriedigung der Externverweise: Die noch offenen Externverweise versucht der Binder aus der Bibliothek TASKLIB zu befriedigen.

Bild 4 Beispiel für die Befriedigung von Externverweisen#pt

Bild 5 Beispiel für die Befriedigung von Externverweisen

Falls nach der 2. Stufe immer noch unbefriedigte Externverweise vorhanden sind, wird eine Liste der unbefriedigten Externverweise und eine Liste der unbefriedigten Externverweise mit dem Namen I$... auf SYSLST und SYSOUT ausgegeben. Diese Listen können in der PROG- und MOD-Anweisung auch unterdrückt werden.

Stapelbetrieb

Im Stapelbetrieb wird der TSOSLNK-Lauf normal beendet und die Liste der unbefriedigten Externverweise ausgegeben.

Dialogbetrieb

– SYSDTA ist einer Datenstation zugewiesen:
  Die Steuerung wird an den Benutzer übergeben. Er kann weitere TSOSLNK-Anweisungen eingeben.
– SYSDTA ist einer katalogisierten Datei zugewiesen: TSOSLNK gibt die Meldung

```
%    LNK0026 DO YOU WANT TO REDIRECT SYSDTA? Y/N
```

  aus.

  Antwort Y: Ein BKPT-Makro wird abgesetzt. Der Benutzer kann SYSDTA auf die Datenstation zuweisen und nach einem RESUME-Kommando weitere TSOSLNK-Anweisungen eingeben.

  Antwort N: Falls der Wert LET=YES im TSOSLNK-Lauf gesetzt wurde, wird der Lauf fortgesetzt; ansonsten wird er abgebrochen.

  Werden im Dialogbetrieb weitere Anweisungen eingegeben, so wird der Bindevorgang für die unbefriedigten Externverweise wiederholt.

Nachdem der Binder die Externverweise befriedigt hat, liegen ihm sämtliche benötigten Bindemodule vor, aus denen er eine ladbare Einheit bilden soll. Die Bindemodule stehen in der Reihenfolge, in der man sie in den INCLUDE-Anweisungen angegeben hat. Anschließend fügt der Binder die Module an, die aufgrund der Autolink-Funktion gefunden wurden (Autolinkmodule).

Bei größeren Programmen kann der Benutzer durch die Reihenfolge der Binderanweisungen und damit der Reihenfolge der Bindemodule ggf. die Seitenwechselrate beeinflussen.

Behandlung der COMMON-Bereiche

Anschließend an die Autolinkmodule, d.h. an das Programmende, fügt der Binder evtl. vorhandene COMMON-Bereiche an. Dies sind Abschnitte, die zum Zeitpunkt des Bindens noch keine Daten oder Befehle enthalten, sondern nur Platz dafür reservieren. Diese Bereiche können bei Programmlauf als Daten-Verständigungsbereiche zwischen verschiedenen Modulen des Programms verwendet oder als Platz für Programmabschnitte eingesetzt werden.

Der Binder TSOSLNK weist COMMON-Bereichen, die in verschiedenen Modulen definiert sind und alle den gleichen Namen haben, einen gemeinsamen Speicherbereich am Programmende zu. Die Länge dieses Bereichs wählt er so groß, daß der längste COMMON-Bereich dieses Namens gerade hineinpaßt. Hat ein Programmabschnitt (CSECT) den gleichen Namen wie COMMON-Bereiche, so gibt der Binder diesem Programmabschnitt die Adresse des COMMON-Bereichs, d.h. der COMMON-Bereich wird beim Laden des Programms mit diesem Programmabschnitt initialisiert.

Unbenannte COMMON-Bereiche behandelt der Binder wie benannte mit der Ausnahme, daß er das Namensfeld (8 Leerzeichen) nicht mit den Namen von Programmabschnitten vergleicht. Text in unbenannten Programmabschitten wird also nicht verwendet, um einen COMMON-Bereich zu initialisieren.

Hinweis

Besonderheiten, die sich aus der Verwendung von Überlagerungsstrukturen ergeben, sind auf Seite 51ff gesondert behandelt.

Relativierung der Adressen

Unter Relativierung versteht man den Teil des Bindevorgangs, bei dem der Binder sämtliche virtuellen Adressen in den Bindemodulen des Programms ihrer relativen Lage anpaßt. Hat ein Modul z.B. die relative Lage 500 bezogen auf den Anfang des Programms, so werden sämtliche Adreßbezüge im Modul um 500 erhöht.

Im Standardfall legt der Binder die Adressen fest. Die Daten in der Programmdatei sind dann ein Abbild von dem, was zur Ladezeit im Speicher steht. Man bezeichnet das Format dieser Programme daher auch als **Speicherabbildformat**.

Der Binder erstellt kein Speicherabbildformat, wenn in der PROGRAM-Anweisung COREIM=N angegeben wird. In diesem Fall übernimmt der Binder Relativierungsinformationen in das Programm, d.h. die Adreßkonstanten erhalten noch nicht ihre endgültigen Werte.

Behandlung von Eingabebibliotheken

TSOSLNK unterstützt als Eingabe Bindemodulbibliotheken (OML, Object Modul Libraries) und Programmbibliotheken (PL, Program Libraries, Elementtyp R).

Der Binder erkennt beim Eröffnen einer Bibliothek, welches Format die Bibliothek hat.

Verhalten bei Zugriffssperren auf Bibliotheken

Stellt der Binder fest, daß ein Zugriff zu einer Bibliothek nicht möglich ist, z.B. weil sie von einer anderen Task exklusiv angefordert wurde, oder weil eine andere Task auf die Bibliothek schreibend zugreift, so verfährt TSOSLNK wie folgt:

Stapelbetrieb

TSOSLNK versucht innerhalb von zehn Minuten mehrfach, Zugriff zur Bibliothek zu erhalten. Ist dies nicht möglich, so wird der Binderlauf mit einer Fehlermeldung abgebrochen.

Dialogbetrieb

TSOSLNK gibt eine Meldung aus, daß ein Zugriff zur betreffenden Bibliothek oder zu dem betreffenden Element nicht möglich ist und fragt dann ab, ob weitere Zugriffsversuche erfolgen sollen.

Gibt der Benutzer N ein, so setzt TSOSLNK den Bindelauf mit einer Fehlermeldung fort.

Gibt der Benutzer Y ein, so versucht TSOSLNK innerhalb einer Minute mehrfach, Zugriff zur Bibliothek zu erhalten. Ist dies nicht möglich, so wird wiederum abgefragt, ob weitere Zugriffsversuche erfolgen sollen.

Ist TSOSLNK im Dialogbetrieb innerhalb einer Prozedur aufgerufen worden, so verhält sich der Binder bei einer Zugriffssperre wie im Stapelbetrieb.

Verhalten bei kennwortgeschützten Bibliotheken

Erkennt der Binder, daß das Schreib-Kennwort für eine kennwortgeschützte Bibliothek nicht eingegeben wurde, wird der Lauf

- im Dialogbetrieb mit der Meldung LNK0264 unterbrochen. Der Benutzer kann anschließend das Kennwort eingeben und mit dem Kommando RESUME das Programm wieder fortsetzen.
- im Stapelbetrieb mit einer Fehlermeldung abgebrochen.

Behandlung von Namenskonflikten

Namenskonflikte können entstehen, wenn in den Bindemodulen mehrere ESD-Einträge (External Symbolic Dictionary) denselben Namen haben.

Nicht jede Namensgleichheit ist auch ein Namenskonflikt.

In Tabelle 1 ist das Verhalten des TSOSLNK bei Namensgleichheit dargestellt.

	CS	EN	EX	CM	VC	DS	PR	XDD	XDR	WX
CSECT	1	11	2	3	2	4	4	4	4	2
ENTRY		11	2	4	2	4	4	4	4	2
EXTRN			5	2	5	4	4	4	4	5
COMMON				6	2	4	4	4	4	2
V-CONST					5	4	4	4	4	5
DSECT						7	4	4	4	4
P-REG							8	4	4	4
XDSEC D								9	10	4
XDSEC R									7	4
WXTRN										5

Tabelle 1 Verhalten des TSOSLNK bei Namensgleichheiten

(1) Namenskonflikt. TSOSLNK benutzt den ersten ESD-Eintrag, um die Externverweise zu befriedigen. Bei der MODULE-Anweisung wird auf Namenskonflikte dieser Art überprüft, wenn der Operand NA-COL den Wert STD oder ABORT hat. Vorgebundene Module mit Namenskonflikten kann der dynamische Bindelader DBL nicht weiterverarbeiten.

(2) Die Externverweise werden befriedigt.

(3) Wenn ein Lademodul erzeugt wurde, wird der COMMON-Bereich mit dem Inhalt des Programmabschnitts (CSECT) initialisiert. Bei der MODULE-Anweisung wird auf Namenskonflikte überprüft, wenn der Operand NA-COL den Wert STD oder ABORT hat. Vorgebundene Module mit Namenskonflikten kann der dynamische Bindelader DBL nicht weiterverarbeiten.

(4) Die Namensgleichheit ist nicht von Bedeutung, da die Art der ESD-Einträge verschieden ist.

(5) Beide Externverweise werden durch denselben Programmabschnitt (CSECT), Einsprungpunkt (ENTRY) oder COMMON-Bereich befriedigt.

(6) Die Länge des COMMON-Bereichs wird so ausgedehnt, daß der längste Bereich hineinpaßt.

(7) Namensgleichheiten werden nicht entdeckt. Beim Binden mit der MODULE-Anweisung bleibt nur ein ESD-Eintrag erhalten.

(8) Bei Pseudoregistern wird die Definition gewählt, die die stärksten Forderungen bezüglich Länge und Alignment beinhaltet.

(9) Namensgleichheiten werden nicht entdeckt. Es dürfen nicht mehrere XDSEC D gleichen Namens in den Bindemodulen vorkommen.

(10) Der Externverweis wird mit dem XDSEC D-Feld befriedigt.

(11) Namensgleichheiten werden nicht als Konflikte betrachtet.

Ausgaben von TSOSLNK

Als Ergebnis des Bindevorgangs hinterlegt TSOSLNK

- ein Programm aus einem oder mehreren Segmenten in einer katalogisierten Datei bzw. als ein Element einer Programmbibliothek (PROGRAM-Anweisung) oder
- einen einzigen Bindemodul in der temporären EAM-Bindemoduldatei oder als ein Element einer Programmbibliothek (MODULE-Anweisung).

Listen mit Informationen über die gebundenen Module schreibt TSOSLNK in die Systemdatei SYSLST. Ein Protokoll des Binderlaufs und eventuelle Fehlermeldungen gibt TSOSLNK in die Systemdatei SYSOUT aus.

Programme

Die Anweisung PROGRAM veranlaßt, daß eingelesene Bindemodule zu einem Programm gebunden werden.

Das Programm wird nach dem Binden in ein Format umgesetzt, das vom statischen Lader ELDE verarbeitet werden kann und in einer PAM-Datei oder in einem Bibliothekselement abgespeichert wird.

Existieren die PAM-Datei oder die Bibliothek noch nicht, so werden sie eingerichtet. Für die PAM-Datei wird dann ein FILE-Makro abgesetzt. Dieser FILE-Makro legt auch den vom Binder errechneten, maximal benötigten Speicherbereich fest.
Bereiche, die nicht mit Textinformation belegt sind (z.B. durch die Assembler-Anweisungen ORG oder DS freigehalten), werden nicht in die Datei übernommen. Die für diese Bereiche berechneten PAM-Seiten werden nach dem Schreiben der Programmdatei wieder freigegeben.

Wenn diese Bereiche sehr groß sind, kann es effektiver sein, wenn der Benutzer die Primärzuweisung des Speicherplatzes mit dem /FILE-Kommando selbst vornimmt.

Den Namen und die Zugriffberechtigung für diese Datei kann der Anwender in der PROGRAM-Anweisung bestimmen. Informationen für den Lader ELDE kann der Anwender ebenfalls in der PROGRAM-Anweisung angeben, z.B. den Startpunkt für den Programmlauf oder Speicherplatzanforderungen.

Pamkey-Eliminierung

In Zukunft unterstützt das BS2000 nur noch Platten mit festen Blockgrößen (2Kbyte, 4Kbyte...). Diese Blockgrößen verhindern die einfache Unterbringung der PAM-Schlüssel bei PAM-Dateien. Aus diesem Grund entfällt zukünftig der PAM-Schlüssel. Dieser Vorgang wird als Pamkey-Eliminierung bezeichnet.

Werden vom Binder TSOSLNK Programme in einer PAM-Datei abgespeichert, kann der Benutzer in der PROGRAM-Anweisung mit dem Operanden PAM-KEY das Format der PAM-Datei (mit oder ohne PAM-Schlüssel) festlegen. Für das Format der PAM-Datei kann auch ein Standardwert festgelegt werden.
Dieser ist durch den Wert des KLasse-2-Systemparameters BLKCTRL bestimmt, der bei der Systeminstallation festgelegt wird (siehe Handbuch "Systeminstallation" [4]). Ist BLKCTRL nicht angegeben, wird in eine PAM-Datei mit PAM-Schlüssel geschrieben.

Vorgebundene Bindemodule (Großmodule)

Mit der MODULE-Anweisung werden mehrere Bindemodule zu einem einzigen Bindemodul gebunden. Dieser vorgebundene Bindemodul (Großmodul) kann in die temporäre EAM-Bindemoduldatei der laufenden Task oder in eine Programmbibliothek geschrieben werden und wieder als Eingabe für TSOSLNK und DBL dienen. Dieses Vorbinden beschleunigt den späteren Ladevorgang.
Im Zusammenhang mit dem Modulbinden kann mit der LINK-SYMBOLS-Anweisung auch bestimmt werden, wie die Symbole CSECT und ENTRY zu behandeln sind, d.h. in welchem Umfang sie für spätere Binderläufe erkennbar sein sollen.

Für den vorgebundenen Modul gilt folgendes:

- Wenn bei der Anweisung LINK-SYMBOLS die Operanden *HIDE oder *KEEP angegeben wurden, so enthält der Modul ein komplettes ESD, in dem jedoch CSECTs und ENTRYs maskiert werden können.
- Wenn bei der Anweisung LINK-SYMBOLS der Operand *NOESD eingegeben wurde, so enthält der Modul nur einen ESD-Satz für den Modulnamen. Der Binder verhält sich dann beim Modulbinden wie die TSOSLNK Versionen ≤ 16.4.
- Testhilfedaten werden nicht erzeugt, d.h. das Internadreßbuch (ISD) fehlt.
- Wenn bei der Anweisung LINK-SYMBOLS die Operanden *HIDE oder *KEEP gesetzt wurden, darf in der MODULE-Anweisung der Name des ersten Programmabschnitts angegeben werden, nicht jedoch der Name eines anderen ESD-Eintrags.

Dieses Bindeverfahren - Vorbinden mit TSOSLNK, binden mit DBL und direkter Aufruf - wird häufig dann gewählt, wenn ein Programm getestet werden soll oder wenn ein Programm so selten gebraucht wird, daß ein Abspeichern des gebundenen Programms zu aufwendig wäre.

Bild 6 Eingabemodule und Ausgabemodule des Binders TSOSLNK

Listen

Listen mit Informationen über die gebundenen Module schreibt TSOSLNK in die Systemdatei SYSLST. Ein Protokoll des Binderlaufs und eventuelle Fehlermeldungen gibt TSOSLNK in die Systemdatei SYSOUT aus.

Welche Informationslisten ausgegeben werden, bestimmt der Anwender in der PROGRAM-Anweisung oder in der MODULE-Anweisung und in der XREF-Anweisung:

Der Operand MAP	fordert die Programmübersicht an.
Der Operand LIST	fordert die Kurzform der Programmübersicht über SYSOUT an.
Der Operand XREF	oder die XREF-Anweisung fordern eine Querverweisliste an.
Der Operand SORT	fordert eine alphabetisch sortierte Liste aller Programmabschnittsnamen und Einsprungstellen an.
Der Operand PR	gibt den Inhalt der Pseudoregister aus.
Der Operand XDSEC	listet die externen Pseudoabschnitte auf.
Der Operand CMAP	fordert eine Programmübersicht und eine Querverweisliste an.

Mit dem Auftragsschalter 4 kann bei der Ausgabe auf SYSLST der Seitenvorschub gesteuert werden. Ist der Auftragsschalter 4 gesetzt, wird nach jeder vollen Seite ein Seitenvorschub durchgeführt. Ist der Auftragsschalter 4 nicht gesetzt, wird für jede Informationsliste mit einer neuen Seite begonnen.

Die **Programmübersicht** enthält auf ihrer ersten Seite eine Übersicht über das gesamte Programm, nämlich über die Anzahl der Segmente (Lademodule) und der darin eingebundenen Bindemodule, die Anzahl der Externverweise und Einsprungstellen, Ladeadresse, Startadresse und Länge des gesamten Programms. An diese Aufstellung schließt sich eine Beschreibung der einzelnen Segmente (Lademodule) an, aus denen das Programm besteht. Hier gibt der Binder die einzelnen Bindemodule an und beschreibt Name, Ladeadresse, Länge, Attribute und Herkunft.

Die **Querverweisliste** beschreibt Einsprungstellen und Externverweise für jeden Bindemodul des Programms. Für die Einsprungstellen werden Namen und Adressen aufgeführt. Bei den Externverweisen werden Namen, Adressen usw. angegeben und wie sie befriedigt wurden.

Die **alphabetische Liste der Programmabschnitte und Einsprungstellen** enthält - aufsteigend sortiert - die Namen aller Programmabschnitte und Einsprungstellen. Sie weist für jeden Eintrag aus:

- den Namen des Programmabschnitts (CSECT Name) oder der Einsprungstelle (ENTRY Name).
- die Anfangsadresse in sedezimaler und dezimaler Form (ADRESS).
- eine Kennzeichnung als Programmabschnitt (CSECT) oder Einsprungstelle (ENTRY).
- den Namen des Moduls (MODULE CONTAINING) und/oder des Segments (SEGMENT ENTRY), in dem der Programmabschnitt oder die Einsprungstelle liegen.

Die Liste wird aus Platzgründen zweispaltig gedruckt. Sie ist daher in folgender Reihenfolge zu lesen:

Eintrag 1	Eintrag 2
Eintrag 3	Eintrag 4
...	...

Das **Protokoll** (Ausgabe auf SYSOUT) enthält die eingegebenen Binderanweisungen und eventuelle Fehlermeldungen.

Beispiel für Binderlisten eines unsegmentierten Programms

```
/START-PROGRAM FROM-FILE=$TSOSLNK
%  BLS0500 PROGRAM 'TSOSLNK', VERSION '21.0D00' OF '90-05-10' LOADED.
*PROGRAM PGM,FILENAM=UNSEG.LADE,XREF=Y,LIST=Y,SORT=Y
PROGRAM PGM,FILENAM=UNSEG.LADE,XREF=Y,LIST=Y,SORT=Y
*INCLUDE EINXEINS,LMS.BIBL
INCLUDE EINXEINS,LMS.BIBL
*RESOLVE ,$RZ.COB1MODLIB
RESOLVE ,$RZ.COB1MODLIB
*END
END
%  LNK0500 PROGRAM BOUND
%  LNK0503 PROG FILE WRITTEN: UNSEG.LADE
%  LNK0504 10 PAM PAGES USED. TSOSLNK RUN FINISHED
```

Nach Eingabe der oben genannten Anweisungen gibt der Binder TSOSLNK aus:

- eine Kurzform der Programmübersicht nach SYSOUT (wegen LIST=Y in der PROGRAM-Anweisung).
- die vollständige Programmübersicht nach SYSLST (wegen MAP=Y als Standardwert in der PROGRAM-Anweisung).
- die Querverweisliste nach SYSLST (wegen XREF=Y in der PROGRAM-Anweisung).
- eine alphabetische Liste der Programmabschnitte (CSECTs) und Einsprungstellen (ENTRYs) nach SYSLST (wegen SORT=Y in der PROGRAM-Anweisung).

Kurzform der Programmübersicht

Die Kurzform der Programmübersicht ist bei den Listen "Programmübersicht und Querverweisliste" erklärt (siehe Seite 37ff).

```
********************************************************************************
**                                                                            **
**                                                                            **
** PROGRAM:  PGM                                                              **
**    FILE:  UNSEG.LADE                                                       **
**                                                                            **
**                                                                            **
********************************************************************************
```

	DEC		HEX	DEC
NO. OF SEGMENTS:	1	LOAD ADDR.:	00000000	0
NO. OF OVERLAY PTS.:	0	EXEC. START ADDR.:	00000000	0
NO. OF REGIONS:	0	COMPUTED LENGTH:	000039A0	14.752
NO. OF MODULES:	13	MAXIMUM LENGTH:	000039A0	14.752
NO. OF EXTRNS:	18			
NO. OF ENTRY PTS.:	21			
CLASS: 2		CONTROL: Y		
COREIM: Y		TEST AID:N		
START NAME: EINXEINS				

```
*******************************************************************************
**                                                                           **
**   SEGMENT: %ROOT                                  HEX           DEC       **
**   ------------------                              ---------------------   **
**                                LOAD ADDR.:        00000000            0  **
**   SEGMENT NUMBER:     1       SEGMENT LENGTH:    000039A0       14.752  **
**                                                                           **
**                                                                           **
*******************************************************************************
```

MODNAME / DATE	ADDR.	LENGTH	ATTRIBUTE	BIND METHOD
EINXEINS/861110	00000000	0000043D	SS	EXPLICIT
ITC0DSA0/860710	00000440	000001B8	SS	AUTOLINK
ITC0ACA0/860710	000005F8	000001C8	SS	AUTOLINK
ITCCTCU1/860710	000007C0	00000118	SS	AUTOLINK
ITC0END0/860710	000008D8	000004F0	SS	AUTOLINK
ITC0POVH/860710	00000DC8	00000878	SS	AUTOLINK
ITC0BEG0/860710	00001640	00000390	SS	AUTOLINK
ITCCMSG3/860710	000019D0	00000370	SS	AUTOLINK
ITC0UPC2/860710	00001D40	00000434	SS	AUTOLINK
ITC0UPS3/860710	00002178	00000290	SS	AUTOLINK
ITCCACA0/860710	00002408	00000C90	SS	AUTOLINK
ITCCDSA0/860710	00003098	000006C0	SS	AUTOLINK
ITCDMSG3/860710	00003758	00000248	SS	AUTOLINK

Liste der Abkürzungen

In dieser Liste sind die Abkürzungen erklärt, die in der Programmübersicht und Querverweisliste verwendet werden (z.B. für die Attribute AMODE/RMODE).

```
BS2000  LINK EDITOR ── PROGRAM MAP AND CROSS REFERENCE LISTING  11:28:40   1990-10-14  PAGE  2


LINKNAME   LIB.TYPE   FILENAME

IOSLK001     P.L      LMS.BIBL
IOSLK002     OML      $RZ.COB1MODLIB


ABBREVIATIONS USED IN THE MAP AND XREF:

ATTRIBUTES :

┌─────────────────────────────────────┐
: A : R : A : A : I : N : N : N : N :
: M : M : L : C : N : O : O : O : O :
: O : O : I : C : V :   :   :   :   :
: D : D : G : E : I : U : U : U : U :
: E : E : N : S : S : S : S : S : S :
:   :   :   : S :   : E : E : E : E :
├─────────────────────────────────────┤
: S :   :   :   :   :   :   :   :   : 24 BITS
: X :   :   :   :   :   :   :   :   : 31 BITS
: A :   :   :   :   :   :   :   :   : ANY (24 OR 31)
: - :   :   :   :   :   :   :   :   : NO MODE SPECIFIED
:   : S :   :   :   :   :   :   :   : 24 BITS
:   : A :   :   :   :   :   :   :   : ANY
:   : - :   :   :   :   :   :   :   : NO MODE SPECIFIED
:   :   : P :   :   :   :   :   :   : PAGE ALIGNED
:   :   : O :   :   :   :   :   :   : OTHER ALIGNEMENT
:   :   : - :   :   :   :   :   :   : NO ALIGNEMENT SPECIFIED
:   :   :   : R :   :   :   :   :   : READ ONLY
:   :   :   : - :   :   :   :   :   : READ-WRITE
:   :   :   :   : I :   :   :   :   : INVISIBLE
:   :   :   :   : - :   :   :   :   : VISIBLE
└─────────────────────────────────────┘

THE INVISIBLES ENTRIES ARE DISPLAYED IN PARENTHESES

BIND METHOD: EXPL=EXPLICIT, IMPL=IMPLICIT, ALTL=ALTERNATE LIBRARY, AUTO=AUTOLINK, PROM=PROMOTED

SYMBOL TYPE: VC=V-CONSTANT, ER=EXTERNAL REFERENCE, WX=WEAK EXTERNAL REFERENCE

-> : SPECIAL SIGN TO MARK A MODULE
```

Programmübersicht und Querverweisliste

```
   BS2000  LINK EDITOR --- PROGRAM MAP AND CROSS REFERENCE LISTING      11:28:40   1990-10-14  PAGE  3

*********************************************************************************************************
**                                                                                                     **
**                                                                                                     **
**       PROGRAM: PGM                                                  (01) -------- CLASS:  2   CONTROL:  Y   **
**       FILE:    UNSEG.LADE                                                         COREIM: Y   TEST AID: N   **
**                                                                                                     **
**                                                                                                     **
*********************************************************************************************************

                                              DEC                                 HEX           DEC
                                             -----                               -------------------------

                       {    NO. OF SEGMENTS:        1    LOAD ADDR.:             00000000          0 ------- (04)
(02) ---------         {    NO. OF OVERLAY PTS.:    0    EXEC. START ADDR.:      00000000          0 ------- (05)
                       {    NO. OF REGIONS:         0    COMPUTED LENGTH:        000039A0     14.752 } ----- (06)
                        {   NO. OF MODULES:        13    MAXIMUM LENGTH:         000039A0     14.752 }
(03) ----------         {   NO. OF EXTRNS:         18
                        {   NO. OF ENTRY PTS.:     21
                        {   START NAME: EINXEINS

*********************************************************************************************************
**                                                                                                     **
**       SEGMENT: %ROOT --------- (07)                                    HEX           DEC            **
**       ----------------                                                -------------------------     **
**                                                   LOAD ADDR.:         00000000          0           **
**       SEGMENT NUMBER:          1                  SEGMENT LENGTH:     000039A0     14.752           **
**                                                                                                     **
**                                                                                                     **
*********************************************************************************************************

  CONTROL SECTION                                 : OBJECT MODULE
  NAME     ADDRESS   LENGTH   ATTRIBUTE ESID : LENGTH   BIND  DATE   LINKNAME,NAME/VERSION(VARIANT)
  -------  --------  -------- --------- ---- : -------- ----  ------ ------------------------------
                                              :

->EINXEINS 00000000 0000043D SS        0001 : 0000043D EXPL 861110 IOSLK001,EINXEINS/@(0001)      }
->ITC0DSA0 00000440 00000030 SS        0001 : 000001B8 AUTO 860710 IOSLK002,ITC0DSA0              }
  ITCDDSA0 00000470 00000188 SS        0003 :                                                     }
                                            : ENTRIES:  I$DDSA0  00000470                         }
->ITC0ACA0 000005F8 00000030 SS        0001 : 000001C8 AUTO 860710 IOSLK002,ITC0ACA0              }
  ITCDACA0 00000628 00000198 SS        0003 :                                                     }
                                            : ENTRIES:  I$DACA0  00000628                         }
->ITCCTCU1 000007C0 00000118 SS        0001 : 00000118 AUTO 860710 IOSLK002,ITCCTCU1              }
->ITC0END0 000008D8 000004F0 SS        0001 : 000004F0 AUTO 860710 IOSLK002,ITC0END0              } ------- (08)
->ITC0POVH 00000DC8 00000878 SS        0001 : 00000878 AUTO 860710 IOSLK002,ITC0POVH              }
->ITC0BEG0 00001640 00000390 SS        0001 : 00000390 AUTO 860710 IOSLK002,ITC0BEG0              }
->ITCCMSG3 000019D0 00000370 SS        0001 : 00000370 AUTO 860710 IOSLK002,ITCCMSG3              }
->ITC0UPC2 00001D40 00000434 SS        0001 : 00000434 AUTO 860710 IOSLK002,ITC0UPC2              }
                                            : ENTRIES:  ITC0DBE0 00001D40  , I$ITCUPC 00001D40    }
->ITC0UPS3 00002178 00000290 SS        0001 : 00000290 AUTO 860710 IOSLK002,ITC0UPS3              }
                                            : ENTRIES:  I$ITCUPS 00002178                         }
->ITCCACA0 00002408 00000C90 SS        0001 : 00000C90 AUTO 860710 IOSLK002,ITCCACA0              }
```

```
   BS2000  LINK EDITOR ── PROGRAM MAP AND CROSS REFERENCE LISTING 11:28:40   1990-10-14  PAGE     4

  CONTROL SECTION                              : OBJECT MODULE
  NAME      ADDRESS   LENGTH   ATTRIBUTE ESID : LENGTH    BIND  DATE   LINKNAME,NAME/VERSION(VARIANT)
                                               :

->ITCCDSA0 00003098 000006C0 SS        0001 : 000006C0 AUTO 860710 IOSLK002,ITCCDSA0                  }
->ITCDMSG3 00003758 00000248 SS        0001 : 00000248 AUTO 860710 IOSLK002,ITCDMSG3                  }  ──── (08)
                                            : ENTRIES:  I$DMSG3  00003758

  EXTRN NAME (TYPE , ESID) IN MODULE IS SATISFIED BY MODULE IN SEGMENT  (NUMBER) SATISFACTION ──────────── (09)

  ITC0POVH     ER    0002   EINXEINS   00000DC8    ITC0POVH   %ROOT          1   IN SEGMENT
  ITC0END0     ER    0003   EINXEINS   000008D8    ITC0END0   %ROOT          1   IN SEGMENT
  ITCCTCU1     ER    0004   EINXEINS   000007C0    ITCCTCU1   %ROOT          1   IN SEGMENT
  ITC0ACA0     ER    0005   EINXEINS   000005F8    ITC0ACA0   %ROOT          1   IN SEGMENT
  ITC0DSA0     ER    0006   EINXEINS   00000440    ITC0DSA0   %ROOT          1   IN SEGMENT
  ITCCDSA0     ER    0002   ITC0DSA0   00003098    ITCCDSA0   %ROOT          1   IN SEGMENT
  ITCCACA0     ER    0002   ITC0ACA0   00002408    ITCCACA0   %ROOT          1   IN SEGMENT
  ITC0UPS3     ER    0002   ITC0END0   00002178    ITC0UPS3   %ROOT          1   IN SEGMENT
  ITC0UPC2     ER    0008   ITC0END0   00001D40    ITC0UPC2   %ROOT          1   IN SEGMENT
  ITCCMSG3     ER    0004   ITC0POVH   000019D0    ITCCMSG3   %ROOT          1   IN SEGMENT
  I$ITCTRC     WX    0005   ITC0POVH   FFFFFFFF                                  SUPPRESSED
  ITC0BEG0     ER    0006   ITC0POVH   00001640    ITC0BEG0   %ROOT          1   IN SEGMENT
  ITC0END0     ER    0007   ITC0POVH   000008D8    ITC0END0   %ROOT          1   IN SEGMENT
  I$ITCSEG     WX    0008   ITC0POVH   FFFFFFFF                                  SUPPRESSED
  I$ITCTOM     WX    0009   ITC0POVH   FFFFFFFF                                  SUPPRESSED
  ITCDMSG3     ER    0002   ITCCMSG3   00003758    ITCDMSG3   %ROOT          1   IN SEGMENT
  ITC0UPS3     ER    0002   ITC0UPC2   00002178    ITC0UPS3   %ROOT          1   IN SEGMENT
  ITC0UPC2     ER    0003   ITC0UPS3   00001D40    ITC0UPC2   %ROOT          1   IN SEGMENT
```

Erklärung

(01) Klasse-II-Programm, d.h. seitenwechselbar.

(02) Nur für segmentierte Programme von Bedeutung. Unsegmentierte Programme bestehen immer aus einem einzigen Segment (Lademodul).

(03) Angaben zum Aufbau des Programms. Es enthält 13 Bindemodule, 18 Externverweise und 21 Einsprungstellen.

(04) Ladeadresse 0, d.h. der Anfang des Klasse-6-Benutzerspeichers.

(05) Programmstart bei Adresse 0.

(06) Programmlänge.

(07) Name des Lademoduls ("Grundsegment"): %ROOT.

(08) Aufbau des Lademoduls %ROOT.

(09) Querverweisliste.
Sie enthält die Externverweise für jeden Bindemodul des Lademoduls %ROOT mit Angaben, ob und wie sie befriedigt werden.

Alphabetische Liste der Programmabschnitte und Einsprungstellen

```
   BS2000  LINK EDITOR      SORTED ENTRY POINTS LISTING  11:28:40   1990-10-14  PAGE  5

**********************************************************************************************************
**                                                                                                      **
**                                                                                                      **
**                                      PROGRAM:  PGM                                                   **
**                                                                                                      **
**                                                                                                      **
**********************************************************************************************************
```

CSECT NAME / ENTRY NAME	ADDRESS HEX	ADDRESS DEC	TYPE	MODULE CONTAINING	SEGMENT ENTRY	CSECT NAME / ENTRY NAME	ADDRESS HEX	ADDRESS DEC	TYPE	MODULE CONTAINING	SEGMENT ENTRY
EINXEINS	000000	0	CSECT	EINXEINS	%ROOT	I$DACA0	000628	1.576	ENTRY	ITC0ACA0	%ROOT
I$DDSA0	000470	1.136	ENTRY	ITC0DSA0	%ROOT	I$DMSG3	003758	14.168	ENTRY	ITCDMSG3	%ROOT
I$ITCUPC	001D40	7.488	ENTRY	ITC0UPC2	%ROOT	I$ITCUPS	002178	8.568	ENTRY	ITC0UPS3	%ROOT
ITCCACA0	002408	9.224	CSECT	ITCCACA0	%ROOT	ITCCDSA0	003098	12.440	CSECT	ITCCDSA0	%ROOT
ITCCMSG3	0019D0	6.608	CSECT	ITCCMSG3	%ROOT	ITCCTCU1	0007C0	1.984	CSECT	ITCCTCU1	%ROOT
ITCDACA0	000628	1.576	CSECT	ITC0ACA0	%ROOT	ITCDDSA0	000470	1.136	CSECT	ITC0DSA0	%ROOT
ITCDMSG3	003758	14.168	CSECT	ITCDMSG3	%ROOT	ITC0ACA0	0005F8	1.528	CSECT	ITC0ACA0	%ROOT
ITC0BEG0	001640	5.696	CSECT	ITC0BEG0	%ROOT	ITC0DBE0	001D40	7.488	ENTRY	ITC0UPC2	%ROOT
ITC0DSA0	000440	1.088	CSECT	ITC0DSA0	%ROOT	ITC0END0	0008D8	2.264	CSECT	ITC0END0	%ROOT
ITC0POVH	000DC8	3.528	CSECT	ITC0POVH	%ROOT	ITC0UPC2	001D40	7.488	CSECT	ITC0UPC2	%ROOT
ITC0UPS3	002178	8.568	CSECT	ITC0UPS3	%ROOT						

Beispiel für die Kurzform der Programmübersicht bei einem segmentierten Programm

```
/START-PROGRAM FROM-FILE=$TSOSLNK
%  BLS0500 PROGRAM 'TSOSLNK', VERSION '21.0D00' OF '90-05-10' LOADED.
*PROGRAM SGMT,FILENAM=SGMT.LADE,LIST=Y
PROGRAM SGMT,FILENAM=SGMT.LADE,LIST=Y
*INCLUDE SEGA,LMS.BIBL
INCLUDE SEGA,LMS.BIBL
*OVERLAY K,S2
OVERLAY K,S2
*INCLUDE SEGA40,*
INCLUDE SEGA40,*
*OVERLAY K,S3
OVERLAY K,S3
*INCLUDE SEGA50,LMS.BIBL
INCLUDE SEGA50,LMS.BIBL
*OVERLAY K,S4
OVERLAY K,S4
*INCLUDE SEGA60,LMS.BIBL
INCLUDE SEGA60,LMS.BIBL
*RESOLVE ,$RZ.COB1MODLIB
RESOLVE ,$RZ.COB1MODLIB
*END
END
%  LNK0500 PROGRAM BOUND
%  LNK0503 PROG FILE WRITTEN: SGMT.LADE
%  LNK0504 19 PAM PAGES USED. TSOSLNK RUN FINISHED
```

Nach Eingabe dieser Steueranweisungen gibt der Binder wegen der Vereinbarung LIST=Y in der PROGRAM-Anweisung folgende Kurzform der Programmübersicht aus:

```
*******************************************************************************
**                                                                           **
**                                                                           **
** PROGRAM:  SGMT                                                            **
**    FILE:  SGMT.LADE                                                       **
**                                                                           **
**                                                                           **
*******************************************************************************

                       DEC                          HEX           DEC
                      -----                      ----------------------
  NO. OF SEGMENTS:       4   LOAD ADDR.:         00000000             0
  NO. OF OVERLAY PTS.:   1   EXEC. START ADDR.:  00000208           520
  NO. OF REGIONS:        1   COMPUTED LENGTH:    00005921        22.817
  NO. OF MODULES:       24   MAXIMUM LENGTH:     00005921        22.817
  NO. OF EXTRNS:        49
  NO. OF ENTRY PTS.:    35
  CLASS:  2                  CONTROL: Y
  COREIM: Y                  TEST AID:N
  START NAME: SEGA
```

```
*******************************************************************************
**                                                                           **
**   SEGMENT: %ROOT                                    HEX          DEC      **
**   ------------------                                ---          ---      **
**                                LOAD ADDR.:          00000000           0  **
**   SEGMENT NUMBER:     1       SEGMENT LENGTH:      000055B8      21.944  **
**                                                                           **
**                                                                           **
*******************************************************************************
```

MODNAME / DATE	ADDR.	LENGTH	ATTRIBUTE	BIND METHOD
SEGA /871013	00000208	00000E03	SS	EXPLICIT
ITCXINT0/860710	00001010	000000CA	SS	AUTOLINK
ITC0DSA0/860710	000010E0	000001B8	SS	AUTOLINK
ITCMSCL0/860710	00001298	00000210	SS	AUTOLINK
ITC0SPC0/860710	000014A8	00000150	SS	AUTOLINK
ITCMSWR0/860710	000015F8	000003D4	SS	AUTOLINK
ITCMSRD0/860710	000019D0	000001E8	SS	AUTOLINK
ITCMSOP0/860710	00001BB8	00000308	SS	AUTOLINK
ITC0END0/860710	00001EC0	000004F0	SS	AUTOLINK
ITC0SEG0/860710	000023B0	00000080	SS	AUTOLINK
ITCMSIN1/860710	00002430	00000460	SS	AUTOLINK
ITC0POVH/860710	00002890	00000878	SS	AUTOLINK
ITC0BEG0/860710	00003108	00000390	SS	AUTOLINK
ITCCMSG3/860710	00003498	00000370	SS	AUTOLINK
ITCXERR1/860710	00003808	00000883	SS	AUTOLINK
ITCCSEG0/860710	00004090	0000024A	SS	AUTOLINK
ITC0UPC2/860710	000042E0	00000434	SS	AUTOLINK
ITC0UPS3/860710	00004718	00000290	SS	AUTOLINK
ITCXXIT1/860710	000049A8	00000304	SS	AUTOLINK
ITCCDSA0/860710	00004CB0	000006C0	SS	AUTOLINK
ITCDMSG3/860710	00005370	00000248	SS	AUTOLINK

```
*******************************************************************************
**                                                                           **
**   SEGMENT: S2                                       HEX          DEC      **
**                                                                           **
**                         LOAD ADDR.:        00005800         22.528      **
**   SEGMENT NUMBER:     2     SEGMENT LENGTH:    0000001A             26      **
**   OVL. NODE PT.: K        REGION:    1 HIGHER SEGMENT IN PATH:  %ROOT       **
**                                                                           **
*******************************************************************************

MODNAME / DATE     ADDR.    LENGTH   ATTRIBUTE   BIND METHOD

SEGA40  /-------  00005800  0000001A  SS          EXPL/EAM

*******************************************************************************
**                                                                           **
**   SEGMENT: S3                                       HEX          DEC      **
**                                                                           **
**                         LOAD ADDR.:        00005800         22.528      **
**   SEGMENT NUMBER:     3     SEGMENT LENGTH:    00000121            289      **
**   OVL. NODE PT.: K        REGION:    1 HIGHER SEGMENT IN PATH:  %ROOT       **
**                                                                           **
*******************************************************************************

MODNAME / DATE     ADDR.    LENGTH   ATTRIBUTE   BIND METHOD

SEGA50  /871013  00005800  00000121  SS          EXPLICIT

*******************************************************************************
**                                                                           **
**   SEGMENT: S4                                       HEX          DEC      **
**                                                                           **
**                         LOAD ADDR.:        00005800         22.528      **
**   SEGMENT NUMBER:     4     SEGMENT LENGTH:    000000FD            253      **
**   OVL. NODE PT.: K        REGION:    1 HIGHER SEGMENT IN PATH:  %ROOT       **
**                                                                           **
*******************************************************************************

MODNAME / DATE     ADDR.    LENGTH   ATTRIBUTE   BIND METHOD

SEGA60  /871013  00005800  000000FD  SS          EXPLICIT
```

Binden segmentierter Programme

Bedeutung der Segmentierung von Programmen

Während der Rechner ein größeres Programm ausführt, benötigt er meistens nicht gleichzeitig alle Programmteile. Diese Tatsache wird im BS2000 von zwei Verfahren ausgenutzt, um den Speicher zu entlasten:

- Der Ablaufteil des BS2000 gliedert ein Programm in Teile von je 4096 Byte (Seiten) und bringt während der Programmausführung nur die Seiten in den Hauptspeicher, die gerade für den Ablauf benötigt werden. Ein Abbild des gesamten Programms befindet sich währenddessen auf dem Seitenspeicher und wird bei Bedarf aktualisiert. Diesen Seitenwechsel führt das System für alle Programme durch.
- Daneben kann der Benutzer für sein Programm einzelne Teile, sog. Segmente, definieren, die getrennt geladen und unabhängig voneinander ausgeführt werden können. Zunächst bringt der Lader nur das Grundsegment in den Speicher, wo es während des ganzen Programmlaufs bleibt. Die anderen Segmente kann der Benutzer nachladen lassen, sobald sie für den Programmablauf erforderlich sind. Segmente können sich gegenseitig überlagern, d.h. sie belegen nacheinander einen gemeinsamen Speicherbereich bzw. Adreßraum; sie haben z.B. die gleiche Anfangsadresse (siehe Bild 7). Sie können geladen, überlagert, wieder geladen werden, so oft die Programmlogik dies erfordert. Segmentierte Programme unterliegen ebenfalls dem Seitenwechsel.

Die folgende Tabelle stellt Seitenwechsel und Segmentierung gegenüber.

Seitenwechsel	Segmentierung
Erfolgt automatisch bei allen Programmen. Keine Maßnahmen bei der Programmierung erforderlich.	Ist beim Programmieren und Binden zu berücksichtigen, d.h. zusätzlicher Aufwand für Benutzer.
Programmteile von je 4 KB werden aus dem Seitenspeicherbereich von Platten in den Hauptspeicher gebracht. Den Zeitpunkt legt der BS2000-Ablaufteil fest. Er ist nicht auf das einzelne Programm bezogen, sondern hat eine Optimierung der Hauptspeicherauslastung zum Ziel.	Programmteile variabler Größe, die eine ladbare Einheit bilden, werden aus der Programmdatei in den Hauptspeicher gebracht. Den Zeitpunkt bestimmt das Programm.
Der Hauptspeicher wird entlastet, da hintereinander der gleiche Speicherbereich (Seite) für verschiedene Programmteile verwendet werden kann.	Der virtuelle Adreßraum und der Seitenspeicher wird entlastet.
Programm kann mit TSOSLNK oder DBL gebunden werden.	Programm muß mit TSOSLNK gebunden werden.

Die Gegenüberstellung zeigt, daß man die Segmentierung nur benötigt, wenn der virtuelle Adreßraum nicht ausreicht, das gesamte Programm einschließlich der Daten aufzunehmen.

Es gibt eine weitere Möglichkeit, den Speicher zu entlasten:
Mit dem BIND-Makro oder LINK-Makro können SHARE-Module in einen Memory Pool (Klasse-6-Speicher) nachgeladen werden, die von mehreren Tasks benutzbar sind (siehe Handbuch "BLS" [1]).

Der Binder behandelt beliebig komplexe segmentierte Programme. Grenzen setzen nur das System (virtueller Adreßraum) und die Anlagenkapazität.

Bild 7 Beispiel für die Belegung des Adreßbereichs ohne und mit Überlagerungsstruktur

Aufbauen einer Überlagerungsstruktur

Wenn der Benutzer sich für die Segmentierung entscheidet, d.h. sein Programm entsprechend unterteilen und Segmente nachladen lassen will, muß er dies beim Programmieren und beim Binden berücksichtigen.

Bei Planung und Programmierung legt der Benutzer die Struktur des Programms fest, d.h. die Gliederung in einzelne Segmente. Er vereinbart die Stellen, an denen nachgeladen werden soll, wobei er den logischen Verlauf des Programms berücksichtigen muß. Durch die Wahl entsprechender Anweisungen entscheidet sich der Benutzer für eine der beiden Nachlademethoden: Bei der einen ruft das Programm den Lader ELDE mit dem LPOV-Makro direkt auf; bei der anderen erfolgt das Nachladen - vom Programm her gesehen - "automatisch", d.h. ein vom Binder eingefügter Programmteil bearbeitet die Nachladeanforderungen und aktiviert dann evtl. den Lader. Diese zweite Methode verwendet im Programm V-Konstanten bzw. CALL- und SEGLD-Makro (siehe Seite 63ff).

Beim Binden beschreibt der Benutzer dem TSOSLNK die gewünschte, für das Programm sinnvolle Überlagerungsstruktur mit Hilfe von OVERLAY-Anweisungen. Dies ist notwendig, weil für ein segmentiertes Programm meist mehrere Überlagerungsstrukturen denkbar sind (siehe Bild 7). Entsprechend den OVERLAY-Vereinbarungen vergibt der Binder dann die Ladeadressen der einzelnen Segmente, so daß z.B. während des Programmlaufs Segmente sich überlagern, weil sie beim Binden dieselbe Ladeadresse erhielten. Ebenfalls auf den Aufbau der Überlagerungsstruktur beziehen sich die Operanden CONTROL=... und XCAL=... in der PROGRAM-Anweisung sowie die Bindeanweisungen NOCTL und XCAL.

Diagrammdarstellung einer Überlagerungsstruktur

Eine Hilfe für den Benutzer ist die Darstellung der gewünschten Überlagerungsstruktur als Diagramm, das die Beziehungen zwischen den einzelnen Segmenten graphisch darstellt (siehe Bild 7).

Jede vertikale Linie in einem dieser Diagramme stellt ein Segment dar, d.h. eine ladbare Einheit, ein Lademodul, der aus einem oder mehreren Bindemodulen besteht. Das oberste Segment, das Grundsegment, wird zu Beginn der Programmausführung geladen und nie überlagert, während die übrigen Segmente erst bei Bedarf geladen werden.

Jede horizontale Linie bezeichnet man als Knoten oder auch als Adreßpegel. Alle Segmente, die an demselben Knoten anfangen, haben dieselbe Ladeadresse, die sich an das vorausgehende Segment anschließt. Sie können sich also während des Programmlaufs überlagern. Man bezeichnet sie daher auch als exklusive Segmente.

Alle Segmente, durch die sich eine verzweigungsfreie Linie ziehen läßt, liegen in einem gemeinsamen Ast. Alle Segmente zwischen Grundsegment und einem Segment X bezeichnet man als "höher als X" in dem betreffenden Ast des Diagramms. Alle Segmente unterhalb eines Segments X nennt man "niedriger als X", bezogen auf dieses Segment. Das Grundsegment ist also höher gegenüber allen übrigen Segmenten. Alle Überlagerungssegmente sind niedriger, bezogen auf das Grundsegment.

Jedem Segment ist eine **Stufennummer** zugeordnet, die durch die Anzahl der höheren Segmente in dem betreffenden Ast bestimmt ist. Das Grundsegment hat die Stufe Null.

Eine neue **Region** beginnt bei einem Knoten (Adreßpegel), der so liegt, daß eine Überlagerung durch vorangehende Segmente ausgeschlossen ist.

Beispiel für Diagrammdarstellungen bei Überlagerungsstrukturen

Bild 8 zeigt drei Beispiele für Überlagerungsstrukturen eines Programms, das aus 5 Segmenten besteht. Durch den logischen Programmablauf wird bestimmt, welche der möglichen Strukturen die sinnvollste ist.
Dem Binder TSOSLNK beschreibt man die Überlagerungsstrukturen mit der in Bild 8 angegebenen Folge von OVERLAY-Anweisungen. Die INCLUDE-Anweisungen beziehen sich auf die jeweils vorangehende OVERLAY-Anweisung und geben die Bindemodule an, die das betreffende Segment aufbauen.
Die Segmente, dargestellt durch vertikale Linien, sind mit %ROOT, S2, S3, S4, S5 bezeichnet. %ROOT ist das Grundsegment. Dieser Name ist vorgegeben. Die Knoten, im Diagramm durch horizontale Linien dargestellt, sind mit K1 und K2 bezeichnet und geben einen Adreßpegel an.

Bild 8 Beispiel für Überlagerungsstrukturen bei einem Programm aus 5 Segmenten

Beispiel A

Bei dieser Struktur sind S2, S3, S4 und S5 exklusive Überlagerungssegmente, d.h. Segmente mit der gleichen Anfangsadresse, die sich während des Programmlaufs gegenseitig überlagern. Sie haben alle 4 die Stufe 1.

Beispiel B

Bei dieser Anordnung sind die Segmente S4 und S5 von der Stufe 1, S2 und S3 von der Stufe 2 im Diagramm. S4, S3 und S2 liegen in einem Ast, wobei S4 z.B. höher als S2 ist, S3 als niedriger bezüglich S4 gilt.

Beispiel C

In dieser Überlagerungsstruktur bilden die Segmente S4 und S5 eine neue Region. Ihre Anfangsadressen liegen so, daß sie durch die vorangehenden Segmente nicht überlagert werden können.

Auflösen von Externverweisen bei segmentierten Programmen

Die Befriedigung von Externverweisen verläuft in segmentierten Programmen dreistufig wie bei den übrigen Programmen (siehe Seite 18ff). Alle Autolinkmodule bindet der TSOSLNK an das Ende des Grundsegments. Um einen Externverweis in der 0. Stufe zu befriedigen, sucht der Binder passende Namen von Programmabschnitten und Einsprungstellen, und zwar der Reihe nach:

1. in dem Segment, das den Externverweis enthält.
2. in den Segmenten, die im zugehörigen Ast des Diagramms höher liegen.
3. in den Segmenten, die in dem betreffenden Ast des Diagramms niedriger sind.
4. in Segmenten anderer Regionen.
5. in den Segmenten der gleichen Region und der gleichen Stufe, falls für den Binderlauf XCAL=Y in der PROGRAM-Anweisung vereinbart wurde. Für XCAL=N unterbleibt diese Suche.

Beim automatischen Nachladen (siehe Seite 53ff) werden in den Fällen 3., 4., und 5. Externverweise nicht direkt mit der Adresse des passenden Programmabschnitts oder Einsprungspunkts befriedigt, sondern mit Informationen versehen, die das automatische Nachladen während des Programmlaufs auslösen. Für das automatische Nachladen werden zum Erstellen der ENTAB-Einträge nur V-Konstanten (nicht EXTRN und A-Konstanten) verwendet.

Hinweis zu V-Konstanten

Bei segmentierten Programmen dürfen maximal 256 V-Konstanten auf andere Segmente verweisen.

Bearbeitung von COMMON-Bereichen bei segmentierten Programmen

Der Benutzer kann dem Sprachübersetzer unbenannte und benannte COMMON-Bereiche angeben. Der Binder versieht alle unbenannten und alle gleichnamigen COMMON-Bereiche mit jeweils derselben Anfangsadresse. Wenn es auch einen gleichnamigen Programmabschnitt (CSECT) gibt, dann erhält der gleichnamige COMMON-Bereich die Programmabschnittsadresse als Anfangsadresse (siehe auch Seite 23ff). Der gemeinsame Speicherbereich wird stets neu initialisiert, sobald der Programmabschnitt (CSECT) gleichen Namens nachgeladen wird.

In segmentierten Programmen reserviert der Binder den COMMON-Bereich im 1. Segment, das höher im Diagramm liegt als alle Segmente, die eine COMMON-Angabe oder einen Programmabschnitt dieses Namens enthalten. Dadurch ist der gemeinsame Bereich für alle betroffenen Segmente erreichbar und gleichzeitig mit ihnen geladen.

Beispiel für die Bearbeitung von COMMON-Bereichen in segmentierten Programmen

Ein Programm enthält 7 Segmente, 2 COMMON-Bereiche namens A und B sowie gleichnamige Programmabschnitte (CSECT). Bild 9 zeigt die Bereiche A und B an den Stellen, an denen sie entsprechend ihrer Reihenfolge bei der Eingabe in den Binder liegen. Das 2. Diagramm in Bild 9 stellt dar, daß der Binder die gemeinsamen Bereiche A und B so legt, daß sie für alle betroffenen Segmente erreichbar sind.

Hinweis zum Ladevorgang

Beim Laden ist der Bereich B des Grundsegments %ROOT noch leer. Der Text von Programmabschnitt B wird erst in den Bereich B gebracht, wenn Segment S7, das diesen Programmabschnitt enthält, geladen wird.

Entsprechend bleibt beim Laden des Segments S2 der Bereich A zunächst leer. Erst wenn Segment S3 geladen wird, füllt sich Bereich A mit dem Inhalt des Programmabschnitts A aus Segment S3. Sobald Segment S6 geladen wird, überschreibt der Text von Programmabschnitt A aus diesem Segment den 1. Teil des Bereichs A.

Bild 9 COMMON-Bereiche in segmentierten Programmen

Nachlademethoden

Das Laden und Nachladen von Segmenten (Lademodule) führt der Lader ELDE durch. Zunächst lädt er nur das Grundsegment in den Speicher und reserviert dabei so viel Speicherplatz, daß auch der längste Ast des Programms untergebracht werden kann. Der Verlauf beim Nachladen hängt weitgehend davon ab, welche Nachlademethode verwendet wird. Bereits bei der Programmierung muß der Benutzer sich für eine bestimmte Nachlademethode entscheiden. Je nach Nachlademethode baut der Binder TSOSLNK das Programm unterschiedlich auf. Die folgende Übersicht zeigt die verschiedenen Nachlademethoden.

Übersicht über die Nachlademethoden

	Direktes Nachladen	Automatisches Nachladen	
	LPOV-Makro	CALL-Makro V-Konstante	SEGLD-Makro
Nachladen der Segmente...	... erfolgt unabhängig davon, ob sie bereits im Speicher stehen.	... entfällt, wenn sie sich bereits im Speicher befinden.	... erfolgt unabhängig davon, ob sie bereits im Speicher stehen.
Anzahl der nachgeladenen Segmente:	ein einziges Segment, das im Makroaufruf explizit angegeben werden muß.	ein oder mehrere Segmente, und zwar das unmittelbar betroffene Segment und alle höheren Segmente desselben Astes.	
Speicherung des aktuellen Programmzustandes	———	In einer Tabelle (SEGTAB) speichert der vom Binder eingebundene Überlagerungssteuermodul den aktuellen Zustand des Programms im Speicher, d.h. die geladenen Segmente sind dort markiert.	
Fortsetzung des Programms nach dem Laden...	... mit dem Befehl, der auf den Makroaufruf folgt oder mit der im Makroaufruf angegebenen Adresse.	... mit der im Makroaufruf angegebenen Einsprungstelle.	... mit dem Befehl, der auf den Makroaufruf folgt oder mit der im Makroaufruf genannten Adresse.

	Direktes Nachladen	Automatisches Nachladen	
	LPOV-Makro	CALL-Makro V-Konstante	SEGLD-Makro
Für den Binder TSOSLNK erforderliche Angaben:	Beschreibung der Überlagerungsstruktur durch eine Folge von OVERLAY- und INCLUDE-Anweisungen. ———	CONTROL=Y in der PROGRAM-Anweisung (bei CONTROL=N werden V-Konstanten wie A-Konstanten behandelt, d.h. automatisches Nachladen ist unmöglich).	

Direktes Nachladen

Das direkte Nachladen löst der Makroaufruf LPOV (Load Program Overlay) aus. Der Lader ELDE wird dadurch aktiviert und lädt das im Makroaufruf angegebene Segment. Sobald dieses im Speicher steht, geht die Steuerung an das Programm zurück.

Vorsicht

Bei Kombinationen des direkten mit dem automatischen Nachladen im selben Programm ist zu beachten, daß durch den Makro LPOV die SEGTAB-Liste des Überlagerungssteuermoduls (s.u.) nicht aktualisiert wird und daher nicht mehr das tatsächliche Bild des Programms im Speicher wiedergibt.

Der Makroaufruf LPOV ist auf Seite 158f ausführlich beschrieben.

Automatisches Nachladen

Das automatische Nachladen von Segmenten kann ausgelöst werden:

- mit einer V-Konstante,
- durch den Makroaufruf CALL,
- durch den Makroaufruf SEGLD.

Da für CALL und SEGLD der Assembler im wesentlichen ebenfalls eine V-Konstante generiert, wird im folgenden meist nur die V-Konstante genannt. Das Gesagte gilt aber auch für diese beiden Makroaufrufe, die auf Seite 63ff ausführlich beschrieben sind.

Soll ein Programm seine Überlagerungssegmente automatisch nachladen, so wird es um zusätzliche Module erweitert, die den aktuellen Stand des Programms im Speicher kennen, aufgrund dessen entscheiden, welche Segmente zusätzlich geladen werden müssen und die den Lader ELDE dazu aufrufen. Der Binder TSOSLNK stellt diese Module zur Verfügung, paßt sie der Programmstruktur an und bindet sie in das Programm ein. Der Benutzer ruft diese Funktion des Binders mit der Angabe von CONTROL=Y (Standardwert) in der PROGRAM-Anweisung auf.

V-Konstanten sind Adreßkonstanten, die ein automatisches Nachladen von Überlagerungssegmenten auslösen können, d.h. sie werden eingesetzt, um die Steuerung zwischen Segmenten zu übergeben. Nicht bei allen V-Konstanten eines Programms muß ein Nachladen ausgelöst werden:

– V-Konstanten, die sich auf Einsprungstellen beziehen, die gleich oder höher in dem Ast des Segments mit der V-Konstante liegen, werden vom Binder wie Externverweise behandelt. Das ist sinnvoll, weil der Überlagerungssteuermodul dafür sorgt, daß die im Ast höher liegenden Segmente stets gleichzeitig mit dem Segment geladen sind, in dem die V-Konstante steht (siehe Bild 10).

– Liegt eine V-Konstante vor, die sich auf eine Einsprungstelle bezieht, die niedriger im Ast ist oder außerhalb des Astes von dem Segment mit der V-Konstante sich befindet, so prüft der Binder, ob mit dem Standardwert CONTROL=Y in der PROGRAM-Anweisung das automatische Nachladen von Segmenten verlangt wurde. Wenn dies der Fall ist, erweitert der Binder das Programm um die Überlagerungssteuermodule. Die V-Konstante wird dann nicht wie bei Externverweisen mit der Adresse der Einsprungstelle befriedigt, sondern mit einer Adresse, die im Überlagerungssteuermodul liegt. Während des Programmlaufs geht dann die Steuerung an den Überlagerungssteuermodul über. Dieser entscheidet, ob nachgeladen werden muß und welche Segmente davon betroffen sind.

Der nächste Abschnitt schildert den Aufbau des Steuermoduls und die sich daraus ergebenden Mechanismen beim Nachladen. Für den Benutzer sind diese Informationen in Ausnahmefällen nützlich, meist jedoch nicht erforderlich.

Beispiel für die Verwendung einer V-Konstanten

In einem Assemblerprogramm stehen die Befehle

```
L   15,=V(symbol)
BR  15
```

symbol ist eine Einsprungadresse in dem Segment, das geladen werden soll.

Bild 10 Verschiedene Auswirkungen von V-Konstanten

V-Konstante in Segment S5	Erläuterung
A B	Der Binder behandelt diese beiden V-Konstanten wie einen Externverweis, da %ROOT und S3 stets mit S5 im Speicher stehen.
C D	Ein Nachladen ist für diese beiden V-Konstanten stets erforderlich, da S5, S6 und S2 exklusive Segmente sind, d.h. nicht gleichzeitig im Speicher stehen. Für die Auflösung von C und D ist der XCALL-Operand notwendig.
E	Für diese V-Konstante kann ein Nachladen erforderlich sein. S8 kann aber auch bereits im Speicher stehen.

Aufbau der Überlagerungssteuermodule

Falls in einer PROGRAM-Anweisung der Standardwert CONTROL=Y gilt und der Binder auf bestimmte V-Konstanten trifft, baut der Binder für das Programm Überlagerungssteuermodule auf: An den Anfang des Grundsegments bindet er einen Steuermodul, eine Tabelle zur Beschreibung sämtlicher Programmsegmente (SEGTAB) und eine Tabelle der Einsprungstellen (ENTAB) für das Grundsegment ein. An den Anfang jedes Segments, das entsprechende V-Konstanten enthält, bringt der Binder eine Tabelle der Einsprungstellen (ENTAB) für dieses Segment (siehe Bild 11).

Der Binder behandelt die **V-Konstanten**, für die evtl. nachgeladen werden muß, nicht wie Externverweise, sondern vergibt für die 4 Byte lange Konstante folgende Werte:

1. Byte: Nummer des ENTAB-Eintrags, der zu diesen V-Konstanten gehört.
2. bis 4. Byte: Adresse des ENTAB-Moduls, der sich am Anfang des Segments befindet, das die V-Konstante enthält.

Der Inhalt der V-Konstanten bleibt während des Programmlaufs unverändert.

Hinweis

Der Aufbau der V-Konstanten erlaubt nicht mehr als 256 Einträge in der ENTAB. Ist die Anzahl überschritten, wird eine Fehlermeldung ausgegeben und der Binderlauf abgebrochen.

Die **Tabelle der Segmente (SEGTAB)** beschreibt mit Hilfe von Verkettungen ihrer Einträge den Aufbau der Überlagerungsstruktur und den aktuellen Stand des Programms im Speicher. Die in das Grundsegment eingebundene Tabelle enthält einen Eintrag von je 14 Byte für jedes Programmsegment (einschließlich Grundsegment).

Bild 11 Beispiel für den Aufbau einer Überlagerungsstruktur mit Steuermodulen

Bild 12 Tabelle zur Beschreibung der Segmente (SEGTAB)

Alle Felder der Segmenttabelle, mit Ausnahme von Zeiger C, bleiben während des Programmlaufs unverändert.

Zeiger A ist eine Distanzadresse, die auf einen SEGTAB-Eintrag für ein anderes Segment derselben Stufe weist. Gibt es auf derselben Stufe kein weiteres Segment, weist der Zeiger auf das eigene Segment. Alle SEGTAB-Einträge derselben Stufe sind also mit Hilfe von Zeiger A miteinander verkettet.

Zeiger B ist eine Distanzadresse, die auf den SEGTAB-Eintrag für das benachbarte, höhere Segment im gleichen Ast zeigt. Für die SEGTAB-Einträge des Grundsegments und der Überlagerungssegmente, die am 1. Knoten nach dem Grundsegment anfangen, weist Zeiger B auf den SEGTAB-Eintrag des Grundsegments. Alle SEGTAB-Einträge eines Astes sind also mit Hilfe von Zeiger B miteinander verkettet.

Zeiger C ist ein Feld, dessen Inhalt sich während des Programmablaufs ändert. Er wird vom Steuermodul aktualisiert, sobald das zugehörige Segment bzw. wenn Segmente niederer Stufe im gleichen Ast nachgeladen werden. Wird Zeiger C gelöscht, verschwindet für den Steuermodul das zugehörige Segment logisch aus dem Speicher.

Zeiger C	Segment-Zustand
Null	Segment ist nicht im Speicher. Diesen Wert trägt der Binder für den Programmstart ein.
Distanzadresse, die auf ein Segment niedrigerer Stufe im gleichen Ast weist	Segment ist im Speicher. Ein Segment niedrigerer Stufe des gleichen Astes befindet sich ebenfalls dort.
Distanzadresse des eigenen SEGTAB-Eintrags	Segment ist im Speicher. Kein Segment niedrigerer Stufe des gleichen Astes ist geladen.

Eine **Tabelle der Einsprungstellen (ENTAB)** erzeugt der Binder TSOSLNK für jedes Programmsegment, das mindestens eine V-Konstante verwendet, die sich nicht auf ein Segment höherer Stufe des gleichen Astes bezieht, für die also ein Nachladen erforderlich werden kann. Für jede dieser V-Konstanten erstellt der Binder einen Eintrag von je 4 Byte, doppelt vorhandene Konstanten erhalten nur einen Eintrag. Maximal dürfen 256 V-Konstanten auf andere Segmente verweisen. Außerdem enthält jede ENTAB einen Befehlsteil von 24 Byte Länge. Die Tabelle kann zusätzlich um 4 Byte erweitert werden, damit der nachfolgende Bindemodul auf Doppelwortgrenze anfangen kann.

Bild 13 Tabelle der Einsprungstellen (ENTAB)

Der Zeiger im ENTAB-Eintrag enthält die Nummer des SEGTAB-Eintrags für das Segment, das die von der V-Konstante bezeichnete Einsprungstelle enthält. Die nächsten 3 Byte bilden die programmrelative Adresse der Einsprungstelle.

Die gesamte ENTAB bleibt während des Programmlaufs unverändert.

Ablauf des automatischen Nachladens

Das Benutzerprogramm lädt den Inhalt der V-Konstanten nach Register 15 und verzweigt dorthin. Damit geht die Steuerung wegen der vom Binder in die V-Konstante eingetragenen Adresse an das erste Byte der ENTAB des Segments, das die V-Konstante enthält. Am Anfang der ENTAB stehen Befehle, die Register 9 mit der Adresse des Steuermoduls (Programmanfang) laden und anschließend dorthin verzweigen.

Der Steuermodul stellt Register 8 bis 14 sicher und verwendet den Inhalt von Register 15, d.h. den Inhalt der V-Konstanten, um den zugehörigen Eintrag in der ENTAB zu finden. Dieser Eintrag enthält den ENTAB-Zeiger, mit dessen Hilfe der Steuermodul den SEGTAB-Eintrag desjenigen Segments aufsucht, in dem die zur V-Konstanten gehörige Einsprung-
stelle liegt.

Als nächstes hat der Steuermodul zu entscheiden, ob das ermittelte Segment überhaupt nachgeladen werden muß (Zeiger C der SEGTAB). Ist dies der Fall, muß ermittelt werden, ob als Folge des Nachladens in der SEGTAB Segmente zu löschen sind (Zeiger A und C) und ob weitere Segmente in den Speicher gebracht werden müssen (Zeiger B). Dabei geht aus der Stellung eines Schalters hervor, ob der Makroaufruf CALL oder SEGLD (siehe Seite 63ff) die Bearbeitung verursachte.

Ein LPOV-Makroaufruf im Steuermodul veranlaßt schließlich den Lader ELDE, das jeweils ermittelte Segment in den Speicher zu laden. Der Steuermodul setzt danach für die neu geladenen Segmente den Zeiger C in der SEGTAB. Dann lädt er die Rücksprungadresse in Register 15, stellt den ursprünglichen Inhalt der Register 8 bis 14 wieder her und verzweigt in den Benutzerteil des Programms.

Wird eine V-Konstante vom Binder wie ein Externverweis behandelt, weil für sie kein Nachladen erforderlich ist, so werden die Überlagerungssteuermodule nicht benötigt, und die Steuerung geht direkt an die in der V-Konstante angegebene Einsprungadresse über.

Bild 14 Beispiel für automatisches Nachladen

Makroaufrufe zum Nachladen von Segmenten

LPOV Segment laden

Mit dem Makroaufruf LPOV legt man ein Segment fest, das vom Lader ELDE in den Speicher geladen werden soll. Die Ausführung erfolgt unabhängig davon, ob sich das Segment schon im Speicher befindet.

Operation	Operanden
LPOV	segment [,adresse]

segment
: Symbolischer Name des zu ladenden Segments, der aus höchstens 6 alphanumerischen Zeichen bestehen darf. Welche Bindemodule dieses Segment aufbauen, vereinbart der Benutzer mit OVERLAY- und INCLUDE-Anweisungen des Binders TSOSLNK.

adresse
: Maximal 8 Zeichen lange symbolische Adresse im aufrufenden Bindemodul oder ein Externverweis, d.h. eine Adresse in einem anderen Modul. Bei dieser Adresse wird das Programm nach dem Laden fortgesetzt. Der Benutzer muß dafür sorgen, daß der Bindemodul, in dem sich die angegebene Adresse befindet, nach der Ausführung des LPOV-Makros im Speicher steht.

  Läßt man diesen Operanden weg, so wird das Programm mit dem Befehl fortgesetzt, der auf den LPOV-Makroaufruf folgt.

Der Makroaufruf LPOV ist ausführlich auf Seite 158ff beschrieben.

CALL Segmente laden

Mit dem Makroaufruf CALL legt man eine Einsprungstelle fest und löst damit das Nachladen des zugehörigen Segments aus, falls sich dieses noch nicht im Speicher befindet. Sämtliche Segmente, die im selben Ast zwischen Grundsegment und dem betreffenden Segment liegen, werden ebenfalls geladen, wenn sie nicht bereits im Speicher stehen.

Operation	Operanden
CALL	einsprungstelle

einsprungstelle Symbolische Adresse einer Einsprungstelle in einem Segment. Für sie wird eine 4 Byte lange V-Konstante erzeugt. Mit diesem Operanden wird

- implizit das Segment angegeben, das nachgeladen werden soll, und gleichzeitig
- die Adresse vereinbart, bei der nach der Ausführung des Makros das Programm weiterläuft.

Funktionsweise

Der Assembler erzeugt aus der symbolischen Adresse (Einsprungstelle) des Makroaufrufs CALL eine V-Konstante. Aufgrund dieser V-Konstanten und wegen CONTROL=Y (Standardwert) in der PROGRAM-Anweisung des Binders TSOSLNK erzeugt dieser Überlagerungssteuermodule, die er in das Programm einbindet. Die Steuermodule führen zur Ablaufzeit des Programms das Nachladen aus: Bei der Ausführung von CALL werden sie aktiviert, ermitteln das zur Einsprungstelle gehörende Segment und prüfen, ob es bereits im Speicher steht. Wenn das der Fall ist, erhält sofort der Modul die Steuerung, der die im Makro genannte Einsprungstelle enthält. Ein Segment wird also nur geladen, wenn es nötig ist. Stellt der Überlagerungssteuermodul fest, daß das Segment noch geladen werden muß, so wird es mit allen höheren Segmenten des gleichen Astes geladen, sofern sich diese nicht bereits im Speicher befinden.

Hinweis

Beim Entwurf der Überlagerungsstruktur eines Programms, in dem man Segmente automatisch nachladen läßt, sollte man den zusätzlichen Speicherbedarf für die Überlagerungssteuermodule (mit den Tabellen SEGTAB und ENTAB) berücksichtigen (siehe Seite 57ff).

Vorsicht

Der Makroaufruf CALL muß in einem Programmbereich liegen, der durch eine USING-Anweisung überdeckt wird.

Register 15 wird vom Makro CALL zum Laden der Segmente verwendet. Es nimmt die Einsprungadresse in den Überlagerungssteuermodul auf. Register 15 sollte daher nicht als Basisregister für den Programmteil verwendet werden, in dem der Makroaufruf CALL liegt.

Beispiel für die Makroauflösung von CALL

```
         CALL  EINXEINS
*                      VER014 REV00 80-03-14 80420301 BAER SP331
         L     15,ITUA0001  VERSION 001  LOAD VCON INTO REG 15
         BR    15            BRANCH ON REGISTER 15
ITUA0001 DC V(EINXEINS)
```

Dieser Makroaufruf CALL legt als Einsprungstelle die symbolische Adresse EINXEINS fest. Bei Programmablauf wird dann Register 15 mit dem Inhalt der V-Konstanten V(EINXEINS) geladen und nach Register 15 verzweigt.

Der Binder TSOSLNK sorgt beim Binden dafür, daß die V-Konstante eine Adresse im Überlagerungssteuermodul enthält, falls in der Binderanweisung folgendes angegeben wurde:

```
PROGRAM... ,CONTROL=Y
```

SEGLD Segmente laden

Mit dem Makroaufruf SEGLD legt man eine Einsprungstelle fest und löst damit das Nachladen des zugehörigen Segments aus, auch wenn sich dieses bereits im Speicher befindet. Sämtliche Segmente, die im Ast zwischen Grundsegment und dem nachzuladenden Segment liegen, werden ebenfalls geladen.

Operation	Operanden
SEGLD	einsprungstelle [,adresse]

einsprungstelle — Symbolische Adresse einer Einsprungstelle in dem zu ladenden Segment. Für sie wird eine 4 Byte lange V-Konstante erzeugt.

adresse — Symbolische Adresse, bei der das Programm nach Bearbeitung des SEGLD-Makros fortgesetzt werden soll. Sie kann im aufrufenden oder einem anderen Bindemodul liegen. Im 2. Fall muß der Benutzer dafür sorgen, daß der Modul, der die "adresse" enthält, nach dem Laden durch den SEGLD-Makro im Speicher steht.

Fehlt dieser Operand, so läuft das Programm mit dem auf SEGLD folgenden Befehl weiter.

Funktionsweise

Der Assembler erzeugt aus der angegebenen symbolischen Adresse der Einsprungstelle eine V-Konstante. Aufgrund dieser V-Konstanten und von CONTROL=Y (Standardwert) in der PROGRAM-Anweisung des Binders TSOSLNK erzeugt dieser Überlagerungssteuermodule, die er in das Programm einbindet. Die Steuermodule führen zur Ablaufzeit des Programms das Nachladen aus: Bei der Ausführung von SEGLD werden sie aktiviert und laden das Segment, das die im Makroaufruf angegebene Einsprungstelle enthält, sowie alle Segmente, die zwischen dem betreffenden Segment und dem Grundsegment innerhalb des gleichen Astes liegen. Dabei werden die Segmente also in jedem Fall neu geladen, auch wenn sie bereits im Speicher stehen.

Anschließend geht die Steuerung an den Befehl, der dem Makroaufruf folgt bzw. der im Makroaufruf genannt ist.

Hinweis

Beim Entwurf der Überlagerungsstruktur eines Programms, in dem man Segmente automatisch nachladen läßt, sollte man den zusätzlichen Speicherbedarf für die Überlagerungssteuermodule (mit den Tabellen SEGTAB und ENTAB) berücksichtigen (siehe Seite 57ff).

Vorsicht

- Der Makroaufruf SEGLD muß in einem Programmbereich liegen, der durch eine USING-Anweisung überdeckt wird.
- Register 15 wird vom Makro SEGLD zum Laden der Segmente verwendet. Es nimmt die Einsprungadresse in den Überlagerungssteuermodul auf. Register 15 darf daher nicht als Basisregister für den Programmteil verwendet werden, in dem der Makroaufruf SEGLD liegt.
- Der Makro SEGLD darf nur in Programmen verwendet werden, die unterhalb 16Mbyte geladen werden.
- Ein Modul, der einen Makro SEGLD enthält, darf nicht in einen Bindelademodul (LLM) eingefügt werden (siehe Handbuch "BLS" [1]).

Beispiel für die Makroauflösung von SEGLD

```
         SEGLD EINXEINS,MELD2
*                     VER014 REV00 80-03-14 80420302 BAER SP331
         CLI   ITUB0001,X'00' VERSION 001.    VCON CAUSE ENTAB ITEM
         BH    ITUA0001     NO-IN LINE, YES-BRANCH
         L     15,ITUC0001  PUT RETURN ADDR. IN REG. 15
         BR    15              RETURN TO USER
ITUA0001 L  15,ITUB0001  LOAD VCON INTO REG. 15
         MVI   20(15),X'01'  SET SEGLD FLAG IN ENTAB
         MVC   12(4,15),ITUC0001  SAVE RETURN ADDR. IN ENTAB
         BR    15           BRANCH TO ENTAB
ITUB0001 DC V(EINXEINS)               VCON FOR LINKAGE
ITUC0001 DC A(MELD2)              RETURN ADDRESS
```

Der Makroaufruf SEGLD legt als Einsprungstelle die symbolische Adresse EINXEINS fest und als Rücksprungadresse für die Makrobearbeitung MELD2. Der MVI markiert den Aufruf als SEGLD, der anders als der CALL-Makroaufruf bearbeitet werden muß.

Anweisungen an TSOSLNK

Nach dem Programmaufruf erwartet TSOSLNK die Eingabe von Anweisungen aus der Systemdatei SYSDTA, also entweder direkt von der Datenstation oder aus einer katalogisierten SAM- oder ISAM-Datei bzw. aus einem Bibliothekselement.

Mit Ausnahme von LINK-SYMBOLS (266 Zeichen) kann eine Anweisung maximal 220 Zeichen lang sein. Fortsetzungszeichen dürfen nicht eingegeben werden.

TSOSLNK überprüft bereits während der Eingabe alle Anweisungen auf formale Fehler. Beim Auftreten eines Formalfehlers weist TSOSLNK die gesamte Anweisung zurück, gibt die fehlerhafte Eingabezeile nach SYSOUT aus und markiert den Fehler mit dem Zeichen "^".

Syntaxbeschreibung

Bei der Darstellung von Formaten werden bestimmte Zeichen (sogenannte Metazeichen) verwendet und Vereinbarungen getroffen, die in der folgenden Tabelle erläutert sind:

Formale Darstellung	Erläuterung	Beispiel
GROSSBUCHSTABEN	Großbuchstaben bezeichnen Konstanten, die in dieser Form vom Benutzer eingegeben werden müssen.	CMAP=CSECTS Einzugeben ist: CMAP=CSECTS
Kleinbuchstaben	Kleinbuchstaben bezeichnen Variablen, die bei der Eingabe vom Benutzer durch aktuelle Werte ersetzt werden müssen, d.h. ihr Inhalt kann von Fall zu Fall verschieden sein.	LIBRARY=bibliothek Einzugeben ist: LIBRARY=LIB1 LIBRARY=X usw.
{ }	Geschweifte Klammern schliessen Alternativen ein, d.h. aus den eingeschlossenen Größen muß eine Angabe ausgewählt werden.	NA-COL={STD / ABORT} Einzugeben ist: NA-COL=STD oder NA-COL=ABORT
|	Ein senkrechter Strich trennt Alternativen, von denen eine ausgewählt werden muß.	PAM-KEY={Y|N} Einzugeben ist: PAM-KEY=Y oder PAM-KEY=N
——	Die Unterstreichung hebt den Standardwert (Voreinstellung) hervor. Das ist der Wert, den das System einsetzt, wenn der Benutzer keine Angabe macht.	NA-COL={STD / ABORT}
...	Punkte bedeuten eine Wiederholung. Sie zeigen an, daß die davor stehende Einheit mehrmals hintereinander wiederholt werden kann.	(modul,...) Einzugeben ist: (MODUL1) oder (A,B,C) oder (A1,A2,A3,A4) usw.

Formale Darstellung	Erläuterung	Beispiel
[ ]	Eckige Klammern schließen Wahlangaben ein, d.h. Angaben, die man weglassen darf. Steht bei Wahlangaben das Komma innerhalb der Klammer, so wird es nur bei Verwendung dieser Wahlangabe verlangt und kann beim ersten Operanden weggelassen werden. Steht es außerhalb der Klammer, so muß es stets geschrieben werden, auch wenn keine Wahlangabe gemacht wird. (Runde Klammern müssen eingegeben werden.)	CMAP=CS[ECTS] Einzugeben ist: CMAP=CSECTS oder abgekürzt CMAP=CS

Übersicht über die Anweisungen an TSOSLNK

In der folgenden Aufstellung sind die Anweisungen zu Funktionsgruppen zusammengefaßt:

Einlesen von Bindemodulen, die gebunden werden sollen

Die INCLUDE-Anweisung bestimmt die Bindemodule, die TSOSLNK binden soll.

Diese Bindemodule können stehen in

1. Der temporären EAM-Bindemoduldatei und
2. einer Bindemodulbibliothek.

Mit der ALTLIB-Anweisung kann man eine Bibliothek zuweisen, die sich auf eine nachfolgende INCLUDE-Anweisung bezieht.

Operation	Operanden	Beschreibung
ALTLIB	{modul,bibliothek (modul,...),bibliothek ,bibliothek ,*NO}	weist eine Bibliothek zu, wenn der Modul nicht aus der mit der INCLUDE-Anweisung festgelegten Bibliothek gelesen werden soll
INCLUDE	{modul[(version)][,bibliothek] (modul[(version)],...)[,bibliothek] ,bibliothek modul,* (modul,...),* *}	veranlaßt das Einlesen von zu bindenden Bindemodulen

Binden eines Lademoduls (ablauffähigen Programms)

Die PROGRAM-Anweisung legt fest, daß die Bindemodule zu einem Programm gebunden werden sollen, das in einer katalogisierten Datei gespeichert wird.

Die Lademodule des Programms werden vom statischen Lader ELDE in den Speicher geladen.

Operation	Operanden	Beschreibung
PROG[RAM]	programm [,VERSION=version] [,COPYRIGHT=(name,jahr)]	legt Namen und Eigenschaften des gebundenen Programms fest
	[,{FILENAM=datei[,SHARE={Y|N}] LIB[RARY]=bibliothek[,ELEMENT=element[(version)]]}]	
	[,P[AM]-K[EY]={Y|N}]	legt das Format der PAM-Datei für das Programm fest
	[,ARM[ODE]-C[HECK]={IGN[ORE]|WARN[ING]|ABORT}]	prüft die CSECTs auf Attribute AMODE und/oder RMODE ungleich ANY
	[,{IDA={Y|N} SYMTEST={ALL|N|MAP}}] [,XS-C[HECK]={Y|N}]	
	[,CLASS={2|E}] [,COREIM={Y|N}] [,CONTROL={Y|N}] [,XCAL={Y|N}] [,LOADPT={adresse|*XS}] [,MAX=proglänge] [,PL1={Y|N}] [,LET={Y|N} [,ADD=länge]	steuert indirekt den Ladevorgang
	[,{ENTRY|START}={programmabschnitt|einsprungstelle}]	

Operation	Operanden	Beschreibung
PROG[RAM] (Forts.)	[,PR={Y|N}]	regelt die Ausgabe von Listen und Protokollen.
	[,XDSEC={Y|N}] [,MAP={Y|N}]	
	[,XREF={Y|N}] [,LIST={Y|N}]	
	[,SYSLST={Y|N}] [,UNSAT={Y|N|S}]	
	[,WUNSAT={Y|N}] [,SORT={Y|N}]	
	[,CMAP={ {ALL | NO} | ([{CS[ECTS] | NOCS[ECTS]}] [,{EN[TRYS] | NOEN[TRYS]}] [,{COM[MONS] | NOCOM[MONS]}]) | ([,{X[REF] | NO[XREF]}] [,{EJ[ECT] | NOEJ[ECT]}] [,MOD[ULES]]) }]	
	[,LINE=number]	legt die Anzahl der Zeilen pro Seite bei Ausgabe auf SYSLST fest
	[,LINEAR={Y|N}]	verhindert die Bildung einer OVERLAY-Struktur (unvereinbar mit der MODULE-Anweisung)

Eigenschaften des Programms festlegen

Beim Binden können Eigenschaften des Programms fest vereinbart werden.

1. Speicherklasse

   Das Programm soll zu einem Systemprogramm (Teil des Organisationsprogramms) gebunden werden.
   Dies kann festgelegt werden:

   1. in der CLASS-Anweisung oder
   2. im Operanden CLASS der PROGRAM-Anweisung.

Operation	Operanden	Beschreibung
CLASS	2|E	bestimmt die Programmklasse des Lademoduls; wirkt wie CLASS=... in der PROGRAM-Anweisung (unvereinbar mit der MODULE-Anweisung)

2. Startpunkt für den Ablauf des Programms festlegen

   Bereits beim Binden kann festgelegt werden, bei welchem Programmabschnitt oder bei welcher Einsprungstelle das Programm nach dem Laden starten soll.
   Dies kann festgelegt werden:

   1. in der ENTRY-Anweisung oder
   2. im Operanden ENTRY oder START der PROGRAM-Anweisung.

Operation	Operanden	Beschreibung
ENTRY	{ programmabschnitt / einsprungstelle }	vereinbart den Startpunkt des Programmlaufs (unvereinbar mit der MODULE-Anweisung)

3. Angaben zur Mehrbenutzbarkeit

   Hier kann festgelegt werden, ob ein Programm nur unter der Eigentümerkennung ablaufen darf oder ob auch Benutzer fremder Kennungen darauf zugreifen dürfen. Die Festlegung geschieht:

   1. in der SHARE-Anweisung oder
   2. im Operanden SHARE der PROGRAM-Anweisung.

Operation	Operanden	Beschreibung
SHARE		markiert neu erstellte Programmdateien als mehrfach benutzbar; wirkt wie SHARE=Y in der PROGRAM-Anweisung (unvereinbar mit der MODULE-Anweisung)

Binden zu einem einzigen Bindemodul

Die MODULE-Anweisung bindet mehrere Bindemodule zu einem einzigen Bindemodul und hinterlegt diesen in der temporären EAM-Bindemoduldatei oder in einer Programmbibliothek.

Dieser Bindemodul kann vom Binder TSOSLNK und vom dynamischen Bindelader mit weiteren Bindemodulen gebunden und geladen werden.

Mit der LINK-SYMBOLS-Anweisung wird bestimmt, wie die CSECTs und ENTRYs beim Binden im ESD abgelegt werden.

Operation	Operanden	Beschreibung
MOD[ULE]	[modul1] [,ENDC=modul2] [,LIB[RARY]=bibliothek] [,ELEM[ENT]=element[(version)]]	legt Namen und Eigenschaften des erzeugten Bindemoduls fest
	[,ARM[ODE]-C[HECK]={IGN[ORE] | WARN[ING] | ABORT}]	prüft die CSECT's auf Attribute AMODE und/oder RMODE ungleich ANY
	[,LET={Y|N}] [,PL1={Y|N}] [,MAP={Y|N}] [,XREF={Y|N}]	regelt die Ausgabe von Listen und Protokollen
	[,LIST={Y|N}] [,PR={Y|N}] [,XDSEC={Y|N}] [,SYSLST={Y|N}] [,UNSAT={Y|N|S}] [,WUNSAT={Y|N}] [,SORT={Y|N}]	
	[,CMAP={ {ALL | NO} | ([{CS[ECTS] | NOCS[ECTS]}] [,{EN[TRYS] | NOEN[TRYS]}] [,{COM[MONS] | NOCOM[MONS]}]) [,{X[REF] | NO[XREF]}] [,{EJ[ECT] | NOEJ[ECT]}] [,MOD[ULES]]) }]	

Operation	Operanden	Beschreibung
MOD[ULE] (Forts.)	[,LINE=number]	legt die Anzahl der Zeilen pro Seite bei Ausgabe auf SYSLST fest
	[,NA-COL={STANDARD|STD / IGN[ORE] / ABORT}]	legt die Behandlung von Namenskonflikten fest.
{LINK-SYMBOLS / LI-SYM}	[{*HIDE / {KEEP / HIDE}={symbol / (symbol,...)} / *KEEP / *NOESD}]	definiert, ob und wie CSECTs und ENTRYs beim Vorbinden im ESD abgelegt werden

Auflösen von Externverweisen

TSOSLNK versucht beim Binden, zuerst alle Externverweise den Einsprungstellen zuzuordnen, die in den Eingabe-Bindemodulen enthalten sind (0. Stufe).

In der 1. und 2. Stufe sucht TSOSLNK nach Externverweisen, deren Auflösung direkt in einer Anweisung angegeben wurde.
Solche Anweisungen sind:

1. die RESOLVE-Anweisung. Sie gibt eine Bindemodulbibliothek an, die nach passenden Einsprungstellen durchsucht werden soll.
2. die ERREXIT-Anweisung. Sie ordnet allen Externverweisen, die nicht aufgelöst werden können, einen festen Wert zu.

Einschränkende Anweisungen

– Die EXCLUDE-Anweisung nennt die Externverweise, die von TSOSLNK nicht aufgelöst werden sollen.
– Die NCAL-Anweisung bestimmt, daß die Bindemodulbibiothek TASKLIB nicht nach passenden Einsprungstellen durchsucht werden soll.

Operation	Operanden	Beschreibung
RESOLVE	[{externverweis / (externverweis,...)}],bibliothek	gibt Quellen zur Befriedigung von Externverweisen an
EXCLUDE	[{externverweis / (externverweis,...)}],bibliothek	legt Externverweise fest, die der Binder nicht befriedigen soll
NCAL		schließt die Bindemodulbibliothek TASKLIB für die Befriedigung von Externverweisen aus; wirkt wie EXCLUDE_,TASKLIB
ERREXIT	{A={nnnnnn / X'xxxxxxxx'} / E={programmabschnitt / einsprungstelle}}	weist allen Externverweisen, die der Binder nicht befriedigen kann einen Wert zu.

Die Anweisungen RESOLVE, EXCLUDE und NCAL beziehen sich auf die Autolink-Funktion.

Bei segmentierten Programmen kann man die Befriedigung spezieller Externverweise mit dem Operanden XCAL=Y in der PROGRAM-Anweisung bzw. mit der XCAL-Anweisung steuern (s.u.).

Änderungen in den Bindemodulen beim Binden

Noch beim Binden können Bindemodule geändert werden:

1. Namensänderungen

   Die RENAME-Anweisungersetzt den Namen eines Programmabschnitts, einer Einsprungstelle oder eines Externverweises durch einen anderen Namen.

2. Die REP-Anweisung verändert bestimmte Bereiche in einem Bindemodul.

Operation	Operanden	Beschreibung
RENAME	{programmabschnitt / einsprungstelle / externverweis},name	ändert Namen in Bindemodulen
REP	{adress_X'sedezimal-daten'_modul / adress_[C]'zeichen'_modul}[_kommentar]	ändert Daten in eingelesenen Bindemodulen

Behandlung von Programmabschnitten

Für Programmabschnitte in den Programmen oder dem gebundenen Bindemodul können, wenn mit LINK-SYMBOLS *HIDE oder *KEEP gearbeitet wird, beim Binden bestimmte Merkmale vereinbart werden:

1. Ausrichten auf Seitengrenze

   Die Ausrichtung eines Programmabschnitts auf Seitengrenze übernehmen die PAGE-Anweisung oder der Operand PAGE in der TRAITS-Anweisung.

2. Zugriffsschutz

   Ob ein Programmabschnitt zum Zeitpunkt des Ablaufs nur gelesen werden darf oder ob darin auch geschrieben werden darf, legt ebenfalls die TRAITS-Anweisung fest.

Operation	Operanden	Beschreibung
TRAITS	[programmabschnitt] [,ALIGN=nummer] [,READONLY={Y|N}] [,PAGE={Y|N}] [,AMODE={24|31|ANY}] [,RMODE={24|ANY}]	verändert die Attribute eines Programmabschnitts
PAGE	programmabschnitt	richtet einen Programmabschnitt auf Seitengrenze aus; wirkt wie PAGE=Y in der TRAITS-Anweisung

Aufbau von Überlagerungsstrukturen

Bei segmentierten Programmen wird der Aufbau von Überlagerungsstrukturen mit der OVERLAY-Anweisung gesteuert. Sie definiert die Überlagerungsstruktur eines segmentierten Programmes.

Die NOCTL-Anweisung verhindert bei segmentierten Programmen, daß ein Überlagerungssteuermodul und die zugehörigen Tabellen erzeugt werden. Die gleiche Funktion hat der Operand CONTROL=NO in der PROGRAM-Anweisung.

Die XCAL-Anweisung bewirkt, daß Externverweise in segmentierten Programmen zwischen voneinander unabhängigen Segmenten befriedigt werden dürfen. Der Operand XCAL=YES in der PROGRAM-Anweisung hat die gleiche Wirkung.

Der Operand LINEAR=Y in der PROGRAM-Anweisung verhindert die Bildung einer OVERLAY-Struktur. Die OVERLAY-Anweisung wird ignoriert.

Operation	Operanden	Beschreibung
OVERLAY	knotenpunkt [,REGION][,segment] [,LOADPT={adresse|*XS}]	definiert die Überlagerungsstruktur eines segmentierten Programms(unvereinbar mit der MODULE-Anweisung)
NOCTL		verhindert bei segmentierten Programmen die Erzeugung eines Überlagerungssteuermoduls und der zugehörigen Tabellen; wirkt wie CONTROL=N in der PROGRAM-Anweisung (unvereinbar mit der MODULE-Anweisung)
XCAL		erlaubt bei segmentierten Programmen die Befriedigung von Externverweisen zwischen unabhängigen Segmenten;wirkt wie XCAL=Y in der PROGRAM-Anweisung (unvereinbar mit der MODULE--Anweisung)

Steuerung des Binder-Laufs

Angaben darüber, wie der Ablauf des Bindens fortgesetzt werden soll, wenn Externverweise nicht befriedigt werden können, machen folgende Anweisungen:

BIND- oder CONTINUE-Anweisung
Sie veranlassen, daß das Binden fortgesetzt werden soll, auch wenn nicht alle Externverweise befriedigt werden können.

LET-Anweisung
Im Dialogbetrieb fragt das Programm an, wie der Programmlauf des Binders fortgesetzt werden soll. Dann können z.B. noch Externverweise im Dialog aufgelöst werden.

Im Stapelbetrieb wird das Binden auch dann fortgesetzt, wenn nicht alle Externverweise aufgelöst werden können.

Operation	Operanden	Beschreibung
BIND		veranlaßt das Binden, auch wenn der Binder nicht alle Externverweise befriedigen kann; wirkt wie die CONTINUE-Anweisung
CONTINUE		veranlaßt das Binden, auch wenn der Binder nicht alle Externverweise befriedigen kann; wirkt wie die BIND-Anweisung
LET		verhindert einen Abbruch des Bindens, auch wenn der Binder nicht alle Externverweise befriedigen kann; wirkt wie LET=Y in der PROGRAM- bzw. MODULE-Anweisung

Beenden der Eingabe von Anweisungen

Die END-Anweisung schließt die Eingabe von Anweisungen an TSOSLNK ab. Danach beginnt der Bindelauf.

Operation	Operanden	Beschreibung
END		beendet die Eingabe in den Binder und löst das Binden aus

Die STOP-Anweisung beendet den Programmlauf sofort.

Operation	Operanden	Beschreibung
STOP		beendet den Binderlauf sofort

Einfügen einer Kommentarzeile

Mit der COMMENT-Anweisung können Kommentarzeilen in den TSOSLNK-Lauf eingefügt werden.

Operation	Operanden	Beschreibung
COMMENT	[beliebiger Kommentar]	hat keinen Einfluß auf den TSOSLNK-Lauf; dient nur der Information des Benutzers

Steuern der Listenausgabe

Mit der NOMAP- und XREF-Anweisung kann die Listenausgabe des Binders gesteuert werden. Die Anweisungen wirken wie die Operanden MAP=N oder XREF=Y in der PROGRAM- oder MODULE-Anweisung.

Operation	Operanden	Beschreibung
NOMAP		Es wird keine Programmübersicht erstellt
XREF		Es wird eine Querverweisliste erstellt

Beschreibung der Anweisungen

ALTLIB-Anweisung

Sie weist eine Bindemodulbibliothek zu. In nachfolgenden INCLUDE-Anweisungen angegebene Bibliotheken werden nicht durchsucht, wenn sich der Modul in der ALTLIB befindet.

Operation	Operanden
ALTLIB	{ modul,bibliothek (modul,...),bibliothek ,bibliothek ,*NO }

modul — definiert die Bindemodule, die von der zugewiesenen Bindemodulbibliothek gelesen werden sollen.
Werden mehrere Module angegeben (maximal 20), müssen sie in Klammern gesetzt werden. Wird kein Modul angegeben, so werden alle Module zunächst in der angegebenen Bibliothek gesucht.

bibliothek — Name der Bindemodulbibliothek, aus der der Binder Module entnehmen soll.
Diese Angabe ist obligatorisch.

*NO — hat keine Wirkung auf den TSOSLNK-Lauf. Durch *NO wird eine Schein-ALTLIB-Anweisung generiert, die UGEN benutzt.

Hinweis

- Die Bindemodulbibliothek, die mit der ALTLIB-Anweisung zugewiesen wird, wird nur von den INCLUDE-Anweisungen benutzt, die nach der ALTLIB-Anweisung eingegeben werden.
- Wurde für einen Bindemodul durch eine ALTLIB-Anweisung explizit oder implizit festgelegt, daß er in einer bestimmten Bibliothek gesucht werden soll, so werden Bibliotheksangaben in der INCLUDE-Anweisung für diesen Modul ignoriert.
- Wird ein Modul in mehreren ALTLIB-Anweisungen angegeben, so wird er von der letzten mit der ALTLIB-Anweisung zugewiesenen Bibliothek gelesen, die ihn enthält.
- Werden Module von der temporären EAM-Bindemoduldatei gelesen, so wird die Bindemodulbibliothek, die mit der ALTLIB-Anweisung zugewiesen wurde, nicht berücksichtigt.

- Wurde ein Modul aus einer mit ALTLIB zugewiesenen Bindemodulbibliothek gebunden, so wird als Bindemethode "ALTERNATE" ausgegeben.
- Ist die mit der ALTLIB-Anweisung zugewiesene Bibliothek eine *Programmbibliothek*, gilt folgendes:
  - Nur der Modul wird ausgewählt, der zu dem angegebenen Modulnamen die höchste Versionsbezeichnung besitzt. Alle anderen Versionen des Moduls werden übergangen.
  - Wird in der INCLUDE-Anweisung eine Versionsbezeichnung für den Modul angegeben, muß die angegebene Version die höchste Versionsbezeichnung in der Bibliothek haben. Bei allen anderen Versionsangaben wird die mit der ALTLIB-Anweisung zugewiesene Bibliothek übergangen.

Beispiel

In den folgenden Beispielen ist vorausgesetzt, daß in der Bindemodulbibliothek LIB1 die Bindemodule MOD1, MOD2 und MOD3 gespeichert sind.

```
1. ALTLIB  MOD1,LIB1
   INLCUDE MOD1
```

Der Modul MOD1 wird aus der Bindemodulbibliothek LIB1 eingelesen.

```
2. ALTLIB (MOD1,MOD2),LIB1
   INCLUDE (MOD1,MOD2)
```

Die Module MOD1 und MOD2 werden aus der Bindemodulbibliothek LIB1 eingelesen.

```
3. ALTLIB,LIB1
   INCLUDE (MOD1,MOD2)
   INCLUDE MOD3
```

Die Module MOD1, MOD2 und MOD3 werden aus der Bindemodulbibliothek LIB1 eingelesen.

```
4. ALTLIB MOD1,LIB1
   ALTLIB ,LIB2
   INCLUDE MOD1,LIB3
```

Zuerst wird geprüft, ob der Modul MOD1 in der mit der letzten ALTLIB-Anweisung zugewiesenen Bindemodulbibliothek LIB2 gespeichert ist. Ist er in LIB2 enthalten, wird er von dort eingelesen. Ist MOD1 nicht in LIB2 enthalten, wird er aus der mit der vorangehenden ALTLIB-Anweisung zugewiesenen Bindemodulbibliothek LIB1 eingelesen. Die Bibliotheksangabe LIB3 in der INCLUDE-Anweisung wird übergangen. Sind LIB1 und LIB2 *Programmbibliotheken*, wird der Modul MOD1 mit der höchsten Versionsbezeichnung eingelesen.

```
5. ALTLIB MOD1,LIB1
   INCLUDE MOD1(007)
```

Der Modul MOD1 wird nur dann aus der Bindemodulbibliothek LIB1 eingelesen, wenn er dort mit der *höchsten* Versionsbezeichnung "007" eingetragen ist. Ist dies nicht der Fall, wird die mit der ALTLIB-Anweisung zugewiesene Bindemodulbibliothek LIB1 übergangen.

BIND-Anweisung

Sie löst sofortiges Binden aus. Im Gegensatz zur END-Anweisung verzichtet hier der Benutzer auf die Möglichkeit, weitere Anweisungen eingeben zu können.

Der TSOSLNK verarbeitet dann die eingelesenen Bindemodule, auch wenn er nicht alle Externverweise befriedigen kann.

Die BIND-Anweisung hat die gleiche Wirkung wie die CONTINUE-Anweisung.

Operation	Operanden
BIND	[beliebiger Kommentar]

Beispiel

```
/START-PROGRAM FROM-FILE=$TSOSLNK
%  BLS0500 PROGRAM 'TSOSLNK', VERSION '21.0D00' OF '90-05-10' LOADED.
*PROGRAM BIND,FILENAM=BIND.LADE,IDA=YES
*INCLUDE EINXEINS,*
*END
 UNRESOLVED EXTRNS: ------------------------------------------------------ (01)
 ITC0DSA0  ITC0ACA0  ITCCTCU1  ITC0END0  ITC0POVH
%  LNK0026 END STMT READ FROM SYSDTA. DO YOU WANT TO REDIRECT SYSDTA ?
%  REPLY (Y(BKPT);N (CONTINUE))?
Y ------------------------------------------------------------------------ (02)
*BIND ------------------------------------------------------------------- (03)
 UNRESOLVED EXTRNS:
 ITC0DSA0  ITC0ACA0  ITCCTCU1  ITC0END0  ITC0POVH
%  LNK0055 PROG BOUND INSPITE OF UNRESOLVED EXTERN'S
%  LNK0503 PROG FILE WRITTEN: BIND.LADE
%  LNK0504 3 PAM PAGES USED. TSOSLNK RUN FINISHED
```

(01) Der Binder meldet unlösbare Externverweise namens ITC0DSA0, ITC0ACA0, usw.

(02) Der Benutzer entscheidet, ob er weitere Binderanweisungen eingeben oder den Binderlauf beenden soll.
Mit der Antwort Y werden weitere Binderanweisungen angefordert.
Mit der Antwort N würde der Binderlauf ohne Binden des Programms mit der Meldung LNK0054 beendet werden.

(03) Mit der BIND-Anweisung wird erreicht, daß das Programm trotz unbefriedigter Externverweise gebunden wird (Meldung LNK0055).

CLASS-Anweisung

Sie bestimmt die Programmklasse des Programms. Sie wirkt wie der Operand CLASS= in der PROGRAM-Anweisung. Mit dem Modulbinden (siehe MODULE-Anweisung, Seite 98ff) ist sie unvereinbar und wird vom Binder mit einer Fehlermeldung zurückgewiesen.

Operation	Operanden
CLASS	2|E

COMMENT-Anweisung

Sie hat keinen Einfluß auf den TSOSLNK-Lauf. Sie dient nur der Information des Benutzers.

Operation	Operanden
COMMENT	[beliebiger Kommentar]

CONTINUE-Anweisung

Sie veranlaßt das Binden, auch wenn der Binder nicht alle Externverweise befriedigen kann. Sie wirkt wie die BIND-Anweisung.

Operation	Operanden
CONTINUE	[beliebiger Kommentar]

END-Anweisung

Sie beendet die Eingabe für den Binder. Entdeckt er im Dialogbetrieb anschließend jedoch unbefriedigte Externverweise oder ist keine PROG-Anweisung angegeben, so nimmt er den Dialog wieder auf und fordert weitere Steueranweisungen an. Im Stapelbetrieb hängt es vom Operanden LET in der PROGRAM- bzw. MODULE-Anweisung ab, ob trotz unbefriedigter Externverweise gebunden oder der Binderlauf abgebrochen wird.

Sonderfall

SYSDTA weist auf eine katalogisierte Datei. Erkennt der Binder das Dateiende, so wirkt das wie die END-Anweisung, die in diesem Fall also überflüssig ist. Sobald er unbefriedigte Externverweise findet, fragt der Binder im Dialog, ob man die Zuweisung von SYSDTA ändern möchte. Dann kann man ggf. SYSDTA mit dem SYSFILE-Kommando auf seine Datenstation zurücklegen und von dort weitere Binderanweisungen eingeben.

Operation	Operanden
END	[beliebiger Kommentar]

ENTRY-Anweisung

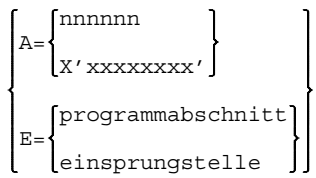
Sie legt den Startpunkt des Programmlaufs fest. Sie wirkt wie der Operand START= bzw. ENTRY= in der PROGRAM-Anweisung. Mit dem Modulbinden (siehe MODULE-Anweisung, Seite 98ff) ist sie unvereinbar und wird vom Binder mit einer Fehlermeldung zurückgewiesen.

Operation	Operanden
ENTRY	{programmabschnitt / einsprungstelle}

ERREXIT-Anweisung

Sie vereinbart für alle Externverweise, die der Binder nicht befriedigen kann, einen gemeinsamen Wert. Fehlt die Anweisung, so erhalten die unbefriedigten Externverweise den Sedezimalwert FFFFFFFF.

Gibt man mehrere ERREXIT-Anweisungen ein, so verwendet der Binder die zuletzt eingelesene.

Operation	Operanden
ERREXIT	

A

=nnnnnn legt einen 6stelligen Dezimalwert fest, den der Binder allen Externverweisen, die er nicht befriedigen kann, zuweisen soll.

=X'xxxxxxxx' legt einen 8stelligen Sedezimalwert fest, den der Binder allen Externverweisen, die er nicht befriedigen kann, zuweisen soll.

E

=programmabschnitt
gibt dem Binder den Namen eines Programmabschnitts (CSECT) an, mit dem er alle unlösbaren Externverweise befriedigen soll.

=einsprungsstelle
legt eine Einsprungstelle (ENTRY) in einem der Bindemodule des zu bindenden Programms fest, die der Binder zur Befriedigung aller unlösbaren Externverweise verwenden soll.

EXCLUDE-Anweisung

Die EXCLUDE-Anweisung verhindert, daß der Binder Externverweise aus der in ihr angegebenen Bindemodulbibliothek befriedigt. Die Anweisung bewirkt also das Gegenteil der RESOLVE-Anweisung (siehe RESOLVE-Anweisung, Seite132ff).

Operation	Operanden
EXCLUDE	[{externverweis / (externverweis,...)}],bibliothek

externverweis — vereinbart den Namen eines Externverweises, den der Binder nicht aus der zugewiesenen Bindemodulbibliothek befriedigen soll. Höchstens 20 Externverweise sind pro EXCLUDE-Anweisung erlaubt. Gibt man keinen Namen an, so befriedigt der Binder überhaupt keine Externverweise aus der angegebenen Bibliothek.

bibliothek — legt den Namen einer Bindemodulbibliothek fest, die der Binder nicht benutzen soll, um Externverweise zu befriedigen. Diese Angabe ist obligatorisch.

Hinweis

Im Gegensatz zur RESOLVE-Anweisung prüft der Binder nicht, ob er die angegebene Bindemodulbibliothek eröffnen kann (OPEN).

Beispiel

```
EXCLUDE ,TASKLIB
```

Diese Steueranweisung für den Binder schließt die Bindemodule in der Bibliothek TASKLIB für die Befriedigung der Externverweise aus (siehe Seite 19).

(Weiteres Beispiel siehe RESOLVE-Anweisung, Seite 133).

INCLUDE-Anweisung

Sie veranlaßt den Binder, einen oder mehrere Bindemodule aus der angegebenen Eingabequelle einzulesen und beim Binden dem Programm bzw. dem Bindemodul anzufügen. Eine Folge von INCLUDE-Anweisungen legt also fest, welche Bindemodule in welcher Reihenfolge eingebunden werden sollen.

Die INCLUDE-Anweisung darf in der Bindereingabe nur fehlen, wenn man Bindemodule aus der Systemdatei SYSDTA einlesen läßt.

Operation	Operanden
INCLUDE	{ modul[(version)][,bibliothek] (modul[(version)],...)[,bibliothek] ,bibliothek modul,* (modul,...),* * }

modul — legt den Bindemodul fest, der aus der zugewiesenen Bindemodulbibliothek bzw. aus der temporären EAM-Bindemoduldatei eingelesen werden soll. Gibt man mehrere Bindemodule an (maximal 20), muß man sie in runde Klammern einschließen.

version — gibt die Versionsbezeichnung des Bindemoduls "modul" an. Die Versionsbezeichnung ist nur für Programmbibliotheken gültig.
Läßt man die Angabe "version" weg, wird der Bindemodul mit der höchsten Versionsbezeichnung eingebunden.

bibliothek — gibt als Eingabequelle eine Bindemodulbibliothek dieses Namens an. Wird kein "modul" angegeben, liest der Binder alle Bindemodule der Bibliothek ein. Läßt man die Angabe "bibliothek" weg, so sucht der Binder den "modul" in der Bindemodulbibliothek TASKLIB.

* — gibt die temporäre EAM-Bindemoduldatei der laufenden Task an. Wird "modul" weggelassen, liest der Binder alle Bindemodule ein, die sich zu diesem Zeitpunkt in der temporären EAM-Bindemoduldatei befinden.

Beispiel

1. `INCLUDE MODA,LIBX`

   Der Modul MODA mit der höchsten Versionsbezeichnung soll aus der Bibliothek LIBX eingebunden werden.

2. `INCLUDE (MODA,MODB),LIBX`

   Die Module MODA und MODB mit der höchsten Versionsbezeichnung sollen aus LIBX eingebunden werden.

3. `INCLUDE (MODC,MODD),*`

   Die Module MODC und MODD sollen aus der temporären EAM-Bindemoduldatei der laufenden Task eingebunden werden.

4. `INCLUDE ,LIBX`

   Sämtliche Module mit der höchsten Versionsbezeichnung sollen aus der Bibliothek LIBX eingebunden werden.

5. `INCLUDE MODX(007),LIB9`

   Der Modul MODX mit der Versionsbezeichnung 007 soll aus der Bibliothek LIB9 eingebunden werden. Wird der Modul MODX mit dieser Versionsbezeichnung nicht gefunden, wird die Bibliothek LIB9 übergangen.

LET-Anweisung

Sie verhindert den Abbruch des Bindens, wenn der Binder nicht alle Externverweise befriedigen kann. Sie wirkt wie der Operand LET=Y in der PROGRAM- bzw. MODULE-Anweisung.

Operation	Operanden
LET	[beliebiger Kommentar]

LINK-SYMBOLS-Anweisung

Mit der LINK-SYMBOLS-Anweisung wird TSOSLNK beim Modulbinden mitgeteilt, in welchem Umfang die Symbole für die Einsprungstellen (ENTRYs) und die Programmabschnitte (CSECTs) für einen späteren Binderlauf sichtbar sein sollen oder ob sie maskiert werden sollen. Maskierte Symbole werden bei späteren Binderläufen nicht mehr zum Befriedigen von Externverweisen herangezogen. Die LINK-SYMBOLS-Anweisung darf nur nach einer MODULE-Anweisung angegeben werden.

Operation	Operanden
LINK-SYMBOLS LI-SYM	*HIDE [KEEP / HIDE] = symbol / (symbol,...) *KEEP *NOESD

*HIDE — Der Binder führt ein komplettes ESD (External Symbolic Dictionary). Alle ENTRYs und CSECTs - bis auf den 1. CSECT - werden für späteres Binden "unsichtbar" gemacht, d.h. sie können nicht mehr zur Befriedigung von Externverweisen herangezogen werden.

KEEP
=symbol — Die mit symbol angegebenen Programmabschnitte bzw. Einsprungstellen und der 1. CSECT bleiben im ESD für spätere Binderläufe erkennbar. Alle anderen werden maskiert.

HIDE
=symbol — Alle Symbole, bis auf die mit HIDE=symbol explizit angegebenen, bleiben für spätere Binderläufe erkennbar.

*KEEP — Der Binder führt ein komplettes ESD. Die CSECTs und die ENTRYs bleiben für späteres Binden erkennbar.

*NOESD — Der Binder legt nur einen ESD-Satz für den Namen des Moduls an. Wird dieser Operand angegeben, so verhält sich der Binder beim Modulbinden wie Versionen ≤ 16.4 des TSOSLNK. Der erzeugte Modul enthält keine Externverweise oder symbolische Informationen mehr. Alle Programmabschnitte gehen verloren.

Hinweis

- In einer LINK-SYMBOLS-Anweisung dürfen maximal 30 CSECT- bzw. ENTRY-Namen eingegeben werden.
- Ein Symbol zu maskieren heißt, daß es bei späteren Binderläufen von TSOSLNK nicht mehr aufgefunden wird. Dies gilt auch besonders für Autolink-Funktionen. Das Symbol wird zwar real im ESD gespeichert. Es wird aber mit einem Kennzeichen versehen, das es "unsichtbar" macht. "Unsichtbar machen" eines Symbols im ESD wirkt sich lediglich so aus, daß dieses Symbol in späteren Binderläufen oder vom DBL nicht mehr zur Befriedigung von Externverweisen hergenommen wird. Namenskonflikte werden durch Maskierung von Symbolen jedoch nicht ausgeschlossen.
- Die maskierten Symbole können nicht mehr zur Befriedigung von Externverweisen bei späteren Binderläufen und beim dynamischen Binden mit dem DBL verwendet werden.
- Sie werden nicht ins Sekundärinhaltsverzeichnis (DIRECTORY 2) der Bindemodul- oder Programmbibliothek aufgenommen. Das trägt auch zur Beschleunigung des Bindevorgangs bei.
- Wird die Anweisung LINK-SYMBOLS nicht verwendet oder werden die Operanden *HIDE, HIDE=, *KEEP oder KEEP= angegeben, so enthält der Modul ein komplettes ESD, in dem jedoch CSECTs oder ENTRYs maskiert sein können.

MODULE-Anweisung

Sie veranlaßt den Binder, die einzulesenden Bindemodule zu einem einzigen vorgebundenen Bindemodul (Großmodul) zusammenzufügen.

Die MODULE-Anweisung muß man stets als erste Anweisung eingeben. Sie ist unvereinbar mit der PROGRAM-, CLASS-, ENTRY-, NOCTL-, OVERLAY-, SHARE- und XCAL-Anweisung, d.h. sobald der Binder die MODULE-Anweisung akzeptiert hat, weist er diese Anweisungen zurück. Mit der LINK-SYMBOLS-Anweisung kann beim Modulbinden angegeben werden, in welchem Umfang die Symbole für ENTRYs und CSECTs für spätere Binderläufe mit TSOSLNK oder DBL benutzt werden können.

Die Anweisung kann wiederholt werden, wenn für den gleichen Modul Operanden geändert oder hinzugefügt werden sollen.

Operation	Operanden
MOD[ULE]	[modul1] [,ENDC=modul2] [,LET={Y|N}]
	[,LIB[RARY]=bibliothek] [,ELEM[ENT]=element[(version)]]
	[,ARM[ODE]-C[HECK]={IGN[ORE] | WARN[ING] | ABORT}]
	[,MAP={Y|N}] [,XREF={Y|N}] [,LIST={Y|N}]
	[,XDSEC={Y|N}] [,PR={Y|N}] [,PL1={Y|N}]
	[,SYSLST={Y|N}] [,UNSAT={Y|N|S}]
	[,WUNSAT={Y|N}] [,SORT={Y|N}]
	[,CMAP={ {ALL | NO} | ([{CS[ECTS] | NOCS[ECTS]}] [,{EN[TRYS] | NOEN[TRYS]}] [,{COM[MONS] | NOCOM[MONS]}]) | ([,{X[REF] | NO[XREF]}] [,{EJ[ECT] | NOEJ[ECT]}] [,MOD[ULES]]) }]
	[,LINE=number]
	[,NA-COL={STANDARD|STD | IGN[ORE] | ABORT}]

modul1

legt den Namen fest, den der neu erstellte Bindemodul erhalten soll. Der Modulname darf bis zu 8 Zeichen lang sein und aus Buchstaben, Ziffern und den Sonderzeichen $, # und @ bestehen. Das erste Zeichen muß ein Buchstabe oder eines der Sonderzeichen sein.

Wird die LINK-SYMBOLS-Anweisung nicht angegeben oder wird in der LINK-SYMBOLS-Anweisung der Operand *NOESD nicht angegeben, gilt:
Mit Ausnahme des ersten Programmabschnittnamens darf kein Name eines anderen ESD-Eintrags für "modul1" benutzt werden, auch wenn dieser Eintrag maskiert wurde.

Gibt man "modul1" nicht an, so benennt der Binder den Modul nach dem Namen des ersten Programmabschnitts.

ENDC
=modul2

veranlaßt den Binder, den END-Satz des Bindemoduls "modul2" an das Ende des neu erstellten Bindemoduls zu setzen. Wird diese Angabe weggelassen, schreibt der Binder den END-Satz des ersten eingelesenen Moduls an das Ende des neu erstellten Moduls.

LET
=Y

legt fest, daß im Stapelbetrieb der Modul auch dann gebunden werden soll, wenn der Binder unbefriedigte Externverweise entdeckt.

=N

legt für den Stapelbetrieb fest, daß der Binderlauf abgebrochen werden soll, sobald der Binder auf unbefriedigte Externverweise stößt.

Im Dialogbetrieb nimmt sowohl für LET=Y als auch für LET=N der Binder den Dialog wieder auf, sobald er unbefriedigte Externverweise entdeckt, so daß der Benutzer weitere Binderanweisungen eingeben kann.

LIBRARY
=bibliothek

gibt den Namen einer Programmbibliothek an, in die der Modul geschrieben werden soll. Der Modul wird nicht in die Bibliothek geschrieben, die als Eingabebibliothek im selben TSOSLNK-Lauf verwendet wird. Existiert die angegebene Bibliothek nicht, wird sie von TSOSLNK als Programmbibliothek eingerichtet. Wird keine Bibliothek angegeben, wird der Modul in die temporäre EAM-Bindemoduldatei geschrieben.

Der Bibliotheksname darf maximal 54 Zeichen lang sein.

ELEMENT

Name und Versionsbezeichnung, unter denen der Modul in die Programmbibliothek aufgenommen werden soll.
Für den Namen und die Versionsbezeichnung muß der Benutzer die Namenskonventionen von LMS beachten (siehe Handbuch "LMS" [3]). Werden die Namenskonventionen nicht eingehalten, kann evtl. LMS das Element nicht mehr bearbeiten. Dieser Operand kann nur dann angegeben werden, wenn gleichzeitig auch der Operand LIBRARY angegeben ist, da eine Versionsführung in der EAM-Bindemoduldatei nicht moglich ist.

=element — gibt den Namen des Elements an, unter dem der Modul in die Programmbibliothek geschrieben wird. Fehlt der Elementname, wird der Modulname als "element" angenommen.

Der Elementname darf maximal 8 Zeichen lang sein. TSOSLNK verarbeitet zwar Elementnamen mit einer maximalen Länge von 41 Zeichen. Diese Elemente können aber von TSOSLNK oder DBL in späteren Bindeläufen nicht weiterverarbeitet werden.

version — gibt die Versionsbezeichnung des Elements "element" an. Die Versionsbezeichnung ist nur für Programmbibliotheken gültig und darf max. 24 Zeichen lang sein.
Läßt man die Angabe "version" weg, wird der Standardwert für die Versionsbezeichnung bei Programmbibliotheken angenommen (siehe Handbuch "LMS" [3]).

ARM[ODE]-C[HECK]

Legt fest, daß die CSECT’s auf Attribute AMODE und/oder RMODE ungleich ANY überprüft werden.

=IGN[ORE] — Es wird nicht geprüft, ob die CSECT’s die Attribute AMODE und/oder RMODE ungleich ANY enthalten.

=WARN[ING] — Für jede CSECT, die Attribute AMODE und/oder RMODE ungleich ANY enthält, wird die Warnungsmeldung LNK0065 ausgegeben. Anschließend wird der Binderlauf fortgesetzt.

=ABORT — Für jede CSECT, die Attribute AMODE und/oder RMODE ungleich ANY enthält, wird die Warnungsmeldung LNK0065 ausgegeben. Anschließend wird der Binderlauf fehlerhaft beendet.

MAP

=Y — vereinbart die Erzeugung einer Modulübersicht, die Informationen über die Größe des neuen Bindemoduls, über Länge und Adressen der eingebundenen Module enthält. Der Binder gibt diese Übersicht in die Systemdatei SYSLST aus.

=N unterdrückt die Erzeugung der Programmübersicht.

XREF

=Y gibt an, daß der Binder eine Querverweisliste des gebundenen Bindemoduls erzeugen soll. Die Liste enthält die Adressen aller Programmabschnitte, Einsprungstellen und Externverweise sowie Angaben über die Befriedigung der Externverweise. Die Ausgabe erfolgt nach SYSLST.

=N verhindert die Ausgabe der Querverweisliste.

LIST

=Y Nur wirksam im Dialogbetrieb.
Der Binder gibt die Anweisungen und eine Kurzform der Modulübersicht des gebundenen Moduls nach SYSOUT aus. Der Operand wirkt sich bereits auf die MODULE-Anweisung aus, in der er steht.

=N bewirkt, daß der Binder die eingegebenen Anweisungen nicht protokolliert und keine Kurzform der Modulübersicht nach SYSOUT gibt.

XDSEC

=Y vereinbart die Ausgabe einer Liste in die Systemdatei SYSLST, in der sämtliche Verweise von externen Pseudoabschnitten nach Modulen geordnet aufgeführt sind. Zu jedem Externverweis gibt der Binder dabei an, mit Hilfe welcher Bindemodule er ihn befriedigen konnte.

=N beschränkt die Ausgabe auf die Verweise externer Pseudoabschnitte, die der Binder nicht befriedigen kann.

PR

=Y veranlaßt, daß der Binder sämtliche Pseudoregister nach Modulen geordnet in die Systemdatei SYSLST ausgibt, und zwar mit den ihnen zugewiesenen Offsetwerten.

=N beschränkt die Ausgabe auf eine Liste der unbefriedigten Pseudoregister.

PL1

=Y vereinbart die Verarbeitung simulierter Q-Konstanten (siehe Handbuch "Assembler" [2]), d.h. der Binder weist allen Externverweisen, die mit den Zeichen IQ... anfangen, keine Adresse, sondern einen Pseudoregister-Offsetwert zu.

=N verhindert die Verarbeitung simulierter Q-Konstanten.

SYSLST

=Y veranlaßt den Binder, in die Systemdatei SYSLST auszugeben.

=N unterdrückt die Ausgabe nach SYSLST.

UNSAT

=Y veranlaßt den Binder, eine Liste aller unbefriedigten Externverweise auszugeben.

=N Die Liste der unbefriedigten Externverweise wird nicht ausgegeben.

=S veranlaßt den Binder, eine alphabetisch sortierte Liste der unbefriedigten Externverweise auszugeben.

WUNSAT

=Y veranlaßt den Binder, eine Liste der unbefriedigten Externverweise auszugeben, deren ESD-Typ X'F9' oder X'F2' bzw. X'F4' ist und deren Namen mit I$ beginnen.

ESD-Typen

X'F2': ER: Externe Verknüpfungsadresse (EXTRN-Anweisung)
X'F4': VC: Externe Verknüpfungsadresse (V-Konstante)
X'F9': WX: Bedingte Extern-Adresse (WXTRN-Anweisung)

=N Es wird keine Liste der unbefriedigten Externverweise ausgegeben.

Ist der Operand UNSAT=N gesetzt, wird die Angabe im Operanden WUNSAT ignoriert, d.h. es werden keinerlei unbefriedigte Externverweise ausgegeben.

SORT

=YES erzeugt eine zusätzliche Liste, in der die Namen aller Programmabschnitte und Einsprungstellen alphabetisch aufsteigend sortiert enthalten sind. Die Liste enthält für jeden Eintrag:

- den Namen des Programmabschnitts (CSECT-Name) oder der Einsprungstelle (ENTRY-Name).
- die Anfangsadresse in sedezimaler und dezimaler Form (ADDRESS).
- eine Kennzeichnung als Programmabschnitt (CSECT) oder Einsprungstelle (ENTRY).
- den Namen des Moduls (MODULE CONTAINING) und/oder des Segments (SEGMENT ENTRY), in dem der Programmabschnitt oder die Einsprungstelle liegen.

Die Ausgabe dieser sortierten Namensliste ist unabhängig von anderen Ausgabelisten von TSOSLNK, insbesondere unabhängig von der Ausgabe der Externverweisliste.

=NO — Eine alphabetische Liste mit den Namen der Programmabschnitte und Einsprungstellen wird nicht ausgegeben.

CMAP

=ALL — Es wird eine vollständige Programmübersicht und eine Querverweisliste ausgegeben.

=NO — Unterdrückt die Ausgabe der vollständigen Programmübersicht und der Querverweisliste.

=(...) — Maximal dürfen nur 6 der nachfolgenden Optionen angegeben werden, die in runde Klammern einzuschließen sind.

=CSECT — Eine Liste der Bindemodule und der Programmabschnitte wird ausgegeben.

=NOCSECT — Es wird keine Liste der Programmabschnitte ausgegeben.

=ENTRYS — Eine Liste der Programmabschnitte und der Einsprungstellen wird ausgegeben.

=NOENTRYS — Es wird keine Liste der Einsprungstellen ausgegeben.

=COMMONS — Eine Liste der Module und COMMON-Bereiche wird ausgegeben.

=NOCOMMONS — Es wird keine Liste der COMMON-Bereiche ausgegeben.

=XREF — Eine Liste der Querverweise und der Module wird ausgegeben.

=NOXREF — Es wird keine Querverweisliste ausgegeben.

=EJECT — Bei jedem neuen Segment wird ein Seitenvorschub gemacht.

=NOEJECT — Es wird kein Seitenvorschub gemacht.

=MODULES — Eine Liste der Module wird ausgegeben.

Eine vollständige Programmübersicht und eine Querverweisliste werden unabhängig von der Angabe CMAP ausgegeben:

- in einer BS2000-Version ≥9.0 oder
- in einer BS2000-Version <9.0 und einer Ladeadresse oberhalb 16 Mbyte.

LINE

=number legt die Anzahl der Zeilen pro Seite bei Ausgabe von Listen in die Systemdatei SYSLST fest.
Für "number" dürfen der Wert 0 und alle Werte ≥30 angegeben werden. Andere Werte werden abgewiesen.
Standardwert ist 54.
Wenn der Wert 0 angegeben ist, wird kein Seitenvorschub durchgeführt.

NA-COL

=STANDARD/STD

Die Bindemodule werden auf Namenskonflikte hin untersucht. Namenskonflikte der Form CSECT - CSECT werden erkannt und es wird eine Meldung ausgegeben. Der Binderlauf wird fortgesetzt.

Die Bindemodule werden *nicht* auf Namenskonflikte untersucht, wenn in der LINK-SYMBOLS-Anweisung der Operand *NOESD angegeben wurde (Kompatibilität zu früheren Binderversionen).

=IGN[ORE] Die Bindemodule werden nicht auf Namenskonflikte hin untersucht.

=ABORT Die Bindemodule werden auf Namenskonflikte hin untersucht. Namenskonflikte der Form CSECT - CSECT werden erkannt und es wird eine Meldung ausgegeben. Der Binderlauf wird fortgesetzt, bis alle Bindemodule untersucht sind und dann abgebrochen.

Hinweis

- Bei Namenskonflikten benutzt TSOSLNK den ersten auftretenden Namen zur Befriedigung der Externverweise. Der Name einer CSECT hat Vorrang vor dem Namen eines ENTRY.
- Der dynamische Bindelader DBL kann von TSOSLNK vorgebundene Module mit Namenskonflikten nicht weiterverarbeiten, d.h. der Operand NA-COL=IGNORE ist in diesem Fall nicht sinnvoll.

Hinweis für 31-Bit-Schnittstelle

Damit der dynamische Bindelader DBL einen vorgebundenen Modul als Einheit laden kann, ist es notwendig, die Lage des Moduls im Adreßraum (oberhalb oder unterhalb 16Mbyte) im gesamten festzulegen. Zu diesem Zweck wird für den Modul beim Binden ein "Pseudo-RMODE" festgelegt, der aus den Attributen RMODE der einzelnen CSECTs wie folgt bestimmt wird:

- Der Bindemodul erhält nur dann das Attribut (Pseudo-)RMODE=ANY, wenn alle enthaltenen CSECTs das Attribut RMODE=ANY besitzen.
- Enthält mindestens eine CSECT das Attribut RMODE=24, erhält auch der Modul das eingeschränkte Attribut (Pseudo-)RMODE=24.

Das Attribut AMODE wird durch den Programmabschnitt (CSECT) bestimmt, der die Einsprungstelle des Bindemoduls enthält.

*Eigenschaften des vorgebundenen Moduls, wenn die Anweisung LINK-SYMBOLS nicht oder bei der Anweisung LINK-SYMBOLS der Operand *NOESD nicht angegeben wurde*

Der vorgebundene Modul enthält ein komplettes ESD, d.h. er enthält alle ursprünglichen CSECTs und ENTRYs. Das ESD enthält auch alle COMMONs, XDSEC-Definitionen, Pseudoregister und IQ-Konstanten. Falls manche von diesen Einträgen doppelt vorkommen, wird jedoch nur der mit den höchsten Werten (Länge, Kennzeichen) gespeichert. Unbefriedigte Externverweise werden ebenfalls übernommen.

XDSEC R wird ebenfalls gespeichert. Nur wenn XDSEC D auch vorhanden ist, werden die Verweise in den TXT-Sätzen aufgelöst und XDSEC R aus dem ESD entfernt.

Befriedigte V-CONs, EXTRNs, WXTRNs und I$-Externverweise werden nicht in das ESD übernommen. Sie werden wie A-Konstanten behandelt.

Treten Externverweise in mehreren Formen auf, so werden sie im ESD folgendermaßen geführt:

Erste aufgetretene Form	andere Formen	Eintrag im ESD
VCON	EXTRN (und WXTRN)	VCON und EXTRN
VCON	WXTRN	VCON und WXTRN
EXTRN	WXTRN	EXTRN

EXTRN und WXTRN können nicht als V-Konstanten gespeichert werden.

Gewöhnlich erhalten die ENTRYs dieselbe Bezeichnung (ESID) wie die zugehörige CSECT.

Ein zusätzlicher ESD-Eintrag - der erste - wird dann erzeugt, wenn in der MODULE-Anweisung ein Modulname angegeben wird. Wenn dieser Name nicht mit dem des 1. CSECT übereinstimmt, wird der so erzeugte ESD-Eintrag zum 1. CSECT mit der Länge Null und ohne Attribute.

Der in der MODULE-Anweisung angegebene Modulname darf nicht gleichzeitig der Name eines anderen ESD-Eintrages sein.

Verarbeitung von ESD-Einträgen des vorgebundenen Moduls (Großmoduls):

ESD vor dem Binden	ESD nach dem Binden
CSECT	CSECT
ENTRY	ENTRY
unbefriedigte VCON	VCON
unbefriedigte EXTRN	EXTRN
unbefriedigte WXTRN	WXTRN (oder EXTRN)
befriedigte Externverweise	unterdrückt
COMMON	COMMON
Pseudoregister (PR)	PR
XDSEC D	XDSEC D
R - mit XDSEC D	unterdrückt
- ohne XDSEC D	XDSEC R
DSECT	unterdrückt

Die Bearbeitung von IQ- und I$-Externverweisen bleibt unverändert.

Testhilfedaten werden nicht erzeugt.
Testhilfen und Korrekturanweisungen können auf CSECT-Namen aufsetzen.

Hinweis

– Die vorgebundenen Module können ohne weitere Einschränkungen wie von Sprachübersetzern erzeugte Module verwendet werden.

– Im ESD können CSECTs und ENTRYs maskiert werden, d.h. sie werden bei späteren Binderläufen mit TSOSLNK nicht zur Befriedigung von Externverweisen verwendet (siehe Anweisung LINK-SYMBOLS, Seite 96).

Beispiel

```
/START-PROGRAM FROM-FILE=$TSOSLNK
%  BLS0500 PROGRAM 'TSOSLNK', VERSION '21.0D00' OF '90-05-10' LOADED.
*MODULE EINXBD,XREF=Y ------------------------------------------------ (01)
*INCLUDE EINXEINS,LMS.BIBL ------------------------------------------- (02)
*RESOLVE ,$RZ.COB1MODLIB --------------------------------------------- (03)
*END
%  LNK0501 MOD BOUND
%  LNK0505 MODULE 'EINXBD' WRITTEN TO EAM OMF. TSOSLNK RUN FINISHED ---- (04)
/
```

(01) Die MODULE-Anweisung veranlaßt den Binder, einen Bindemodul aufzubauen und in die temporäre EAM-Bindemoduldatei der laufenden Task auszugeben. Neben der Modulübersicht (Standardfall) soll eine Querverweisliste erstellt werden.

(02) Modul EINXEINS aus der Bibliothek LMS.BIBL soll als erster eingebunden werden.

(03) Der Binder soll versuchen, unbefriedigte Externverweise mit Hilfe der Bibliothek $RZ.COB1MODLIB aufzulösen, d.h. evtl. benötigte Module mit einzubinden.

(04) Nachdem der Binder alle eingelesenen Bindemodule zu einem einzigen Bindemodul gebunden hat, gibt er diesen in die temporäre EAM-Bindemoduldatei der Task aus.

NCAL-Anweisung

Sie schließt die TASKLIB für die Befriedigung von Externverweisen aus. Sie wirkt wie die Anweisung EXCLUDE ,TASKLIB und verhindert, daß der Binder die Bindemodulbibliothek TASKLIB nach unbefriedigten Externverweisen durchsucht.

Operation	Operanden
NCAL	[beliebiger Kommentar]

NOCTL-Anweisung

Sie verhindert bei segmentierten Programmen die Erzeugung eines Überlagerungssteuermoduls und der zugehörigen Tabellen. Sie wirkt wie der Operand CONTROL=N in der PROGRAM-Anweisung. Mit dem Modulbinden (siehe MODULE-Anweisung, Seite 98ff) ist sie unvereinbar und wird mit einer Fehlermeldung zurückgewiesen.

Operation	Operanden
NOCTL	[beliebiger Kommentar]

NOMAP-Anweisung

Sie unterdrückt die Erzeugung einer Programmübersicht. Sie wirkt wie der Operand MAP=N in der PROGRAM- bzw. MODULE-Anweisung.

Operation	Operanden
NOMAP	[beliebiger Kommentar]

OVERLAY-Anweisung

Mit der OVERLAY-Anweisung vereinbart man für das zu erzeugende Programm eine Überlagerungsstruktur. Jede Anweisung bezeichnet Namen und Lage eines Segments, in das der Binder diejenigen Bindemodule einbindet, die in den nachfolgenden INCLUDE-Anweisungen angegeben sind. Die Reihenfolge der Overlay-Anweisungen steuert den Aufbau, d.h. je nach Anordnung der Anweisungen erzeugt der Binder unterschiedliche Überlagerungsstrukturen. Wahlweise kann man den Anfang einer neuen Region markieren (siehe auch Seite 47).

Beim Modulbinden (siehe MODULE-Anweisung, Seite 98ff) wird die OVERLAY-Anweisung mit einer Fehlermeldung zurückgewiesen.

Wenn der Operand LINEAR=Y in der PROG-Anweisung gesetzt wurde, wird die OVERLAY-Anweisung ignoriert. Der Überlagerungssteuermodul (siehe Seite 53ff) wird nicht erzeugt.

Operation	Operanden
OVERLAY	knotenpunkt [,REGION] [,segment] [,LOADPT={adresse|*XS}]

knotenpunkt — vereinbart den Namen eines Überlagerungsknotens, d.h. eines Punktes in der Überlagerungsstruktur, an dem mehrere Segmente aneinandergrenzen. Der Name darf bis zu 8 Zeichen lang sein und aus Buchstaben, Ziffern und den Sonderzeichen $, # und @ bestehen. Das erste Zeichen muß ein Buchstabe oder eines der Sonderzeichen sein.

Wird in der OVERLAY-Anweisung der Name des Knotenpunktes zum erstenmal genannt, so fügt der Binder das betreffende Segment adreßmäßig an das zuletzt bearbeitete Segment an. Wird ein bereits definierter Knotenpunkt genannt, so wird das zugehörige Segment mit der gleichen Anfangsadresse wie das Segment der ersten Nennung versehen.

REGION — markiert den Anfang einer neuen Region, d.h. stellt sicher, daß das zugehörige Segment andere nicht überlagert. Der Binder legt dann den Knotenpunkt anschließend an das bis dahin adreßmäßig höchste Segment.

Die Zeichenfolge REGION darf man weder als Name eines Knotenpunktes noch als Segmentnamen verwenden.

segment — legt den Namen eines Segments fest, d.h. eines Programmteils, der unabhängig von anderen Teilen geladen und ausgeführt werden kann. Ein Segmentname muß eindeutig und darf nicht länger als 8 Zeichen sein. Bei Segmenten, deren erste 6 Zeichen gleich sind, wird die Meldung LNK0006 ausgegeben.

Fehlt die Angabe "segment", so verwendet der Binder die ersten 6 Zeichen vom Namen des ersten Bindemoduls, der anschließend eingelesen wird. Besteht dieser Name aus Leerzeichen oder ist er nicht eindeutig, so erzeugt der Binder einen Namen AAAnnn, wobei nnn eine Zahl von 001 bis 999 sein kann.

Werden Knotenpunkt und Segment nochmals gemeinsam genannt, so wird in einem bereits begonnenen Segment fortgefahren.

LOADPT

=adresse — definiert eine sedezimale Adresse (X'...') im virtuellen Adreßraum, ab der das Segment geladen werden soll. Die Adresse muß auf Seitengrenze ausgerichtet sein. Ist dies nicht der Fall, so wird das Segment ab der nächst höheren Seitengrenze geladen.

=*XS — legt die Ladeadresse des zugehörigen Segments im Adreßraum oberhalb 16 MB fest. Die Ladeadresse ist von evtl. weiteren OVERLAY-Anweisungen abhängig.

Warnung

Beim Binden eines Programms, das auf einem Vektorrechner ablaufen soll, dürfen nur die virtuellen Adressen 0 oder 16Mbyte angegeben werden. Andere Werte führen zu einem Fehler beim Starten des Programms.

Beispiel

Die folgenden Anweisungen veranlassen den Binder, eine Überlagerungsstruktur zu erzeugen.

Einzelne Buchstaben, z.B. A, B, C, bedeuten im Beispiel die Namen von Bindemodulen. Knoten sind mit N1, N2, N3 und N4 bezeichnet. S2 bis S9 sind Segmentnamen.

Statt der INCLUDE-Anweisungen könnten auch die Bindemodule selbst eingegeben werden.

Bild 15 zeigt die Überlagerungsstruktur, die sich aus den nachfolgenden Anweisungen ergibt.

Zeile Nr.	
1	PROGRAM EXAMPLE
2	INCLUDE (A,B,C),LIB
3	OVERLAY N1,S2
4	INCLUDE D,LIB
5	OVERLAY N1,S3
6	INCLUDE F,LIB
7	OVERLAY N2,S4
8	INCLUDE (G,H),LIB
9	OVERLAY N1,S2
10	INCLUDE E,LIB
11	OVERLAY N1,S6
12	INCLUDE L,LIB
13	OVERLAY N2,S5
14	INCLUDE (I,K),LIB
15	OVERLAY N3,REGION,S7
16	INCLUDE M,LIB
17	OVERLAY N4,S8
18	INCLUDE (O,P),LIB
19	OVERLAY N4,S9
20	INCLUDE (R),LIB

Zeile 1 bezeichnet den Programmnamen.

Zeile 2 bindet die Module A, B und C aus der Bibliothek LIB in das Grundsegment des Programms ein.

Zeile 3 definiert den Knotenpunkt N1 und das Segment S2.

Zeile 4 bindet den Modul D aus LIB in das Segment S2.

Zeile 5 definiert das Segment S3, das denselben Knotenpunkt (also auch dieselbe Ladeadresse) wie Segment S2 hat.

Zeile 6 bindet den Modul F in S3.

Zeile 7 definiert einen neuen Knotenpunkt N2 am Ende von S3 und einen Segmentnamen S4.

Zeile 8 bindet die Module G und H in Segment S4.

Zeile 9 weist den Binder an, die folgenden Bindemodule in das Segment S2 einzubinden, das in Zeile 3 bereits definiert wurde.

Zeile 10 bindet E in S2 anschließend an D, der bereits eingebunden wurde.

Zeile 11 definiert Segment S6 am Knotenpunkt N1.

Zeile 12 bindet in das Segment S6.

Zeile 13 definiert Segment S5, das am selben Knotenpunkt beginnt wie S4.

Zeile 14 bindet I und K in S5.

Zeile 15 definiert eine neue Region am Ende des längsten vorausgegangenen Zweiges, legt dort den Knoten N3 fest und definiert das Segment S7.

Zeile 16 bindet M in das Segment S7.

Zeile 17 definiert den Knoten N4 am Ende S7 und den Segmentnamen S8.

Zeile 18 bindet O und P in S8.

Zeile 19 definiert das Segment S9 am Knoten N4.

Zeile 20 bindet R in das Segment S9.

Weitere Beispiele zur OVERLAY-Anweisung finden Sie auf Seite 40 und in Bild 7 (Seite 46).

Bild 15 Beispiel für eine Überlagerungsstruktur

Bild 16 Namensvergabe durch TSOSLNK in PROGRAM- und OVERLAY-Anweisungen

PAGE-Anweisung

Die PAGE-Anweisung richtet einen Programmabschnitt beim Binden auf Seitengrenze aus. Sie wirkt ab dem Eingabezeitpunkt wie der Operand PAGE=Y in der TRAITS-Anweisung.

Operation	Operanden
PAGE	programmabschnitt

PROGRAM-Anweisung

Sie beschreibt die Ausgabe des Binders. Für die Operanden, die in ihr nicht explizit genannt werden, gelten die Angaben, die zuletzt in vorangegangenen PROGRAM-Anweisungen gemacht wurden, andernfalls Standardwerte. Sind nicht alle Operanden in einer einzigen PROGRAM-Anweisung unterzubringen, muß man mehrere eingeben und dabei den Programmnamen in jeder von ihnen wiederholen.

Die PROGRAM-Anweisung und die MODULE- bzw. LINK-SYMBOLS-Anweisung sind unvereinbar, d.h. der Binder weist die PROGRAM-Anweisung mit einer Fehlermeldung zurück, sobald er eine MODULE-Anweisung bearbeitet hat.

Operation	Operanden
PROG[RAM]	programm [,VERSION=version] [,COPYRIGHT=(name,jahr)] [,{FILENAM=datei[,SHARE={Y|N}] / LIB[RARY]=bibliothek[,ELEM[ENT]=element[(version)]]}] [,P[AM]-K[EY]={Y|N}] [,ARM[ODE]-C[HECK]={IGN[ORE] / WARN[ING] / ABORT}] [,{IDA={Y|N} / SYMTEST={ALL|N|MAP}}]

Operation	Operanden
PROGRAM (Forts.)	[,LET={Y|N}] [,XS-C[HECK]={Y|N}] [,CLASS={2|E}] [,COREIM={Y|N}] [,CONTROL={Y|N}] [,XCAL={Y|N}] [,LOADPT={adresse|*XS}] [,MAX=proglänge] [,ADD=länge] [,ENTRY={programmabschnitt|einsprungstelle}][,START={programmabschnitt|einsprungstelle}] [,PL1={Y|N}] [,PR={Y|N}] [,XDSEC={Y|N}] [,MAP={Y|N}] [,XREF={Y|N}] [,LIST={Y|N}] [,SYSLST={Y|N}] [,UNSAT={Y|N|S}] [,WUNSAT={Y|N}] [,SORT={Y|N}] [,CMAP={ {ALL|NO} | ([{CS[ECTS]|NOCS[ECTS]}][,{EN[TRYS]|NOEN[TRYS]}][,{COM[MONS]|NOCOM[MONS]}]) | [,{X[REF]|NO[XREF]}][,{EJ[ECT]|NOEJ[ECT]}][,MOD[ULES]]) }] [,LINE=number] [,LINEAR={Y|N}]

programm
: vereinbart einen bis zu 41 Zeichen langen Namen, den der Binder als Name des gebundenen Programms in die Programmdatei einträgt. Falls man weder den Operanden FILENAM=... noch die Operanden LIB=... und ELEM=... angibt, wird "programm" zusätzlich zum Namen der Datei, die das Programm enthält. Falls man nur den Operanden LIB= angibt, so wird "programm" zusätzlich zum Namen des Elements, das das Programm enthält.

VERSION
=version
: gibt eine Zeichenkette von 10 Zeichen an, die als Versionsbezeichnung des gebundenen Programms in den Phasen-Header der Programmdatei eingetragen werden soll.
Diese Versionsnummer wird zusammen mit dem Programmnamen beim Laden des Programms in der Meldung

```
% BLS0500 PROGRAM 'program', VERSION 'version' LOADED
```

ausgegeben. Wird das Programm als Element einer Programmbibliothek gespeichert, so ist "version" gleichzeitig die Versionsbezeichnung des Bibliothekselementes. Wird der Operand nicht angegeben, werden die ersten 10 Zeichen der in der Programmbibliothek eingetragenen Versionsbezeichnung angenommen. Fehlt die Angabe in der Programmbibliothek, wird keine Versionsnummer eingetragen.

COPYRIGHT
=(name,jahr)
: "name" bezeichnet eine Zeichenkette von bis zu 12 alphanumerischen Zeichen einschließlich "." und "-".
"jahr" bezeichnet eine Zahl von bis zu 4 Stellen. Die Operandenwerte werden in das Programm eingetragen und beim Aufruf des Programms mit ELDE innerhalb der Meldung BLS0500 oder BLS0551 ausgegeben. Sind für "name" nur Leerzeichen angegeben, wird keine Copyright-Information in das Programm eingetragen. Beim Aufruf des Programms mit ELDE wird dann keine Meldung ausgegeben.

  Standardwert für "name" ist der Wert des Klasse-2-Systemparameters BLSCOPYR, der bei der Systeminstallation festgelegt wird (siehe Handbuch "Systeminstallation" [4]), und für "jahr" die gegenwärtige Jahreszahl.

FILENAM
=datei
: legt den Namen der PAM-Datei fest, in die der Binder das gebundene Programm schreiben soll. Ist eine Datei namens "datei" bereits vorhanden, wird das gebundene Programm dort eingetragen und ihr alter Inhalt zerstört. Sonst richtet der Binder eine neue Datei unter diesem Namen ein (siehe Bild 16).

SHARE

=Y veranlaßt den Binder, falls er für das gebundene Programm eine neue Datei aufbaut, diese als mehrbenutzbar (SHARE=YES im Katalogeintrag) zu markieren.

=N erzeugt die neue Datei nicht mehrbenutzbar, d.h. Benutzer mit einer Benutzerkennung, die von der des Eigentümers abweicht, sind vom Zugriff ausgeschlossen.

XS-CHECK

=Y Jede 24-Bit-Adresse wird durch die Meldung LNK0218 angezeigt.

=N Es findet keine Überprüfung der Adressen statt.

LIBRARY

=bibliothek gibt den Namen einer Programmbibliothek an, in der das Programm gespeichert werden soll. Existiert die Bibliothek vor dem Aufruf von TSOSLNK noch nicht, so wird sie eingerichtet.

ELEMENT Name und Versionsbezeichnung des Elements, in dem das Programm in der Programmbibliothek gespeichert werden soll.
Für den Namen und die Versionsbezeichnung muß der Benutzer die Namenskonventionen von LMS beachten (siehe Handbuch "LMS" [3]). Werden die Namenskonventionen nicht eingehalten, kann evtl. LMS das Element nicht mehr bearbeiten.

=element gibt den Namen des Elementes an, in dem das Programm gespeichert werden soll. Der Elementname darf bis zu 41 Zeichen lang sein. Programme werden unter dem Elementtyp C gespeichert.

version gibt die Versionsbezeichnung des Elementes "element" in der Programmbibliothek an. Die Versionsbezeichnung darf bis zu 24 Zeichen lang sein. Die explizite Angabe der Versionsbezeichnung "@" ist nicht erlaubt. Ist der Operand nicht angegeben, wird die Versionsbezeichnung des Operanden VERSION angenommen. Fehlt der Operand VERSION, wird der Standardwert für die Versionsbezeichnung bei Programmbibliotheken angenommen (siehe Handbuch "LMS" [3]).

Sind weder der Operand FILENAM noch die Operanden LIBRARY und ELEMENT angegeben, wird das Programm in einer PAM-Datei mit dem Namen "programm" gespeichert. Ist nur der Operand LIBRARY angegeben, wird "programm" als Elementname angenommen.

P[AM]-K[EY] legt das Format der PAM-Datei fest, in die der Binder das gebundene Programm schreiben soll.
Standardwert ist der Wert des Klasse-2-Systemparameters BLKCTRL, der bei der Systeminstallation festgelegt wird (siehe Handbuch "Systeminstallation" [4]).
Ist BLKCTRL nicht angegeben, wird "Y" angenommen.

=Y Das gebundene Programm wird in eine PAM-Datei mit PAM-Schlüssel geschrieben (keine Pamkey-Eliminierung).

=N Das gebundene Programm wird in eine PAM-Datei ohne PAM-Schlüssel geschrieben (Pamkey-Eliminierung). Dieser Wert ist ungültig, wenn gleichzeitig der Operand LIBRARY angegeben wird.

Hinweis

- Wird das gebundene Programm in eine Datei ohne PAM-Schlüssel geschrieben (PAMKEY=NO oder BLKCTRL=NONKEY), kann es nicht vom Lader ELDE der Versionen ≤10.0A geladen werden.
- Bei einer Platte ohne PAM-Schlüssel wird die Erzeugung einer Programmdatei mit dem PAM-Schlüssel 1 abgewiesen (Meldung LNK0270).

ARM[ODE]-C[HECK]
legt fest, daß die CSECT's auf Attribute AMODE und/oder RMODE ungleich ANY überprüft werden.

=IGN[ORE] Es wird nicht geprüft, ob die CSECT's die Attribute AMODE und/oder RMODE ungleich ANY enthalten.

=WARN[ING] Für jede CSECT, die Attribute AMODE und/oder RMODE ungleich ANY enthält, wird die Warnungsmeldung LNK0065 ausgegeben. Anschließend wird der Binderlauf fortgesetzt.

=ABORT Für jede CSECT, die Attribute AMODE und/oder RMODE ungleich ANY enthält, wird die Warnungsmeldung LNK0065 ausgegeben. Anschließend wird der Binderlauf fehlerhaft beendet.

IDA

=Y vereinbart, daß der Benutzer beim Testen mit der Dialogtesthilfe die symbolischen Namen des Quellprogramms verwenden kann. Der Benutzer muß dafür gesorgt haben, daß beim Übersetzen des Quellprogramms die entsprechenden Compilerparameter gesetzt waren (siehe Benutzerhandbücher der Compiler).

=N gibt an, daß keine Namen des Quellprogramms beim Programmtest benutzt werden sollen.

SYMTEST — Dieser Operand wird zum Testen mit AID benötigt.
(Zu AID siehe Handbuch "AID" [10]).

=ALL — SYMTEST=ALL erlaubt die Anwendung von symbolischen Adressen beim Testen des Programms mit AID Kommandos. Symbolisch können nur Programme getestet werden, für die beim Übersetzen symbolische Informationen erzeugt wurden (siehe Benutzerhandbücher der Compiler).

=N — Das Programm kann nicht symbolisch getestet werden.

=MAP — Aus den symbolischen Informationen erzeugt der Binder eine Objekt-Strukturliste, die mit in den Lademodul geschrieben wird. Mit dieser Information ist ein minimales symbolisches Testen möglich, d.h. es können Segmentnamem angesprochen werden. Ausgewertet wird die Objekt-Strukturliste auch beim IDA-Kommando
DISPLAY %MAP.
Für das volle symbolische Testen muß AID weitere symbolische Informationen noch nachladen.

LET

=Y — vereinbart für den Stapelbetrieb, daß gebunden werden soll, auch wenn der Binder unlösbare Externverweise entdeckt.

=N — legt für den Stapelbetrieb fest, daß der Binderlauf abzubrechen ist, sobald der Binder auf unlösbare Externverweise stößt.

Im Dialogbetrieb nimmt sowohl für LET=Y als auch für LET=N der Binder den Dialog wieder auf, sobald er unlösbare Externverweise entdeckt, so daß der Benutzer weitere Binderanweisungen eingeben kann, z.B. die BIND-, RESOLVE-, STOP-Anweisung.

CLASS

=2 — veranlaßt die Erzeugung eines seitenwechselbaren Programms (Klasse-II-Programm).

=E — muß man angeben, wenn ein Systemprogramm, d.h. ein Teil des BS2000-Organisationsprogramms, gebunden werden soll.

Ein Programm, das mit CLASS=E gebunden wurde, kann nicht mit dem Kommando START-PROGRAM oder LOAD-PROGRAM geladen werden. Dieses Programm hat nur Verwendung beim System-STARTUP.

COREIM

=Y veranlaßt den Binder, das Programm bereits so aufzubauen, wie es nach dem Laden im Speicher steht (Speicherabbildformat).

=N hat zur Folge, daß der Binder Relativierungsinformationen in das Programm übernimmt, das Programm also nicht als Speicherabbild erzeugt wird.

Hinweis

Das Größenverhältnis bei den beiden Programmtypen ist abhängig vom internen Aufbau des Programms.
Bei einem Programm mit vielen kleinen Lücken (DS-Anweisungen) kann das Speicherabbildformat *kleiner* als das Format mit Relativierungsinformation sein.
Bei einem Programm mit wenigen großen Lücken (DS-Anweisungen) kann das Speicherabbildformat *größer* als das Format mit Relativierungsinformation sein.

CONTROL

=Y hat nur bei segmentierten Programmen eine Bedeutung, falls V-Konstanten bzw. die Makroaufrufe CALL oder SEGLD darin verwendet werden. Der Binder baut dann einen Steuermodul auf, der das Nachladen der Überlagerungssegmente während des Programmlaufs steuert und bindet ihn in das Grundsegment ein.

=N verhindert den Aufbau eines Überlagerungssteuermoduls. Das Programm muß dann zum Nachladen den Makroaufruf LPOV verwenden.

XCAL

=Y bezieht sich auf segmentierte Programme und veranlaßt den Binder, Externverweise zwischen unabhängigen Segmenten zu befriedigen, d.h. der Benutzer nimmt evtl. auftretende Fehler in Kauf.

=N gibt an, daß der Binder Externverweise nur meldet und sie nicht befriedigt, falls er Verweise zwischen voneinander unabhängigen Segmenten feststellt.

LOADPT

=adresse — legt eine virtuelle Adresse (sedezimal: X'...') fest, an die der Lader ein Klasse-II- oder ein Klasse-E-Programm (siehe Operand CLASS) laden soll. Die Adresse muß an einer Seitengrenze liegen, d.h. ein Vielfaches von 4096 (X'1000') sein. Wenn die Adresse nicht an einer Seitengrenze liegt, wird sie automatisch auf Seitengrenze ausgerichtet. Wird LOADPT angegeben und die Operanden COREIM=Y und CLASS=E gesetzt, dann wird das Programm im Speicherabbildformat erstellt.
Fehlt dieser Operand, dann wird die virtuelle Adresse X'000000' genommen.

=*XS — legt die Ladeadresse des Programmes im Adreßraum oberhalb 16Mbyte fest.
Die CSECT muß das Attribut RMODE=ANY haben.

Warnung
Beim Binden eines Programms, das auf einem Vektorrechner ablaufen sol, dürfen nur die virtuellen Adressen 0 oder 16Mbyte angegeben werden. Andere Werte führen zu einem Fehler beim Starten des Programms.

MAX

=proglänge — Dezimalwert, der dem Lader die Länge des Programms vorgibt, falls die tatsächliche Länge nicht größer als "proglänge" ist.

ADD

=länge — Dezimalwert, der zur errechneten Programmlänge addiert wird, so daß der Lader zusätzlichen Speicherplatz für das Programm anfordert.

ENTRY=programmabschnitt
START=programmabschnitt

legt den Namen eines Programmabschnitts fest, an dessen Anfangsadresse nach dem Laden der Programmlauf beginnen soll. Bei segmentierten Programmen muß dieser Abschnitt zum Grundsegment gehören.

ENTRY=einsprungstelle
START=einsprungstelle

gibt den Namen einer Einsprungstelle an, d.h. den Namen einer Adresse, an der nach dem Laden der Programmlauf beginnen soll. Bei segmentierten Programmen muß die Einsprungstelle im Grundsegment liegen.
Läßt man den ENTRY- oder START-Operanden weg, wird die Startadresse des Programms aus dem END-Satz des ersten eingegebenen Bindemoduls entnommen. Fehlt dort eine derartige Angabe, startet das Programm mit dem 1. Byte des 1. Bindemoduls.

PL1

=Y vereinbart die Verarbeitung simulierter Q-Konstanten (siehe Handbuch "Assembler" [2]), d.h. der Binder weist allen Externverweisen, die mit IQ... anfangen, keine Adresse, sondern einen Pseudoregister-Offsetwert zu.

=N verhindert die Verarbeitung simulierter Q-Konstanten.

PR

=Y Der Binder gibt sämtliche Pseudoregister nach Modulen geordnet in die Systemdatei SYSLST aus, und zwar mit den ihnen zugewiesenen Offsetwerten.

=N beschränkt die Ausgabe auf eine Liste der unbefriedigten Pseudoregister.

XDSEC

=Y vereinbart die Ausgabe auf eine Liste in die Systemdatei SYSLST, in der sämtliche Verweise von externen Pseudoabschnitten nach Modulen geordnet aufgeführt sind. Zu jedem Externverweis gibt der Binder dabei an, mit Hilfe welcher Bindemodule er ihn befriedigen konnte.

=N beschränkt die Angabe auf die Verweise externer Pseudoabschnitte, die der Binder nicht befriedigen kann.

MAP

=Y vereinbart die Erzeugung einer Programmübersicht, die Informationen über Größe, Länge und Adressen der eingebundenen Bindemodule enthält. Der Binder gibt sie in die Systemdatei SYSLST aus.

=N unterdrückt die Erzeugung der Programmübersicht.

XREF

=Y gibt an, daß der Binder eine Querverweisliste des gebundenen Programms erzeugen soll. Die Liste enthält die Adressen aller Programmabschnitte, Einsprungstellen und Externverweise sowie Angaben über die Befriedigung der Externverweise. Die Angabe erfolgt nach SYSLST.

=N verhindert die Ausgabe der Querverweisliste.

LIST

=Y Nur wirksam im Dialogbetrieb.
Der Binder gibt die Anweisungen und eine Kurzform der Programmübersicht des gebundenen Programms nach SYSOUT aus. Der Operand wirkt sich bereits auf die PROGRAM-Anweisung aus, in der er steht.

=N bewirkt, daß der Binder die eingegebenen Anweisungen nicht protokolliert und keine Kurzform der Programmübersicht nach SYSOUT gibt.

SYSLST

=Y veranlaßt den Binder, in die Systemdatei SYSLST auszugeben.

=N unterdrückt die Ausgabe nach SYSLST.

UNSAT

=Y veranlaßt den Binder, eine Liste aller unbefriedigten Externverweise auszugeben.

=N Die Liste der unbefriedigten Externverweise wird nicht ausgegeben.

=S veranlaßt den Binder, eine alphabetisch sortierte Liste der unbefriedigten Externverweise auszugeben.

WUNSAT

=Y veranlaßt den Binder, eine Liste der unbefriedigten Externverweise auszugeben, deren ESD-Typ X'F9' oder X'F2' bzw. X'F4' ist und deren Namen mit I$ beginnen.

ESD-Typen

X'F2': ER: Externe Verknüpfungsadresse (EXTRN-Anweisung)
X'F4': VC: Externe Verknüpfungsadresse (V-Konstante)
X'F9': WX: Bedingte Extern-Adresse (WXTRN-Anweisung)

=N — Es wird keine Liste dieser unbefriedigten Externverweise ausgegeben.

Ist der Operand UNSAT=N gesetzt, wird die Angabe im Operanden WUNSAT ignoriert, d.h. es werden keinerlei unbefriedigte Externverweise ausgegeben.

SORT

=Y — erzeugt eine zusätzliche Liste, in der die Namen aller Programmabschnitte und Einsprungstellen alphabetisch aufsteigend sortiert enthalten sind. Die Liste enthält für jeden Eintrag:

- den Namen des Programmabschnitts (CSECT NAME) oder der Einsprungstelle (ENTRY NAME).
- die Anfangsadresse in sedezimaler und dezimaler Form (ADDRESS).
- eine Kennzeichnung als Programmabschnitt (CSECT) oder Einsprungstelle (ENTRY).
- den Namen des Moduls (MODULE CONTAINING) und/oder des Segments (SEGMENT ENTRY), in dem der Programmabschnitt oder die Einsprungstelle liegen.

Die Ausgabe dieser sortierten Namensliste ist unabhängig von anderen Ausgabelisten von TSOSLNK, insbesondere unabhängig von der Ausgabe der Externverweisliste.

=N — Eine alphabetische Liste mit den Namen der Programmabschnitte und Einsprungstellen wird nicht ausgegeben.

CMAP

=ALL — Es wird eine vollständige Programmübersicht und eine Querverweisliste ausgegeben.

=NO — Unterdrückt die Ausgabe der vollständigen Programmübersicht und der Querverweisliste.

=(...) — Maximal dürfen nur 6 der nachfolgenden Optionen angegeben werden, die in runde Klammern einzuschließen sind.

=CSECT — Eine Liste der Bindemodule und der Programmabschnitte wird ausgegeben.

=NOCSECT — Es wird keine Liste der Programmabschnitte ausgegeben.

=ENTRYS Eine Liste der Programmabschnitte und der Einsprungstellen wird ausgegeben.

=NOENTRYS Es wird keine Liste der Einsprungstellen ausgegeben.

=COMMONS Eine Liste der Module und COMMON-Bereiche wird ausgegeben.

=NOCOMMONS
Es wird keine Liste der COMMON-Bereiche ausgegeben.

=XREF Eine Liste der Querverweise und der Module wird ausgegeben.

=NOXREF Es wird keine Querverweisliste ausgegeben.

=EJECT Bei jedem neuen Segment wird ein Seitenvorschub gemacht.

=NOEJECT Es wird kein Seitenvorschub gemacht.

=MODULES Eine Liste der Module wird ausgegeben.

Eine vollständige Programmübersicht und eine Querverweisliste werden unabhängig von der Angabe CMAP ausgegeben:
- in einer BS2000-Version ≥9.0 oder
- in einer BS2000-Version <9.0 und einer Ladeadresse oberhalb 16 Mbyte.

LINE
=number legt die Anzahl der Zeilen pro Seite bei Ausgabe von Listen in die Systemdatei SYSLST fest.
Für "number" dürfen der Wert 0 und alle Werte ≥30 angegeben werden. Andere Werte werden abgewiesen.
Standardwert ist 54.
Wenn der Wert 0 angegeben ist, wird kein Seitenvorschub durchgeführt.

LINEAR
=Y Während des Binderlaufs werden vorhandene OVERLAY-Anweisungen ignoriert.

=N OVERLAY-Anweisungen werden ausgeführt.

Beispiel

1. `PROGRAM ABC,FILENAM=FOR.TEST.ABC,XREF=Y`
   Ein Programm ABC wird in die Datei FOR.TEST.ABC geschrieben und dabei neben einer Programmübersicht (MAP=Y, Standardfall) eine Querverweisliste (XREF=Y) erzeugt.

2. `PROG TEST,LOADPT=X'3000',IDA=Y`
   Das Programm TEST wird in die Datei TEST ausgegeben, d.h. der Programmname wird auch als Dateiname verwendet. Als Ladeadresse wird Seite 3 des Klasse-6-Benutzerspeichers vereinbart. Das vom Übersetzer erzeugte Internadreßbuch soll in den Lademodul übernommen werden (IDA=Y).

RENAME-Anweisung

Sie veranlaßt den Binder, den Namen eines Programmabschnitts, einer Einsprungstelle oder eines Externverweises durch einen anderen Namen zu ersetzen. Der Binder sucht nach dem zu ändernden Namen nur in den Bindemodulen, die der Anweisung in der Eingabe folgen. Jede RENAME-Anweisung wirkt nur einmal.

Operation	Operanden
RENAME	{programmabschnitt / einsprungstelle / externverweis},name

programmabschnitt
: vereinbart den Namen eines Programmabschnitts (CSECT), der zu ändern ist.

einsprungstelle
: gibt den Namen einer Einsprungstelle (ENTRY) an, der ersetzt werden soll.

externverweis
: legt den Namen eines Externverweises (EXTRN, WXTRN, V-Konstante) fest, den der Binder ändern soll.

name
: soll den Namen ersetzen, den man im 1. Operanden angegeben hat.

Beispiel

a)
```
RENAME EXT,ABC
INCLUDE MODA,*
```

Wenn der Modul MODA eingebunden wird, prüft der Binder, ob MODA einen Programmabschnitt, eine Einsprungstelle oder einen Externverweis mit dem Namen EXT enthält. Den 1. Namen EXT, den er findet, benennt er in ABC um und löscht anschließend die RENAME-Anweisung.

Findet er den Namen EXT nicht, so speichert er die RENAME-Anweisung, d.h. sie kann sich auf alle später eingelesenen Bindemodule beziehen.

b) Bindemodul MODE und MODF sollen Einsprungstellen namens EXA enthalten. Der Name EXA soll in den Modulen MODA, MODB und MODC nicht vorkommen.

```
RENAME   EXA,ABC
INCLUDE  (MODA,MODB,MODC),*
INCLUDE  (MODE,MODF)
```

c) Die temporäre EAM-Bindemoduldatei der laufenden Task soll mehrere Bindemodule enthalten. Der 1. Modul besitzt eine Einsprungstelle namens XXX, der 2. ebenfalls und der 3. Modul einen Externverweis XXX.

```
INCLUDE  (MODA,MODB),BIB ———————————————————————————— (01)
RENAME   MODA,XYZ
RENAME   XXX,AB
RENAME   XXX,XXX
RENAME   XXX,AB
INCLUDE  * ——————————————————————————————————————————— (02)
INCLUDE  (MODA,MODC),BIB ———————————————————————————— (03)
```

(01) Die nachfolgenden RENAME-Anweisungen wirken sich auf diese INCLUDE-Anweisung nicht aus.

(02) Der Binder benennt die Einsprungstelle XXX im 1. Modul aus der temporären EAM-Bindemoduldatei in AB um. Im 2. Modul bleibt der Name der Einsprungstelle XXX unverändert. Im 3. Modul der temporären EAM-Bindemoduldatei wird der Externverweis XXX in AB umbenannt.

(03) Der Name MODA soll auch der Name des 1. Programmabschnitts (CSECT) im Modul MODA sein. Der Binder ändert diesen Namen wegen der ersten RENAME-Anweisung in XYZ.

REP-Anweisung

Sie veranlaßt den Binder, einen Bereich im angegebenen Bindemodul mit einem anderen Text zu überschreiben. Die bearbeiteten Anweisungen protokolliert er in der Systemdatei SYSLST.

Mit dem Bibliotheksprogramm LMS (siehe Handbuch "LMS" [3]) kann man in einen Bindemodul REP-Sätze eintragen lassen, die der Binder wie REP-Anweisungen behandelt.

Operation	Operanden
REP	{ adresse_X'sedezimal-daten'_modul adresse_[C]'zeichen'_modul } [_kommentar]

adresse — bezeichnet die sedezimale Adresse desjenigen Byte im Bindemodul "modul", ab dem der Binder den Text "sedezimal-daten" schreiben soll. Die Adresse darf höchstens 8 Zeichen lang sein.

sedezimal-daten oder zeichen — stellt den Text (Daten oder Befehle) dar, der den im Bindemodul "modul" enthaltenen Text ersetzen soll. Er kann in sedezimaler Form X'...' oder als Zeichenfolge C'...' angegeben werden.
Sedezimal darf der Text bis zu 32 Zeichen lang sein, im Zeichenformat bis zu 16 Zeichen.
Das 'C' darf auch weggelassen werden.

modul — gibt den Bindemodul an, in dem Text ersetzt werden soll. Gibt es mehrere Module dieses Namens, gilt die REP-Anweisung für den zuerst gefundenen Modul.

kommentar — bezeichnet eine beliebige Zeichenfolge, die in die REP-Sätze als Kommentar eingefügt werden soll.

Zwischen den Operanden dürfen beliebig viele Leerzeichen stehen, aber kein Komma.

Hinweis

Wenn ein Bindemodul mehrfach in einen Lademodul eingebunden wird, werden die REPs nur beim ersten Einbinden des Bindemoduls verarbeitet.

RESOLVE-Anweisung

Sie gibt eine Bindemodulbibliothek an, aus deren Bindemodulen der Binder versuchen soll, unbefriedigte Externverweise zu befriedigen (Autolink-Funktion).

Operation	Operanden
RESOLVE	[{externverweis / (externverweis,...)}],bibliothek

externverweis — gibt den Namen eines Externverweises an, den der Binder aus der angegebenen Bindemodulbibliothek befriedigen soll. Höchstens 20 Externverweise sind pro RESOLVE-Anweisung erlaubt. Wird keiner angegeben, so versucht der Binder, alle bisher unbefriedigten Externverweise aus der angegebenen Bibliothek zu befriedigen.

bibliothek — legt den Namen der Bindemodulbibliothek fest, in der vom Binder nach passenden Namen von Programmabschnitten oder Einsprungstellen gesucht werden soll, um unbefriedigte Externverweise zu befriedigen.

Hinweis

- Werden mehrere RESOLVE- und EXCLUDE-Anweisungen angegeben, so wird eine Reihenfolge durch die Nennung des Bibliotheksnamens bei der Eingabe bestimmt. Dabei ist es gleichgültig, ob der Name zuerst in einer RESOLVE- oder in einer EXCLUDE-Anweisung auftritt.
- Zuerst bearbeitet der Binder die RESOLVE-Anweisungen, in denen Externverweise explizit aufgeführt sind, und zwar die Bibliotheken in der Reihenfolge, in der sie in den Anweisungen genannt wurden (von vorne nach hinten). Dann erst durchsucht der Binder die Bibliotheken der übrigen RESOLVE-Anweisungen, diesmal in umgekehrter Nennungsreihenfolge (von hinten nach vorne). Dabei läßt er die Externverweise weg, die in EXCLUDE-Anweisungen explizit genannt sind.
- Regel: Widersprechen sich RESOLVE- und EXCLUDE-Anweisungen, so gilt die zuletzt eingelesene.
- Der Binder prüft, ob er die angegebene Bindemodulbibliothek eröffnen kann (OPEN). Bei einem OPEN-Fehler wird die Warnungsmeldung LNK0254 ausgegeben.

Beispiel

A, B, C und D sollen die Namen von Externverweisen sein, LIBA, LIBB, LIBC und LIBD die Namen von Bindemodulbibliotheken.

Zeile Nr.	Anweisung
1	RESOLVE ,LIBA
2	RESOLVE ,LIBB
3	EXCLUDE (A,B),LIBB
4	RESOLVE C,LIBA
5	RESOLVE ,LIBC
6	RESOLVE ,LIBA
7	RESOLVE D,LIBD
8	RESOLVE B,LIBB

Der Autolink-Mechanismus (siehe Seite 18ff) des Binders arbeitet in der Reihenfolge:

1) Beim Einlesen werden die Namen der Bibliotheken in der Reihenfolge gespeichert, in der sie in RESOLVE- oder EXCLUDE-Anweisungen auftreten; in diesem Beispiel LIBA, LIBB, LIBC, LIBD.

2) Dann bearbeitet der Binder die Anweisungen, in denen Externverweise explizit aufgeführt sind, d.h. die Zeilen 3, 4, 7 und 8, und zwar von hinten nach vorne, bezogen auf die gespeicherten Bibliotheksnamen:

   Zuerst wird versucht, B aus LIBB zu befriedigen. Die EXCLUDE-Anweisung für B ist unwirksam, da sie früher als Zeile 8 eingelesen wurde. Falls Externverweis D dann noch unbefriedigt ist, versucht der Binder, ihn mit Hilfe von LIBD zu lösen.

3) Als nächstes bearbeitet der Binder die Anweisungen, in denen keine Externverweise angegeben sind, d.h. die Zeilen 1, 2, 5 und 6 in umgekehrter Reihenfolge (von hinten nach vorne) bezüglich der Bibliotheksnamen.

   Bis zu diesem Zeitpunkt noch unbefriedigte Externverweise werden also in der Bibliothek LIBC gesucht, dann (mit Ausnahme von A und B) in LIBB, danach noch in LIBA (mit Ausnahme von Externverweis C).

4) Schließlich versucht der Binder, alle bis dahin unbefriedigt gebliebenen Externverweise aus der TASKLIB aufzulösen.

SHARE-Anweisung

Sie markiert neu erstellte Lademoduldateien als mehrfach benutzbar und wirkt wie der Operand SHARE=Y in der PROGRAM-Anweisung. Sie wird beim Modulbinden (siehe MODULE-Anweisung, Seite 98ff) mit einer Fehlermeldung zurückgewiesen.

Operation	Operanden
SHARE	[beliebiger Kommentar]

STOP-Anweisung

Sie beendet den Binderlauf sofort und kann jederzeit eingegeben werden.

Operation	Operanden
STOP	[beliebiger Kommentar]

Beispiel

```
/START-PROGRAM FROM-FILE=$TSOSLNK
%  BLS0500 PROGRAM 'TSOSLNK', VERSION '21.0D00' OF '90-05-10' LOADED.
*PROG STONAM,FILENAM=STONAM.LADE
*INCLUDE EINXEINS,LMS.BIBL
*STOP
%  LNK0050 *** TSOSLNK RUN ABORTED ***
%  EXC0732 ABNORMAL PROGRAM TERMINATION. ERROR CODE 'NRT0101': /HELP NRT0101,INF=D
/
```

TRAITS-Anweisung

Sie vereinbart beim Binden Merkmale für Programmabschnitte in zu erzeugenden Lade- bzw. Bindemodulen, wenn mit LINK-SYMBOLS *HIDE oder *KEEP gearbeitet wird. Sie wirkt ab dem Eingabezeitpunkt bis zum Ende des Binderlaufs, wenn man sie nicht mit einer neuen TRAITS-Anweisung ändert.

Operation	Operanden
TRAITS	[programmabschnitt] [,READONLY={Y|N}][,PAGE={Y|N}]
	[,ALIGN=nummer]
	[,AMODE={24|31|ANY}]
	[,RMODE={24|ANY}]

programmabschnitt

legt den Namen eines Programmabschnitts (CSECT) fest, auf den sich die übrigen Operanden beziehen.

Fehlt der Operand "programmabschnitt", so gilt die TRAITS-Anweisung für alle Programmabschnitte (CSECTs), die nicht in anderen TRAITS-Anweisungen angegeben sind.
Eine TRAITS-Anweisung ohne Angabe eines Programmabschnitts wird durch eine neue TRAITS-Anweisung ungültig.

Ausrichtung eines Großmoduls
Entspricht der erste Programmabschnitt (CSECT) dem in der MODULE-Anweisung angegebenen Modulnamen, bezieht sich der Wert des Operanden PAGE auf diese erste CSECT, wobei alle übrigen Operanden der TRAITS-Anweisung ignoriert werden.

READONLY

=Y gibt an, daß der betreffende Abschnitt zur Ablaufzeit des Programms nur gelesen werden soll.
Bei einem Vektorrechner kann dieser Operand nicht angegeben werden. Falls der Operand angegeben ist, wird dieses Attribut zur Ablaufzeit des Programms ignoriert.

=N erlaubt während des Programmlaufs auch das Schreiben in dem angegebenen Programmabschnitt.

Fehlt der Operand READONLY, so behält der Programmabschnitt das Merkmal, das ihm bei der Übersetzung zugewiesen wurde.

PAGE

=Y bestimmt, daß der betreffende Programmabschnitt auf Seitengrenze ausgerichtet werden soll, d.h. seine Ladeadresse wird ein Vielfaches von dezimal 4096 bzw. sedezimal 1000.

=N läßt Seitengrenzen unberücksichtigt, d.h. der Programmabschnitt beginnt bei der nächsten Doppelwortadresse, die sich beim Binden ergibt.

Fehlt der Operand PAGE, so bleibt das Merkmal, das dem Programmabschnitt bei der Übersetzung zugewiesen wurde, unverändert erhalten.

ALIGN

=nummer bestimmt die minimale Ausrichtung des Programmabschnitts. "nummer" kann jede Zweierpotenz zwischen 8 und 4096 sein. Die kleinste Ausrichtung kann auf Doppelwortgrenze, die größte auf Seitengrenze erfolgen.

AMODE

=24 Programmabschnitte können im 24-Bit-Modus adressiert werden. Bei RMODE=ANY kann AMODE=24 nicht angegeben werden.

=31 Programmabschnitte können im 31-Bit-Modus adressiert werden.

=ANY Programmabschnitte können entweder im 24- oder im 31-Bit-Modus adressiert werden.

RMODE

=24 legt eine Adresse unterhalb 16Mbyte fest, ab der Programmabschnitte geladen werden.

=ANY Programmabschnitte können ab einer Adresse unterhalb oder oberhalb 16Mbyte geladen werden. Bei AMODE=24 kann RMODE=ANY nicht angegeben werden.

Hinweis

– Eine Seite im Speicher kann nicht teilweise ’nur lesbar’ und teilweise beschreibbar sein. Diese Eigenschaft kann sich auf die für den Lademodul erzeugten Adressen auswirken: Ist irgendein Programmabschnitt als ’nur lesbar’ gekennzeichnet, so muß ihm der Binder eine Anfangsadresse geben, die sicherstellt, daß der Abschnitt beim Laden in eine ’nur lesbare’ Seite kommen kann.

  Besteht z.B. die Eingabe aus 3 Programmabschnitten A, B und C (jeder kleiner als 4096 Byte), von denen B als ’nur lesbar’ markiert ist, so vergibt der Binder folgende Adressen:

  A – Adresse $000000_{(16)}$ in einer beschreibbaren Seite
  B – Adresse $001000_{(16)}$ in einer nur lesbaren Seite
  C – Adresse $002000_{(16)}$ in einer beschreibbaren Seite

  A^x80ist der 1. Abschnitt des Programms, weswegen er auf Seitengrenze ausgerichtet ist. Abschnitt B muß eine andere Seite als A belegen, denn er ist als ’nur lesbar’ gekennzeichnet. Programmabschnitt C wiederum muß eine andere Seite als B belegen, denn er soll beschreibbar sein.

– Gibt man TRAITS- und RENAME-Anweisungen zusammen ein, so muß sich die TRAITS-Anweisung auf den neuen Namen des Programmabschnitts beziehen. Der Binder bearbeitet nämlich zuerst die RENAME-Anweisungen für den betreffenden Bindemodul und erst dann dessen TRAITS-Anweisungen. Die Reihenfolge, in der RENAME- und TRAITS-Anweisungen eingegeben werden, ist deshalb nicht von Bedeutung.

– Änderung der Attribute AMODE und RMODE sind nur zulässig, wenn sie gegenüber den Attributen in der CSECT eingeschränkt werden. Treten Konflikte zwischen den Attributen in der CSECT und der TRAITS-Anweisung auf, übergeht der Binder die Attribute in der TRAITS-Anweisung.

– Die TRAITS-Anweisung gilt auch für COMMON-Bereiche.

– Übergeht der Binder die Angaben in einer TRAITS-Anweisung, gibt er am Ende des Binderlaufs die Warnungsmeldung LNK0232 aus.

Beispiel

```
a) INCLUDE  (A,B,C),LIB ———————————————————————————— (01)
   TRAITS   A,READONLY=N,PAGE=Y
   INCLUDE  (B,A,X),LIB ———————————————————————————— (02)
   TRAITS   A,READONLY=Y ———————————————————————————— (03)
```

(01) Diese Anweisung bearbeitet der Binder zuerst. Er berücksichtigt dabei also keine der nachfolgenden TRAITS-Anweisungen.

(02) Auf die 2. INCLUDE-Anweisung wirkt sich die vorausgegangene TRAITS-Anweisung aus, d.h. Programmabschnitt A in Bindemodul A wird als beschreibbar gekennzeichnet und auf Seitengrenze ausgerichtet

Für Testzwecke kann es z.B. notwendig werden, einen Programmabschnitt als beschreibbar zu kennzeichnen, da die Dialogtesthilfe IDA bei 'nur lesbaren' Abschnitten nicht eingesetzt werden kann.

(03) Folgen auf diese TRAITS-Anweisung weitere Programmabschnitte namens A, markiert der Binder sie als 'nur lesbar' und beläßt das bei der Übersetzung zugewiesene Seitenmerkmal.

b) Modul A soll einen Programmabschnitt namens ABC enthalten.

```
RENAME   ABC,X
TRAITS   X,PAGE=Y
INCLUDE  (A,L),LIB
```

Bevor der Binder den Bindemodul A einbindet, sucht er evtl. vorhandene, bis dahin unbearbeitete RENAME-Anweisungen, die sich auf A beziehen und bearbeitet sie. Er gibt also dem Programmabschnitt ABC den neuen Namen X. Anschließend berücksichtigt der Binder die TRAITS-Anweisungen, findet eine, die sich auf Programmabschnitt X bezieht und richtet X auf Seitengrenze aus.

Man kann die RENAME- und die TRAITS-Anweisung auch in umgekehrter Reihenfolge eingeben, ohne daß sich das Ergebnis ändert.

XCAL-Anweisung

Sie erlaubt bei segmentierten Programmen die Befriedigung von Externverweisen zwischen unabhängigen Segmenten. Sie wirkt wie der Operand XCAL=Y in der PROGRAM-Anweisung. Sie wird beim Modulbinden (siehe MODULE-Anweisung, Seite 98ff) mit einer Fehlermeldung zurückgewiesen.

Operation	Operanden
XCAL	[beliebiger Kommentar]

XREF-Anweisung

Sie veranlaßt die Erstellung einer Querverweisliste. Sie wirkt wie der Operand XREF=Y in der PROGRAM- bzw. MODULE-Anweisung.

Operation	Operanden
XREF	[beliebiger Kommentar]

Meldungen des Binders TSOSLNK

Die Meldungen des Binders TSOSLNK sind im Handbuch "Systemmeldungen" [9] aufgeführt.

Der Lader ELDE

Aufgaben des Laders

Der Lader ELDE ist eine Routine des BS2000-Organisationsprogrammes, die Lademodule eines Programms in den Speicher bringt, so daß sie dort ablaufen können. Der Lader hat also die Aufgabe, vom Binder TSOSLNK erzeugte und in einer PAM-Datei oder in einem Bibliothekselement gespeicherte Programme einzulesen. Dabei hat der Lader Relativierungsinformationen auszuwerten, falls der Lademodul zu diesem Zeitpunkt noch kein Speicherabbildformat besitzt (siehe COREIM=N in der PROGRAM-Anweisung des Binders TSOSLNK, Seite 116ff).

Der Ablaufteil des BS2000 sorgt während des Programmablaufs dafür, daß die für den Ablauf benötigten Programmteile sich jeweils im Hauptspeicher befinden (siehe Bild 17).

Bild 17 Laden von Programmen

Aufruf des Laders

Der Lader ELDE kann vom Benutzer mit verschiedenen Kommandos und Makros aufgerufen werden (siehe Bild 17):

– ELDE wird bei der Bearbeitung der Kommandos START-PROGRAM und LOAD-PROGRAM aktiviert, wenn im Kommando eine Datei oder ein Bibliothekselement genannt wird, das ein vom Binder TSOSLNK aufgebautes Programm enthält.

– ELDE wird durch den Makro LPOV (Load Program Overlay) aufgerufen, der das Nachladen eines Überlagerungssegments veranlaßt. Die Makroaufrufe CALL und SEGLD führen indirekt zur Ausführung des LPOV-Makros (siehe auch Beschreibung der Makroaufrufe auf Seite 63ff).

Segmentierte Programme (siehe Seite 44ff) müssen mit dem Binder TSOSLNK gebunden sein und werden deshalb immer mit dem Lader ELDE geladen oder nachgeladen.

Kommandos für den Aufruf des Laders

Dieser Abschnitt enthält die Beschreibung der Kommandos START-PROGRAM und LOAD-PROGRAM für den Aufruf des Laders ELDE. Im Kommandoformat und in der Operandenbeschreibung sind alle Operanden, die den dynamischen Bindelader DBL aufrufen, weggelassen. Diese Operanden sind beim DBL beschrieben (siehe Handbuch "BLS" [1], Kommando START-PROGRAM).

Syntaxbeschreibung

Die Syntax der SDF-Kommandossprache wird im folgenden in 3 Tabellen erklärt.

Tabelle 1: Metasyntax

In den Kommandoformaten werden bestimmte Zeichen und Darstellungsformen verwendet, deren Bedeutung in Tabelle 1 erläutert wird.

Tabelle 2: Datentypen

Variable Operandenwerte werden in SDF durch Datentypen dargestellt. Jeder Datentyp repräsentiert einen bestimmten Wertevorrat. Die Anzahl der Datentypen ist beschränkt auf die in Tabelle 2 beschriebenen Datentypen.

Die Beschreibung der Datentypen gilt für alle Kommandos. Deshalb werden bei den entsprechenden Operandenbeschreibungen nur noch Abweichungen von Tabelle 2 erläutert.

Tabelle 3: Zusätze zu Datentypen

Zusätze zu Datentypen kennzeichnen weitere Eingabevorschriften für Datentypen. Die Zusätze schränken den Wertevorrat ein oder erweitern ihn. Im Handbuch werden folgende Zusätze in gekürzter Form dargestellt:

```
lower-case              low
wildcards               wild
generation              gen
version                 vers
cat-id                  cat
user-id                 user
```

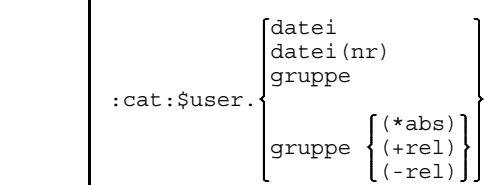
Die Beschreibung der Zusätze zu den Datentypen gilt für alle Kommandos. Deshalb werden bei den entsprechenden Operandenbeschreibungen nur noch Abweichungen von Tabelle 3 erläutert.

Tabelle 1: Metasyntax

Kennzeichnung	Bedeutung	Beispiele
GROSSBUCHSTABEN	Großbuchstaben bezeichnen Schlüsselwörter. Einige Schlüsselwörter beginnen mit *	CPU-LIMIT = JOB-REST MONJV = *NONE
GROSSBUCHSTABEN in Halbfett	Großbuchstaben in Halbfett kennzeichnen garantierte Abkürzungen der Schlüsselwörter.	**VERS**ION = *STD
=	Das Gleichheitszeichen verbindet einen Operandennamen mit den dazugehörenden Operandenwerten.	TEST-OPTIONS = AID
< >	Spitze Klammern kennzeichnen Variablen, deren Wertevorrat durch Datentypen und ihre Zusätze beschrieben wird (siehe Tabellen 2 und 3).	MINIMUM = <integer 0..254>
Unterstreichung	Der Unterstrich kennzeichnet den Standardwert eines Operanden.	TEST-OPTIONS = NONE / AID
/	Der Schrägstrich trennt alternative Operandenwerte.	TEST-OPTIONS = NONE / AID
(...)	Runde Klammern kennzeichnen Operandenwerte, die eine Struktur einleiten.	FROM-FILE = *PHASE(...)
Einrückung	Die Einrückung kennzeichnet die Abhängigkeit zu dem jeweils übergeordneten Operanden.	FROM-FILE = *PHASE(...) *PHASE(...) VERSION = STD

Kennzeichnung	Bedeutung	Beispiele
|	Der Strich kennzeichnet zusammengehörende Operanden einer Struktur. Sein Verlauf zeigt Anfang und Ende einer Struktur an. Innerhalb einer Struktur können weitere Strukturen auftreten. Die Anzahl senkrechter Striche vor einem Operanden entspricht der Strukturtiefe.	RESIDENT-PAGES = PAR(...) PAR(...) | | MINIMUM = STD
,	Das Komma steht vor weiteren Operanden der gleichen Strukturstufe.	,TEST-OPTIONS = AID ,MONJV = *NONE
list-poss(n):	Aus den list-poss folgenden Operandenwerten kann eine Liste gebildet werden. Ist (n) angegeben, können maximal n Elemente in der Liste vorkommen. Enthält die Liste mehr als ein Element, muß sie in runde Klammern eingeschlossen werden.	list-poss(2): CSECT / ENTRY

Tabelle 2: Datentypen

Datentyp	Zeichenvorrat	Besonderheiten
alphanum-name	A...Z 0...9 $,#,@	
c-string	EBCDIC-Zeichen	ist in Hochkommata einzuschließen; der Buchstabe C kann vorangestellt werden; Hochkommata innerhalb des c-string müssen verdoppelt werden.
full-filename	A...Z 0...9 $,#,@ Bindestrich Punkt	Eingabeformat:  :cat: wahlfreie Angabe der Katalogkennung; Zeichenvorrat auf A...Z und 0...9 eingeschränkt; max. 4 Zeichen; ist in Doppelpunkte einzuschließen; Standardwert ist die Katalogkennung, die der Benutzerkennung laut JOIN-Eintrag zugeordnet ist. $user. wahlfreie Angabe der Benutzerkennung; Zeichenvorrat auf A...Z und 0...9 eingeschränkt; max. 8 Zeichen; $ und Punkt müssen angegeben werden; Standardwert ist die eigene Benutzerkennung. $. (Sonderfall) System-Standardkennung

Datentyp	Zeichenvorrat	Besonderheiten
		datei Datei- oder Jobvariablenname; letztes Zeichen darf kein Bindestrich oder Punkt sein; max. 41 Zeichen; muß mindestens einen Buchstaben enthalten. #datei (Sonderfall) @datei (Sonderfall) # oder @ als erstes Zeichen kennzeichnet je nach Systemgenerierung temporäre Dateien oder Jobvariablen. datei(nr) Banddateiname nr: Versionsnummer; Zeichenvorrat ist A...Z, 0...9, $,#,@. Klammern müssen angegeben werden. gruppe Name einer Dateigenerationsgruppe (Zeichenvorrat siehe unter "datei") gruppe{(*abs) (+rel) (-rel)} Name einer Dateigeneration (Zeichenvorrat siehe unter "datei") (*abs) absolute Generationsnummer (1-9999) * und Klammern müssen angegeben werden. (+rel) (-rel) relative Generationsnummer (0-99); Vorzeichen und Klammern müssen angegeben werden.
integer	0...9,+,-	+ bzw. - kann nur erstes Zeichen sein.
text	beliebig	Das Eingabeformat ist den jeweiligen Operandenbeschreibungen zu entnehmen.

Tabelle 3: Zusätze zu Datentypen

Zusatz	Bedeutung
without	Schränkt die Angabemöglichkeiten für einen Datentyp ein.
-gen	Die Angabe einer Dateigeneration oder Dateigenerationsgruppe ist nicht erlaubt.
-vers	Die Angabe der Version (datei(nr)) ist bei Banddateien nicht erlaubt.
-cat	Die Angabe einer Katalogkennung ist nicht erlaubt.
-user	Die Angabe einer Benutzerkennung ist nicht erlaubt.

Beschreibung der Kommandos

START-PROGRAM Programm laden und starten

Dieses Kommando lädt Programme (Lademodule) in den Hauptspeicher und startet sie. Wenn der Benutzer die Programme nur laden, aber noch nicht starten möchte, kann er anstelle des Kommandos START-PROGRAM das Kommando LOAD-PROGRAM verwenden.

```
START-PROGRAM

 FROM-FILE = <full-filename 1..54 without-gen> / *PHASE(...)

   *PHASE(...)
      | LIBRARY = <full-filename 1..54 without-gen>
      |,ELEMENT = <full-filename 1..41 without-cat-user-gen-vers>
      |,VERSION = *STD / <text 1..24>

,CPU-LIMIT = JOB-REST / <integer 1..32767>

,TEST-OPTIONS = NONE / IDA / AID

,MONJV = *NONE / <full-filename 1..54 without-gen>

,RESIDENT-PAGES = PARAMETERS(...)

   PARAMETERS(...)
      | MINIMUM = STD / <integer 0..32767>
      |,MAXIMUM = STD / <integer 0..32767>

,VIRTUAL-PAGES = STD / <integer 0..32767>

,VP-SPACE = PARAMETERS(...)

   PARAMETERS(...)
      | VP-BELOW = STD / <integer 0..32767>
      |,VP-ABOVE = STD / <integer 0..32767>

,VP-WAIT = TASK-STD / <integer 0..43200>
```

FROM-FILE =
Gibt die Eingabequelle an.

FROM-FILE = <full-filename 1..54 without-gen>
Als Eingabequelle ist die Datei anzugeben, die das Programm (Lademodul) enthält, das vom Binder TSOSLNK gebunden wurde.

FROM-FILE = *PHASE(...)
Als Eingabequelle ist eine Programmbibliothek anzugeben, die das Programm (Lademodul) als Element vom Typ C enthält, das vom Binder TSOSLNK gebunden wurde.

LIBRARY = <full-filename 1..54 without-gen>
Dateiname einer Programmbibliothek, die als Eingabequelle verwendet wird.

ELEMENT = <full-filename 1..41 without-cat-user-gen-vers>
Name des Programms (Lademodul), das in der angegebenen Programmbibliothek als Element vom Typ C gespeichert ist.

VERSION =
Gibt die Elementversion an.

VERSION = *STD
Der Standardwert für die Elementversion bei Programmbibliotheken wird übernommen (siehe Handbuch "LMS" [3]).

VERSION = <text 1..24>
Explizite Angabe der Elementversion.

CPU-LIMIT =
Weist dem Programm eine in Sekunden anzugebende CPU-Zeit zu, die auch die Zeit einschließt, die während des Programmlaufs für Systemaktivitäten (z.B. Makros) benötigt wird.
Überschreitet das Programm beim Ablauf diese Zeit, wird der Benutzer im Dialogbetrieb vom System benachrichtigt. Bei Fortsetzung des Laufs wird das Programm bis zum Ende (maximal 32767 CPU-Sekunden) bearbeitet. Im Stapelbetrieb wird das Programm beendet.

CPU-LIMIT = JOB-REST
Legt die restliche CPU-Zeit für die Task fest. Die überwachte Zeit ist abhängig vom Operanden TIME im LOGON-Kommando (siehe Hinweis zum Operanden CPU-LIMIT, Seite 155).

CPU-LIMIT = <integer 1..32767>
Explizite Angabe der CPU-Zeit in Sekunden. Der angegebene Wert muß kleiner als der bei JOB-REST verwendete Wert sein; andernfalls wird das Kommando abgewiesen.
Die überwachte Zeit ist abhängig vom Operanden TIME im LOGON-Kommando (siehe Hinweis zum Operanden CPU-LIMIT, Seite 155).

TEST-OPTIONS =
Gibt an, ob symbolische Adressen im Quellprogramm beim Testen mit IDA oder AID verwendet werden dürfen.
Mit symbolischen Adressen können nur Programme getestet werden, für die beim Übersetzen LSD-Informationen bzw. ein Internadreßbuch (ISD) erzeugt wurde (siehe Benutzerhandbücher der Sprachübersetzer).

TEST-OPTIONS = NONE
Das lokale symbolische Adreßbuch bzw. das Internadreßbuch wird nicht berücksichtigt.

TEST-OPTIONS = IDA
Erlaubt die Verwendung von symbolischen Adressen des Quellprogramms beim Testen des Programms mit DTH-Kommandos.

TEST-OPTIONS = AID
Erlaubt die Verwendung von symbolischen Adressen des Quellprogramms beim Testen des Programms mit AID (siehe Handbuch "AID" [8]).

MONJV = *NONE / <full-filename 1..54 without-gen>
Name einer Jobvariable (JV), die das Programm überwachen soll. Während des Programmlaufs setzt dann das System die JV auf entsprechende Werte:

$R Programm läuft (START-PROGRAM)
$T Programm erfolgreich beendet
$A Programm fehlerhaft beendet

Bei Angabe *NONE wird das Programm nicht mit einer JV überwacht.

Hinweis
Dieser Operand steht nur dem Benutzer mit dem Softwareprodukt JV zur Verfügung (siehe Handbuch "Jobvariablen" [6]).

RESIDENT-PAGES =
Anzahl residenter Speicherseiten (im Klasse-6-Speicher), die für den Programmlauf benötigt werden.
Dieser Operand ist nur sinnvoll, wenn im Programm (Lademodul) mit einem CSTAT-Makroaufruf Seiten resident gemacht werden sollen (siehe Handbuch "Makroaufrufe" [7]).
Die angegebene Anzahl darf die Klasse-6-Speichergrenze nicht überschreiten.
Die zulässige Anzahl an residenten Speicherseiten kann der Operateur beeinflussen.

RESIDENT-PAGES = PARAMETERS(...)

MINIMUM =
Minimal benötigte Anzahl residenter Speicherseiten zusätzlich zur Programmlänge.

MINIMUM = STD
Die minimale Anzahl residenter Speicherseiten wird dem Anfangssatz des Programms entnommen.

MINIMUM = <integer 0..32767>
Explizite Angabe der minimalen Anzahl residenter Speicherseiten.

MAXIMUM =
Maximal benötigte Anzahl residenter Speicherseiten zusätzlich zur Programmlänge.

MAXIMUM = STD
Die maximale Anzahl residenter Speicherseiten wird dem Anfangssatz des Programms entnommen.

MAXIMUM = <integer 0..32767>
Explizite Angabe der maximalen Anzahl residenter Speicherseiten.

VIRTUAL-PAGES =
Anzahl Speicherseiten insgesamt (resident und seitenwechselbar), die für den Programmlauf benötigt werden.

VIRTUAL-PAGES = STD
Die Speicheranforderungen werden dem Anfangssatz des Programms entnommen.

VIRTUAL-PAGES = <integer 0..32767>
Explizite Angabe der Speicherseiten. Die Klasse-6-Speichergrenze darf nicht überschritten werden.

VP-SPACE =
Anzahl der Vektorseiten oberhalb oder unterhalb 16 MByte, die für einen Programmlauf benötigt werden. Der Operand ist nur zulässig für einen Vektorrechner.

VP-SPACE = PARAMETERS(...)

VP-BELOW =
Anzahl der Vektorseiten unterhalb 16 MByte.

VP-BELOW = STD
Der Wert 0 wird angenommen.

VP-BELOW = <integer 0..32767>
Explizite Angabe der benötigten Vektorseiten.

VP-ABOVE =
Anzahl der Vektorseiten oberhalb 16 MByte.

VP-ABOVE = STD
Der Wert 0 wird angenommen.

VP-ABOVE = <integer 0..32767>
Explizite Angabe der benötigten Vektorseiten.

VP-WAIT =
Legt die maximale Wartezeit (in Sekunden) fest, die der Job für die Reservierung von Speicherplatz benötigen darf. Der Operand ist nur zulässig für einen Vektorrechner.

VP-WAIT = TASK-STD
Im *Dialogbetrieb* wird das Kommando abgewiesen, wenn die Reservierung von Vektorspeicherplatz nicht möglich ist.
Im *Stapelbetrieb* ist die Wartezeit nicht begrenzt.

VP-WAIT = <integer 0..43200>
Explizite Angabe der maximalen Wartezeit in Sekunden.

Hinweis zum Operanden CPU-LIMIT

Die folgende Tabelle zeigt die Abhängigkeit des Operanden CPU-LIMIT vom Operanden TIME im LOGON-Kommando.

		LOGON-Kommando		
		TIME=t	TIME=NTL	TIME=STD
CPU-LIMIT= JOB-REST	Dialogbetrieb	Zeit "t"	keine Zeit-beschränkung	Zeit 59 Stunden 2)
	Stapelbetrieb	restliche Task-Zeit 1)	keine Zeit-beschränkung	restliche Task-Zeit 4)
CPU-LIMIT= <integer..>	Dialogbetrieb	Zeit <integer..> 2)	keine Zeit-beschränkung	Zeit <integer..> 2)
	Stapelbetrieb	Zeit <integer..> 3)	keine Zeit-beschränkung	Zeit <integer..> 3)

1) Die restliche Task-Zeit ist die Differenz zwischen der LOGON-Zeit "t" und der bereits verbrauchten CPU-Zeit.

2) In diesem Fall werden die LOGON-Zeit und die verbrauchte CPU-Zeit nicht überprüft.

3) Das Kommando START-PROGRAM oder LOAD-PROGRAM wird abgewiesen, wenn die im Operanden CPU-LIMIT verlangte Zeit die restliche Task-Zeit überschreitet.

4) Die restliche Task-Zeit ist die Differenz zwischen der Standard-LOGON-Zeit (wird bei der Systeminstallation festgelegt) und der verbrauchten CPU-Zeit.

LOAD-PROGRAM Programm laden

Dieses Kommando lädt Programme (Lademodule) in den Hauptspeicher. Das geladene Programm wird erst gestartet, wenn dies durch ein Kommando RESUME-PROGRAM gefordert wird.

Wenn der Benutzer das Programm laden *und* starten möchte, kann er anstelle der Kommandos LOAD-PROGRAM und RESUME-PROGRAM das Kommando START-PROGRAM verwenden.

Das Kommando LOAD-PROGRAM wird abgewiesen, wenn die Datei mit dem Lademodul oder die Programmbibliothek mit dem Element vom Typ C durch ein Lesekennwort geschützt ist und das Kennwort nicht in die Kennwortliste der Task eingetragen wurde (Kommando ADD-PASSWORD).

```
LOAD-PROGRAM

 FROM-FILE = <full-filename 1..54 without-gen> / *PHASE(...)

   *PHASE(...)
      | LIBRARY = <full-filename 1..54 without-gen>
      |,ELEMENT = <full-filename 1..41 without-cat-user-gen-vers>
      |,VERSION = *STD / <text 1..24>
,CPU-LIMIT = JOB-REST / <integer 1..32767>
,TEST-OPTIONS = NONE / IDA / AID
,MONJV = *NONE / <full-filename 1..54 without-gen>
,RESIDENT-PAGES = PARAMETERS(...)
   PARAMETERS(...)
      | MINIMUM = STD / <integer 0..32767>
      |,MAXIMUM = STD / <integer 0..32767>
,VIRTUAL-PAGES = STD / <integer 0..32767>
```

Fortsetzung>

Fortsetzung

```
,VP-SPACE = PARAMETERS(...)

   PARAMETERS(...)
      |  VP-BELOW = STD / <integer 0..32767>
      |  ,VP-ABOVE = STD / <integer 0..32767>

,VP-WAIT = TASK-STD / <integer 0..43200>
```

Bedeutung der Operanden siehe Kommando START-PROGRAM

Makroaufrufe für den Aufruf des Laders

LPOV Segment laden

Mit dem Makroaufruf LPOV legt man ein Segment fest, das vom Lader ELDE in den Speicher geladen werden soll. Die Ausführung erfolgt unabhängig davon, ob sich das Segment schon im Speicher befindet.

Im Gegensatz zum automatischen Laden von Segmenten wird mit dem Makro LPOV stets nur ein einziges, nämlich das angegebene Segment geladen, d.h. für jedes zu ladende Segment muß man einen eigenen LPOV-Makroaufruf geben.

Operation	Operanden
LPOV	segment[,adresse]

segment — Symbolischer Name des zu ladenden Segments, der aus höchstens 6 alphanumerischen Zeichen bestehen darf. Welche Bindemodule dieses Segment aufbauen, vereinbart der Benutzer mit OVERLAY- und INCLUDE-Anweisungen des Binders TSOSLNK.

adresse — Maximal 8 Zeichen lange symbolische Adresse im aufrufenden Programm. Bei dieser Adresse wird das Programm nach dem Laden fortgesetzt. Der Benutzer muß dafür sorgen, daß das Segment, in dem sich die angegebene Adresse befindet, nach der Ausführung des LPOV-Makros im Speicher steht.

Läßt man diesen Operanden weg, so wird das Programm mit dem Befehl fortgesetzt, der auf den LPOV-Makroaufruf folgt.

Funktionsweise

Der Makroaufruf LPOV aktiviert den Lader ELDE, der das vereinbarte Segment in den Speicher lädt. Danach wird der Programmlauf an der vereinbarten Adresse bzw. an dem auf LPOV folgenden Befehl fortgesetzt.

Hinweis

Während des Programmlaufs kann sich der Benutzer den Ladezeitpunkt des Segments, das mit LPOV geladen wird, anzeigen lassen, indem er das DTH-Kommando `"AT %LPOV"` gibt. Nach dem Laden des Segments kann er dann DTH-Kommandos geben, die sich auf Adressen im nachgeladenen Segment beziehen.

Vorsicht

Der Makroaufruf LPOV ruft den Lader ELDE direkt auf und arbeitet daher unabhängig von einem evtl. vorhandenen Überlagerungssteuermodul, der zum automatischen Laden von Segmenten vom Binder TSOSLNK aufgebaut wurde (siehe auch Makroaufrufe CALL und SEGLD, Seite 64ff).
LPOV aktualisiert deshalb nicht die Tabellen des Steuermoduls.

Rückinformation und Fehleranzeigen

Treten Fehler während des Ladens eines Überlagerungssegments auf, führt der Ablaufteil folgendes aus:

Im niederwertigsten Byte von Register 15 wird ein Returncode RC abgespeichert.

Das Programm wird mit dem Befehl fortgesetzt, der LPOV folgt.

Tabelle der Rückkehrinformation siehe nächste Seite.

RC	Bedeutung
X'00'	Das Segment wurde geladen.
X'04'	Falscher Nur-Lese-Modifikationssatz.
X'08'	Falscher Satzcode im Text/Modifikationsblock oder ein Modifikationssatz steht vor dem ersten Textsatz.
X'10'	Fehler während eines Systemaufrufs oder ein Segment wurde nicht gefunden.
X'14'	Fehler während eines Systemlaufs.
X'0C'	- Fehler beim Laden von AID-Testhilfedaten oder - zuviele Nur-Lese-Modifikationssätze sind vorhanden oder - der Segmentname kann in keinem der Informationssätze für Lademodule (≙ Segment) gefunden werden oder - keine Operandenliste für LPOV definiert oder - der ELDE wurde nicht aufgerufen.
X'18'	- PAM-Schlüsselinformation falsch oder - Fehler beim Lesen der Programmbibliothek.
X'1C'	Unterschiedliche Segmentnamen im Programm und in der Operandenliste.
X'20'	Inkonsistenz in der PAM-Schlüsselinformation oder ungültiges Format bei READ-ONLY
X'24'	Fehler beim Lesen eines Elements vom Typ C in der Programmbibliothek.
X'F0'	Ungültige CFID in einem PAM-Schlüssel.

Meldungen des Laders ELDE

Die Meldungen des Laders ELDE sind im Handbuch "Systemmeldungen" [9] aufgeführt.

XS-Unterstützung von Binder und Lader (31-Bit-Adressierung)

Dieses Kapitel wendet sich an Anwender, die auf einer 31-Bit-Anlage (XS31) den Adreßraum oberhalb 16Mbyte nutzen wollen. Zum Verständnis dieses Kapitels werden Grundbegriffe der XS-Programmierung vorausgesetzt, die im Handbuch "Einführung in die XS-Programmierung" [10] behandelt sind.

Der Binder TSOSLNK (ab Version 21.0B), der Lader ELDE und der dynamische Bindelader DBL unterstützen ab BS2000 Version 9.0 die 31-Bit-Adressierung. Damit ist es möglich, Programme und Bindemodule zu binden und zu laden, die teilweise oder ganz im Adreßraum oberhalb 16Mbyte liegen. Die Segmentgröße bei Programmen ist dabei auf 16Mbyte begrenzt.

Binder TSOSLNK

Für das Binden von Programmen, die im Adreßraum oberhalb 16Mbyte liegen, sind die Funktionen der PARAM-, OVERLAY- und TRAITS-Anweisung erweitert.

PROGRAM-Anweisung

Die Ladeadresse eines gebundenen Programmes steuert der Operand LOADPT der PROGRAM-Anweisung.

Operation	Operanden
PROGRAM	LOADPT={adresse|*XS}

LOADPT

=adresse — legt die Ladeadresse des Programms fest.
Angabe sedezimal X'...'. Standardwert ist LOADPT=0.

=*XS — legt die Ladeadresse des Programmes im Adreßraum oberhalb 16Mbyte fest. Die CSECT muß das Attribut RMODE ANY haben.

OVERLAY-Anweisung

Bei segmentierten Programmen steuert der Operand LOADPT der OVERLAY-Anweisung die Ladeadresse eines einzelnen Segments.

Operation	Operanden
OVERLAY	LOADPT={adresse|*XS}

LOADPT

=adresse — legt die Ladeadresse des zugehörigen Segments fest. Die sedezimale Adresse X'...' wird auf Seitengrenze ausgerichtet, wenn sie nicht auf Seitengrenze angegeben wurde.

=*XS — legt die Ladeadresse des zugehörigen Segments im Adreßraum oberhalb 16Mbyte fest. Die Ladeadresse ist von evtl. weiteren OVERLAY-Anweisungen abhängig.

Hinweis

– Im Adreßraum oberhalb 16Mbyte sind keine "echten" Überlagerungsstrukturen möglich, d.h. der Binder strukturiert die Segmente adreßmäßig hintereinander.
– Im Adreßraum unterhalb 16Mbyte werden Überlagerungsstrukturen behandelt wie auf Seite 44ff beschrieben.

Binden eines vorgebundenen Moduls (MODULE-Anweisung)

Damit der dynamische Bindelader DBL einen vorgebundenen Modul als Einheit laden kann, ist es notwendig, die Lage des Moduls im Adreßraum (oberhalb oder unterhalb 16Mbyte) im gesamten festzulegen. Zu diesem Zweck wird für den Modul beim Binden ein "Pseudo-RMODE" festgelegt, der aus den Attributen RMODE der einzelnen CSECTs wie folgt bestimmt wird:

– Der Bindemodul erhält nur dann das Attribut (Pseudo-)RMODE=ANY, wenn alle enthaltenen CSECTs das Attribut RMODE=ANY besitzen.
– Enthält mindestens eine CSECT das Attribut RMODE=24, erhält auch der Modul das eingeschränkte Attribut (Pseudo-)RMODE=24.

Das Attribut AMODE wird durch den Programmabschnitt (CSECT) bestimmt, der die Einsprungstelle des Bindemoduls enthält.

TRAITS-Anweisung

Die TRAITS-Anweisung bietet die Möglichkeit, beim Binden die Werte der Attribute AMODE und RMODE für einzelne oder alle Programmabschnitte (CSECTs) zu ändern. Änderungen der Attribute AMODE und RMODE sind nur zulässig, wenn sie gegenüber den Attributen in der CSECT eingeschränkt werden. Treten Konflikte zwischen den Attributen in der CSECT und der TRAITS-Anweisung auf, übergeht der Binder die Attribute in der TRAITS-Anweisung.

Lader ELDE

Der Lader ELDE lädt die Programme ab der Ladeadresse, die in der PROGRAM- bzw. OVERLAY-Anweisung des Binders mit dem Operanden `LOADPT=adresse|*XS` (bzw. dem Standardwert) festgelegt ist (siehe Seite 161ff).

Der Adressierungsmodus ist abhängig vom Attribut AMODE der CSECT, die die Einsprungstelle enthält. Abhängig von AMODE wird der Adressierungsmodus wie folgt eingestellt:

AMODE (CSECT)	24	31	ANY
Adressierungs-modus	24-Bit	31-Bit	31-Bit: falls alle CSECTs mit AMODE=ANY 24-Bit: falls eine oder mehrere CSECTS mit AMODE=24.

Der 31-Bit-Adressierungsmodus wird eingestellt, wenn die CSECT das Attribut AMODE=31 hat. Bei AMODE=24 wird der 24-Bit-Adressierungsmodus eingestellt.

Wird ein Programm vom Binder nicht als Speicherabbild gebunden (Operand COREIM=N in der PROGRAM-Anweisung) und haben alle CSECTs das Attribut AMODE=ANY, lädt der ELDE das Programm oberhalb 16Mbyte.

Literatur

[1] BS2000
Binder-Lader-Starter (BLS)
Benutzerhandbuch

Zielgruppe
Software-Entwickler
Inhalt
Das System Binder-Lader-Starter (BLS) besteht aus den Funktionseinheiten
- Binder BINDER
- dynamischer Bindelader DBL
- statischer Lader ELDE

Die einzelnen Abschnitte enthalten jeweils eine Beschreibung aus funktioneller Sicht mit Beispielen, sowie einen Nachschlageteil mit den Anweisungen, Kommandos und ggf. Makroaufrufen.

[2] **Assembler (BS2000)**
Beschreibung

Zielgruppe
Assembler-Anwender im BS2000
Inhalt
Assembler-Charakteristik, Assemblersprache, Makrosprache, Handhabung des Assemblers, Meldungen bzw. Fehlermeldungen, Flags.
Beschreibung des Assembler-Diagnoseprogramms ADIAG.

[3] **LMS** (BS2000)
Beschreibung

Zielgruppe
BS2000-Anwender
Inhalt
- Beschreibung der LMS-Anweisungen im ISP-Format zum Erstellen und Verwalten von PLAM-Bibliotheken
- Speicherung mit Delta-Technik
- Aufruf als Unterprogramm aus COBOL- und Assembler-Programmen

Einsatz

Dialog- und Stapelbetrieb

[4] BS2000
Systeminstallation
Benutzerhandbuch

Zielgruppe
BS2000-Systemverwalter
Inhalt
Neuinstallation, Versionsumstellung, Generierung eines neuen Public-Volume-Sets, Generierung eines Subsystemkatalogs, Anweisungen für SIR und UGEN.
Einsatz
Systemverwaltung, Rechenzentrum

[5] BS2000
Benutzer-Kommandos (SDF-Format)
Beschreibung

Zielgruppe
BS2000-Anwender
Inhalt
Benutzer-Kommandos des BS2000 in der Syntax der Dialogschnittstelle SDF (System Dialog Facility)
Einsatz
BS2000-Dialogbtrieb und -Stapelbetrieb mit SDF

[6] BS2000
Jobvariablen
Beschreibung

Zielgruppe
BS2000-Benutzer
Inhalt
Anwendungsmöglichkeiten für Jobvariablen zur Steuerung und Überwachung von Aufträgen und Programmläufen;
Bedingungsabhängige Auftragssteuerung;
Alle erforderlichen Kommandos und Makroaufrufe;
Anwendungsbeispiele;
Einsatz
BS2000-Teilnehmerbetrieb

[7] BS2000
Makroaufrufe an den Ablaufteil
Beschreibung

Zielgruppe
BS2000-Assembler-Programmierer (nicht privilegiert); Systemverwalter
Inhalt
Alle Makroaufrufe an den Ablaufteil in lexikalischer Reihenfolge mit Hinweisen und Beispielen, einschließlich ausgewählter Makroaufrufe für das DVS und für TIAM;
Zusammenstellung der Makroaufrufe nach Anwendungsgebieten. Ausführlicher Lernteil über Ereignissteuerung, Serialisation, Inter-Task-Kommunikation, Contingencies;
Einsatz
BS2000-Anwendungsprogramme

[8] **AID** (BS2000)
Advanced Interactive Debugger
Beschreibung

Zielgruppe
AID-Erstanwender
Inhalt
Beschreibung des Leistungsumfangs und der Anwendungsmöglichkeiten, sowie der Bedienung von AID bei Fehlerdiagnose, Fehlerbehebung und Test

[9] BS2000
Systemmeldungen
Beschreibung

Zielgruppe
BS2000-Anwender
Inhalt
Standardmeldungen BS2000 - Zentralsystem inklusive SPOOL, RSO, SDF;
Standardmeldungen der Softwareprodukte DCAM, TIAM, RBAM;

[10] **BS2000**
Einführung in die XS-Programmierung
(für ASSEMBLER-Programmierer)
Benutzerhandbuch

Zielgruppe
Programmierer, System-Programmierer
Inhalt
Die Adressierungsmodi der ab BS2000 V9.0 unterstützten Zentraleinheiten und wie sie sich auf die ASSEMBLER-Befehle auswirken;
Programmierhinweise für ASSEMBLER-Programmierer, die den erweiterten Adreßraum von XS-Anlagen nutzen oder ihre Programme adressierungsmodus-unabhängig und portabel gestalten wollen;
Einsatz
TU-, TPR-Programme

[11] **BS2000**
Programmiersystem
Technische Beschreibung

Zielgruppe
- BS2000-Anwender und -Betreiber, die sich für den technischen Hintergrund ihres Systems interessieren (Softwareentwickler, Systemanalytiker, RZ-Leiter, Systemverwalter)
- Informatiker, die ein konkretes "General-Purpose"-Betriebssystem studieren wollen

Inhalt
Funktionen und Realisierungsprinzipien
- des Binders
- des Laders
- des Binde-Laders
- der Test- und Diagnosehilfen
- des Programmbibliothekssystems

Bestellnummer
U3216-J-Z53-1

Bestellen von Handbüchern

Die aufgeführten Handbücher finden Sie mit ihren Bestellnummern im *Druckschriftenverzeichnis Datentechnik*. Dort ist auch der Bestellvorgang erklärt. Neu erschienene Titel finden Sie in den *Druckschriften-Neuerscheinungen Datentechnik*.

Beide Veröffentlichungen erhalten Sie regelmäßig, wenn Sie in den entsprechenden Verteiler aufgenommen sind. Wenden Sie sich bitte hierfür an eine Geschäftsstelle unseres Hauses.

Stichwörter