# Sie haben

uns zu diesem Handbuch etwas mitzuteilen?
Schicken Sie uns bitte Ihre Anregungen unter
Angabe der Bestellnummer dieses Handbuches.

Siemens Nixdorf Informationssysteme AG
Manualredaktion STM QM 2
Otto-Hahn-Ring 6
W-8000 München 83

Fax: (089) 636-40443

email im EUnet:
man @ sieqm2.uucp

Dieses Handbuch ist auf chlorfrei gebleichtem Papier gedruckt.

# LMS (BS2000)

ISP-Format

Benutzerhandbuch




Ausgabe Januar 1992 (LMS V2.0A)

# ...und Schulung?

Zu dem nachstehend beschriebenen Produkt, wie zu fast allen DV-Themen, bieten unsere regionalen Training Center in Berlin, Essen, Frankfurt, Hannover, Hamburg, München, Mainz, Stuttgart, Wien und Zürich Kurse an.

**Auskunft und Info-Material:**

| | |
|---|---|
| Systemfamilien 7.500 und 8890 | Telefon (089) 636-48987 |
| Ein- und Mehrplatzsysteme | Telefon (089) 636-42480 |

Siemens Nixdorf Training Center
Postfach 830951, W-8000 München 83

# Inhalt

# Einleitung

Dieses Handbuch beschreibt den Funktionsumfang und die Arbeitsweise des Bibliotheksprogramms LMS.

## Kurzbeschreibung des Produkts

Das Bibliotheksverwaltungssystem LMS (Library Maintenance System) erstellt und verwaltet Programmbibliotheken und bearbeitet die darin enthaltenen Elemente.

Programmbibliotheken sind PAM-Dateien des BS2000, die mit der Bibliotheks-Zugriffsmethode PLAM (Program Library Access Method) bearbeitet werden. Daher werden sie auch als PLAM-Bibliotheken bezeichnet.

Der grundlegende Nutzen besteht darin, daß

- alle Elementtypen in einer Bibliothek mit einheitlichen Anweisungen bearbeitet werden können,
- gleichnamige Elemente existieren können, die sich nur durch Typ- oder Versionsbezeichnung unterscheiden,
- auf die Bibliothek von mehreren Benutzern gleichzeitig auch schreibend zugegriffen werden kann,
- für die meisten während eines SW-Entwicklungsprozesses anfallenden Datenelemente eine einheitliche Datenhaltung mit einheitlichen Zugriffsfunktionen existiert und
- die Dienstprogramme und Compiler auf diese Datenhaltung zugreifen und die einzelnen Elemente direkt bearbeiten können.

Dadurch werden viele Probleme beseitigt, die bei der Programmerstellung, -pflege und -dokumentation entstehen.

# Zielgruppen des Handbuchs

Dieses Handbuch richtet sich an alle BS2000-Anwender, die ihre Daten mit Hilfe von Bibliotheken verwalten.

Sie sollten über BS2000 Kenntnisse verfügen, insbesondere mit den wichtigsten Kommandos vertraut sein. Als Unterlage siehe Einführung in die Systemanwendung [6].

Zu dem Produkt LMS gibt es ein eigenes Taschenbuch [13]. Das Taschenbuch ist als Kurzanleitung für den erfahreren LMS-Benutzer gedacht. Es enthält eine Zusammenstellung aller LMS-Anweisungen und Verarbeitungsoperanden.

# Konzept des Handbuchs

### Teilung des Handbuchs

Zur Version des LMS V2.0A wurde das Handbuch erstmals in das hier vorliegende Handbuch "LMS (BS2000) Benutzerhandbuch" und "LMS Unterprogramm-Schnittstelle" geteilt. Neben den immer mehr zunehmenden Funktionalitäten des LMS und des damit zunehmenden Umfanges des Handbuchs, war die Unabhängigkeit des Unterprogramms von der Benutzeroberfläche des LMS der Hauptgrund für die Teilung.

### Aufbau des Handbuchs

Im einzelnen finden Sie im Kapitel

- **Definitionen und Konventionen**
  die Elementtypen und Bibliotheksformate, die Sie mit LMS bearbeiten können
- **Funktionen von LMS**
  einen Umriß der Möglichkeiten, die LMS bietet
- **Anweisungen**
  alle Anweisungen in alphabetischer Reihenfolge
- **Verarbeitungsoperanden**
  alle Verarbeitungsoperanden in alphabetischer Reihenfolge
- **Beispiele**
  ausgewählte Beispiele für die Anwendung von LMS
- **Alte LMS-Unterprogrammschnittstelle**
  die Beschreibung der Konventionen und ein Beispiel
- **Meldungen**
  die von LMS ausgegebenen Meldungen in der Reihenfolge ihrer Schlüssel

Das Stichwortverzeichnis am Ende des Handbuchs dient dem schnelleren Auffinden von Problem- und Begriffserläuterungen.

Literaturhinweise werden im Text in Kurztiteln angegeben. Der vollständige Titel jeder Druckschrift, auf die durch eine Nummer verwiesen wird, ist im Literaturverzeichnis hinter der entsprechenden Nummer aufgeführt. Hinter diesem Verzeichnis finden Sie Hinweise zur Bestellung von Druckschriften.

# Änderungen gegenüber der vorigen Handbuchausgabe

Das Handbuch zur LMS-Version 2.0A enthält gegenüber der Vorgängerversion (LMS V1.4A) folgende Neuerungen:

- Das ehemalige Kapitel 8 "LMS als Unterprogrammschnittstelle" ist jetzt in einem eigenen Handbuch [15] beschrieben.
- Das Kapitel "Pamkey-Eliminierung" (ab Seite 75) wurde ergänzt.
- Der Elementtyp F für IFG-Formatmasken und der Elementtyp U für IFG-Benutzer-Profile wurde neu eingeführt. F und U können bei den Anweisungen DEL, DUP, LST, NAM und TOC verwendet werden.
- Der Elementtyp L für Bindelademodule (LLMs) wurde neu eingeführt. L kann bei den Anweisungen DEL, DUP, LST, NAM, TOC und UPD verwendet werden.
- Für den Elementtyp L gibt es folgende neue Verarbeitungsoperanden: PAR CSECT, PAR PATH und PAR SLICE.
- Die Korrekturanweisungen für Bindemodule wurden um die Korrekturanweisung *REM (Seite 178) erweitert.
- Das Kapitel "LMS als Dialoganschluß" wurde in "alte LMS-Unterprogramm-Schnittstelle" (Seite 255) umbenannt.
- Die BS2000-Kommandos wurden von ISP auf SDF umgestellt.

# Einführung in LMS

LMS erstellt und verwaltet Programmbibliotheken und bearbeitet die darin enthaltenen Elemente. Eine Programmbibliothek ist eine Datei mit Unterstruktur. Sie enthält Elemente und ein Inhaltsverzeichnis der gespeicherten Elemente.

Ein Element ist eine logisch zusammengehörige Datenmenge wie z.B. eine Datei, eine Prozedur, ein Bindemodul oder ein Quellprogramm. Jedes Element ist in der Bibliothek einzeln ansprechbar.

Jede Bibliothek hat einen Eintrag im Systemkatalog. Der Benutzer kann den Namen und andere Dateimerkmale, wie z.B. die Schutzfrist oder die Mehrbenutzbarkeit festlegen.

Das Speichern mehrerer Dateien in einer Bibliothek entlastet den Systemkatalog, da dort nur die Bibliothek eingetragen ist und nicht jedes Element. Außerdem spart es auch Speicherplatz, da die Elemente in der Bibliothek in komprimierter Form gespeichert werden.

Bei Verwendung der Delta-Technik werden von mehreren Versionen eines Elementes nur die Unterschiede (Deltas) zur Vorgängerversion gespeichert, was zusätzlich weiter Speicherplatz einsparen hilft. Beim Lesen solcher Elementversionen werden diese Deltas von LMS wieder an die entsprechenden Stellen eingemischt. Dem Benutzer steht somit wieder das komplette Element zur Verfügung.

Bindemodule können von allen Compilern und Lademodule von TSOSLNK direkt in Programmbibliotheken abgelegt werden, LMS kann aber auch Bindemodule aus dem EAM Bereich und Lademodule aus Dateien in die Programmbibliothek kopieren.

LMS führt folgende Funktionen aus:

- Bibliotheken einrichten,
- Elemente in eine Bibliothek aufnehmen,
- Elemente editieren,
- Elemente in Dateien ausgeben,
- Elemente in eine andere Bibliothek kopieren,
- Elemente auflisten,
- Elemente löschen,
- Elemente korrigieren,
- Elemente umbenennen,
- Elemente neunumerieren,
- Elemente vergleichen,
- Unterschiede (Deltas) von Elementversionen bilden und speichen,
- Inhaltsverzeichnis einer Bibliothek ausgeben.

LMS bearbeitet folgende Bibliotheksformate:

- **Programmbibliotheken** zum Speichern von Quellprogrammen, Makros, Binde- und Lademodulen, Listen, Prozeduren, Text usw. Diese Bibliotheken werden mit der Bibliothekszugriffsmethode PLAM bearbeitet.
- **Sequentielle Bibliotheken** zum Speichern von Quellprogrammen, Bindemodulen, Makros und BS1000-Phasen auf Bändern.

Daneben können auch die noch vorhandenen MLU-, LMR- und COBLUR-Bibliotheken von LMS bearbeitet werden.

Die Umstellung der bisherigen Verfahren auf PLAM wird somit wesentlich erleichert und vereinfacht.

Mit der Unterprogrammschnittstelle von LMS [15] stehen dem Benutzer komfortable Möglichkeiten zur Bearbeitung von LMS-Bibliotheken und deren Inhalten, direkt aus seinem Hauptprogramm zur Verfügung. Dabei wird LMS dynamisch nachgeladen. Diese Unterprogrammschnittstelle kann auch im Extended-System-Bereich (XS) verwendet werden.

Das folgende Bild stellt die Ein- und Ausgabemöglichkeiten von LMS dar:

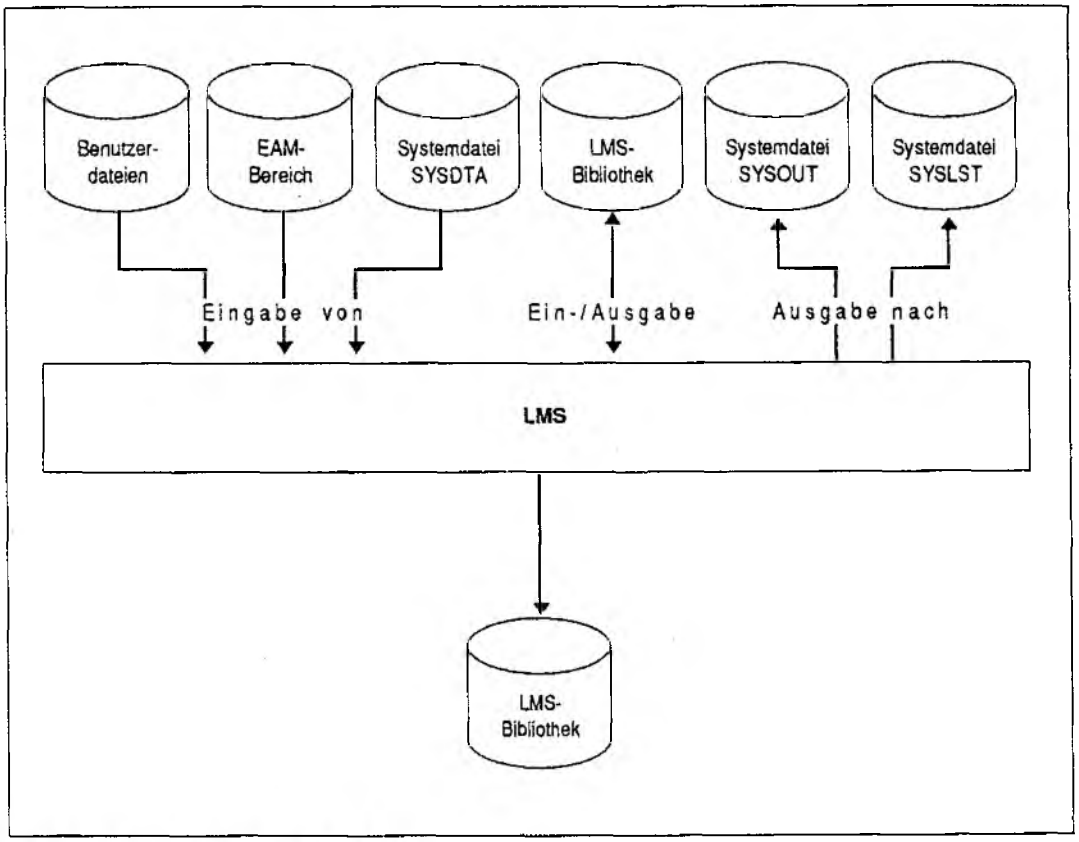



Bild 1 Zugriffsmöglichkeiten von LMS

*Beispiel für einen LMS-Lauf*

```
/SHOW-FILE-ATTRIBUTES A.
00000003 :N:$BENUTZER.A.BEISPIEL
00000003 :N:$BENUTZER.A.QUELL.A
:N:    PUBLIC:     2  FILES RES=        6  FREE=        3  REL=        0 PAGES
/START-PROGRAM $LMS ------------------------------------------------------ (01)
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG. 1990.
%  ALL RIGHTS RESERVED
%  LMS0310 LMS VERSION V02.0A10 LOADED
$LIB FILE=UEB.BIBL,BOTH,NEW ---------------------------------------------- (02)
$ADDS A.QUELL.A ---------------------------------------------------------- (03)
$PAR LOG=MED ------------------------------------------------------------- (04)
$ADDD A.BEISPIEL>BSP ----------------------------------------------------- (05)
INPUT  FILE
OUTPUT LIBRARY= :N:$BENUTZER.UEB.BIBL,DEV=DISK                  ---------- (06)
         ADD A.BEISPIEL AS (D)BSP/@(0001)/1991-07-24
$TOC* * ------------------------------------------------------------------ (07)
INPUT  LIBRARY= :N:$BENUTZER.UEB.BIBL,DEV=DISK
TYP NAME VER (VAR#) DATE
(D) BSP      (0001) 1991-07-24
     1 (D)-ELEMENT(S) IN THIS TABLE OF CONTENTS                    ---------- (08)
TYP NAME      VER (VAR#) DATE
(S) A.QUELL.A     (0001) 1991-07-24
     1 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS
$END --------------------------------------------------------------------- (09)
%  LMS0311 LMS V02.0A10 ENDED NORMALLY
/
```

(01) LMS wird aufgerufen.

(02) LMS richtet UEB.BIBL als Programmbibliothek neu ein und weist sie als Ein- und Ausgabebibliothek zu.

(03) Die Datei A.QUELL.A wird als Element vom Typ S mit dem Elementnamen A.QUELL.A in die Bibliothek aufgenommen.

(04) Der Verarbeitungsoperand LOG=MED bestimmt, daß LMS nicht nur Fehlermeldungen, sondern auch Erfolgsmeldungen ausgibt.

(05) Die Datei A.BEISPIEL wird als Element vom Typ D mit dem Elementnamen BSP in die Bibliothek aufgenommen.

(06) Erfolgsmeldung: Da der Verarbeitungsoperand LOG=MED gesetzt ist, bestätigt LMS die Aufnahme der Datei A.BEISPIEL als Element BSP.

(07) Das Inhaltsverzeichnis der Programmbibliothek UEB.BIBL soll aufgelistet werden.

(08) Inhaltsverzeichniseintrag der Programmbibliothek UEB.BIBL.

(09) LMS wird beendet.

### LMS im Dialog- und Stapelbetrieb

LMS läuft sowohl im Dialog- als auch im Stapelbetrieb ab. Im Dialogbetrieb liest LMS die Anweisungen standardmäßig mit dem Makro WRTRD von der Datensichtstation. Während des LMS-Laufes läßt sich die Anweisungseingabe mit CTL auf die Systemdatei SYSDTA oder ein Bibliothekselement vom Typ J (S bei Quellprogrammbibliothek) umschalten.

Vor dem LMS-Lauf muß der Schalter 1 (/MODIFY-JOB-SWITCHES ON=1) gesetzt werden, wenn LMS bei Aufruf in einer Prozedur auch die LMS-Anweisungen aus der Prozedur lesen soll. Sonst wird LMS zwar aufgerufen, erwartet aber die Anweisungen von der Datensichtstation.

Im Stapelbetrieb liest LMS die Anweisungen von der Systemdatei SYSDTA. Mit CTL läßt sich die Anweisungseingabe auf ein Bibliothekselement vom Typ J (Typ S bei Quellprogrammbibliothek) umschalten.

Ist eine Bibliothek noch geschlossen, gibt im Stapelbetrieb LMS die Meldung

```
FILE (ELEMENT oder TYP) IS LOCKED.NEXT ATTEMPT AFTER 6 SECONDS!
```

maximal hundertmal aus. Nach hundert Versuchen wird automatisch zum nächsten Programmschritt übergegangen.

Das LMS-Protokoll wird in die Systemdatei SYSOUT (d.h. im Dialog die Datensichtstation) bzw. in das Medium ausgegeben, das mit PRT vereinbart ist (Systemdatei SYSLST oder Bibliothekselement). Wenn LMS außer Fehler- auch Erfolgsmeldungen ausgeben soll, muß der Verarbeitungsoperand PAR LOG=MED gesetzt werden.

# Definitionen und Konventionen

## Was ist eine Bibliothek?

Eine Bibliothek ist eine Datei mit Unterstruktur. Sie enthält Elemente und ein Inhaltsverzeichnis. Jedes neu aufgenommene Element wird automatisch in das Inhaltsverzeichnis eingetragen.

Ein Element ist eine logisch zusammengehörige Datenmenge wie z.B. eine Datei, eine Prozedur, ein Bindemodul oder ein Quellprogramm. Jedes Element ist in der Bibliothek einzeln ansprechbar.

Das Speichern von Dateien als Elemente in einer Bibliothek entlastet den Systemkatalog, da jede Bibliothek nur einen Katalogeintrag hat. Es wird Speicherplatz gespart, da die Elemente in der Bibliothek grundsätzlich in komprimierter Form gespeichert werden. Darüberhinaus können die Elemente auch als Delta-Elemente gespeichert werden (siehe Kapitel "*Archivieren mit Delta-Technik*", *Seite 58*).

Werden Bindemodule aus dem EAM-Bereich gespeichert, müssen die Quellprogramme nicht immer wieder neu übersetzt werden.

Jede Bibliothek hat einen Eintrag im Systemkatalog. Der Benutzer kann den Namen und andere Dateimerkmale, wie z.B. die Schutzfrist oder die Mehrbenutzbarkeit, festlegen. Der Eintrag im Katalog bzw. Änderungen darin werden vom Benutzer über Systemkommandos vorgenommen.

### Aufbau einer Bibliothek

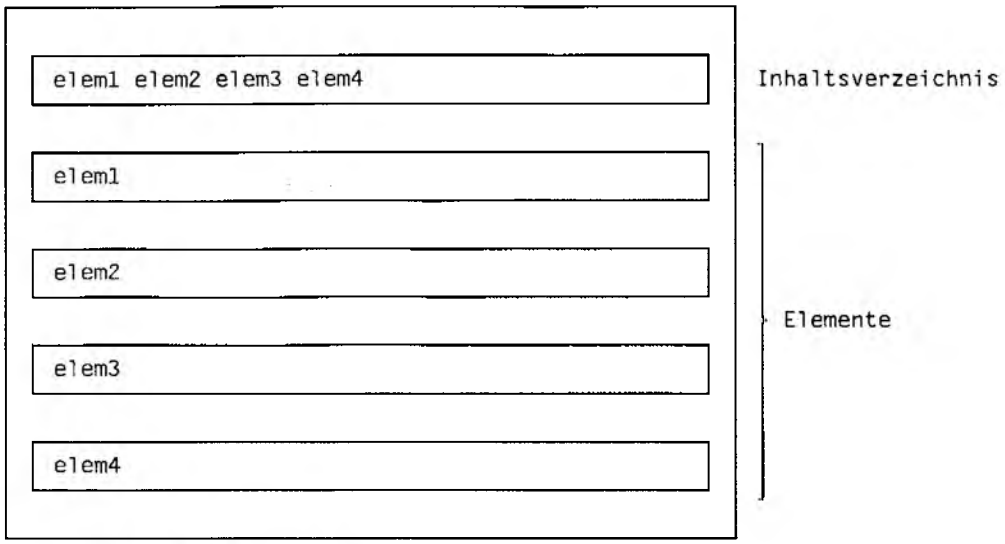



Bild 2 Aufbau einer Bibliothek

### Ein- und Ausgabebibliotheken

LMS bearbeitet eine Bibliothek als Ein- und/oder Ausgabebibliothek. Die **Eingabebibliothek** dient LMS als Eingabemedium, in die **Ausgabebibliothek** gibt LMS Elemente aus. Eine Ein- bzw. Ausgabebibliothek wird mit LIB zugewiesen.

LMS eröffnet eine Ausgabebibliothek zum Lesen und Schreiben. Eine Eingabebibliothek wird nur mit DEL oder NAM auch zum Schreiben eröffnet, sonst nur zum Lesen.

## Welche Bibliotheken gibt es?

LMS bearbeitet verschiedene Bibliotheksformate:

- Programmbibliotheken
- Quellprogrammbibliotheken
- Makrobibliotheken
- Bindemodulbibliotheken
- sequentielle Bibliotheken

### Programmbibliotheken (PL)

Programmbibliotheken sind PAM-Dateien, die mit der Bibliotheks-Zugriffsmethode PLAM bearbeitet werden. Daher werden sie auch als PLAM-Bibliotheken bezeichnet.

Die grundlegenden Vorteile zu den anderen Bibliotheksformaten bestehen darin, daß

- alle Elementtypen in einer Bibliothek abgelegt werden können,
- gleichnamige Elemente existieren können, die sich durch Typ- oder Versionsbezeichnung unterscheiden,
- auf die Bibliothek von mehreren Benutzern gleichzeitig auch schreibend zugegriffen werden kann.

### Mehrere Elementtypen in einer Bibliothek

Programmbibliotheken können alle von LMS unterstützten Elementtypen enthalten.

Der Elementtyp gibt an, wie der Inhalt der Elemente von LMS zu interpretieren ist und in welche Ablageeinheit das Element gehört:

| Typ | Inhalt des Elementes |
|---|---|
| S | Quellprogramme |
| M | Makros |
| R | Bindemodule |
| J | Prozeduren |
| P | druckaufbereitete Daten |
| C | Lademodule |
| D | Textdaten |
| X | Daten beliebigen Formates |
| H | Compiler-Ergebnisinformationen |
| L | Bindelademodule (LLM) |
| F | IFG-Formatmasken |
| U | IFG-Benutzer-Profile |

Durch die Eigenschaften der Programmbibliothek können zu einem Projekt sämtliche Daten, vom Quellprogramm, über Binde- und Lademodule, Übersetzungsprozeduren, Testdaten bis zur Dokumentation, in den entsprechenden Ablageeinheiten einer Bibliothek gespeichert werden.

Elemente vom Typ S, M, J, P und D können auch mit Hilfe der Delta-Technik archiviert werden. Bei Verwendung der Delta-Technik werden von mehreren Versionen eines Elementes nur die Unterschiede (Deltas) zur Vorgängerversion gespeichert, was zusätzlich weiteren Speicherplatz sparen hilft. Beim Lesen solcher Elementversionen werden diese Deltas von LMS wieder an die entsprechenden Stellen eingemischt. Dem Benutzer steht somit wiederum das komplette Element zur Verfügung. Außerdem können hierarchische Beziehungen zwischen den Elementen hergestellt werden (Delta-Sequenz, Delta-Baum).

### Mehrere Versionen je Elementtyp und -name

In Programmbibliotheken wird ein Element durch seinen Typ, seinen Namen und eine Versionsbezeichnung eindeutig bestimmt. Weiter ist es möglich, zu einem Elementtyp und -namen mehrere Versionen zu speichern.

Falls der Benutzer bei der Bearbeitung keine Angaben zur Version macht, führt LMS standardmäßig folgende Aktionen aus:

- Beim Lesen
  wird das Element gesucht, das zu dem angegebenen Namen die höchste Versionsbezeichnung besitzt. Das Datum wird dabei nicht berücksichtigt.
- Beim Schreiben
  hängt die Versionsbehandlung von der Anweisung ab:

  ADD, PRT
  Das Element wird mit der höchsten Versionsnummer X'FF' erzeugt bzw. überschrieben. LMS stellt diese Version mit @ dar.

  COR, DUP, EDT, EDR, NAM, NUM, UPD
  Das Ausgabeelement erhält die Versionsbezeichnung des Eingabeelementes.

  Wird dabei ein gleichnamiges Element überschrieben, wird die interne Variantennummer um 1 erhöht. Sie dient als Schreibzugriffszähler.

Durch die Einführung der Delta-Technik besteht für den Benutzer zusätzlich die Möglichkeit, zwischen zwei Arten der Speicherungsform zu wählen. Gesteuert wird die Speicherung durch den Operanden BASEVERSION bei ADD bzw. DUP. Wird er angegeben, wird das Element als Delta-Element gespeichert. Fehlt er, wird das Element als Voll-Element in die Bibliothek aufgenommen.

### Mehrfachzugriff auf Programmbibliotheken

Eine Bibliothek kann von einem oder mehreren Benutzern sowohl lesend als auch schreibend eröffnet werden.

Ein Voll-Element kann gleichzeitig von mehreren Benutzern gelesen, jedoch nur von einem geschrieben werden. Während ein Voll-Element zum Schreiben eröffnet ist, kann kein anderer Zugriff - auch kein lesender - auf dieses Element erfolgen, jedoch auf andere Voll-Elemente der Bibliothek.

Bei den Korrekturfunktionen (COR, EDR, EDT, UPD) wird die Bibliothek gleichzeitig als Ein- und Ausgabebibliothek zugewiesen. in diesem Fall kann ein Benutzer sowohl lesend als auch schreibend auf das Voll-Element zugreifen.

Delta-Elemente können jeweils nur von einem Benutzer bearbeitet werden. Der Behälter (eine Ablageeinheit einer Bibliothek) ist sowohl beim Schreiben als auch beim Lesen für andere Benutzer gesperrt.

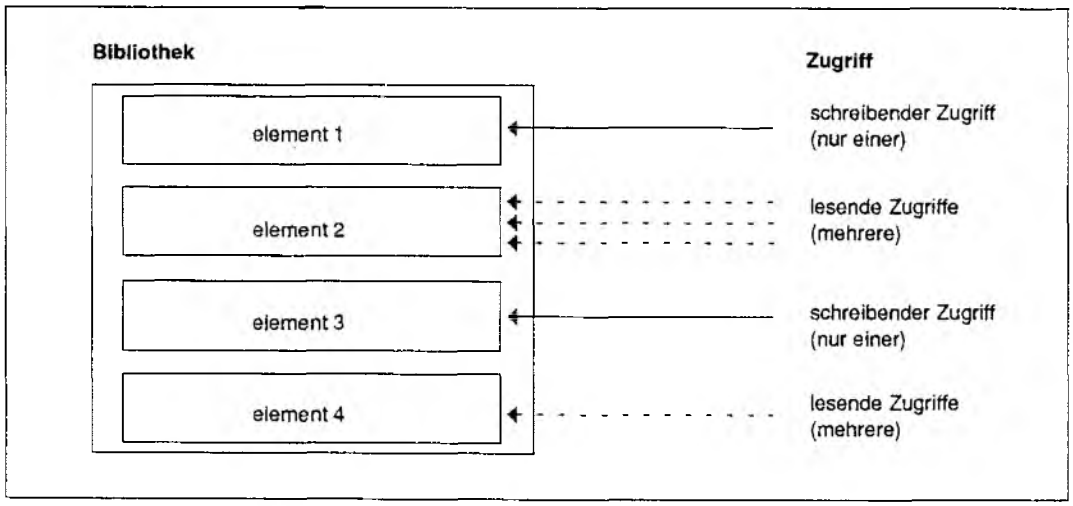



Bild 3 Mehrfachzugriff auf Elemente

Durch den Mehrfachzugriff auf eine Bibliothek ist es möglich, daß ein Element beim Auflisten des Inhaltsverzeichnisses noch vorhanden ist, beim anschließenden Zugriff jedoch nicht mehr existiert. Ein anderer Benutzer hat es in der Zwischenzeit gelöscht.
Das Auflisten des Inhaltsverzeichnisses (siehe TOC) zeigt also nur den momentanen Zustand der Eingabebibliothek.
Für die logische Koordination der Zugriffe auf Programmbibliothekselemente ist der Benutzer selbst verantwortlich.

### Einschränken des Mehrfachzugriffes

LMS eröffnet eine Programmbibliothek immer mit SHARED-UPDATE=YES. Ein /SET-FILE-LINK-Kommando mit SHARED-UPDATE=NO, das sich auf diese Bibliothek bezieht, wird nicht wirksam. Der Benutzer kann jedoch den Mehrfachzugriff durch folgende Kommandos einschränken:

/SECURE-RESOURCE-ALLOCATION-Kommando:
Während des Prozesses, in dem das /SECURE-RESOURCE-ALLOCATION-Kommando gegeben wird, kann kein anderer Benutzer mehr auf die Bibliothek zugreifen.

/MODIFY-FILE-ATTRIBUTES-Kommando:
Dieses Kommando schränkt den Mehrfachzugriff durch die Vergabe von Schreib- und Lesekennworten bzw. durch die Operanden USER-ACCESS=OWNER-ONLY oder ACCESS=READ nach Bedarf ein.

Beschreibung dieser Kommandos, siehe Benutzer-Kommandos [7].

### Typbezogene Bibliotheken

Folgende Bibliotheken können jeweils nur einen Elementtyp aufnehmen:

- Quellprogrammbibliotheken
- Makrobibliotheken
- Bindemodulbibliotheken

Im Unterschied zu den Programmbibliotheken können diese Bibliotheken nicht mehrere Elemente mit gleichem Namen aufnehmen.

LMS trägt keine Dateischutzmerkmale für Bibliotheken bzw. Elemente ein. Die Angabe einer Schutzfrist (RETENTION-PERIOD) im /SET-FILE-LINK-Kommando wird von LMS beim Anlegen einer Bibliothek in den Katalogeintrag übernommen.

Ein Parallelzugriff auf Quellprogramm-, Makro- und Bindemodulbibliotheken ist möglich. Sie können gleichzeitig von verschiedenen Prozessen gelesen werden.

Eine Bibliothek wird von LMS zum Lesen und Schreiben eröffnet (OPEN=INOUT), wenn sie als Standard-Ausgabebibliothek (LIB ...,USAGE=OUT) für den LMS-Lauf zugewiesen oder bei DEL, NAM oder PRT angegeben wurde.

#### Quellprogrammbibliotheken (OSM)

Quellprogrammbibliotheken sind ISAM-Dateien (KEY-POSITION=5, KEY-LENGTH=8) und kennen nur den Elementtyp S. In Quellprogrammbibliotheken können jedoch auch Prozeduren, Protokolle und Textdaten als Elementtyp S abgelegt werden.

Bei Übersetzungsläufen können Quellprogramme in Bibliotheken als Eingabe für Sprachübersetzer zugewiesen werden. Dies kann mit dem Kommando `/ASSIGN-SYSDTA TO-FILE=*LIBR-ELEM(LIBRARY=bibliothek,ELEM=element)` oder in der gleichen Weise in der *COMOPT-Anweisung erfolgen.

Über Prozeduraufrufe können Kommandofolgen in Quellprogrammbibliotheken zum Ablauf gebracht werden.

#### Makrobibliotheken (OSM)

Makrobibliotheken sind ISAM-Dateien (KEY-POSITION=5, KEY-LENGTH=8) und kennen nur den Elementtyp M.

Der Assembler entnimmt die im Programm angesprochenen Makros der Makrobibliothek. Die Makrobibliothek muß mit einem /SET-FILE-LINK-Kommando zugewiesen sein.

### Bindemodulbibliotheken (OML)

Bindemodulbibliotheken sind Dateien im PAM-Format. Sie nehmen die von Sprachübersetzern erzeugten Bindemodule aus dem EAM-Bereich als Element mit dem Typ R auf.

Der Binder und der dynamische Bindelader können Elemente aus Bindemodulbibliotheken lesen. Es können bis zu 3380 Module aufgenommen werden. Die Anzahl der CSECTs/ENTRYs/COMMONs ist beschränkt (zwischen 380 und 800, je nach Belegung der Bibliothek).
Die Bindemodulbibliothek kann 32500 PAM-Seiten nicht überschreiten.

### Sequentielle Bibliotheken (Archivbibliotheken)

Sequentielle Bibliotheken befinden sich auf Magnetbändern. Diese Bibliotheken sind Banddateien mit Standardkennsätzen und einer Blockgröße von 2048 Byte.

LMS bearbeitet sequentielle Bibliotheken mit der Zugriffsmethode BTAM.

In einer sequentiellen Bibliothek können BS1000-Phasen, Bindemodule, Makros und Quellprogramme (bzw. Prozeduren und andere Texte) stehen. Elemente des gleichen Typs werden in je einem Bibliotheksabschnitt zusammengefaßt.
Die Reihenfolge der Abschnitte ist zwingend:
BS1000-Phasen, Bindemodule, Makros, Quellprogramme.

Die maximale Satzlänge für Bindemodule, Makros und Quellprogramme beträgt 80 Byte.

Das Inhaltsverzeichnis sequentieller Bibliotheken unterscheidet sich von dem anderer Bibliotheken:
Vor jedem Element steht ein Block, in dem die Elementbezeichnung des Elementes steht.

In eine sequentielle Bibliothek können mehrere Elemente des gleichen Typs mit dem gleichen Namen geschrieben werden. Wenn sich die Versionsnummer und/oder das Datum unterscheiden, können sie mit LMS einzeln gelesen werden.

Für sequentielle Bibliotheken gelten folgende Einschränkungen:

- Sie können nicht mit DEL und NAM bearbeitet werden.
- Listenelemente können nicht in Bandbibliotheken aufgenommen werden, da sie eine Satzlänge > 80 Zeichen haben.
  Längere Sätze werden abgeschnitten.
- Die Erstellung von Folgebändern ist im BS2000 nicht möglich.
  Im BS1000 erstellte Folgebänder können jedoch verarbeitet werden (siehe Seite 341).

LMS trägt keine Dateischutzmerkmale für Bibliotheken bzw. Elemente ein. Die Angabe einer Schutzfrist (RETENTION-PERIOD) im /SET-FILE-LINK-Kommando wird von LMS beim Anlegen einer Bibliothek in den Dateianfangskennsatz (HDR1) übernommen.

### Richtlinien für Elementbezeichnungen in sequentiellen Bibliotheken

elementname maximal 8 Zeichen

Zeichenvorrat
Buchstaben : A-Z
Sonderzeichen : $ # @ & % - (Bindestrich) _ (Unterstreichung)
Ziffern : 0-9

Das erste Zeichen muß ein Buchstabe, $, # oder @ sein.
Ausnahme: BS1000-Jobmakros

version genau dreistellige Versionsbezeichnung

Zeichenvorrat
Buchstaben : A-Z
Sonderzeichen : keine
Ziffern : 0-9

Buchstaben dürfen nur als erstes Zeichen verwendet werden.

### Behandlung des Datums in der Elementbezeichnung

Zur Elementauswahl kann auch das Benutzerdatum herangezogen werden, das für jedes Element geführt wird. Beim Erstellen eines Elementes kann das Datum vom Benutzer vorgegeben werden.
Standardmäßig setzt LMS das aktuelle Tagesdatum ein. Durch die Angabe DATE wird das Tagesdatum anstelle von datum eingesetzt.

Richtlinien für die Vergabe des Datums in sequentiellen Bibliotheken

```
datum          6 Zeichen:  zzzzzz
               Bedeutung:  JJMMTT
                           JJ  Jahreszahl
                           MM  Monat
                           TT  Tag
```

### Behandlung der Version

Alle Korrekturfunktionen (COR, EDT, EDR, UPD) erhöhen bei dem korrigierten Element die Variantennummer um 1. Dies gilt auch für die Numerierungsfunktion (NUM).

# Was enthält eine Programmbibliothek?

LMS bearbeitet in Programmbibliotheken folgende Elementtypen:

| Typ | Inhalt des Elementes |
|---|---|
| S | Quellprogramme |
| M | Makros |
| R | Bindemodule |
| J | Prozeduren |
| P | druckaufbereitete Daten |
| C | Lademodule |
| D | Textdaten |
| X | Daten beliebigen Formates |
| H | Compiler-Ergebnisinformationen |
| L | Bindelademodule (LLM) |
| F | IFG-Formatmasken |
| U | IFG-Benutzer-Profile |

Mit der Angabe Stern '*' als Elementtyp können bei DEL, DUP, LST, NAM und TOC alle Elementtypen angesprochen werden, die in der Bibliothek gespeichert sind. Möchte man alle Elemente einer Programmbibliothek auflisten, ist entweder TOC oder TOC* */* anzugeben.

Ein Stern '*' als Elementtyp kann nur bei der Bearbeitung von Programm-, Quellprogramm-, Makro- und Bindemodulbibliotheken angegeben werden. Für Bandbibliotheken gilt der Typ '*' nicht.

## Beschreibung der Elementtypen

Die Satzlänge der Elemente in Programmbibliotheken beträgt maximal 32 KByte (inkl. Satzkopf). Der EDT bearbeitet Textelemente mit einer maximalen Satzlänge von 256 Byte, der EDOR solche mit einer maximalen Satzlänge von 244 Byte.

### Elementtyp S - Quellprogramme

Quellprogramme in Bibliotheken dienen den Compilern und dem Assembler bei Übersetzungsläufen als Eingabe.

### Elementtyp M - Makros

Der Assembler entnimmt der zugewiesenen Bibliothek die im Programm angesprochenen Makroelemente.

### Elementtyp R - Bindemodule

Von Compilern und dem Assembler erzeugte Bindemodule werden standardmäßig im temporären EAM-Bereich abgelegt. Diese Bindemodule können mit LMS in einer Programmbibliothek als Elemente vom Typ R abgelegt werden. Wahlweise können die von Compilern und dem Assembler erzeugten Bindemodule auch direkt in einer Programmbibliothek abgelegt werden.

Dem Binder TSOSLNK und dem dynamischen Bindelader DBL dienen diese Elemente als Eingabe.

### Elementtyp J - Prozeduren

In diesem Elementtyp werden BS2000-Prozeduren und LMS-Anweisungen abgelegt.

Mit CTL (siehe Seite 118) kann gesteuert werden, daß LMS-Anweisungen direkt aus Prozedurelementen gelesen werden. Diese Elemente dürfen Sätze beliebiger Länge haben, bei BS2000-Prozeduren ist jedoch zu beachten, daß Sätze mit mehr als 80 Byte nicht unterstützt werden.

Der Aufruf von BS2000-Prozeduren aus einem Element vom Typ J direkt aus der Bibliothek ist von der Systemumgebung abhängig.

*Hinweis*

Auftragsschalter 1 muß gesetzt sein, wenn LMS in einer DO- oder CALL-Prozedur aufgerufen wird und dabei die Anweisungen ebenfalls aus der Prozedur lesen soll.

### Elementtyp P - Listenelemente

Als Listenelemente werden druckaufbereitete Daten bezeichnet. Das erste Zeichen des Satzes muß ein gültiges Vorschubsteuerzeichen sein; dies wird bei der Ausgabe in die Systemdatei SYSLST geprüft.
Elemente dieses Typs können mit ADDP (siehe Seite 91), DUPP (siehe Seite 122), EDTP/EDRP (siehe Seite 128) und PRT (siehe Seite 150) erzeugt werden.

Listenelemente werden unter Berücksichtigung des Vorschubsteuerzeichens mit LST ausgedruckt, wenn vorher PRT (LST) gegeben wurde.

### Elementtyp C - Lademodule

Ein vom Binder TSOSLNK erzeugter Lademodul wird standardmäßig in einer Datei abgelegt. Diese Datei kann mit LMS in einer Programmbibliothek als Element vom Typ C abgelegt werden. Wahlweise können die vom Binder erzeugten Lademodule auch direkt in einer Programmbibliothek abgelegt werden.

Ein in der Programmbibliothek gespeicherter Lademodul dient dem statischen Lader ELDE als Eingabe und wird wie folgt aufgerufen:

```
/START-PROGRAM FROM-FILE(LIB=bibliothek,ELEM=element[,(VER[SION]=version)]
```

### Elementtyp D - Textdaten

In Elemente vom Typ D kann beliebiger Text geschrieben werden. Es sind die gleichen Funktionen wie bei Elementtyp S möglich.

### Elementtyp X - Daten beliebigen Formates

In Elemente vom Typ X können ISAM-, SAM- und PAM-Dateien aufgenommen werden.

### Elementtyp H - Compiler-Ergebnisinformationen

Elemente dieses Typs werden von den Compilern und vom Assembler erzeugt und in Programmbibliotheken abgelegt. Näheres können Sie den entsprechenden Benutzerhandbüchern entnehmen.

### Elementtyp L - LLMs

In Elemente vom Typ L legt der Binder BINDER [2] die erzeugten Bindelademodule (LLMs) ab. Dieser Elementtyp wird ab der BS2000 Version 10.0A unterstützt.

### Elementtyp F - IFG-Formatmasken

Elemente dieses Typs werden zukünftig von IFG erzeugt und in Programmbibliotheken abgelegt. Dieser Elementtyp kann jedoch nicht von LMS erzeugt werden.

### Elementtyp U - IFG-Benutzer-Profile

Elemente dieses Typs werden in Zukunft von IFG erzeugt und in Programmbibliotheken abgelegt. Dieser Elementtyp kann jedoch nicht von LMS erzeugt werden.

### Elementbezeichnung

Elemente sind in Programmbibliotheken über ihren Elementtyp und ihre Elementbezeichnung einzeln ansprechbar.

Die Elementbezeichnung setzt sich zusammen aus Name, Version und Datum und wird in folgender Form angegeben:

```
elementname[/version[/datum]]
```

oder

```
elementname[//datum]
```

Die Angabe der Version und des Datums ist wahlfrei. Wird in einer Anweisung kein Wert für version eingegeben, wird standardmäßig das Element mit der höchsten Version ausgewählt. Wird in einer Anweisung kein Wert für datum eingegeben, wird standardmäßig das aktuelle Tagesdatum eingesetzt.

Die Elementbezeichnung wird als Operand angegeben:

```
operationx  elem
```

| | |
|---|---|
| operation | Name der Anweisung |
| x | Elementtyp |
| elem | Elementbezeichnung, bestehend aus elementname und wahlweise version und datum |

#### Richtlinien für Elementbezeichnungen in Programmbibliotheken

elementname maximal 64 Zeichen

Zeichenvorrat

| | | |
|---|---|---|
| Buchstaben | : | A-Z |
| Sonderzeichen | : | $ # @ . (Punkt) - (Bindestrich) _ (Unterstreichung) |
| Ziffern | : | 0-9 |

Die Zeichen Bindestrich, Unterstreichung und Punkt dürfen nicht erstes oder letztes Zeichen sein, und zwei gleiche der eben genannten Sonderzeichen dürfen nicht unmittelbar nebeneinander stehen.
Der Bindestrich darf nicht unmittelbar rechts neben einem der Zeichen $, @, #, Unterstreichung und Punkt stehen. Der Elementname muß mindestens einen Buchstaben oder eines der Sonderzeichen @, #, $ enthalten.

version maximal 24 Zeichen

Zeichenvorrat
Buchstaben : A-Z
Sonderzeichen : . (Punkt) - (Bindestrich) @ (Klammeraffe)
Ziffern : 0-9

Die Sonderzeichen Punkt und Bindestrich dürfen nicht erstes oder letztes Zeichen sein. Gleiche Sonderzeichen dürfen nicht unmittelbar nebeneinander stehen.
Der Bindestrich darf nicht unmittelbar rechts von einem Punkt stehen.

Wird der Klammeraffe explizit angegeben, darf sonst kein weiteres Zeichen in der Versionsbezeichnung angegeben werden.

Wird beim schreibenden Zugriff in einer Elementbezeichnung keine Angabe zur Version gemacht, wird standardmäßig @ eingesetzt. Der Klammeraffe bedeutet die größtmögliche Version. Gibt der Benutzer explizit eine Version an, die mit Vz. (z=Ziffer) beginnt, wird der Ziffer eine 0 vorangestellt (also V0z). Das soll gewährleisten, daß z.B. V9.x=V09.x <V10.x ist.

Wird beim lesenden Zugriff in einer Elementbezeichnung keine Angabe zur Version gemacht, wird die höchste vorhandene Version gelesen.

### Behandlung des Datums in der Elementbezeichnung

Zur Elementauswahl kann auch das Benutzerdatum herangezogen werden, das für jedes Element geführt wird. Beim Duplizieren und beim Umbenennen eines Elementes wird das Datum des ursprünglichen Elementes übernommen, falls nicht explizit ein neues Datum angegeben wurde.
Bei der Neuerstellung eines Elementes setzt LMS standardmäßig das aktuelle Tagesdatum ein, falls nicht explizit ein anderes Datum angegeben wurde.

### Richtlinien für die Vergabe des Datums bei Programmbibliotheken

datum

```
10 Zeichen:  zzzz-zz-zz
Bedeutung :  JJJJ-MM-TT
             JJJJ  Jahreszahl
             MM    Monat
             TT    Tag
```

Beim Übertragen von Elementen aus anderen Bibliotheken wird das Datum standardmäßig in dieses Format umgewandelt.
Standardwert: aktuelles Tagesdatum
Durch die Angabe DATE wird das Tagesdatum anstelle von datum eingesetzt.

### Behandlung der Version und der Variantennummer

Die bei Quellprogramm-, Makro- und Bindemodulbibliotheken übliche automatische Versionsfortschaltung bei den Korrekturfunktionen und der Numerierungsfunktion ist bei den Programmbibliotheken nicht mehr gegeben.

Sie führen statt dessen eine Variantennummer (max. vierstellig, numerisch), die die Funktion eines Schreibzugriffszählers erfüllt. Pro Element gibt es jeweils nur eine Variante. Wird die Variantennummer 9999 erreicht, muß das Element zur weiteren Bearbeitung kopiert werden.

Bei den Programmbibliotheken erfolgt keine Versionsfortschaltung, sondern die Erhöhung der Variante eines Elementes um 1 bei jedem schreibenden Zugriff auf das Element. Das heißt, daß bei ADD, COR, EDT, EDR, UPD und NUM die Variante hochgezählt wird, wenn dabei ein Element gleichen Typs, Namens und gleicher Version überschrieben wird.

### Protokollierung der Elementbezeichnung

Elementbezeichnungen werden wie folgt protokolliert:

```
(Typ)Elementname/Version[(Variantennummer)]
```

Die Variantennummer wird nur bei Elementen in Programmbibliotheken geführt. Sie ist standardmäßig auf (0001) gesetzt und wird durch ADD, NUM, UPD, COR, EDT und EDR um jeweils 1 erhöht.

Bei TOC wird zusätzlich das Datum ausgegeben:

```
(Typ)Elementname/Version[(Variantennummer)]/Datum
```

# Was enthalten typbezogene Bibliotheken?

Typbezogene Bibliotheken enthalten jeweils nur einen Elementyp.

| Typ | Inhalt der Elemente | Bibliothek |
|---|---|---|
| S | Quellprogramme, Prozeduren, Protokolle und Textdaten | Quellprogrammbibliothek |
| M | Makros | Makrobibliothek |
| R | Bindemodule | Bindemodulbibliothek |

## Beschreibung der Elementtypen

Die Elementtypen S und M haben eine maximale Satzlänge von 251 Byte, der Elementtyp R hat eine maximale Satzlänge von 80 Byte.

### Elementtyp S

Dieser Elementtyp enthält Quellprogramme, Prozeduren, Protokolle und Textdaten.

Die Elemente können jedoch auch mit dem Elementtyp P, D oder J angesprochen werden. Dadurch ist es möglich, die Elemente einer bestehenden Quellprogrammbibliothek auf die entsprechenden Elementtypen einer Programmbibliothek zu verteilen.

Quellprogramme in Quellprogrammbibliotheken dienen Compilern als Eingabe.

Soll ein Protokoll unter Berücksichtigung der Vorschubsteuerzeichen mit LST ausgedruckt werden, muß vorher PRT (LST) und der Verarbeitungsoperand PAR FORMAT=P angegeben werden.

### Elementtyp M

Der Assembler entnimmt der zugewiesenen Makrobibliothek die im Programm angesprochenen Makroelemente.

### Elementtyp R

Die von den Compilern erzeugten Bindemodule werden in der Bindemodulbibliothek gespeichert. Sie dienen dem Binder TSOSLNK und dem dynamischen Bindelader DBL als Eingabe.

### Elementbezeichnung

Über die Elementbezeichnung kann jedes Element in einer Bibliothek einzeln angesprochen werden. Die Elementbezeichnung setzt sich zusammen aus Name, Version und Datum und wird in folgender Form angegeben:

```
elementname[/version[/datum]] oder
elementname[//datum]
```

Die Angabe der Version und des Datums ist wahlfrei. Wird in einer Anweisung kein Wert für version eingegeben, wird standardmäßig das Element mit der höchsten Version ausgewählt. Wird in einer Anweisung kein Wert für datum angegeben, wird standardmäßig das aktuelle Tagesdatum eingesetzt.

Die Elementbezeichnung wird als Operand angegeben:

```
operationx  elem
```

| | |
|---|---|
| operation | Name der Anweisung |
| x | Elementtyp |
| elem | Elementbezeichnung bestehend aus elementname und wahlweise version und datum |

**Richtlinien für Elementbezeichnungen in Quellprogramm-, Makro- und Bindemodulbibliotheken**

elementname: maximal 8 Zeichen

Zeichenvorrat
Buchstaben : A-Z
Sonderzeichen : $ # @ - (Bindestrich) _ (Unterstreichung)
Ziffern : 0-9

Das erste Zeichen muß ein Buchstabe, $, # oder @ sein.

version: genau dreistellige Versionsbezeichnung

Zeichenvorrat
Buchstaben : A-Z
Sonderzeichen : keine
Ziffern : 0-9

Buchstaben dürfen nur als erstes Zeichen verwendet werden.

### Behandlung des Datums in der Elementbezeichnung

Zur Elementauswahl kann auch das Benutzerdatum herangezogen werden, das für jedes Element geführt wird. Beim Erstellen und beim Umbenennen eines Elementes kann das Datum vom Benutzer vorgegeben werden.
Standardmäßig setzt LMS das aktuelle Tagesdatum ein. Durch die Angabe DATE wird das Tagesdatum anstelle von datum eingesetzt.

Richtlinien für die Vergabe des Datums bei Quellprogramm-, Makro- und Bindemodulbibliotheken

```
datum          6 Zeichen:  zzzzzz
               Bedeutung:  JJMMTT
                           JJ  Jahreszahl
                           MM  Monat
                           TT  Tag
```

### Behandlung der Version

Quellprogramm-, Makro- und Bindemodulbibliotheken:

Alle Korrekturfunktionen (COR, EDT, EDR, UPD) erhöhen bei dem korrigierten Element die Versionsnummer um 1. Dies gilt auch für die Numerierungsfunktion (NUM).

# Auswahlangabe für Elementbezeichnungen

Die Elementbezeichnung kann genau ein Element ansprechen, sie kann jedoch auch variabel gestaltet werden und somit mehrere Elemente für die Verarbeitung auswählen. Diese Elementbezeichnung bezeichnet man als Auswahlangabe.

Um Elementbezeichnungen variabel gestalten zu können, bietet LMS Symbole an, die in der Auswahlangabe folgende Bedeutung haben:

| Symbol | Bedeutung in der Auswahlangabe |
|---|---|
| ' (Apostroph) | Beliebiges erlaubtes Zeichen in Elementname, Version und Datum. Alle Elemente werden ausgewählt, die an der entsprechenden Position in der Elementbezeichnung ein beliebiges Zeichen haben.<br><br>Ein Apostroph steht für genau ein Zeichen.<br><br>Für abschließende Apostrophe kann auch das Leerzeichen stehen, d.h. die ausgewählten Namen können auch kürzer als die Auswahlangabe sein. |
| * (Stern) | Der Stern als Elementtyp:<br><br>Die Anweisung bezieht sich auf alle Elementtypen der zugewiesenen Eingabebibliothek.<br><br>Der Stern in der Elementbezeichnung:<br><br>- im Elementnamen<br>Der Stern kann in der Kombination mit anderen Zeichen oder als einzelnes Zeichen stehen. Wird er mit anderen Zeichen kombiniert, muß er das letzte Zeichen des Namens sein. Ab dieser Stelle der Zeichenkette darf der Name beliebige Länge und Inhalt haben. Ist * einziges Zeichen darf der Elementname beliebige Länge und Inhalt haben.<br>- als Version<br>Die Anweisung bezieht sich auf alle Versionen des angegebenen Elementnamens.<br><br>Wird * als einziges Zeichen für die gesamte Elementbezeichnung angegeben, bezieht sich die Anweisung auf alle Elemente der jeweils höchsten Version. |

| Symbol | Bedeutung in der Auswahlangabe |
|---|---|
| < (kleiner)<br>> (größer)<br>= (gleich)<br># (steht für ungleich | Mit diesen Symbolen können Grenzwerte für Version und Datum zur Auswahl von Elementen verwendet werden.<br><br>Eines dieser Zeichen wird jeweils zwischen den Schrägstrich und die Version, bzw. zwischen den Schrägstrich und das Datum gesetzt. Zusammen mit den angegebenen Werten für Version und Datum grenzen sie die Auswahl der zu verarbeitenden Elemente ein. |
| -elem | Diese Angabe ist als 'minus Element' zu verstehen. Sie schließt ein Element, das mit der vorangegangenen Auswahlangabe mit ausgewählt worden wäre, von der Verarbeitung aus.<br><br>elem darf auch in diesem Fall eine Auswahlangabe sein, muß aber ein Kennzeichen haben, das eine Gruppe von Elementen oder ein Element innerhalb der vorangegangenen Auswahlangabe anspricht.<br>Zwischen Auswahlangabe und -elem muß ein Komma stehen. |

*Beispiel für die Auswahlangabe*

- Auswahlangabe

  AB'C* Alle Elemente, deren Name mit AB beginnt, an der 3. Stelle ein beliebiges Zeichen, an der 4. Stelle ein C besitzt, werden ausgewählt. Ab der 5. Stelle kann der Elementname beliebigen Inhalt haben.

  "'/B* Alle Elemente mit einer Namenslänge von maximal 3 Zeichen, die an der ersten Stelle der Versionsnummer ein B besitzen, werden ausgewählt.

  */*/>1983* Alle Elemente, die seit dem 1.1.84 eingetragen wurden, werden ausgewählt.

  AB//* Alle Elemente AB der höchsten Version mit beliebigem Datum werden ausgewählt.

- Auswahlangaben mit Grenzwerten

  */>402 Alle Elemente, deren Versionsnummer größer als 402 ist, werden ausgewählt.

  A*//<1982*
  Alle Elemente der höchsten Version, deren Name mit A beginnt, und die älteren Datums als 1.1.1982 sind, werden ausgewählt.

  A*/#B* Alle Elemente, deren Name mit A beginnt, und deren Versionsnummer nicht mit B beginnt, werden ausgewählt.

  AB'/=107 Alle Elemente, deren Name mit AB beginnt, maximal 3 Zeichen lang ist und die die Versionsnummer 107 haben (entspricht der Angabe AB'/107), werden ausgewählt.

- Auswahlangaben mit auszuschließenden Elementen

  AB*,-ABC,C*
  Alle Elemente, deren Name mit AB oder mit C beginnt, außer dem Element ABC, werden ausgewählt.

  L'',-L''/001
  Alle Elemente, deren Name mit L beginnt und maximal 4 Zeichen lang ist, außer denen, die Versionsnummer 1 haben, werden ausgewählt.

# Konstruktionsangabe für Elementbezeichnungen

Einige LMS-Anweisungen benutzen außer der Bezeichnung des zu bearbeitenden Elementes elem noch eine weitere Elementbezeichnung elemu, die hinter einem Trennzeichen (,>=) in der Anweisung angegeben wird. elemu bezeichnet in der Anweisung

COM die Elementbezeichnung des zum Vergleich heranzuziehenden Elementes (=elemu),

ADD, DUP, NAM
die Elementbezeichnung für das neue Element (>elemu).

Diese Elementbezeichnungen können durch die Verwendung von Konstruktionsangaben variabel gestaltet werden.
Die variablen Stellen in der Konstruktionsangabe werden aus dem bearbeiteten Element elem übernommen. LMS verwendet bei den Konstruktionsangaben den Formalismus der Auswahlangabe.

Bei den Konstruktionsangaben haben die in der Elementbezeichnung elemu angegebenen Symbole folgende Bedeutung:

| Symbol | Bedeutung in der Konstruktionsangabe |
|---|---|
| ' (Apostroph) | Die in der Konstruktionsangabe mit Apostroph gekennzeichnete Stelle wird mit dem Zeichen, das an der entsprechenden Stelle in der Elementbezeichnung elem steht, besetzt.<br><br>Ein Apostroph definiert eine Position in der Elementbezeichnung. Steht der Apostroph an letzter Stelle der Konstruktionsangabe, werden nicht mehr Zeichen aus der Elementbezeichnung elem übernommen. |
| * (Stern) | Der Stern kann nur als letztes Zeichen in der Konstruktionsangabe verwendet werden. Er bedeutet, daß ab dieser Stelle alle Zeichen der Elementbezeichnung elem übernommen werden.<br><br>Wenn * als einziges Zeichen angegeben wird, wird die gesamte Elementbezeichnung elem übernommen. |

Alle anderen Stellen werden nicht verändert.

*Beispiel für Konstruktionsangaben*

In einer Bibliothek sind 3 Elemente mit den Namen

1. ABC/001
2. ABCD/234
3. ABCDE/101

| Auszuführende Anweisungen | von LMS gebildete Namen |
|---|---|
| NAMS ABC>''X | 1. ABX/001<br>2. wird nicht umbenannt<br>3. wird nicht umbenannt |
| NAMS AB*>XY*/A02 | 1. XYC/A02<br>2. XYCD/A02<br>3. XYCDE/A02 |
| DUPS AB''>'X'Y/A* | 1. AXCY/A01<br>2. AXCY/A34<br>3. wird nicht dupliziert |

*Hinweis*

Bei Verwendung von Auswahl- und Konstruktionsangabe ist zu beachten:

- Möglicherweise werden verschiedene Eingabebezeichnungen auf dieselbe Ausgabebezeichnung abgebildet. Je nachdem, wie der Verarbeitungsoperand OVERWRITE gesetzt ist, werden die verschiedenen Daten überschrieben (Beispiel: NAMx A*>B).
- Wird durch eine Konstruktionsangabe eine lexikographisch höhere Elementbezeichnung erzeugt, die wiederum der Auswahlangabe genügt, findet LMS dieses Element bei der sequentiellen Abarbeitung des Inhaltsverzeichnisses erneut.
- Kommt eine Auswahlangabe (auch über Teillisten hinweg) mehrmals vor, wird nur die letzte Auswahlangabe bearbeitet, z.B.

  ```
  NAMS A>B,A>C
  ```

  Nur A>C wird bearbeitet.

# Funktionen von LMS

Dieses Kapitel gibt einen Überblick über die Funktionen von LMS. Die Funktionen werden durch Anweisungen ausgelöst.
Verarbeitungsoperanden steuern und beeinflussen sowohl die Anweisungen als auch den gesamten LMS-Lauf. Auftragsschalter, die beim Aufruf von LMS gesetzt waren, beeinflussen den LMS-Lauf ebenfalls. LMS-Anweisungen siehe ab Seite 81, Verarbeitungsoperanden siehe ab Seite 205.

Nach dem Aufrufen von LMS sind alle Verarbeitungsoperanden auf Standardwerte gesetzt. Werden für bestimmte Anweisungen andere Werte von Verarbeitungsoperanden benötigt, muß der Verarbeitungsoperand gesetzt werden, bevor die Anweisung eingegeben wird.

Sollen außer Fehlermeldungen auch die Ausführung jeder LMS-Anweisung (Erfolgsmeldungen) protokolliert werden, muß der Wert des Verarbeitungsoperanden LOG=MED oder LOG=MAX gesetzt werden.

Um Bibliotheken zu bearbeiten, müssen sie zuerst zugewiesen werden. Zugewiesene Bibliotheken können bereits bestehen oder neu eingerichtet werden. Erst nach dem erfolgreichen Zuweisen können Elemente aufgenommen und/oder bearbeitet werden.

# Zuweisen von Bibliotheken

Alle LMS-Funktionen benötigen eine Ein- oder Ausgabebibliothek oder auch beides. Aus der Eingabebibliothek liest LMS Elemente, in die Ausgabebibliothek gibt LMS Elemente aus.

Die LMS-Anweisung LIB weist eine Bibliothek als Ein-, Aus- oder Ein- und Ausgabebibliothek zu. Diese Zuweisung gilt bis zu einem weiteren LIB oder bis zum Ende des LMS-Laufes.

## Zuweisen von Bibliotheken mit LIB

LIB bestimmt,

- welche Bibliothek zugewiesen oder geschlossen wird;
- ob die Bibliothek als Ein-, Aus- oder Ein- und Ausgabebibliothek zugewiesen wird;
- ob die Bibliothek eine Programm-, Quellprogramm-, Makro- oder Bindemodulbibliothek ist;
- ob die Bibliothek bereits besteht oder neu eingerichtet werden muß.

Die Bibliotheken werden bei LIB mit dem Dateinamen der Bibliothek, mit dem Dateikettungsnamen oder mit der Bibliothekskurzbezeichnung angesprochen.

### Dateikettungsname

Soll ein Dateikettungskettungsname (Linkname) für die Bibliothek verwendet werden, muß vor dem Aufruf von LMS ein /SET-FILE-LINK-Kommando gegeben werden, das die Verbindung zum Dateinamen der Bibliothek herstellt:

```
/SET-FILE-LINK FILE-NAME=bibliotheksname, LINK-NAME=dateikettungsname
```

*Beispiel*

```
/SET-FILE-LINK FILE-NAME=BIB,LINK-NAME=TEST
/START-PROGRAM $LMS
$LIB TEST,IN
.
.
.
```

Der Bibliothek A.BIB wird der Dateikettungsname TEST zugewiesen und im LMS-Lauf mit dem Dateikettungsnamen angesprochen und als Eingabebibliothek geöffnet.

### Bibliothekskurzbezeichnung

Mit der Bibliothekskurzbezeichnung kann für eine Anweisung temporär eine andere Eingabebibliothek zugewiesen werden. Diese Zuweisung gilt für diese Anweisung, anschließend gilt wieder die Zuweisung durch LIB.

Mit LIB zugewiesene Bibliotheken gelten solange als Ein- bzw. Ausgabebibliotheken, bis mit LIB neue Bibliotheken zugewiesen werden.
Falls keine andere Bibliothek zugewiesen wird, gelten die Zuweisungen bis zum Ende des LMS-Laufs.

LMS bietet aber auch die Möglichkeit, eine andere Eingabebibliothek für eine Anweisung zu eröffnen. In diesem Fall wird die Kurzbezeichnung der Bibliothek als Operand in der Anweisung angegeben. Diese Bibliothek gilt nur während der Ausführung dieser Anweisung.

In der nächsten Anweisung, die keine Bibliothekskurzbezeichnung als Operand enthält, gilt wieder die mit LIB zugewiesene Eingabebibliothek.

Die Bibliothekskurzbezeichnung wird in einem /SET-FILE-LINK-Kommando mit dem Dateikettungsnamen LIBlib vereinbart:

```
/SET-FILE-LINK FILE-NAME=bibliotheksname, LINK-NAME=LIBlib
```

- Für lib werden die 3 Ziffern verwendet, die in den LMS-Anweisungen die Kurzbezeichnung darstellen.
- Führende Nullen dürfen in den Anweisungen weggelassen werden, im /SET-FILE-LINK-Kommmando jedoch nicht.
- lib ist in den LMS-Anweisungen in Klammern zu setzen.

*Beispiel 1*

```
/SET-FILE-LINK FILE-NAME=BIB,LINK-NAME=LIB001
/START-PROGRAM $LMS
$LIB (1),OUT
.
.
.
```

Der Bibliothek B.BIB wird die Bibliothekskurzbezeichnung (1) zugewiesen und im LMS-Lauf mit der Kurzbezeichnung angesprochen und als Ausgabebibliothek geöffnet.

*Beispiel 2*

```
/SET-FILE-LINK FILE-NAME=EIN.BIB,LINK-NAME=LIB001   ]
/SET-FILE-LINK FILE-NAME=AUS.BIB,LINK-NAME=LIB002   } ——————————— (01)
/SET-FILE-LINK FILE-NAME=PLA1.BIB,LINK-NAME=LIB003  }
/SET-FILE-LINK FILE-NAME=PLA2.BIB,LINK-NAME=LIB004  ]
/START-PROGRAM $LMS ———————————————————————————————————————————— (02)
$LIB (1),IN    ]
$LIB (2),OUT   } ———————————————————————————————————————————— (03)
$DUPS ELEM1>ELEM01]
$COMS ELEM3=ELEM4 } ———————————————————————————————————————————— (04)
$LSTS ELEMENT1(3)]
$TOCS (4)         } ———————————————————————————————————————————— (05)
$LSTS ELEM1       ]
$COMS ELEM1=ELEM3 } ———————————————————————————————————————————— (06)
$END ——————————————————————————————————————————————————————————— (07)
```

(01) Die Ein- und Ausgabebibliotheken werden mit dem /SET-FILE-LINK-Kommando zugewiesen, da sie im LMS-Lauf über Kurzbezeichnungen angesprochen werden.

(02) LMS wird aufgerufen.

(03) Die Bibliotheken, die standardmäßig bei den Anweisungen gelten sollen, werden mit LIB zugewiesen.

(04) Für diese Anweisungen, die keine Bibliothekskurzbezeichnung enthalten, gelten die mit LIB zugewiesenen Bibliotheken.

(05) Das Element ELEMENT1 aus der Bibliothek PLA1.BIB, für die die Kurzbezeichnung (3) im /SET-FILE-LINK-Kommando vereinbart wurde, wird aufgelistet; das Inhaltsverzeichnis der Bibliothek PLA2.BIB für Elemente des Typs S wird ausgegeben.

(06) Für diese Anweisungen gelten wieder die mit LIB zugewiesenen Bibliotheken.

(07) LMS wird beendet.

Die Beschreibung der verwendeten Anweisungen finden Sie in alphabetischer Reihenfolge ab Seite 91.

### Sequentielle Bibliotheken

Sequentielle Bibliotheken müssen mit dem /SET-FILE-LINK-Kommando zugewiesen werden. Im /SET-FILE-LINK-Kommando ist der Operand ACCESS-METHOD=BTAM erforderlich. Falls die Bandbibliothek noch keinen Katalogeintrag hat und sie gelesen werden soll, muß vor dem /SET-FILE-LINK-Kommando ein IMPORT-FILE-Kommando abgesetzt werden:

```
/IMPORT-FILE SUPPORT=TAPE(VOLUME=archivnr, DEVICE-TYP=gerät, FILE-NAME=dateiname
/SET-FILE-LINK LINK-NAME=LIBlib, FILE-NAME=dateiname, ACCESS-METHOD=BTAM
```

Sequentielle Bibliotheken können nicht mit LIB eingerichtet werden. Sie müssen mit LIBOUT und der Kurzbezeichnung (siehe Seite 345) zugewiesen werden.

# Bearbeiten von Elementen

Das folgende Kapitel gibt einen Überblick über die Möglichkeiten, die LMS für die Elementbearbeitung bietet:

LMS kann Elemente

- sowohl als Voll- als auch als Delta-Elemente in Bibliotheken aufnehmen,
- in Dateien ausgeben,
- in andere Bibliotheken ausgeben (duplizieren),
- auflisten,
- löschen,
- numerieren,
- vergleichen,
- umbenennen,
- editieren,
- korrigieren und das Inhaltsverzeichnis der Bibliothek ausgeben.

Die Beschreibung aller in diesem Kapitel erwähnten Anweisungen finden Sie in alphabetischer Reihenfolge ab Seite 91.

### Aufnehmen von Elementen in eine Bibliothek

Folgende Anweisungen geben Elemente in die zugewiesene Ausgabebibliothek aus: ADD, COR, DUP, EDR, EDT, NUM, PRT, UPD.

Der Verarbeitungsoperand OVERWRITE bestimmt dabei, ob ein gleichnamiges Element in der Ausgabebibliothek überschrieben wird oder nicht.

**ADD** nimmt Dateien, Module aus dem EAM-Bereich, Elemente aus FMS-Bibliotheken und Datensätze aus dem LMS-Anweisungsstrom als Elemente in die zugewiesene Ausgabebibliothek auf. Mit dieser Anweisung legt der Benutzer zusätzlich fest, ob das Element als Voll- oder Delta-Element gespeichert wird.
In Programmbibliotheken können Dateien mit einer RECORD-SIZE bis max. 32 KByte aufgenommen werden.

Wenn eine sequentielle Bibliothek (Bandbibliothek) zugewiesen ist, setzt die ADDC-Anweisung BS2000-Programmdateien in BS1000-Phasen um und speichert sie als Elementtyp C.

Wird eine ISAM-Datei aufgenommen, bestimmt der Verarbeitungsoperand KEY, ob die ISAM-Schlüssel und andere Dateimerkmale mit aufgenommen werden. Es können dann ISAM-Schlüssel bis zu einer Länge von 255 Byte abgelegt werden.

Die ISAM-Schlüssel im Element mit aufzunehmen, ist vor allem zur Archivierung geeignet.

Ist der Verarbeitungsoperand KEY gesetzt, ist es auch möglich, Dateien mit RECORD-FORMAT=FIXED aufzunehmen, ansonsten ist nur RECORD-FORMAT=VARIABLE zulässig.

*Hinweis*

- Die ISAM-Schlüssel einer Quellprogrammdatei sollten nicht mit ins Element aufgenommen werden. Der Compiler kann das Quellprogramm, wenn ISAM-Schlüssel enthalten sind, aus diesem Element nicht fehlerfrei übersetzen.
- Wird die Systemdatei SYSDTA einem Element zugewiesen, das die ISAM-Schlüssel gespeichert hat, werden die ISAM-Schlüssel mitgelesen. Die ISAM-Schlüssel müssen dann vom verarbeitenden Programm selbst entfernt werden.

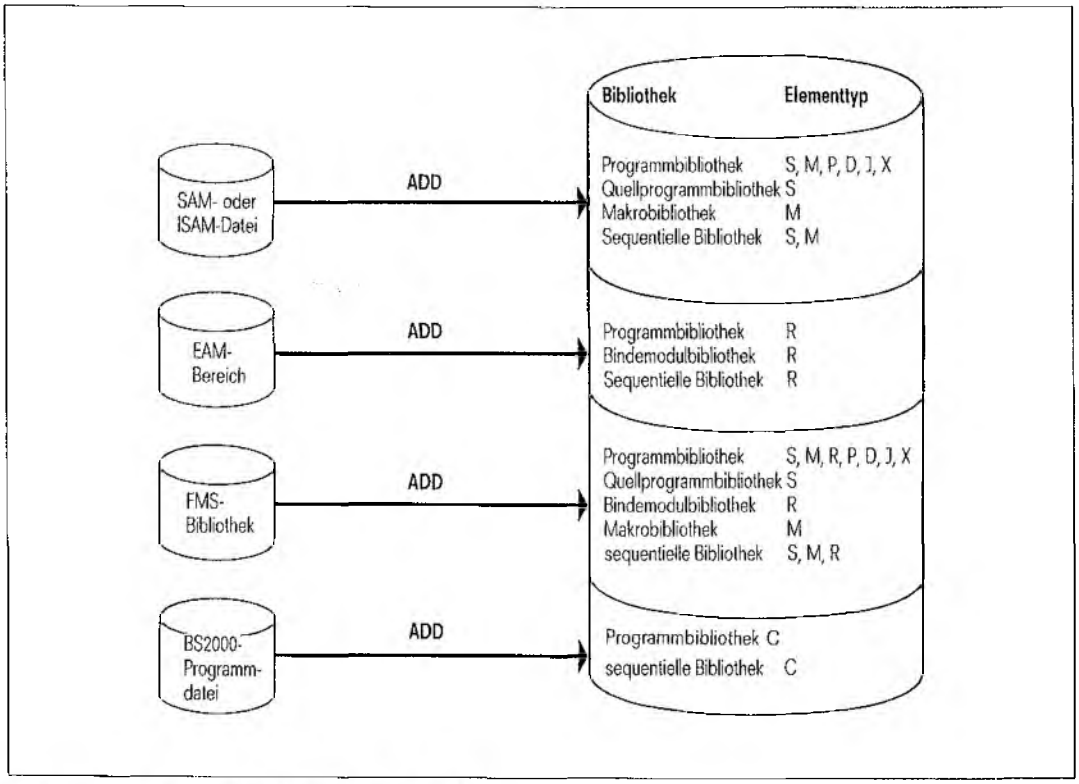



**Bild 4** **Aufnehmen von Elementen mit ADD**

**PRT** kann das LMS-Protokoll in ein Listenelement ablegen.

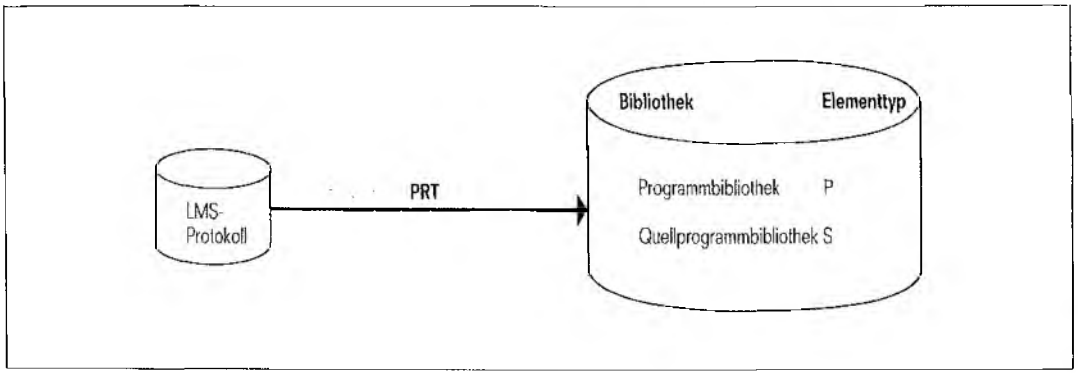



Bild 5 Aufnehmen von Elementen mit PRT

**DUP** dupliziert Elemente von der Eingabebibliothek in die Ausgabebibliothek und legt sie dort, falls gewünscht, mit einer anderen Elementbezeichnung ab. Mit dieser Anweisung legt der Benutzer zusätzlich fest, ob das Element als Voll- oder Delta-Element gespeichert wird.

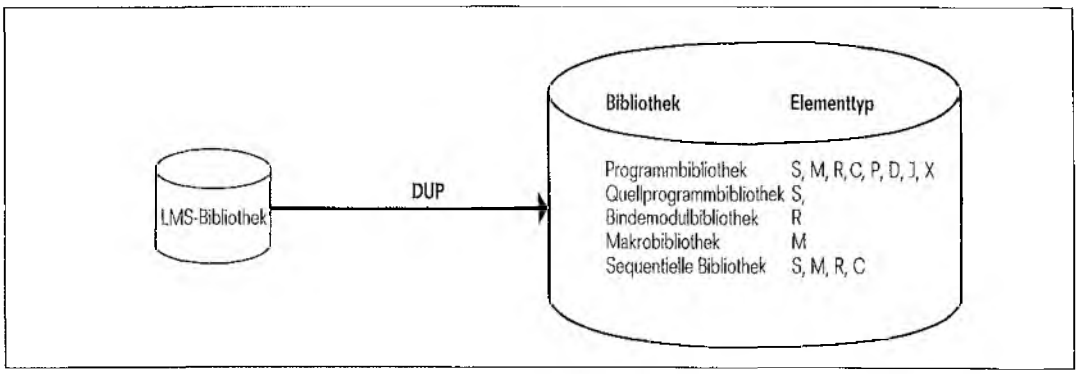



Bild 6 Aufnehmen von Elementen mit DUP

**EDT** bzw. **EDR** verzweigt in den EDT bzw. EDOR, um das angegebene Element der Eingabebibliothek zu bearbeiten. Bei entsprechender Beendigung des EDT bzw. EDOR wird das bearbeitete Element in die zugewiesene Ausgabebibliothek eventuell mit neuem Namen ausgegeben.

EDT bzw. EDR ohne Operanden verzweigt in den EDT bzw. EDOR, ohne ein Element in den Ausgabebereich einzulesen.

## Ausgeben von Elementen

Die Elemente einer Eingabebibliothek werden ausgegeben mit

- SEL in Dateien oder FMS-Bibliotheken;
- DUP in die Ausgabebibliothek.

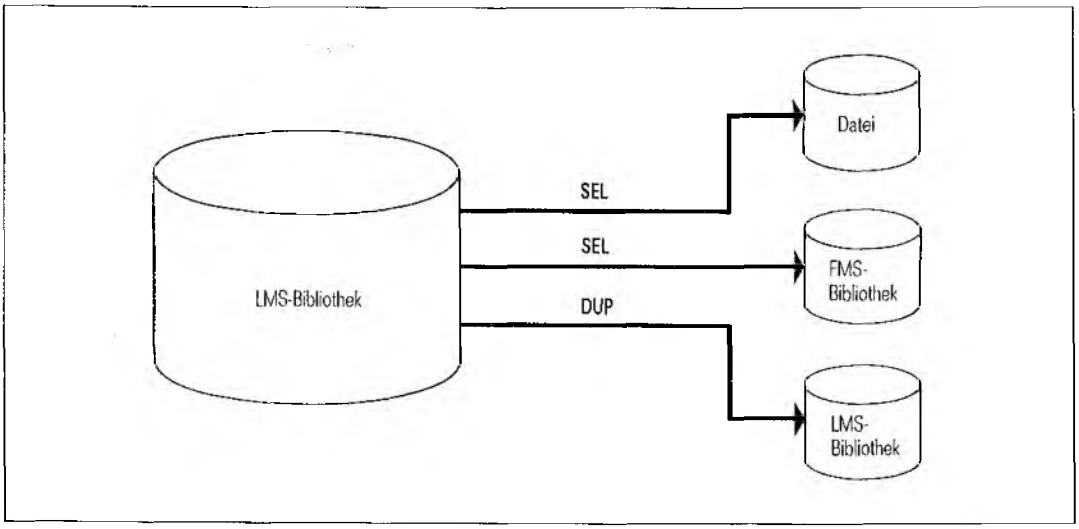



Bild 7 Ausgeben von Elementen

## Auflisten von Elementen

Das Auflisten von Elementen wird durch **LST**- und **PRT** gesteuert. LST bestimmt dabei die Elemente und die Bibliothek, deren Elemente ausgegeben werden sollen.
PRT bestimmt das Ausgabemedium.

Durch Verarbeitungsoperanden wird das Format und der Umfang der Ausgabe gesteuert.

## Löschen von Elementen

DEL löscht ein Element, mehrere oder alle Elemente in der Eingabebibliothek (Programm-, Quellprogramm-, Makro- oder Bindemodulbibliothek).

Bei Programmbibliotheken wird zwischen logischem und physikalischem Löschen unterschieden:

- Logisches Löschen
  Die Einträge im Inhaltsverzeichnis werden gelöscht und der Speicherplatz des entsprechenden Elementes wird freigegeben.
- Physikalisches Löschen
  Zusätzlich zum logischen Löschen wird der Speicherplatz des entsprechenden Elementes mit binären Nullen überschrieben.

Ein Element einer Programmbibliothek wird physikalisch gelöscht, wenn der Verarbeitungsoperand DESTROY=YES gesetzt ist oder im Element einer Programmbibliothek ein Kennzeichen für physikalisches Löschen vorhanden ist. Delta-Elemente werden erst dann physikalisch gelöscht, wenn das letzte Delta-Element eines Delta-Baums, d.h. der komplette Delta-Baum, gelöscht wird.

## Numerierung von Sätzen durch Satznummern und Kennungsfelder

Bei manchen LMS-Funktionen, z.B. Korrigieren von Elementen, ist es nötig, auf bestimmte Sätze zuzugreifen.

LMS bietet dazu zwei Kriterien:

- Satznummern
- Satzkennungen in Kennungsfeldern

### Satznummern

Die Satznummer ist die Position des Elementsatzes bezogen auf den Elementanfang. Die Satznummer wird beim Auflisten eines Elementes nach SYSLST, in das Element oder beim Vergleichsprotokoll (siehe Seite 51) mit ausgegeben. Sie wird als #zahl dargestellt.
zahl ist eine ganze positive Zahl mit maximal 8 Stellen.

Die Nummer des ersten Elementsatzes ist #1, die Nummer des zweiten Satzes ist #2, usw.

*Beispiel einer Elementauflistung*

```
#1>AAAAAAA
#2>BBBBBBB
#3>CCCCCCC
```

### Kennungsfelder

LMS kann in zu bearbeitenden Sätzen Kennungsfelder definieren und bearbeiten. Der Inhalt des Kennungsfeldes wird als Satzkennung bezeichnet.

Satzkennungen identifizieren einzelne Sätze, z.B. beim Korrigieren von Elementen (siehe Seite 53) oder beim Auswerten des Vergleichsprotokolls (siehe Seite 51).

Eine Satzkennung kann sein:

- eine alphabetische Zeichenfolge
- eine numerische Zeichenfolge
- eine Kombination aus beidem
- Leerzeichen

Beim Lesen von Sätzen aus einer Eingabebibliothek oder einer Datei, kann

- ein Kennungsfeld definiert werden (Verarbeitungsoperand CHECK),
- die enthaltene Satzkennung auf aufsteigende Reihenfolge kontrolliert werden (Verarbeitungsoperand CHECK).

Beim Schreiben der Sätze in ein Element kann

- ein Kennungsfeld definiert werden (Verarbeitungsoperand RANGE),
- die Satzkennung mit Zeichenfolgen (Verarbeitungsoperand STRING) und aufsteigender Numerierung (Verarbeitungsoperand VALUE) erzeugt werden,
- wenn es sich um Sätze aus einer ISAM-Datei handelt, der ISAM-Schlüssel als Satzkennung abgelegt werden (Verarbeitungsoperand STRING).

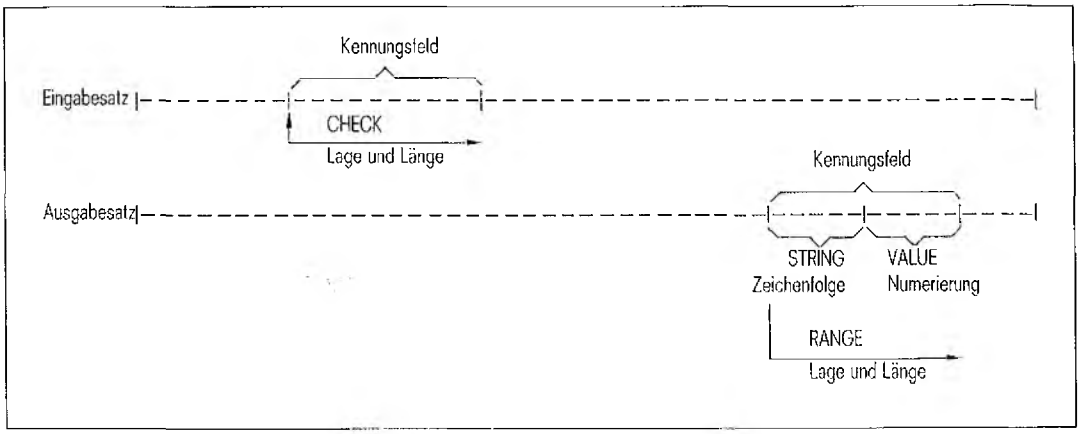



Bild 8 Aufbau eines Kennungsfeldes

Bei bereits existierenden Elementen kann durch NUM

- ein neues Kennungsfeld mit Satzkennung erzeugt werden,
- ein bereits vorhandenes neu numeriert werden.

Der Aufbau des Kennungsfeldes wird durch die Verarbeitungsoperanden RANGE, STRING und VALUE gesteuert.

Bei der Ausgabe eines Elementes in eine ISAM-Datei kann der ISAM-Schlüssel aus dem Kennungsfeld der Elementsätze übernommen werden (Verarbeitungsoperand CHECK), wenn im Element keine ISAM-Schlüssel gespeichert sind.

### Vergleichen von Elementen

Mit **COM** werden Textelemente verglichen. Die Elemente werden dabei Satz für Satz verglichen, wobei über Verarbeitungsoperanden der Vergleichsbereich festgelegt wird.

LMS vergleicht zwei Elemente, wobei das bei COM vor dem Vergleichsoperator angegebene Element als Primärelement, das nach dem Vergleichsoperator als Sekundärelement bezeichnet wird. Der Vergleichsoperator wird durch das Zeichen '=' dargestellt.

Die festgestellten Unterschiede werden auf Anforderung protokolliert. Dieses Protokoll bezeichnet man als Vergleichsprotokoll. Anschließend an das Vergleichsprotokoll gibt LMS die Vergleichsstatistik aus. Das ist eine Tabelle, die das Ergebnis des Vergleichs zahlenmäßig wiedergibt.

Der Vergleichsbereich muß vor der Eingabe von COM mit dem Verarbeitungsoperanden COMPARE festgelegt werden.

Bei der Entscheidung, ob zwei Sätze gleich sind, wird unterschieden zwischen

- formalem und
- logischem Vergleich.

Im logischen Vergleich werden Leerzeichen nicht berücksichtigt.
Im formalen Vergleich werden alle Zeichen des Satzes verglichen.

Das Ergebnis der beiden Vergleichsarten (formal bzw. logisch) wird in der gleichen Weise protokolliert.

Für den Vergleich stehen zwei Algorithmen zur Verfügung, zum einen der Heckel-Algorithmus (Standard ab der LMS-Version V1.3A) und wahlweise auch der Kreuzvergleich.

Untersuchungen haben gezeigt, daß die CPU-Zeit für den Heckel-Algorithmus bei kleinen und mittleren Elementgrößen etwa gleich und bei großen Elementen niedriger als beim Kreuzvergleich ist. Die Ergebnisse des Heckel-Algorithmus bzgl. der Anzahl SAME-lines sind besser. Beim formalen Vergleich ist der Heckel-Algorithmus dem Kreuzvergleich deutlich überlegen. Dieser Vorteil ist bei der Bearbeitung von Delta-Elementen wichtig.

### Vergleichsprotokoll

Standardmäßig wird der Vergleichsbereich der Sätze protokolliert und nicht der komplette Satz.

Im Vergleichsprotokoll bezieht sich das Vergleichsergebnis auf den Vergleich zweier Vergleichsbereiche, wobei der Unterschied immer gegenüber dem Sekundärelement dargestellt wird, d.h. das Primärelement wird als neues Element und das Sekundärelement als altes Element gewertet.

| Vergleichs-ergebnis | Bedeutung |
|---|---|
| SAM (same) | Die Vergleichsbereiche der beiden verglichenen Sätze im Primär- und Sekundärelement entsprechen sich. |
| DEL (deleted) | Der Satz mit diesem Vergleichsbereich ist nur im Sekundärelement vorhanden. |
| INS (inserted) | Der Satz mit diesem Vergleichsbereich ist nur im Primärelement vorhanden. |

Wird im Verarbeitungsoperanden COMPARE eine Zahl für den Synchronisationszähler angegeben, schaltet LMS auf den Kreuzvergleich um. Mit der Anweisung PAR COM= kann man wieder auf den Heckel-Algorithmus zurückschalten.

Durch Setzen des Verarbeitungsoperanden PAR COMPARE=/COR wird das Vergleichsprotokoll in Korrekturanweisungen umgesetzt (siehe Seite 53, Korrekturanweisungen aus dem Vergleichsprotokoll).

### Vergleichsstatistik

Die Vergleichsstatistik gibt über einen Vergleich folgende Auskunft:

- Anzahl aller verglichenen Sätze im Primär- und Sekundärelement,
- Anzahl der eingefügten Sätze,
- Anzahl der gelöschten Sätze,
- Anzahl der gleichen Sätze.

Zusätzlich wird ein Ergebnis des gesamten Vergleichs angegeben:

| Vergleichs-ergebnis | Bedeutung |
|---|---|
| S (same) | Beim Vergleichen wurden keine Unterschiede festgestellt. |
| C (changed) | Beim Vergleichen wurden Unterschiede festgestellt. |
| I (inserted) | Das Sekundärelement wurde nicht gefunden. |
| D (deleted) | Das Primärelement wurde nicht gefunden. |
| ERR (error) | Während des Vergleichs trat ein Fehler auf. |

Durch Setzen des Verarbeitungsoperanden PAR SUM=YES wird die Vergleichsstatistik gespeichert.

So gespeicherte Vergleichsstatistiken werden mit

- SUMPRT aufgelistet,
- SUMADD in ein Summenfeld addiert, und mit
- SUMDEL gelöscht.

## Korrigieren von Elementen

Zum Korrigieren von Elementen besitzt LMS verschiedene Korrekturanweisungen:

- COR, EDR und EDT korrigieren Textelemente (Elementtyp S, M, J, P, D, X).
- UPD korrigiert Binde- und Lademodule (Elementtyp R und C) und LLMs (Elementtyp L).

### Korrigieren von Textelementen

Das mit **COR** angegebene Element wird durch Unterfunktionen von COR, den sogenannten Korrekturanweisungen, verändert.
Diese Korrekturanweisungen, die über Satznummern oder Satzkennungen auf einen oder mehrere Elementsätze zugreifen können, führen folgende Veränderungen am Element durch:

- Sätze einfügen,
- Sätze löschen,
- Sätze ersetzen,
- Sätze ändern.

### Korrekturanweisungen aus Vergleichsprotokoll

Beim Vergleichen von Elementen des Typs S oder M über COM bestimmt der Verarbeitungsoperand COMPARE, ob das Vergleichsprotokoll in Korrekturanweisungen umgesetzt wird.

Diese Korrekturanweisungen, die somit die Differenz zwischen den verglichenen Elementen bilden, werden in die Systemdatei SYSOPT geschrieben. Weist man SYSOPT einer Datei zu, kann diese Datei wieder als Prozedurelement aufgenommen werden.

Durch Umweisen der Anweisungseingabe mit CTL (siehe Seite 68) kann der Benutzer dem Prozedurelement die Steuerung des LMS-Laufs übergeben. Die Änderungen am Sekundärelement werden somit automatisch ausgeführt.

Diese Funktion von LMS ermöglicht dem Benutzer, nur eine Version eines Elementes zu speichern. Alle aktuelleren Stände können mit dieser Version verglichen und die Differenz in einem Prozedurelement gespeichert werden.

### Anweisungen EDT und EDR

**EDT** bzw. **EDR** ruft den Editor EDT bzw. EDOR als Unterprogramm auf. Das angegebene Element wird dann mit EDT- bzw. EDOR-Anweisungen bearbeitet. Nach Beenden des Editors wird das korrigierte Element in die Ausgabebibliothek geschrieben.

EDT bzw. EDR ohne Elementtyp und Operanden verzweigt in den EDT bzw. EDOR, ohne ein Element in den Arbeitsbereich einzulesen.

### Korrigieren von Binde- und Lademodulen und LLMs

**UPD** korrigiert das angegebene Element der zugewiesenen Eingabebibliothek. Das korrigierte Element wird dann in die zugewiesene Ausgabebibliothek geschrieben. Es kann dabei eine neue Elementbezeichnung erhalten.

Für das Korrigieren von Binde- und Lademodulen und LLMs hat UPD verschiedene Subanweisungen. Die Subanweisungen werden unmittelbar hinter UPD bis zu *END aus dem Anweisungsstrom gelesen.

Die Funktionen der Subanweisungen für Bindemodule (Elementtyp R) sind:

- Textsätze korrigieren,
- Textkorrekturen rückgängig machen,
- Korrekturen umwandeln, d.h. entweder REP-Sätze in Textkorrekturen oder Textkorrekturen in REP-Sätze,
- REP-Sätze einfügen,
- INCLUDE-Sätze einfügen,
- Merkmale der Programmabschnitte ändern,
- Satzarten aus dem Eingabeelement ausschließen,
- Symbole umbenennen,
- Kontrollzahlen festlegen,
- Identifikationen festlegen,
- Basisadresse festlegen.

Die Funktionen der Subanweisungen für Lademodule (Elementtyp C) sind:

- Textsätze korrigieren,
- Textkorrekturen rückgängig machen,
- Korrekturjournalsätze löschen,
- Kontrollzahlen festlegen,
- Identifikationen festlegen,
- Segmente festlegen,
- Basisadresse festlegen.

Die Funktionen der Subanweisungen für LLMs (Elementtyp L) sind:

- Textsätze korrigieren
- Textkorrekturen rückgängig machen,
- Korrekturjournalsätze löschen,
- Identifikation festlegen.

### Umbenennen von Elementen

**NAM** benennt die angegebenen Elemente der zugeordneten Eingabebibliothek um. Bei dieser Anweisung können auch Elemente umbenannt werden, deren Bezeichnungen den LMS-Konventionen nicht genügen.
Ein Umbenennen von Delta-Elementen ist aus Revisionsgründen nicht zugelassen.

### Inhaltsverzeichnis einer Bibliothek ausgeben

**TOC** protokolliert die Inhaltsverzeichniseinträge der angegebenen Elemente oder der gesamten Eingabebibliothek.

Mit dem Verarbeitungsoperanden SORT wird dabei festgelegt, ob das Inhaltsverzeichnis unsortiert oder nach Namen, nach Versionsnummern, nach Datum oder nach Referenznamen sortiert ausgegeben werden soll. Die Elementbezeichnungen werden standardmäßig nach Namen, Versionsnummer und Datum sortiert ausgegeben.

Inhaltsverzeichnisse von Programmbibliotheken werden immer nach Elementtyp und Name alphabetisch sortiert ausgegeben.

Damit LMS das vollständige Inhaltsverzeichnis einer Programmbibliothek ausgibt, ist TOC oder TOC* */* anzugeben.

## Speichern und Aufrufen von Prozeduren

### Speichern von Prozeduren

LMS ermöglicht dem Benutzer BS2000-Prozeduren und ENTER-Prozeduren als Elemente in Bibliotheken abzulegen (Elementtyp J bei Programmbibliotheken, Elementtyp S bei Quellprogrammbibliotheken).

Bestehende Prozedurdateien können mit ADD als Elemente in Bibliotheken aufgenommen werden.

Durch das Speichern von Prozeduren kann, insbesondere bei kleinen Kommandodateien, Speicherplatz eingespart werden. Die Zahl der Katalogeinträge wird verringert.

Es ist jedoch darauf zu achten, daß aus den Elementen in Programmbibliotheken, die über PAR KEY=YES evtl. vorhandene ISAM-Schlüssel mitgespeichert haben, diese Schlüssel vor dem Aufruf der Prozedur entfernt werden.

Ein Bibliothekselement kann auch als Systemeingabedatei (SYSDTA) zugewiesen werden:

```
/ASSIGN-SYSDTA TO-FILE=(LIB=bibliotheksname,ELEM=element[,VERSION=[vers / *STD]][,TYPE=[type / *STD]])
```

wobei *STD bei VERSION die höchste vorhandene Version und *STD bei TYPE Elementtyp S bedeutet. Weitere Möglichkeiten bei TYPE sind D und M.

### Aufruf von Prozeduren

Für einen Aufruf von Prozeduren, die als Elemente einer Bibliothek gespeichert sind, ist statt des Dateinamens der Bibliotheks- und Elementname anzugeben:

```
/DO
/CALL    } bibliotheksname(element)
/ENTER
```

*Achtung*

Zulässige Elementnamen sind nicht in jedem Fall als Dateinamen zulässig.

Beim Aufruf eines Bibliothekselementes wird vom SYSFILE-Management eine temporäre Kopie des Elementes (SAM-Datei) unter dem Namen:

```
S.IN.bibliotheksname.elementname.tsn.HHMMSS
```

erzeugt.

Der Bibliotheksname wird auf die ersten zwanzig Zeichen abgekürzt, falls er länger ist. HHMMSS ist die Zeit in Stunden, Minuten und Sekunden. Diese Datei wird gelöscht, wenn sie nicht mehr benutzt wird.

*Hinweis*

Auftragsschalter 1 muß gesetzt sein, wenn LMS in einer DO- oder CALL-Prozedur aufgerufen wird und dabei die Anweisungen ebenfalls aus der Prozedur lesen soll.

# Archivieren mit Delta-Technik

Zum Speichern von mehreren Versionen pro Elementname in Programmbibliotheken stehen zwei Methoden zur Verfügung:

- die Voll-Speicherung und
- die Delta-Speicherung.

Bei der **Voll-Speicherung** befindet sich genau ein Element, d.h. alle Sätze des Elementes, in einem eigenen Behälter (einer Ablageeinheit der Bibliothek). Wenn unter diesem Namen eine weitere Version aufgenommen wird, werden alle Sätze dieses Elementes ebenfalls in einem eigenen Behälter abgelegt. Eine eventuelle Beziehung der Elemente zueinander ist LMS nicht bekannt.
Im folgenden werden solche Elemente als Voll-Elemente bezeichnet.

Bei der Bearbeitung, z.B. Lesen eines Voll-Elementes, kann LMS direkt auf alle Sätze des angegebenen Elementes zugreifen und die Aktion ausführen. Diese Methode gilt für alle Elementtypen und eignet sich wegen des schnellen Zugriffs besonders für in Entwicklung bzw. in Änderung befindliche Elemente.

Für die textartigen Elementtypen S, M, J, P und D kann zum Speichern mehrerer Versionen eines Elementes die **Delta-Speicherung** verwendet werden. Bei dieser speicherplatzsparenden Methode wird nur noch das zeitlich erste Element vollständig in einem eigenen Behälter abgelegt. Bei der Aufnahme weiterer Versionen zum gleichen Element werden nicht mehr alle Sätze, sondern nur noch die Differenzmenge zum jeweiligen Vorgänger ermittelt und einsortiert (Vergleich der Elemente). Zusätzlich wird durch die Verknüpfung 'neue Version - Vorgänger-Version' die logische Beziehung der Elemente zueinander bekanntgegeben.
Im folgenden werden solche Elemente als Delta-Elemente bezeichnet.

Bei der Bearbeitung, z.B. Lesen eines Delta-Elementes, werden die entsprechenden Sätze des angesprochenen Elementes herausgefiltert. Dies kann, wenn eine Vielzahl in Beziehung stehender Delta-Elemente existiert, zu einer Verlangsamung der gesamten Aktion führen. Aus diesen Gründen ist die Delta-Speicherung speziell für die platzsparende und übersichtliche Archivierung von Element-Versionen geeignet.

### Delta als Speicherungsform und Ordnungsmittel

Delta-Elemente unterscheiden sich von Voll-Elementen nicht nur durch die platzsparende Speicherungsform, sondern auch dadurch, daß zwischen den Delta-Elementen eine eindeutige Beziehung besteht.

Speicherplatz wird gespart durch die Delta-Bildung:

- Redundante Sätze eines Elementes gegenüber dem Vorgänger-Element werden erkannt und nicht erneut gespeichert.
- Um zu erkennen, ob Sätze redundant sind, wird ein formaler Vergleich zwischen den Sätzen der neuen Version und der angegebenen Basisversion vorgenommen.

Eindeutige Beziehungen werden hergestellt durch:

- Eindeutige Elementnamen
  Delta-Elemente, die zueinander in Beziehung stehen, werden dadurch charakterisiert, daß sie den gleichen Elementnamen tragen. Sie bilden einen Namensraum. Daher müssen Voll-Elemente und Delta-Elemente unterschiedliche Elementnamen haben, das bedeutet,
  - ein Voll-Element wird nur dann erstellt, wenn kein gleichnamiges Delta-Element existiert,
  - ein Delta-Element wird nur dann erstellt, wenn kein gleichnamiges Voll-Element existiert.

  Alle Delta-Elemente gleichen Namens bilden genau ein logisches Gebilde 'Delta-Baum', in seiner einfachsten Form eine 'Delta-Sequenz'.

  Zu einem Namen gibt es genau einen Delta-Baum.
- Eindeutige Versionsbezeichnungen
  Beim Aufnehmen eines Elementes als Delta-Element muß angegeben werden, welches Element der Vorgänger ist, d.h. als (Vergleichs-) Basis für den Vergleich dienen soll.

Die bei der Elementaufnahme getroffenen Vereinbarungen hinsichtlich Elementname und Versionsbezeichnung können aus Revisions- und Konsistenzgründen nicht nachträglich manipuliert werden, ggf. ist durch Duplizieren des Delta-Baums mit gleichzeitiger Umbenennung ein neuer Delta-Baum zu schaffen.

### Aufnehmen von Delta-Elementen

Delta-Elemente können mit **ADD** bzw. **DUP** aufgenommen werden. Mit Hilfe des Operanden BASEVERSION wird

- die Delta-Verarbeitung eingeschaltet und
- gesteuert, zu welchem Vorgänger eine Beziehung aufgebaut werden soll. Zum Vorgänger-Element wird auch der Vergleich durchgeführt.

Fehlt der Operand, wird ein Voll-Element aufgenommen; BASEVERSION muß der letzte Operand der Anweisung sein. Außerdem ist es nicht erlaubt, zwischen BASEVERSION= und dem Operandenwert zu trennen, d.h. Fortsetzungszeilen sind nicht möglich.

Durch die Operandenwerte des Operanden BASEVERSION kann unterschieden werden:

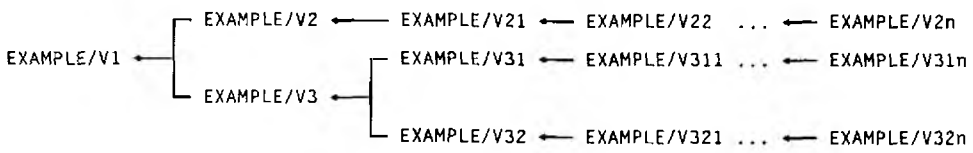
- BASEVERSION=*NONE
  Das Delta-Element hat keinen Vorgänger. Dieses Element ist das erste Element eines Delta-Baums. Es muß einen eindeutigen Namen haben. Alle Sätze werden in einem neuen Behälter abgelegt.
- BASEVERSION=*HIGH
  Der Vorgänger ist das gleichnamige Delta-Element mit der höchsten Versionsbezeichnung. Das neue Delta-Element wird an die vorhandene höchste Version angehängt, die Delta-Menge wird ermittelt und im Behälter abgelegt. Die Versionsbezeichnung des neuen Delta-Elementes sollte höher als die bisher höchste sein.

  *Beispiel für eine Delta-Sequenz (die Pfeile sollen die Beziehungen darstellen)*

  EXAMPLE/V1 ⟵ EXAMPLE/V2 ⟵ EXAMPLE/V3 ...⟵ EXAMPLE/Vn

- BASEVERSION=version
  Der Vorgänger ist das gleichnamige Delta-Element mit der angegebenen Versionsbezeichnung. Hierdurch können baumartige Strukturen aufgebaut werden. Das neue Delta-Element wird (quasi seitlich) an die angegebene Version angehängt, die Delta-Menge wird ermittelt und im Behälter abgelegt. Aus der Versionsbezeichnung des neuen Delta-Elementes sollte die Beziehung zum Vorgänger-Element abzulesen sein.

  *Beispiel für einen Delta-Baum (die Pfeile sollen die Beziehungen darstellen)*



### Übersicht über Delta-Elemente

Für das Erzeugen von Übersichten über Delta-Elemente ist die Wirkung des Verarbeitungsoperanden **TOC** folgende:

- Im Falle TOC=F wird die Ausgabe um das Feld FLAG erweitert. Durch den Eintrag D wird angezeigt, daß dieses Element ein Delta-Element ist.
- Ist der Operandenwert TOC=D gesetzt, gilt:
  Es wird immer ein kompletter Delta-Baum aufgelistet, unabhängig davon, welches Element bei TOC angegeben wurde.
  Zusätzlich zur Elementbezeichnung wird die interne Delta-Nummer der Elemente im Feld DELTA# und die interne Nummer des Vorgängers im Feld BASE# ausgegeben.
  Die Delta-Nummer spiegelt die chronologische Reihenfolge der Elementaufnahme wieder. Beide Nummern beschreiben die Verkettung der Elemente in einem Baum eindeutig, sie können vom Benutzer nicht beeinflußt werden. Der Benutzer kann die Verkettung der Elemente durch geeignete Versionsvergabe sichtbar festschreiben.

### Löschen von Delta-Elementen

Es ist zu unterscheiden zwischen dem logischen und physikalischen Löschen:

- Logisches Löschen
  Hierbei wird lediglich der Eintrag aus dem Inhaltsverzeichnis entfernt. Die Datensätze und die Beziehung zum Vorgänger bleiben erhalten.
- Physikalisches Löschen
  Abhängig vom Verarbeitungsoperanden DESTROY (er muß beim Aufnehmen des Elementes gesetzt werden) wird zusätzlich der Speicherplatz des Elementes mit binären Nullen überschrieben.
  Für Delta-Elemente wird dieser Operand jedoch erst dann wirksam, wenn das letzte Element des Delta-Baumes gelöscht wird. Die Angabe von DESTROY=YES genügt bei einem Element, um den gesamten Behälter zu löschen.

### Sperren von Delta-Elementen

Bei der Bearbeitung

- eines **Voll-Elementes** wird das angegebene Element gesperrt (Element = Behälter);
- eines **Delta-Elementes** werden alle Elemente, die im angesprochenen Behälter liegen, gesperrt (Delta-Baum = Behälter).

Für die Anwendung bedeutet das, daß ein Behälter nur zur Eingabe oder zur Ausgabe benutzt werden kann.

### Besonderheiten der Delta-Technik

Im folgenden sind einige Besonderheiten aufgezählt, die bei der Archivierung von Elementen mit der Delta-Technik zu beachten sind:

- Ein **Umbenennen** von einzelnen Delta-Elementen oder eines ganzen Delta-Baums ist nicht möglich, weil dies die Revisionsfähigkeit eines Archivs unterlaufen würde bzw. die Konsistenz der Daten gefährdet sein kann, wenn eine laufende Aktion, wie auch immer, abgebrochen wird.
- Ein **Überschreiben** von Delta-Elementen im Sinne von 'Ersetzen existierender Elemente' ist nicht zulässig, d.h. der Verarbeitungsoperand OVERWRITE = YES ist wirkungslos bzw. wird mit einer Fehlermeldung abgewiesen.
- Ein **Korrigieren** eines Delta-Elementes ist nicht möglich.

# Steuern des LMS-Laufs

Der LMS-Lauf wird gesteuert durch

- Verarbeitungsoperanden
- Auftragsschalter

## Wirkung der Verarbeitungsoperanden

Die Verarbeitungsoperanden beeinflussen sowohl den Ablauf von LMS als auch einzelne Funktionen. Sie werden mit **PAR** gesetzt.

Die folgende Tabelle stellt dar, welcher Verarbeitungsoperand auf welche Anweisung oder den gesamten LMS-Lauf wirkt.

| Verarbeitungs-operanden | LMS-Anweisungen | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADD | COM | COR | DEL | DUP | EDR | EDT | LST | NAM | NUM | PRT | RST | SEL | SUM | TOC | UPD | LMS-LAUF |
| B[ASE] | | | | | | | | * | | | | | | | | | |
| CH[ECK] | * | * | * | | | * | * | | | * | | | * | | | | |
| COM[PARE] | | * | | | | | | | | | | | | | | | |
| CS[ECT] | | | | | | | | * | | | | | | | | * | |
| DES[TROY] | * | | * | * | * | * | * | | * | * | * | | | | | * | |
| E[RRCONS] | | | | | | | | | | | | | | | | | * |
| FC[BTYPE] | | | | | | | | | | | | | * | | | | |
| FO[RMAT] | | | | | | | | * | | | | | | | | | |
| I[NFO] | | | | | | | | * | | | | | | | | | |
| K[EY] | * | | | | | | | | | | | | | | | | |
| LC[ASE] | | | | | | | | | | | | | | | | | * |
| LIN[E] | | * | | | | | | * | | | * | | | | * | | * |
| LO[G] | | | * | | | | | | | * | | | | | | | * |
| LS[T] | | | | | | | | * | | | | | | | | | |
| N[EWFORM] | | | | | | | | | | | | | | | | | * |

| Verarbeitungs-operanden | LMS-Anweisungen | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ADD | COM | COR | DEL | DUP | EDR | EDT | LST | NAM | NUM | PRT | RST | SEL | SUM | TOC | UPD | LMS- | LAUF |
| O[VERWRITE] | * | | * | | * | * | * | | * | * | * | | * | | | * | | |
| PA[TH] | | | | | | | | * | | | | | | | | * | | |
| PH[ASE] | | | | | | | | | | | | | * | | | | | |
| RA[NGE] | * | | * | | | * | * | | | * | | | | | | | | |
| RE[FERENCE] | | | | * | * | | | * | | | | | | | * | | | |
| SE[GMENT] | | | | | | | | * | | | | | | | | | | |
| SL[ICE] | | | | | | | | * | | | | | | | | * | | |
| SO[RT] | | | | | | | | | | | | | | | * | | | |
| STRIN[G] | * | | * | | | * | * | | | * | | | * | | | | | |
| STRIP | | | | | * | | | | | | | | | | | * | | |
| SU[M] | | * | | | | | | | | | | | | | | | | |
| TER[MINATE] | | | | | | | | | | | | | | | | | * | |
| TES[T] | | | | | | | | | | | | * | | | | | * | |
| TO[C] | | | | | | | | | | | | | | | * | | | |
| TY[PE] | * | * | * | * | * | * | * | * | * | * | | | * | | * | * | | |
| V[ALUE] | * | | * | | | * | * | | | * | | | | | | | | |

Die Verarbeitungsoperanden bestimmen,

- welcher Elementtyp voreingestellt wird (TYPE)
- ob zu löschende Elemente physikalisch gelöscht werden (DESTROY)
- ob die ISAM-Schlüssel und andere Dateimerkmale von ISAM-Dateien mit aufgenommen werden (KEY)
- mit welchem FCB-TYP Ausgabedateien von Textelementen erzeugt werden (FCBTYPE)
- die Werte von Kennungsfeldern und deren Auswertung (CHECK, RANGE, STRING, VALUE)
- den Ablauf von Vergleichen und deren Auswertung (COMPARE, SUM)
- das Überschreiben von Elementen (OVERWRITE)
- Umfang und Format der Ausgabe beim Auflisten und Protokollieren (LINE, BASE, FORMAT, INFO, LOG, NEWFORM, LST, PATH, SLICE, CSECT)
- welche Satzarten bei Elementen von Typ R oder C nicht vom Eingabe- ins Ausgabeelement übernommen werden (STRIP)
- das Protokollformat des Inhaltsverzeichnisses von Programmbibliotheken (TOC)
- ob das Inhaltsverzeichnis einer Bibliothek unsortiert oder nach Namen, Versionsnummer, Datum oder Referenznamen sortiert ausgegeben wird (SORT)
- die Protokollierung und das Verhalten im Fehlerfall (ERRCONS, TERMINATE)
- ob der RUN- oder TEST-Modus eingeschaltet ist (TEST)
- ob nur Elemente bearbeitet werden, die einer Referenzbedingung genügen (REFERENCE)
- das Segment eines Lademoduls, das bearbeitet werden soll (SEGMENT)
- das zu erzeugende Phasenformat (PHASE)
- ob alle Eingaben in Großbuchstaben umgesetzt werden (LCASE)

### Protokollausgabe steuern

Das LMS-Protokoll kann die eingegebenen Anweisungen enthalten, deren Ausführung oder Abbruch, die zugewiesenen Ein- und Ausgabebibliotheken und Listen, die z.B. beim Auflisten oder Vergleichen von Elementen erzeugt wurden.
Das Protokoll wird in die Systemdatei SYSOUT, SYSLST oder in ein Bibliothekselement geschrieben. Wohin LMS ausgibt, legt PRT fest.
Wird das Protokoll in ein Element geschrieben, erzeugt LMS bei Programmbibliotheken ein Element mit dem Typ P, bei Quellprogrammbibliotheken ein Element mit dem Typ S. Falls die Bibliothek, in die das Element geschrieben wird, eine Quellprogrammbibliothek ist, darf sie nicht die standardmäßige Ein- bzw. Ausgabebibliothek des aktuellen LMS-Laufes sein.
Den Umfang und das Format des Protokolls steuern Verarbeitungsoperanden und Auftragsschalter.
War beim Aufruf von LMS der Auftragsschalter 4 gesetzt, werden die Anfangs- und Endemeldungen von LMS unterdrückt. Außerdem wird das LMS-Protokoll auf den minimalen Umfang beschränkt (entsprechend dem Verarbeitungsoperanden PAR LOG = MIN).

Einen Überblick über die Wirkung von Verarbeitungsoperanden auf das LMS-Protokoll gibt folgende Tabelle:

| Verarbeitungs-operand | Funktion |
|---|---|
| LOG | Legt fest, ob alle Anweisungen, Anweisungen nur im Fehlerfall oder nur Meldungen protokolliert werden. |
| FORMAT | Legt die Satzdarstellung beim Auflisten eines Elementes fest. |
| INFO | Legt den Ausgabeumfang beim Auflisten eines Elementes fest. |
| SORT | Legt die Sortierung des Inhaltsverzeichnisses fest. |
| ERRCONS | Definiert, ob und welche Meldungen in die Systemdatei SYSOUT protokolliert werden, wenn ansonsten nicht in die Systemdatei nach SYSOUT protokolliert wird. |
| LINE | Legt die Anzahl der Zeilen und Spalten einer Protokollseite fest. |
| TOC | Legt das Protokollformat des Inhaltsverzeichnisses von Programmbibliotheken fest. |
| NEWFORM | Steuert den Zeilen- oder Seitenvorschub des Protokolls. |
| COMPARE | Definiert den Umfang des Vergleichsprotokolls. |

### Erfolgs- und Mißerfolgsmeldungen

Ist der Wert des Verarbeitungsoperanden LOG=MAX oder LOG=MED, wird die Ausführung jeder LMS-Anweisung, die ein Element betrifft, protokolliert. Wird die Anweisung fehlerfrei ausgeführt, dann gibt LMS eine Erfolgsmeldung aus.

Kann die Anweisung nicht ausgeführt werden, protokolliert LMS dies mit einer Mißerfolgsmeldung und evtl. der entsprechenden LMS-Fehlermeldung (Fehlermeldungen siehe ab Seite 295).

Alle Erfolgs- und Mißerfolgsmeldungen haben folgendes Format:

```
[NO] anweisung elem[wort elem][ursache]
```

| Bedeutung | |
|---|---|
| NO | Die Anweisung wird nicht ausgeführt. |
| anweisung | Anweisungsname. |
| elem | Elementbezeichnung oder Dateiname (bei ADD und SEL). |
| wort | Schlüsselwort: AS, INTO, WITH. |
| ursache | Ergebnis: EXISTING, REPLACED, u.ä. |

*Beispiel*

```
/SHOW-FILE-ATTRIBUTES
 PROQUELL
   .
   .
   .
ADDS PROQUEL>ELEMPRO
       NO ADD (S)PROQUEL AS ELEMPRO, OUTPUT EXISTING
ADDS PROQUELL>ELEMPRO
           ADD PROQUELL AS (S)ELEMPRO
```

### Anweisungseingabe steuern

**CTL** definiert das Medium, von dem die LMS-Anweisungen gelesen werden.

Die Anweisungen können

- von der Datensichtstation,
- von der Systemdatei SYSDTA, oder
- aus einem Bibliothekselement

gelesen werden.

Werden die Anweisungen aus einem Element gelesen, muß das Element bei Programmbibliotheken vom Typ J und bei Quellprogrammbibliotheken vom Typ S sein.

### Bildschirmwechsel steuern

Mit **TCH** macht der Benutzer Angaben zur Überlaufkontrolle des Bildschirms und zum Bildschirmwechsel.

Ein Überlauf ist dann gegeben, wenn nach einer Eingabe an der Datensichtstation mehr Zeilen ausgegeben würden, als im Kommando /MODIFY-TERMINAL-OPTIONS (siehe Benutzer-Kommandos [7]) vereinbart wurde.

TCH vereinbart, ob bei einem anstehenden Überlauf eine Zeit t in Sekunden gewartet, nicht gewartet oder PLEASE ACKNOWLEDGE ausgegeben wird, ehe die neue Ausgabe am Bildschirm erscheint.

Zusätzlich kann angegeben werden, ob bei einem Funktionswechsel oder Quittieren von PLEASE ACKNOWLEDGE die Ausgabe auf einem neuen Bildschirm oder im ROLL-UP-Modus erfolgen soll.

Die Meldung wird in folgender Form ausgegeben:

```
PLEASE ACKNOWLEDGE (NO/TIMER/<ANY INPUT>) / INTERRUPT (NE/NS/NI):
```

Die Eingabemöglichkeiten bedeuten dabei:

NO — Der Bildschirmüberlauf wird nicht kontrolliert, d.h. ist der Bildschirm voll und sollen weitere Daten ausgegeben werden, wird der Bildschirm überschrieben.

TIMER — Das System wartet 6 Sekunden, ehe der nächste Bildschirm ausgegeben wird.

<ANY INPUT> — Jede andere Angabe als NO, TIMER, NE, NS oder NI bewirkt die Ausgabe des nächsten Bildschirms.

NE Die Bearbeitung eines Elementes wird abgebrochen. Anschließend wird das nächste Element bearbeitet, das der aktuellen Auswahlangabe genügt.
(NE : NEXT ELEMENT)

NS Die nächste Anweisung wird ausgeführt. Die aktuelle Anweisung und eventuell alle bisher gesammelten Anweisungen werden vergessen; die nächste Anweisung aus dem Anweisungspuffer bzw. aus dem Prozedurelement, das mit CTL zugewiesen ist, wird ausgeführt.

Sind alle Anweisungen abgearbeitet, wird die nächste Anweisung von dem Medium gelesen, das vor der Eingabe von CTL gültig war.
(NS : NEXT STATEMENT)

NI Der nächste Eingabepuffer wird bearbeitet. Alle Anweisungen, die gerade bearbeitet werden, werden abgebrochen; Anweisungen, die sich noch im Eingabepuffer bzw. im zugewiesenen Prozedurelement befinden, werden ignoriert. War ein Prozedurelement zugewiesen, wird die Anweisungseingabe wieder dem Medium zugewiesen, das vor dem Prozedurelement zugewiesen war.
(NI : NEXT INPUT)

### Ablauf im RUN- und TEST-Modus

LMS unterscheidet zwischen den beiden Ablaufarten RUN und TEST. Im RUN-Modus werden alle Anweisungen ausgeführt.

Im TEST-Modus werden nur CTL, END, LIB, PAR, PRT und SYS ausgeführt. Die übrigen Anweisungen werden nicht ausgeführt, sondern von LMS überprüft, ob

- die Anweisungen formal richtig sind;
- die benötigten Bibliotheken zugewiesen sind;
- die verwendeten Elementnamen zulässig sind;
- bei den Korrekturfunktionen Lage und Länge des Kennungsfeldes und die Numerierung richtig sind.

Im TEST-Modus errechnet LMS auch die Kontrollzahlen für die Korrekturen mit UPD.

Der TEST-Modus ermöglicht somit, Fehler zu finden, bevor eine Funktion ausgeführt wird.

Standardmäßig ist der RUN-Modus eingeschaltet. Der TEST-Modus wird mit dem Verarbeitungsoperanden TEST ein- und ausgeschaltet.

LMS schaltet außerdem in den TEST-Modus um, wenn ein bestimmter Fehler auftritt. Der Verarbeitungsoperand TERMINATE legt fest, nach welchen Fehlern das LMS-interne Abbruch-Kennzeichen gesetzt und in den TEST-Modus umgeschaltet wird. Standardmäßig bricht LMS den RUN-Modus nur bei schwerwiegenden Fehlern ab. Mit RST schaltet LMS dann wieder in den RUN-Modus um.

### Benutzeranschlüsse

LMS bietet dem Benutzer die Möglichkeit, beim Auflisten oder Vergleichen von Elementen in ein Benutzerprogramm zu verzweigen.

Dieses Unterprogramm kann vor der Bearbeitung eines Elementsatzes folgende Aktionen ausführen:

- Manipulieren des aktuellen Elementsatzes.
- Einfügen eigener Sätze vor dem aktuellen Elementsatz bzw. an das Elementende.
- Den aktuellen Elementsatz von der Verarbeitung ausschließen.

Näheres siehe Seite 199, USE.

## Unterbrechen des LMS-Laufs

### Unterbrechen durch den Benutzer

Der Benutzer kann den LMS-Lauf durch Betätigen einer Programmunterbrechungstaste (z.B. K2) unterbrechen.

Die Fortsetzung des LMS-Laufs kann durch das INTR-Kommando gesteuert werden, das wahlweise mit einem Eingabetext versehen werden kann. Dieser Eingabetext wird dann in der Unterbrechungsbehandlung von LMS interpretiert. Die gerade laufende Funktion wird von der Art des Abbruchs unterrichtet.

Folgende Aktionen laufen im Falle einer Unterbrechung ab:

- LMS wird durch BREAK/ESCAPE (K2 oder ähnliches) unterbrochen.
  Falls LMS im Dialog läuft, wird die entsprechende STXIT-Routine aktiviert.
- LMS führt einen BKPT-Makro aus (nur im Dialog).
- Durch die Eingabe eines INTR-Kommandos wird die BREAK-STXIT-Routine verlassen und die INTR-STXIT-Routine aktiviert.
- LMS analysiert den bei dem INTR-Kommando mitgegebenen Text und unterrichtet die laufende Funktion über die Art des gewünschten Abbruchs.
- Die INTR-STXIT-Routine wird mit EXIT-Makro verlassen.
- Die unterbrochene Funktion beendet sich in der gewünschten Form.

### INTR-Kommando

Die Steuerung des Abbruchs laufender Funktionen wird über das INTR-Kommando erreicht. Hier kann wahlweise ein Text mitgegeben werden, der in der Unterbrechungsbehandlung an die unterbrochene Funktion weitergereicht wird. Folgende Texte sind möglich:

INTR NI
: Der nächste Eingabepuffer wird bearbeitet (NEXT INPUT). Alle noch anstehenden Anweisungen in LMS werden vergessen. Dies bedeutet, daß die gerade sich in Bearbeitung befindlichen Anweisungen abgebrochen, sowie Anweisungen, die sich noch im Eingabepuffer bzw. Anweisungen, die sich noch im Eingabeelement befinden, vergessen werden. CTL wird auf das Medium gelegt, das vor der Eingabe von CTL gültig war.

  Alle anstehenden Aktivitäten werden vergessen.

INTR [NS] Nächste Anweisung ausführen (NEXT STATEMENT). Die aktuelle Anweisung und eventuell alle bisher gesammelten Anweisungen werden vergessen; die nächste Anweisung aus dem Anweisungspuffer bzw. aus dem CTL-Element wird ausgeführt.

Sind alle Anweisungen abgearbeitet, wird die nächste Anweisung von dem Medium gelesen, das vor der Eingabe von CTL gültig war.

NS ist auch die Voreinstellung, falls das INTR-Kommando ohne Operanden eingegeben wird.

INTR NE Abbrechen der Bearbeitung eines Elementes (NEXT ELEMENT). Es wird mit der Bearbeitung des nächsten Elementes fortgesetzt, das der aktuellen Auswahlangabe genügt.

**Abbruch des LMS-Laufs durch Fehler**

Die Fehlerbehandlung wird ebenfalls über die STXIT-Routine gesteuert.

Programmbeendigung tritt ein bei

- Programmfehler ("P-Fehler", SVC-Fehler).
- ABEND-Kommando (abnormal end) durch EXECUTE, LOAD, CANCEL, LOGOFF oder Leitungsverlust.
- TIMEOUT (Ablauf der Programm- oder Tasklaufzeit).

In all diesen Fällen wird sichergestellt, daß LMS-Bibliotheken konsistent bleiben. Im wesentlichen sorgt LMS für einen korrekten Abschluß der Bibliotheken.

Für alle Fälle der Programmbeendigung gilt:

- Alle STXIT-Routinen in LMS werden ausgeschaltet, um ein irreguläres Weiterarbeiten durch INTR zu unterbinden.
- LMS simuliert ein END. Dadurch werden alle offenen Bibliotheken geschlossen.

Falls zum Zeitpunkt der Programmbeendigung noch Bibliotheken offen sind, werden diese geschlossen.

### Programmfehler

Vor der Simulation von END wird auf SYSOUT folgende Meldung ausgegeben:

```
PROGRAM ERROR AT loc (IW=iw)
```

loc bedeutet dabei die Unterbrechungsadresse und iw das Unterbrechungsgewicht.

LMS wird mit einem Speicherabzug abgebrochen.

Durch die Programmfehlerbehandlung wird folgendes erreicht:

- Es werden immer Diagnoseunterlagen erstellt.
- Eine Fortsetzung von LMS nach Programmfehlern (dies gilt auch für andere Unterbrechungen) mit RESUME, die in allen Fällen unsinnig ist, wird unterbunden.

### Diagnosehilfsmittel

Durch Setzen des Auftragsschalters 31 wird in LMS eine Testbedingung gesetzt. Im Falle eines Programmfehlers wird LMS im Stapelbetrieb mit einem Speicherabzug abgebrochen. Dabei sind die Register so gesetzt, wie sie zum Unterbrechungszeitpunkt gesetzt waren. Im Dialog wird, falls der Auftragsschalter 31 gesetzt ist, angefragt:

```
DO YOU WISH A BKPT (Y/N)?
```

Im Ja-Fall werden die Register so geladen, wie sie zum Unterbrechungszeitpunkt definiert waren, dann wird ein BKPT-Makro abgesetzt. Die INTR-Unterbrechungsroutine wird ausgeschaltet, um eine Fortsetzung von LMS durch INTR zu unterbinden.

In jedem Fall wird ein Speicherabzug veranlaßt, bevor ein simuliertes END ausgeführt und LMS beendet wird.

Das Absetzen eines BKPT ist nur im Dialog sinnvoll. Dies wird durch das Setzen des Auftragsschalters 31 gefordert.

Eine wichtige Anwendung dieser Funktion ist der Test von Prozeduren mit Benutzerausgängen, bei denen der Benutzer eigene Routinen in den LMS-Lauf einbringen kann.

### Verwenden von Auftragsschaltern

Über BS2000-Auftragsschalter kann der Benutzer auf den Ablauf von LMS Einfluß nehmen. Sie müssen vor dem Laden von LMS mit dem Systemkommando `/MODIFY-JOB-SWITCHES ON=(nr,...)` gesetzt werden.

Folgende Auftragsschalter nehmen Einfluß auf den Ablauf von LMS:

Auftragsschalter 1:
Im Dialogbetrieb werden die LMS-Anweisungen standardmäßig mit dem Makro WRTRD von der Datensichtstation gelesen. Wenn Auftragsschalter 1 gesetzt ist, werden die Anweisungen mit dem Makro RDATA aus der Datei gelesen, die der logischen Systemdatei SYSDTA zugewiesen ist.

Beim Aufruf von LMS in BS2000-Prozeduren ist der Auftragsschalter 1 zu setzen, wenn die LMS-Anweisungen von SYSDTA zu lesen sind.

Auftragsschalter 4:
Wenn Auftragsschalter 4 gesetzt ist, werden die Anfangs- und die Endemeldung von LMS unterdrückt.

Auftragsschalter 9:
Über das Setzen von Auftragsschalter 9 kann der Benutzer zusätzlich benötigten Speicherplatz anfordern. Dadurch können bis zu 12000 ungleiche Sätze in einem Vergleich bearbeitet werden und mit der Funktion TOC größere Inhaltsverzeichnisse zusammenhängend sortiert werden.

Auftragsschalter 31:
Mit dem Auftragsschalter 31 kann eine Testbedingung gesetzt werden, die zu Diagnosezwecken verwendet werden kann (siehe Seite 71).

Die Auftragsschalter werden nur bei der Initialisierung abgefragt, nachträgliches Setzen und Löschen hat für LMS keine Wirkung.

## Pamkey-Eliminierung

In Zukunft werden vom BS2000 nur noch Platten mit fester Blockgröße (2 KByte, 4KByte...) unterstützt. Diese festen Blockgrößen verhindern die einfache Unterbringung der PAM-Schlüssel. Aus diesem Grund wird der PAM-Schlüssel entfallen. Dieser Vorgang wird als Pamkey-Eliminierung bezeichnet.

Ab BS2000 V10 (für ISAM ab BS2000 V9.5) gibt es für SAM-, ISAM- und UPAM-Dateien zwei unterschiedliche Dateiformate auf Platte: das bisherige PAM-Schlüssel behaftete Format (kurz PK) und das PAM-Schlüssel lose Format (kurz NK).
Das Dateiformat wird durch den BLKCTRL-Wert festgelegt. BLKCTRL kann den Wert PAMKEY, DATA oder NO annehmen. Einzelheiten zu den Dateiformaten siehe Handbuch "DVS Einführung und Kommandoschnittstelle" [14].

Bibliotheksdateien

Die Unterscheidung PK <-> NK ist zunächst eine DVS bedingte Unterscheidung. Sie überträgt sich wie folgt auf die interne Dateiorganisation der Bibliotheken :

- PLAM

  Für die Organisation einer PLAM-Bibliothek ist der PAM Key nicht notwendig. Dateiseitig ergibt sich dennoch eine Unterscheidung, die durch das Dateiattribut BLKCTRL repräsentiert wird.

  PLAM Bibliotheken brauchen bei der Migration zwischen PK-Welt und NK-Welt nicht mit PAMCONV umgesetzt werden.

- OML

  In OML's wird der PAM-Schlüssel zur Organisation der Bibliotheksstruktur und -daten benutzt; folgerichtig kann dieses Format nicht mehr in der NK-Welt angeboten werden.
  Eine Migration gibt es nicht. OML's sind durch PLAM-Bibliotheken zu ersetzen.

- OSM

  OSM's beruhen auf der Zugriffmethode ISAM. Das Bibliotheksdienstprogramm MLU benutzt diese Zugriffsmethode und ist daher nur indirekt von der Pamkey-Eliminierung betroffen.

  OSM's müssen bei der Migration zwischen PK-Welt und NK-Welt mit dem Produkt PAMCONV umgesetzt werden.

Elementverarbeitung

### Überblick

Das nachfolgende Bild zeigt einen Überblick über die möglichen Situationen beim Transfer von Daten zwischen Datei und Bibliothekselementen.
Für die Elemente sind die logischen Informationseinheiten aufgeführt; für die Dateien ist der BLKCTRL-Wert angegeben.

Die Pfeile beschreiben die Transfer-Richtung.

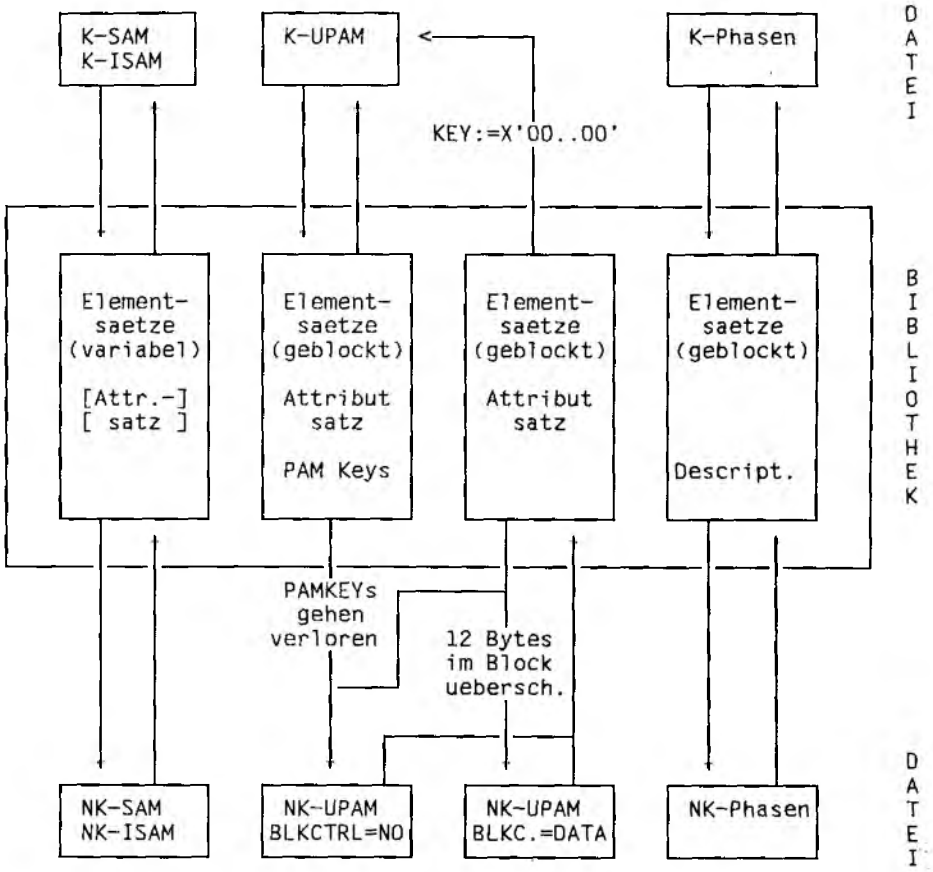



Bild 9 Transfer von Daten zwischen Datei und Bibliothekselementen

### Verhalten bei der ADD-Anweisung

Über die ADD-Anweisung werden Datei-Inhalte in Elemente abgelegt. Im Einzelnen gilt für:

- SAM/ISAM-Dateien

  Bei Aufnahme von SAM- und ISAM-Dateien wird der BLKCTRL-Wert mit abgespeichert, wenn PAR KEY=YES gesetzt ist, d.h. die ursprüngliche, vom BLKCTRL-Wert geprägte Blockstruktur der Datei wird im Attributsatz dokumentiert.

  Die Daten werden über die logische Zugriffsmethode SAM/ISAM gelesen (Einzelsätze) und unverändert ins Element geschrieben (als Sätze variablen Formates).
  Die erzeugte Elementstruktur ist unabhängig von der ursprünglichen BLKCTRL-Eigenschaft.

- PAM-Dateien

  Bei Aufnahme von PAM-Dateien wird der BLKCTRL-Wert generell mit abgespeichert. Die Blöcke der Datei werden über die Zugriffsmethode UPAM gelesen und unverändert als Block im Element gespeichert. Sind PAM-Schlüssel vorhanden, d.h. BLKCTRL=PAMKEY, so werden diese PAM-Schlüssel beim Element abgelegt.
  Das erzeugte Element behält somit die vom BLKCTRL-Wert geprägte Blockstruktur bei.

- Phasen

  Bei Aufnahme von Phasen wird der BLKCTRL-Wert nicht abgespeichert. Die entsprechende Formatangabe ist in der Phaseninformation auf Datei hinterlegt. In der PLAM-Bibliothek haben PK-Phasen und NK-Phasen gleiches Format. Die PAM-Schlüssel Informationen sind in Descriptoren abgelegt.

| ADD datei>element | Datei-Typ | BLKCTRL-Eintrag im Attributsatz | PAMKEY-Speicherung |
|---|---|---|---|
| LMS-Aufruf in V10 (Datei liegt auf NK-Platte) | SAM/ISAM | — 1) | |
| | SAM/ISAM | aus dem Katalog | nein |
| | UPAM | aus dem Katalog | nein |
| LMS-Aufruf in V10 (Datei liegt auf PK-Platte) | SAM/ISAM | — 1) | |
| | SAM/ISAM | aus dem Katalog | nein |
| | UPAM | aus dem Katalog | für BLKCTRL=PAMKEY |
| LMS-Aufruf V9.5 | SAM/ISAM | — 1) | |
| | SAM/ISAM | aus dem Katalog | nein |
| | UPAM | aus dem Katalog (immer PAMKEY) | ja |

1) die Ablage kann über PAR KEY=YES/NO gesteuert werden

**Verhalten bei der SEL-Anweisung**

Über die SEL-Anweisung werden Elementinhalte in Dateien ausgegeben. Der BLKCTRL-Wert wird über folgende Hierarchie bestimmt :

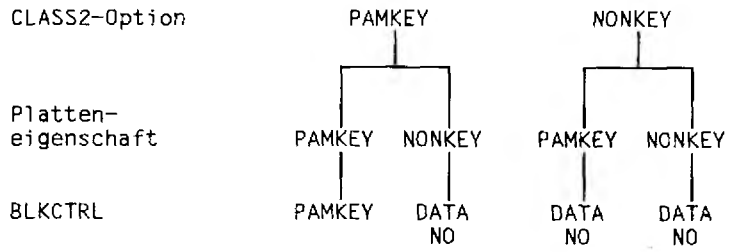
1.) Angabe im Katalogeintrag oder FILE-Kommando oder LMS-Parameter

2.) gespeicherter BLKCTRL-Wert beim Element. Er ist nur bei ursprünglichen PAM-Dateien relevant.

3.) Einstellung der CLASS2-Option PAMKEY oder NONKEY

4.) Platteneigenschaft PAMKEY oder NONKEY

Ist kein Katalogeintrag vorhanden und der BLKCTRL-Wert nicht gespeichert, so entscheidet die CLASS2-Option und die Platteneigenschaft über den BLKCTRL-Wert :




Ist CLASS2-Option = PAMKEY gesetzt, so läßt LMS das System den BLKCTRL-Wert bestimmen, d.h. BLKCTRL = <not specified>.

Ist CLASS2-Option = NONKEY gesetzt, so setzt LMS für SAM- und ISAM-Dateien BLKCTRL = DATA und für PAM-Dateien BLKCTRL = NO.

Im Einzelnen gilt für:

- ISAM-Dateien

  Die variabel langen Elementsätze werden mit der logischen Zugriffsmethode ISAM geschrieben. Der BLKCTRL-Wert der Datei wird nach dem oben beschriebenen Algorithmus bestimmt, wobei allerdings Punkt 2) entfällt, da der beim Element gespeicherte BLKCTRL-Wert nur dokumentarischen Charakter hat; er wird ignoriert.

- SAM-Dateien

  Bei BLKCTRL=DATA tritt ein DVS-Fehler auf, wenn im Element Sätze länger als 32KByte - 16Byte sind. In der PK-Welt dürfen diese Sätze bis zu 32Kbyte - 4Byte lang sein. LMS reicht beim Selektieren zu lange Sätze ungeprüft an DVS weiter.

  Der BLKCTRL-Wert wird wie bei ISAM ermittelt.

- PAM-Dateien

  In der NK-Welt gehen die PAM-Schlüssel verloren. Zusätzlich werden bei BLKCTRL=DATA die ersten 12 Bytes eines jeden logischen Blockes durch das System überschrieben. In beiden Fällen gibt LMS keine Fehlermeldung aus.

- Phasen (Typ C Elemente)

  Phasen (Typ C Elemente) werden gesondert behandelt.
  Neben dem alten Phasenformat (PK-Phase) gibt es ein PAM-Schlüssel freies, neues Phasenformat (NK-Phase) auf Dateiseite. Beim Selektieren kann das Format mit dem Parameter PHASE=PK/NK gesteuert werden (default PHASE=CLASS2-Option).
  Dieser Parameter bestimmt einzig und allein den BLKCTRL-Wert und das Phasenformat.

```
PHASE=PK ——>  Format=PK und BLKCTRL=PAMKEY

PHASE=NK ——>  Format=NK und BLKCTRL=PAMKEY  für BS2000 <V10
                            BLKCTRL=NO      ab  BS2000  V10
```

  Damit können NK-Phasen auch in BS2000 Versionen kleiner 10.0 erzeugt werden (Migrationshilfe). Ablauffähig sind NK-Phasen jedoch erst ab BS2000 V10.

Zusammenfassung

- SAM/ISAM-Dateien

  Das Aufnehmen der Dateien geht immer; ebenso der Selektiervorgang. Ein evtl. gespeicherter BLKCTRL-Wert hat nur dokumentarischen Charakter.

  Die innere Dateiform wird stets durch die Zugriffsmethode SAM/ISAM bestimmt. Diese übernimmt auch die Konversion der Datensätze in das "innere" Blockformat der Datei.

- UPAM-Dateien

  Eine evtl. notwendige Konversion der Daten kann nicht automatisch erfolgen (weder durch die Zugriffsmethode UPAM noch durch LMS), da in diesen Fällen ein Datenverlust eintritt.

  Die Steuerung liegt letztendlich beim Benutzer.

| Datei-Type U P A M | Erzeugter / abgespeicherter BLKCTRL-Eintrag im Attributsatz | | | |
|---|---|---|---|---|
| | PAMKEY | DATA | NO | — |
| LMS-Aufruf in V10 (Datei liegt auf NK-Platte) | 1) 2) | ADD SEL | ADD SEL | — SEL |
| LMS-Aufruf in V10 (Datei liegt auf PK-Platte) | ADD SEL | ADD SEL | ADD SEL | — SEL |
| LMS-Aufruf < V10 (Datei liegt auf NK-Platte) | ADD SEL | — SEL | — SEL | — SEL |

1) der Wert BLKCTRL=PAMKEY ist nicht möglich

2) der Selektiervorgang muß vom Benutzer gesteuert werden, z.B. über Angabe eines Dateikettungsnamen in der Anweisung.

# Anweisungen

### Syntax der Anweisungen

Für die formale Darstellung der Anweisungen und der Verarbeitungsoperanden werden folgende Metazeichen verwendet:

| Formale Darstellung | Erläuterung | Beispiel |
|---|---|---|
| GROSSBUCHSTABEN und Sonderzeichen | Großbuchstaben und Sonderzeichen bezeichnen Konstanten, die in dieser Form vom Benutzer eingegeben werden. | NAMx elem(lib)>elemu<br>Einzugeben ist:<br>NAMR MODLA(1)>AMOD |
| kleinbuchstaben | Kleinbuchstaben bezeichnen Variablen, die bei der Eingabe vom Benutzer durch aktuelle Werte ersetzt werden müssen. | |
| { } | Geschweifte Klammern schließen Alternativen ein, d.h. eine der Angaben muß ausgewählt werden. | PRT {(LST) \| (SYSOUT) \| (BOTH) \| elem[(lib)] \| ?}<br>Einzugeben ist:<br>PRT (LST) oder<br>PRT (SYSOUT) oder<br>PRT (BOTH) oder<br>PRT PROT(1) oder<br>PRT ? |
| [ ] | Eckige Klammern kennzeichnen Wahlangaben. | LSTx elem[(lib)]<br>Einzugeben ist:<br>LSTM MAKRO oder<br>LSTM MAKRO(3) |
| ... | Punkte bedeuten eine Wiederholung; die davor stehende Einheit kann mehrmals hintereinander wiederholt werden. | elem(lib),...<br>Einzugeben ist:<br>A(1) oder<br>A(1),B(2) oder<br>A(1),B(2),C(3) usw. |

| Formale Darstellung | Erläuterung | Beispiel |
|---|---|---|
| _ | Die Unterstreichung hebt den Standardwert hervor. Das ist der Wert, den LMS einsetzt, wenn der Benutzer keine Angabe macht. Fehlt die Unterstreichung, so werden im Dialog- und Stapelbetrieb verschiedene Werte als Standardwerte genommen oder es existiert keine Standardzuweisung. | PAR TEST=[{YES / NO}]<br>Einzugeben ist:<br>PAR TEST=YES oder<br>PAR TEST=NO oder<br>PAR TEST=, wobei letzteres<br>PAR TEST=NO entspricht. |

### Format der Anweisungen

Die LMS-Anweisungen bestehen aus drei Teilen:

- Operation
- Operanden
- Kommentar

### Allgemeines Format:

```
[$]Operation_Operanden_Kommentar
```

An erster Stelle der Anweisung kann ein $-Zeichen stehen (mit Ausnahme der Anweisung $), es muß jedoch nicht angegeben werden.

### Operation

Die Operation muß am Anfang der Anweisung stehen. Sie besteht aus dem Anweisungsnamen und, falls mit der Anweisung Elemente bearbeitet werden, dem Elementtyp. Der Elementtyp braucht nicht angegeben zu werden, wenn im Verarbeitungsoperanden TYPE der entsprechende Elementtyp vereinbart ist.

*Beispiel*

| | |
|---|---|
| ADDx | Darstellung in der Anweisungssyntax |
| ADD | Anweisungsname |
| x | für x ist der Elementtyp anzugeben, z.B. ADDS für Quellprogramme |

Die von der jeweiligen Anweisung unterstützten Elementtypen sind in der Beschreibung der Anweisung aufgeführt.

### Operanden

Der Operation folgen, mindestens durch einen Zwischenraum getrennt, die Operanden. Operanden werden durch Kommata getrennt. In manchen Anweisungen werden zur Trennung von Operanden auch die Trennzeichen "=" und ">" verwendet. Das Zeichen ">" steht als Symbol für einen Pfeil, der die Bearbeitungsrichtung angibt.

Vor dem ersten und nach dem letzten Operanden der vollständigen Anweisung darf kein Trennzeichen stehen. In Operanden, sowie zwischen Operanden und Trennzeichen darf kein Zwischenraum sein. Anweisungen können höchstens 2028 Byte lang sein.

*Beispiel*

```
ADDS QUELL.DAT>QUELL.ELEM
```

Die Datei QUELL.DAT wird in der Bibliothek als Element QUELL.ELEM abgelegt.

### Kommentar

Nach den Operanden kann, durch mindestens einen Zwischenraum getrennt, Kommentar angefügt werden.

In Anweisungen ohne Operandenangabe darf kein Kommentar geschrieben werden.

Soll sich der Kommentar über eine ganze Zeile erstrecken (Kommentarzeilen), sind diese Zeilen mit * in Spalte 1 und Zwischenraum in Spalte 2 zu kennzeichnen.

Im Text des Kommentars dürfen die Zeichen ! und X'15' (NEW-LINE) nicht vorkommen, da sie als Anweisungstrenner interpretiert werden.

### Folgezeilen

Eine Anweisung kann aus einer oder mehreren Zeilen bestehen. Der Operationsteil muß am Anfang der ersten Zeile stehen, der Operandenteil kann sich über mehrere Zeilen erstrecken.

Um eine Fortsetzung anzuzeigen, muß unmittelbar auf eines der Trennzeichen ein Fortsetzungszeichen oder ein Zwischenraum folgen. Das Fortsetzungszeichen muß im Spaltenbereich von 1 bis 72 liegen.

Die Anweisung kann in der Folgezeile an beliebiger Stelle fortgesetzt werden.

Bei Subanweisungen (z.B. *COR nach der UPD-Anweisung) sind *keine* Folgezeilen möglich.

### Fortsetzungszeichen

Das Fortsetzungszeichen kann durch einen Bindestrich "-" oder das Zeichen Plus "+" dargestellt werden. Der Operand BASEVERSION darf nicht durch ein Fortsetzungszeichen von seinem Operandenwert getrennt werden.

Innerhalb einer Anweisung darf nach ",", ">" oder "=" getrennt werden.

### Anweisungen geblockt eingeben

Anweisungen können auch geblockt eingegeben werden.
Das bedeutet, daß im Dialogbetrieb nicht jede Anweisung einzeln übertragen werden muß, sondern für mehrere Anweisungen gemeinsam die Datenübertragung gestartet werden kann.

In einer Zeile können dabei mehrere Anweisungen - durch das Ausrufezeichen "!" getrennt - und Anweisungen, die sich über mehrere Zeilen erstrecken, - durch das logische Zeilenende getrennt - abgeschickt werden.

*Ausnahme:*

Nach einer CTL-Anweisung eingegebene geblockte Anweisungen werden nicht ausgeführt.

Das logische Zeilenende ist das für den jeweiligen Terminaltyp gültige NEW-LINE-Zeichen (NL). Standardmäßig wird es terminalabhängig durch die Zeichen \ oder < dargestellt.

# Übersicht über die Anweisungen

| Anweisung | Anwendungsbereich |
|---|---|
| ADD[x] [dateiname,... \| LINK=dateikettungsname \| [prefix.](name,...)[.suffix]] [>elemu]<br>[,BASEVERSION=[*NONE \| version \| *HIGH]]<br>ADD[R] *OMF[(modul,...)][>elemu]<br>ADD[x] FMS=fmslib(fmselem)[>elemu]<br>[,BASEVERSION=[*NONE \| version \| *HIGH]]<br>ADD[x] [(CTL) \| (CMD) \| ([SYS]DTA)]>elemu[,BASEVERSION=[*NONE \| version \| *HIGH]]<br>ADD[C] [dateiname \| LINK=dateikettungsname] | Aufnehmen von Daten in eine Bibliothek |
| COM[x] [elem,...[(lib)] \| (lib)]=elemu[(libu)][,{...}=...] | Vergleichen von Elementen |
| COR[x] elem[(lib)][>elemu] | Korrigieren von Textelementen |
| CTL [(CMD) \| ([SYS]DTA) \| (RDR) \| elem[(lib)] \| ?] | Steuern der Anweisungseingabe |
| DEL[x] [elem[(lib)] \| (lib)][,...] | Löschen von Elementen |

| Anweisung | Anwendungsbereich |
|---|---|
| DUP[x] {elem,...[(lib)] / (lib)} [>elemu][,{...}[>...]]<br>[,BASEVERSION={*NONE / version / *HIGH}]<br>DUP[x] name1[(lib)][>name2],STRUC=Y[ES] | Duplizieren von Elementen und Delta-Bäumen |
| EDT[x] elem [(lib)][>elemu]<br>EDR[x] elem [(lib)][>elemu]<br>EDT<br>EDR<br>EDT[x] elemu[(lib)]>*DUMMY<br>EDR[x] elemu[(lib)]>*DUMMY | Verzweigen in einen Editor; Erstellen, Korrigieren und Anschauen von Textelementen oder Dateien |
| END | Beenden des LMS-Laufs |
| LIB {FILE=bibliotheksname / LINK=dateikettungsname / [LIBRARY=]name / [LID=](lib)} [,[USAGE=]{IN / OUT / BOTH}]<br>[,[FORMAT=]{PL / OML / OSM}][,[STATE=]{O[LD] / N[EW] / A[NY]}]<br>LIB C[LOSE][,{FILE=bibliotheksname / LINK=dateikettungsname / [LIBRARY=]name / [LID=](lib)}]<br>LIB ? | Zuweisen und Schließen von Bibliotheken |
| LST[x] {elem[(lib)] / (lib)}[,...] | Auflisten von Elementen |

| Anweisung | Anwendungsbereich |
|---|---|
| NAM[x] { elem,...[(lib)] / (lib) } >elemu[,{...}>...] | Umbenennen von Elementen |
| NOP [zeichenfolge] | Leerfunktion |
| NUM[x] { elem[(lib)] / (lib) } [>elemu] | Numerieren von Elementsätzen |
| PAR [ { parname=[ { parwert / ? } ] / ? } ,{...}] | Setzen von Verarbeitungsoperanden |
| PRT { (LST) / (SYSLOUT) / (CON) / (BOTH) / elem[(lib)] / ? } | Steuern der Protokollausgabe |
| RST [STOP] | Verlassen des TEST-Modus |
| SEL[x] elem,...[> { [prefix.](name)[.suffix] / dateiname / LINK=dateikettungsname } ]<br>SEL[x] elem>FMS=fmslib(fmselem) | Ausgeben von Elementen in Dateien und FMS-Bibliotheken |
| SUM ['text'] | Speichern von Vergleichsstatistiken |
| SUMPRT { S1 / S2 } [,'text'] | Ausgeben der Vergleichsstatistik |
| SUMADD { S1 / S2 } > { S1 / S2 } | Addieren von Vergleichsstatistiken |
| SUMDEL { S1 / S2 } | Löschen der Vergleichsstatistik |

| Anweisung | Anwendungsbereich |
|---|---|
| SYS [{'systemkommando' / systemkommando}] | Absetzen von Systemkommandos |
| TCH {[O[FLOW]=[N[O] / A[CK] / T[IMER] / ?]][,N[EWSCREEN]={Y[ES] / N[O] / ?}] / ?} | Steuern des Bildschirmwechsels |
| TOC[x] [{elem[(lib)] / (lib)},...] | Auflisten eines Inhaltsverzeichnis einer Bibliothek |
| UPD[x] elem[(lib)][>elemu] | Korrigieren von Binde- und Lademodulen und LLMs |
| USE{[{LST / COMP / COMS}[={entry / {* / bibliothekb}(entry)}]] / [{EDTLIB / EDORLIB / FMSLIB}=bibliotheku] / ?} | Verzweigen in Benutzerprogramme |
| $ | Ausgeben des Anweisungspuffers |

### Tabelle der zulässigen Elementtypen pro Anweisung

Folgende Tabelle gibt einen Überblick darüber, welche Elementtypen in den einzelnen Anweisungen zulässig sind:

| Elementtyp / Anweisung | S | M | R | J | P | C | D | X | H | L | F | U | * |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADD | + | + | + | + | + | + | + | + | | | | | |
| COM | + | + | | + | + | | + | + | | | | | |
| COR | + | + | | + | + | | + | + | | | | | |
| DEL | + | + | + | + | + | + | + | + | + | + | + | + | + |
| DUP | + | + | + | + | + | + | + | + | + | + | + | + | + |
| EDT/EDR | + | + | | + | + | | + | + | | | | | |
| LST | + | + | + | + | + | + | + | + | + | + | + | + | + |
| NAM | + | + | + | + | + | + | + | + | + | + | + | + | + |
| NUM | + | + | | + | + | | + | + | | | | | |
| SEL | + | + | + | + | + | + | + | + | | | | | |
| TOC | + | + | + | + | + | + | + | + | + | + | + | + | + |
| UPD | | | + | | | + | | | | + | | | |

+: Elementtyp bei Anweisung zulässig

leeres Feld: Elementtyp bei Anweisung unzulässig

### Tabelle der benötigten Bibliotheken

Aus nachfolgender Tabelle ist ersichtlich, welche Bibliotheken von den einzelnen LMS-Anweisungen benötigt werden:

| Funktion | von LMS werden benötigt | |
|---|---|---|
| | Eingabe-Bib. | Ausgabe-Bib. |
| ADD | nein | ja |
| COM 1) | ja | nein |
| COR 3) | ja | ja |
| CTL | ja | nein |
| DEL 2) | ja | nein |
| DUP 3) | ja | ja |
| EDR 3) | ja 4) | nein |
| EDT 3) | ja 4) | nein |
| LST | ja | nein |
| NAM 2) | ja | nein |
| NUM 3) | ja | ja |
| PRT | nein | ja |
| TOC | ja | nein |
| SEL | ja | nein |
| SUM | nein | nein |
| SUMADD | nein | nein |
| SUMDEL | nein | nein |
| SUMPRT | nein | nein |
| UPD 3) | ja | ja |

1) Die Vergleichselemente dürfen auf verschiedenen Eingabebibliotheken liegen. Sie dürfen nicht auf derselben sequentiellen Bibliothek liegen.

2) Die Eingabebibliothek wird zum Schreiben eröffnet.

3) Ein- und Ausgabebibliotheken dürfen mit Ausnahme von sequentiellen Bibliotheken identisch sein.

4) Die Eingabebibliothek wird nur benötigt, falls ein vorhandenes Element mit den Editoren korrigiert wird.

# ADD Aufnehmen von Daten in eine Bibliothek

ADD nimmt als Bibliothekselemente auf:

- Dateien
- Module aus dem EAM-Bereich
- Elemente aus einer FMS-Bibliothek
- Elementsätze aus dem LMS-Anweisungsstrom
- Lademodule (BS2000-Phasen) als BS1000-Phasen in sequentiellen Bibliotheken

Für diese verschiedenen Funktionen hat ADD fünf verschiedene Formate:

### Format 1

Aufnehmen von Dateien:

| Operation | Operanden |
|---|---|
| ADD[x] | {dateiname,...<br>{LINK=dateikettungsname<br>[[prefix.](name,...)[.suffix]]}[>elemu]<br>[,BASEVERSION={*NONE<br>{version<br>{*HIGH}] |

### Format 2

Aufnehmen von Modulen aus dem EAM-Bereich:

| Operation | Operanden |
|---|---|
| ADD[R] | *OMF[(modul,...)][>elemu] |

### Format 3

Übernehmen eines Elementes aus einer FMS-Bibliothek in eine LMS-Bibliothek:

| Operation | Operanden |
|---|---|
| ADD[x] | FMS=fmslib(fmselem)[>elemu][,BASEVERSION={*NONE / version / *HIGH}] |

### Format 4

Erzeugung eines Elementes durch Aufnehmen von Elementsätzen aus dem LMS-Anweisungsstrom:

| Operation | Operanden |
|---|---|
| ADD[x] | [{(CTL) / (CMD) / ([SYS]DTA)}]>elemu[,BASEVERSION={*NONE / version / *HIGH}] |

### Format 5

Aufnehmen von BS2000-Lademodulen als BS1000-Phasen in sequentielle Bibliotheken (Bandbibliotheken):

| Operation | Operanden |
|---|---|
| ADD[C] | {dateiname / LINK=dateikettungsname} |

**Format 1:** Aufnehmen von Dateien

Mit diesem Format von ADD werden Dateien als Elemente in die eröffnete Ausgabebibliothek aufgenommen. Wenn die Ausgabebibliothek eine Programmbibliothek ist, kann ein Element entweder als Voll-Element oder als Delta-Element gespeichert werden. Die Ausgabebibliothek muß vorher mit LIB zugewiesen werden.

Dateien, die mit RECORD-FORMAT=U katalogisiert sind, können ebenfalls in Bibliotheken aufgenommen werden. Dateien, die RECORD-FORMAT=FIXED haben, können nur über PAR KEY=YES abgespeichert werden.
Die Angaben zu BLKSIZE und RECSIZE können beliebig sein. Die maximale Satzlänge von 32 KByte (inkl. Satzkopf) darf allerdings nicht überschritten werden.
Dateigenerationsgruppen können nur über LINK= und eine LMS-gültige Elementbezeichnung aufgenommen werden.

| Operation | Operanden |
|---|---|
| ADD[x] | [dateiname,... ] {LINK=dateikettungsname }[>elemu] [[prefix.](name,...)[.suffix]] [,BASEVERSION={*NONE \| version \| *HIGH}] |

ADDx Name der Anweisung mit Angabe des Elementtyps.

- Für Voll-Elemente sind folgende Elementtypen zulässig:
  S, M, R, J, P, C, D, X
- Für Delta-Elemente sind nur die textartigen Elementtypen zulässig:
  S, M, J, P, D

Der Elementtyp braucht nicht angegeben zu werden, wenn im Verarbeitungsoperanden TYPE der entsprechende Elementtyp vereinbart ist.

dateiname vollqualifizierter Dateiname oder Auswahlangabe.
Für dateiname kann auch pfadname (siehe Benutzer-Kommandos [7]) angegeben werden.
Werden temporäre Dateien aufgenommen, ist eine Auswahlangabe nicht zulässig.
Die Angabe mehrerer Dateinamen ist nur sinnvoll, wenn für elemu * oder eine Konstruktionsangabe gemacht wird.

LINK=dateikettungsname
Dateikettungsname, der auf die Datei verweist.

[prefix.](name,...)[.suffix]

Mit dieser Angabe können mehrere Dateien für eine Aufnahme in die Bibliothek ausgewählt werden.

prefix
gibt die gemeinsamen vorderen Teilnamen der auszuwählenden Dateien an. 'prefix' muß mit einem Punkt aufhören.

suffix
gibt die gemeinsamen hinteren Teilnamen der auszuwählenden Dateien an. 'suffix' muß mit einem Punkt beginnen.

name
gibt den oder die Teilnamen an, mit dem prefix und/oder suffix zu einem oder mehreren vollqualifizierten Dateinamen ergänzt werden. Mit diesen Teilnamen werden die erzeugten Elemente auch in der Bibliothek abgelegt, wenn elemu nicht angegeben wird. Für 'name' ist auch eine Auswahlangabe zulässig.

Die Angabe von teilqualifizierten Dateinamen ist in der Form prefix.(*).suffix möglich.

Läßt man bei (name) die runden Klammern weg, verhält sich ADD wie bei der Angabe von dateiname.

elemu — Bezeichnung des zu erzeugenden Elementes.
Bei der Angabe [prefix.](name,...)[.suffix] oder einer Auswahlangabe für dateiname sind auch Konstruktionsangaben zulässig. elemu darf fehlen, dann erhält das erzeugte Element den Namen der Eingabedatei ohne prefix und suffix.

Für die Elementbezeichnungen der zu erzeugenden Elemente sind die unterschiedlichen Syntaxregeln für Programmbibliotheken (siehe Seite 25) und übrige Bibliothekstypen (siehe Seite 29) zu beachten.

BASEVERSION

bestimmt das Basis-Element bei der Delta-Speicherung. Er bezieht sich auf alle zu erzeugenden Elemente (elemu). Er muß der letzte Operand der Anweisung sein.
Fehlt dieser Operand wird ein Voll-Element erstellt.

=*NONE — elemu wird als erstes Element (=Basis) eines Delta-Baumes abgelegt (Ersterstellung).

=version — Basis-Element ist das Element mit version als Versionsangabe; dieses Element muß existieren und als Delta-Element gespeichert sein.
Eine Konstruktionsangabe ist nicht erlaubt.

=*HIGH Basis-Element ist das Element mit der höchsten Versionsbezeichnung (zum Erstellzeitpunkt des Delta-Elementes). Das erzeugte Delta-Element wird an dieses Basis-Element angehängt.

**Verarbeitungsoperanden**

TYPE bestimmt den Elementtyp, falls bei der Anweisung selbst kein Typ angegeben wird.

KEY bestimmt, ob Dateimerkmale und vorhandene ISAM-Schlüssel mit aufgenommen werden.

DESTROY bestimmt, ob im Ausgabeelement ein Kennzeichen für physikalisches Löschen eingetragen wird.
(Nur bei Programmbibliotheken möglich.)

OVERWRITE bestimmt, ob ein gleichnamiges Voll-Element in der Ausgabebibliothek überschrieben, nicht überschrieben oder mit den neuen Datensätzen erweitert wird.
Dieser Verarbeitungsoperand ist für Delta-Elemente unzulässig.

RANGE bestimmt Lage und Länge des Kennungsfeldes in den Ausgabesätzen von Textelementen.

VALUE bestimmt Anfangswert und Schrittweite für die Neunumerierung bei Textelementen im Kennungsfeld.

STRING bestimmt, ob der ISAM-Schlüssel im Kennungsfeld abgelegt werden soll, bzw. gibt die Zeichenfolge an, die linksbündig in das Kennungsfeld der Ausgabesätze eingetragen wird.

CHECK bestimmt Lage und Länge des Kennungsfeldes in Eingabesätzen und veranlaßt Überprüfung auf aufsteigende Numerierung.

*Beispiel 1*

Aus den Dateien

A.B.C.A A.B.C.B A.B.C.C B.B.C.A C.B.C.A

werden mit

```
ADDx A.B.(*)>X.*       die Dateien A.B.C.A ⟶ Elementname X.A
(nur für Programm-                 A.B.C.B ⟶ Elementname X.B
bibliothek möglich)                A.B.C.C ⟶ Elementname X.C

ADDx (A,C).B.C.A       die Dateien A.B.C.A ⟶ Elementname A
                                   C.B.C.A ⟶ Elementname C
```

ausgewählt.

*Beispiel 2*

```
/START-PROGRAM $LMS
$LIB TESTLIB,OUT
$ADDS S.PROG>SPROG
$END
```

Das Quellprogramm mit dem Dateinamen S.PROG wird in der Programmbibliothek TESTLIB als Element vom Typ S mit der Elementbezeichnung SPROG abgelegt.

*Beispiel 3*

```
/SET-FILE-LINK FILE-NAME=PROG.DAT,LINK-NAME=PROG
/SET-FILE-LINK FILE-NAME=TEST.COB,LINK-NAME=TEST
/START-PROGRAM $LMS
$LIB TESTLIB,OUT
$ADDD LINK=PROG>DDAT
$ADDS LINK=TEST>TCOB
$END
```

Die Dateien PROG.DAT und TEST.COB werden in die Programmbibliothek TESTLIB als Elemente DDAT und TCOB aufgenommen. Sie werden bei ADD über ihre Dateikettungsnamen angesprochen.

**Format 2:** Aufnehmen von Modulen aus dem EAM-Bereich

Mit diesem Format von ADD werden Module aus dem EAM-Bereich des laufenden Prozesses übernommen und als Elemente mit dem Elementtyp R in die zugewiesene Ausgabebibliothek abgelegt. Ausgabebibliotheken können Programm- und Modulbibliotheken und sequentielle Bibliotheken sein.

| Operation | Operanden |
|---|---|
| ADD[R] | *OMF[(modul,...)][>elemu] |

ADDR — Name der Anweisung mit dem Elementtyp R.

Der Elementtyp braucht nicht angegeben zu werden, wenn im Verarbeitungsoperanden TYPE der Elementtyp R vereinbart ist.

modul — maximal 8-stelliger Name eines Moduls im EAM-Bereich.
Sind im EAM-Bereich mehrere gleichnamige Module enthalten, übernimmt LMS den zuletzt übersetzten Modul in die Bibliothek.
Die Angabe modul darf fehlen, LMS nimmt dann alle Module auf, die sich im EAM-Bereich befinden.
Eine Auswahlangabe ist erlaubt.

elemu — Elementbezeichnung des zu erstellenden Elementes.
Eine Konstruktionsangabe ist erlaubt.
elemu darf fehlen. Die Module erhalten dann die Namen, die sie im EAM-Bereich haben.

*Wichtiger Hinweis*

- Wird für elemu eine Elementbezeichnung angegeben, muß für 'modul' der Modulname des aufzunehmenden Moduls angegeben werden, wenn sich mehrere Module im EAM-Bereich befinden.
  Ansonsten werden alle Module aus dem EAM-Bereich unter der Bezeichnung 'elemu' aufgenommen, wobei der jeweils vorher aufgenommene überschrieben wird.
- Die Angabe OVERWRITE = EXTEND führt auf Fehler.

*Beispiel*

```
/START-PROGRAM $LMS
$LIB TESTLIB,OUT
$ADDR *OMF(MOD1,MOD2)>ELM*
  .
  .
  .
$END
```

Die Bindemodule MOD1 und MOD2 aus dem EAM-Bereich werden als Elemente ELM1 und ELM2 in die Ausgabebibliothek TESTLIB übernommen.

**Format 3:** Übernehmen eines Elementes aus einer FMS-Bibliothek

Mit diesem Format von ADD werden Elemente aus einer FMS-Bibliothek (siehe FMS [10]) in die eröffnete Ausgabebibliothek übernommen. Wenn die Ausgabebibliothek eine Programmbibliothek ist, kann ein Element entweder als Voll-Element oder als Delta-Element gespeichert werden. Die Ausgabebibliothek muß vorher mit LIB zugewiesen werden.

Bei dieser Funktion wird intern FMS aufgerufen.

| Operation | Operanden |
|---|---|
| ADD[x] | FMS=fmslib(fmselem)[>elemu][,BASEVERSION=[*NONE / version / *HIGH]] |

ADDx Name der Anweisung mit der Angabe des Elementtyps.

- Für Voll-Elemente sind folgende Elementtypen zulässig: S, M, R, J, P, D, X
- Für Delta-Elemente sind nur die textartigen Elementtypen zulässig: S, M, J, P, D

Der Elementtyp braucht nicht angegeben zu werden, wenn im Verarbeitungsoperanden TYPE der entsprechende Elementtyp vereinbart ist.

fmslib vollqualifizierter Dateiname einer FMS-Bibliothek.
Für fmslib kann auch pfadname (siehe Benutzer-Kommandos [7]) angegeben werden.

fmselem vollständiger Elementname des zu übernehmenden Elementes oder *.
Mit * kann eine komplette FMS-Bibliothek kopiert werden.

elemu Elementbezeichnung des zu erzeugenden Elementes oder *. Mit * werden in die LMS-Bibliothek die Elemente mit den gleichen Namen wie in der FMS-Bibliothek übernommen.

BASEVERSION
bestimmt das Basis-Element bei der Delta-Speicherung. Er muß der letzte Operand der Anweisung sein.
Fehlt dieser Operand wird ein Voll-Element erstellt.

=*NONE elemu wird als erstes Element (=Basis) eines Delta-Baums abgelegt (Ersterstellung).

=version Basis-Element ist das Element mit version als Versionsangabe; dieses Element muß existieren und als Delta-Element gespeichert sein. Eine Konstruktionsangabe ist nicht erlaubt.

=*HIGH Basis-Element ist das Element mit der höchsten Versionsbezeichnung (zum Erstellzeitpunkt des Delta-Elementes). Das erzeugte Delta-Element wird an dieses Basis-Element angehängt.

*Hinweis*

- Ist OVERWRITE=EXTEND angegeben, wird die Funktion mit Fehler abgebrochen.
- Es werden immer Sätze mit variablem Satzformat erwartet.

**Verarbeitungsoperanden**

RANGE bestimmt Lage und Länge des Kennungsfeldes in den Ausgabesätzen von Textelementen.

VALUE bestimmt Anfangswert und Schrittweite für die Neunumerierung bei Textelementen im Kennungsfeld.

STRING bestimmt, ob der ISAM-Schlüssel im Kennungsfeld abgelegt werden soll, bzw. gibt die Zeichenfolge an, die linksbündig in das Kennungsfeld der Ausgabesätze eingetragen wird.

OVERWRITE bestimmt, ob ein gleichnamiges Voll-Element in der Ausgabebibliothek überschrieben, nicht überschrieben oder mit den neuen Datensätzen erweitert wird.
Dieser Verarbeitungsoperand ist für Delta-Elemente unzulässig.

CHECK bestimmt Lage und Länge des Kennungsfeldes in Eingabesätzen und veranlaßt Überprüfung auf aufsteigende Numerierung.

*Beispiel*

```
/START-PROGRAM $LMS
$LIB LMSLIB,OUT,NEW
$ADDS FMS=FMSBIB2(SRC)>SRC1
$END
```

Das Element SRC aus der FMS-Biblitohek FMSBIB2 wird in die neu zu erstellende Programmbibliothek LMSLIB als Element SRC1 aufgenommen.

**Format 4:** Erzeugung eines Elementes durch Aufnehmen von Elementsätzen aus dem LMS-Anweisungsstrom

Mit diesem Format von ADD werden die Datensätze aus einem LMS-Anweisungsstrom in ein Element der eröffneten Ausgabebibliothek aufgenommen. Wenn die Ausgabebibliothek eine Programmbibliothek ist, kann das Element entweder als Voll-Element oder als Delta-Element gespeichert werden. Die Ausgabebibliothek muß vorher mit LIB zugewiesen werden. Die Datensätze müssen direkt auf ADD folgen. Die Folge der Datensätze muß mit ***END** abgeschlossen werden.

Diese Datensätze werden eingegeben von

- der Datensichtstation,
- der Systemdatei SYSDTA,
- einem Bibliothekselement.

| Operation | Operanden |
|---|---|
| ADD[x] | [{(CTL) / (CMD) / ([SYS]DTA)}]>elemu[,BASEVERSION={*NONE / version / *HIGH}] |

ADDx — Name der Anweisung mit Angabe des Elementtyps.

- Für Voll-Elemente sind die folgenden Elementtypen zulässig: S, M, R, J, P, D, X
- Für Delta-Elemente sind nur die textartigen Elementtypen zulässig: S, M, J, P, D

Der Elementtyp braucht nicht angegeben zu werden, wenn im Verarbeitungsoperanden TYPE der entsprechende Elementtyp vereinbart ist.

CTL — Die Datensätze werden aus dem Eingabemedium aufgenommen, das mit CTL festgelegt wurde, d.h. entweder die Datensichtstation oder die Systemdatei SYSDTA oder ein Bibliothekselement.

CMD — Die Datensätze werden im Dialog von der Datensichtstation und im Stapelbetrieb von der Systemdatei SYSDTA gelesen.

SYSDTA — Die Datensätze werden von der Systemdatei SYSDTA gelesen.

*Hinweis:*

Werden Datensätze von der Systemdatei SYSDTA gelesen, so dürfen sie nicht mit "/" beginnen. Der Grund dafür ist, daß der RDATA-Makro solche Sätze als Kommandos interpretiert und daher den Return-Code für EOF übergibt. Es ist damit nicht möglich, System-Kommandos als Datensätze zu übergeben.

elemu — Elementbezeichnung des zu erzeugenden Elementes.
Eine Konstruktionsangabe ist nicht erlaubt.
elemu darf nicht fehlen.

BASEVERSION
bestimmt das Basis-Element bei der Delta-Speicherung. Er muß der letzte Operand der Anweisung sein.
Fehlt dieser Operand wird ein Voll-Element erstellt.

=*NONE — elemu wird als erstes Element (= Basis) eines Delta-Baums abgelegt (Ersterstellung).

=version — Basis-Element ist das Element mit version als Versionsangabe; dieses Element muß existieren und als Delta-Element gespeichert sein.
Eine Konstruktionsangabe ist nicht erlaubt.

=*HIGH — Basis-Element ist das Element mit der höchsten Versionsbezeichnung (zum Erstellzeitpunkt des Delta-Elementes). Das erzeugte Delta-Element wird an dieses Basis-Element angehängt.

**Verarbeitungsoperanden**

TYPE — bestimmt den Elementtyp, falls bei der Anweisung selbst kein Typ angegeben wird.

DESTROY — bestimmt, ob im Ausgabeelement ein Kennzeichen für physikalisches Löschen eingetragen wird.
(Nur bei Programmbibliotheken möglich.)

OVERWRITE — bestimmt, ob ein gleichnamiges Voll-Element in der Ausgabebibliothek überschrieben wird oder nicht. OVERWRITE=EXTEND ist nicht erlaubt. Dieser Verarbeitungsoperand ist für Delta-Elemente unzulässig.

RANGE — bestimmt Lage und Länge des Kennungsfeldes in den Ausgabesätzen von Textelementen.

STRING — bestimmt bei Textelementen, ob der ISAM-Schlüssel im Kennungsfeld abgelegt werden soll, bzw. gibt die Zeichenfolge an, die linksbündig in das Kennungsfeld der Ausgabesätze eingetragen wird.

VALUE — bestimmt Anfangswert und Schrittweite für die Neunumerierung bei Textelementen im Kennungsfeld.

### *END Beenden der Aufnahme

*END beendet die Aufnahme von Datensätzen in ein Element.

| Operation | Operanden |
|---|---|
| *END | |

*END Name der Anweisung

*Beispiel*

```
/START-PROGRAM $LMS
$LIB BIBLIOTHEK,BOTH
$ADDD >BRIEF.A
* Sehr geehrte ...
.
.
.
**END
$END
```

Der Text 'Sehr geehrte...' ist im Element BRIEF.A in Großbuchstaben gespeichert. Möchte man auch Kleinbuchstaben im Element gespeichert haben, muß der Verarbeitungsoperand LCASE verwendet werden (siehe Seite 227).

**Format 5:** Aufnehmen von BS2000-Lademodulen als BS1000-Phasen in sequentielle Bibliotheken (Bandbibliotheken)

Mit diesem Format von ADD werden aus einem BS2000-Lademodul BS1000-Phasen in einer Bandbibliothek erzeugt.

| Operation | Operanden |
|---|---|
| ADD[C] | [dateiname / LINK=dateikettungsname] |

ADDC Name der Anweisung mit dem Elementtyp C.

Der Elementtyp braucht nicht angegeben zu werden, wenn im Verarbeitungsoperanden TYPE der Elementtyp C vereinbart ist.

dateiname vollqualifizierter Dateiname einer mit dem Binder TSOSLNK erzeugten Programmdatei.
Für dateiname kann auch pfadname (siehe Benutzer-Kommandos [7]) angegeben werden.

LINK=dateikettungsname
Dateikettungsname, der auf eine mit dem /SET-FILE-LINK-Kommando zugewiesene BS2000-Programmdatei verweist. Die Datei muß mit dem Binder TSOSLNK erzeugt worden sein.

Das Element erhält in der sequentiellen Bibliothek den Namen, den der Lademodul in der Programmdatei trägt.

Die Bibliothek muß vorher mit LIBOUT ...,NEWLIB (siehe Seite 345) zugewiesen werden.

Bitte beachten Sie auch das Kapitel "Kompatibilität BS2000-BS1000" auf Seite 341.

### Verarbeitungsoperanden

TYPE bestimmt den Elementtyp, falls bei der Anweisung selbst kein Typ angegeben wird.

*Beispiel*

Der BS2000-Lademodul PROG.DAT wird in die Bandbibliothek DOS-LIB aufgenommen.

```
/CREATE-FILE FILE-NAME=DOS-LIB, -
/            SUPPORT=TAPE(VOLUME=E1000A, DEVICE-TYP=T9P)
/SET-FILE-LINK FILE-NAME=DOS-LIB, LINK-NAME=LIB001, ACCESS-METHOD=BTAM
/START-PROGRAM $LMS
$LIBOUT (1),NEWLIB
$ADDC PROG.DAT
  .
  .
  .
$END
```

Die Bandbibliothek DOS-LIB wird mit einem /CREATE-FILE-Kommando erzeugt und mit einem /SET-FILE-LINK-Kommando zugewiesen.
Über LIBOUT (1),NEWLIB wird sie als Ausgabebibliothek festgelegt. Der Modul aus der Datei PROG.DAT wird unter Beibehaltung seines Namens auf das Band übernommen.

# COM Vergleichen von Elementen

Mit COM werden Elemente miteinander verglichen. Die dabei festgestellten Unterschiede werden in einem Vergleichsprotokoll und einer Vergleichsstatistik dargestellt. Die Elemente können sich in verschiedenen Bibliotheken befinden. Die miteinander zu vergleichenden Elemente dürfen nicht in derselben sequentiellen Bibliothek stehen. Mit dem Verarbeitungsoperanden COMPARE wird gesteuert, ob es sich um einen formalen oder logischen Vergleich handelt und mit welchem Vergleichs-Algorithmus verglichen werden soll. Außerdem werden die Vergleichsfelder festgelegt.

Weitere Informationen über Vergleichsprotokoll und Vergleichsstatistik siehe Seite 50. Beispiele zu COM siehe auf den Seiten 264, 276 und 284.

Mit USE kann über ein Benutzerunterprogramm vor dem eigentlichen Vergleich auf die Elementsätze zugegriffen werden.

| Operation | Operanden |
|---|---|
| COM[x] | [elem,... [(lib)]]<br>{ }=elemu[(libu)][,{...}=...]<br>[(lib)] |

COMx — Name der Anweisung mit Angabe des Elementtyps:
S, M, J, P, D, X

Der Elementtyp braucht nicht angegeben zu werden, wenn im Verarbeitungsoperanden TYPE der entsprechende Elementtyp vereinbart ist.

elem — Elementbezeichnung der Primärelemente, die mit dem Element elemu verglichen werden, oder Auswahlangabe.

lib — Kurzbezeichnung der Eingabebibliothek für elem.

elemu — Elementbezeichnung für das Sekundärelement oder Konstruktionsangabe.

libu — Kurzbezeichnung der Eingabebibliothek für elemu.

**Verarbeitungsoperanden**

| | |
|---|---|
| TYPE | bestimmt den Elementtyp, falls bei der Anweisung selbst kein Typ angegeben wird. |
| LINE | bestimmt die Zeilen- und Spaltenanzahl einer Protokollseite bei Ausgabe in die Systemdatei SYSLST oder ein Bibliothekselement. |
| CHECK | nur beim Kreuzvergleich wirksam, d.h. es muß im Verarbeitungsoperanden COMPARE für den Synchronisierungszähler ein Wert zwischen 1 und 9 angegeben werden.<br>Definiert Lage und Länge des Kennungsfeldes in den zu vergleichenden Sätzen und veranlaßt Überprüfung auf aufsteigende Numerierung. |
| COMPARE | steuert Lage und Länge des Vergleichsfeldes, die Vergleichsart, die Synchronisierungszahl, die Protokollart und das Erzeugen von Korrekturanweisungen. |
| SUM | steuert die Ablage der Vergleichsstatistik. |

Die Funktion wird auch dann ausgeführt, wenn nur eines der Vergleichselemente in den angegebenen Bibliotheken gefunden wird. Dies ermöglicht das Zählen der Sätze in Elementen.

Über den Verarbeitungsoperanden SUM kann die Vergleichsstatistik gespeichert werden. Mit SUM, SUMADD, SUMPRT und SUMDEL können gespeicherte Vergleichsstatistiken weiterverarbeitet werden.

Das Setzen des Auftragsschalters 9 ermöglicht eine Synchronisation von bis zu 12000 ungleichen Sätzen (siehe Seite 74).

**Synchronisation**

Beim Vergleichen versucht LMS nach ungleichen Satzfolgen möglichst wieder auf gleichen Satzfolgen aufzusetzen. Dies wird als Synchronisation bezeichnet. Stimmen mindestens soviele Sätze überein wie im Verarbeitungsoperanden COMPARE gefordert werden, gilt der Synchronisierungsversuch als erfolgreich. Solange nicht erfolgreich synchronisiert werden konnte, werden auch gleiche Sätze als ungleich protokolliert.

*Beispiel*

```
/SET-FILE-LINK FILE-NAME=BIBU,LINK-NAME=LIB005
/START-PROGRAM $LMS
$LIB FILE=PLIB
$PAR COMPARE=5/26/L/MAX
$COMS ASRC=ASRC(5)
  .
  .
  .
$END
```

Die Elemente ASRC der Bibliotheken BIBU und PLIB werden verglichen. Mit dem Verarbeitungsoperanden COMPARE (siehe Seite 214) wird der Vergleichsbereich (5. bis 30. Byte des Elementsatzes), die Vergleichsart (logischer Vergleich) und der Umfang des Vergleichsprotokolls (MAX) festgelegt.

Weitere Beispiele siehe ab Seite 255.

# COR Korrigieren von Textelementen

COR korrigiert das angegebene Element und gibt es in die zugewiesene Ausgabebibliothek aus. Die Korrekturen werden über Korrekturanweisungen ausgeführt, die unmittelbar nach COR erwartet werden.

COR bearbeitet Elementsätze mit einer Länge ≤251 Byte. Längere Sätze werden abgeschnitten. In diesem Fall gibt LMS eine Warnung aus.

Ein- und Ausgabebibliotheken dürfen bei sequentiellen Bibliotheken nicht identisch sein. Bei den anderen Bibliothekstypen dürfen Ein- und Ausgabebibliothek identisch sein.

| Operation | Operanden |
|---|---|
| COR[x] | elem[(lib)][>elemu] |

CORx — Name der Anweisung mit Angabe des Elementtyps:
S, M, J, P, D, X

Der Elementtyp braucht nicht angegeben zu werden, wenn im Verarbeitungsoperanden TYPE der entsprechende Elementtyp vereinbart ist.

elem — Elementbezeichnung des zu korrigierenden Elementes oder Auswahlangabe (keine Listenangabe).

lib — Kurzbezeichnung der Eingabebibliothek.

elemu — Elementbezeichnung für das Ausgabeelement oder Konstruktionsangabe.

**Verarbeitungsoperanden**

TYPE — bestimmt den Elementtyp, falls bei der Anweisung selbst kein Typ angegeben wird.

DESTROY — bestimmt, ob im Ausgabeelement ein Kennzeichen für physikalisches Löschen gesetzt wird.
(Nur bei Programmbibliotheken möglich.)

CHECK — definiert Lage und Länge des Kennungsfeldes in Eingabesätzen und veranlaßt Überprüfung auf aufsteigende Numerierung.

RANGE — definiert Lage und Länge des Kennungsfeldes in den Ausgabesätzen.

VALUE — gibt Anfangswert und Schrittweite für die Neunumerierung im Kennungsfeld an.

STRING gibt die Zeichenfolge an, die linksbündig in das Kennungsfeld der Ausgabesätze eingetragen wird.

OVERWRITE steuert das Überschreiben von gleichnamigen Elementen in der Ausgabebibliothek.
Der Verarbeitungsoperand wird jedoch nicht ausgewertet, wenn

```
Eingabebibliothek=Ausgabebibliothek und
                  elem=elemu
```

Das Eingabeelement wird dann überschrieben.

LOG steuert den Umfang des Korrektur-Protokolls.

*Hinweis*

In der Bibliothek gespeicherte PAM-Dateien sind nicht korrigierbar.

Bei der Korrektur wird ein neues Element nur dann erzeugt, wenn LMS nicht im TEST-Modus ist und wenn kein Fehler beim Korrigieren erkannt wurde.

Wurde im Dialog ein Fehler erkannt, muß die Korrektur mit *END beendet und die Korrektur neu durchgeführt werden.

Das korrigierte Element wird mit dem alten Namen und um 1 erhöhter Variantennummer bei Programmbibliotheken (siehe Seite 27) bzw. um 1 erhöhter Versionsnummer bei den übrigen Bibliotheken (siehe Seite 30) und Tagesdatum in die Ausgabebibliothek geschrieben.

Falls ein Kennungsfeld in den Eingabesätzen definiert ist (CHECK ungleich NO), wird ein aufsteigend numeriertes Eingabeelement vorausgesetzt. Mit Zwischenraum gelöschte Kennungsfelder sind erlaubt und führen nicht zu Reihenfolgefehlern. Es sind auch kürzere Sätze erlaubt, die kein CHECK-Feld enthalten.

Die Korrekturanweisungen müssen in aufsteigender Folge vorliegen (sequentielle Abarbeitung im Element).
Soll ein Element nur neu numeriert werden, genügt *END hinter COR.

Die Ausgabe des Korrekturprotokolls wird mit dem Verarbeitungsoperanden LOG gesteuert.

```
Bei  LOG=MIN  werden alle Änderungen aufgelistet;
     LOG=MED  werden zusätzlich die Korrekturanweisungen ausgegeben;
     LOG=MAX  werden auch noch alle unverändert aus dem Eingabeelement
              übernommenen Sätze protokolliert.
```

Korrekturen können entweder über Korrekturanweisungen oder durch die Angabe eines einzufügenden oder zu ersetzenden Datensatzes mit Satzkennung erfolgen.

### Übersicht der Korrekturanweisungen

| Korrekturanweisung | Funktion |
|---|---|
| *INSERT oder Datensatz mit Kennung oder Satznummer | Einfügen von Sätzen |
| *DELETE | Löschen von Sätzen |
| *REPLACE oder Datensatz mit Kennung oder Satznummer | Ersetzen von Sätzen |
| *CHANGE | Ändern von Sätzen |
| *END | Beenden der Korrektureingaben |
| */*/ | Begrenzen eines eingegebenen Satzes |

### Format

Korrekturanweisungen beginnen mit *. Dem Stern in Spalte 1 folgen die Operation, d.h. der Name der Korrekturanweisung, ein Zwischenraum und die Satzbezeichnung. Die Satzbezeichnung wird in der Beschreibung der Korrekturanweisung 'satzbez' genannt. Die Satzbezeichnung wird durch die Satznummer oder die Satzkennung dargestellt (siehe Seite 47).

Satznummer: #zahl

zahl ist eine ganze positive Zahl mit maximal 8 Stellen. Die Satznummer ist die relative Position des Elementsatzes bezogen auf den Elementanfang (aus der Auflistung des Elements mit Angabe PAR LST=art/NUM zu entnehmen). Ist zahl größer als die größte Satznummer des Elementes, wird die Korrektur nach dem letzten Satz fortgesetzt, d.h. Sätze an das Element angefügt.

*Beispiel*

#26 bezeichnet den 26. Satz im Eingabeelement

#0 bezeichnet den Platz vor dem ersten Elementsatz, d.h. ein Satz wird vor dem ersten Satz des Elementes eingefügt.

Kennung: Buchstaben, Ziffern und Sonderzeichen

Lage und Länge der Kennung (Inhalt des Kennungsfeldes) wird mit dem Verarbeitungsoperanden CHECK festgelegt. Deshalb darf diese Angabe für Satzbezeichnung nur bei CHECK ungleich NO verwendet werden. Die Kennung muß in der Länge, die bei CHECK vereinbart wurde, angegeben werden. Nur führende Nullen dürfen entfallen. Sind Zwischenräume enthalten, ist die Kennung durch die Zeichen > und < zu begrenzen.
Wenn die Kennung im Eingabeelement nicht vorkommt, wird korrigiert

- vor dem ersten Satz
- nach dem letzten Satz
- vor dem ersten Satz mit größerer Kennung.

*Beispiel*

| Kennungsfeld im Eingabeelement | Kennung |
|---|---|
| 000315 | 315 |
| xy z | >xy z< |

Um Sätze mit leerem Kennungsfeld in einem nicht numerierten Eingabeelement adressieren zu können, kann man auch Satznummer und -kennung gemischt in einem Korrekturlauf (sogar in einer Korrekturanweisung) verwenden.

Ist der Verarbeitungsoperand CHECK definiert, können Datensätze gemäß ihrem Kennungsfeld in das zu korrigierende Element aufgenommen werden (ohne Benutzung von *INSERT oder *REPLACE).

Gibt es einen Satz mit dem angegebenen Kennungsfeld, wird er durch den Datensatz ersetzt. Gibt es keinen Satz mit dem angegebenen Kennungsfeld, wird der Datensatz vor dem ersten Satz mit höherem Kennungsfeld eingefügt. Datensätze mit leerem Kennungsfeld, oder wenn CHECK=NO gesetzt ist, werden an aktueller Position eingeordnet.

Auf *INS oder *REP folgende Datensätze ohne Satzkennung müssen mit einem Satz der Form */*/ von evtl. folgenden Datensätzen mit Satzkennung abgegrenzt werden, wenn nicht unmittelbar danach *INS, *DEL, *CHA, *REP oder *END folgt.

**Beschreibung der Korrekturanweisungen**

*INSERT Einfügen von Sätzen

Über *INSERT werden Datensätze an gewünschten Stellen eingefügt. Die Sätze können entweder eingegeben oder von einem anderen Element, das als Sekundärelement bezeichnet wird, kopiert werden. Das zu korrigierende Element wird als Primärelement bezeichnet.

| Operation | Operanden |
|---|---|
| *INS[ERT] | satzbez[,[x=]elem[(lib)][:satzbez1[-satzbez2]]] |

*INSERT Name der Korrekturanweisung.

satzbez Nach dem Satz mit satzbez als Satznummer bzw. Kennung werden Datensätze eingefügt. Ist die angegebene Satznummer oder Kennung nicht vorhanden, werden vor dem ersten Satz, dessen Satznummer/Kennung größer ist als satzbez, Datensätze eingefügt. Eingefügt werden diejenigen Datensätze, die im Anschluß an *INSERT eingelesen werden.

x Elementtyp des Sekundärelementes
zugelassen sind: S, M, J, P, D, X

Wird der Typ nicht angegeben, wird der Typ des Primärelementes angenommen.

elem Name des Sekundärelementes.
Eine Auswahlangabe ist nicht erlaubt.

Aus dem angegebenen Sekundärelement werden die Datensätze von satzbez1 bis einschließlich satzbez2 in das Primärelement eingefügt.

lib Kurzbezeichnung der Bibliothek des Sekundärelementes. Die Angabe der Bibliothek kann entfallen, wenn das Sekundärelement auf der Eingabebibliothek des Primärelementes liegt.

satzbez1 Erste oder einzige Satznummer oder Kennung eines Satzes aus dem Sekundärelement, der ins Primärelement eingefügt werden soll.

satzbez2 Alle Sätze von satzbez1 bis einschließlich satzbez2 werden aus dem Sekundärelement ins Primärelement eingefügt. Fehlt die Angabe satzbez1 - satzbez2, wird das gesamte Sekundärelement eingefügt.

Werden aus einem Sekundärelement Datensätze in das Primärelement übernommen, können noch zusätzlich Datensätze nach *INSERT angegeben werden. Sie werden nach den Sätzen des Sekundärelementes in das Primärelement eingefügt.

*DELETE Löschen von Sätzen

*DELETE löscht angegebene Sätze oder Satzbereiche im Element.

| Operation | Operanden |
|---|---|
| *DEL[ETE] | satzbez1[-satzbez2] |

*DELETE Name der Korrekturanweisung.

satzbez1-satzbez2
Satzbereich. Alle Sätze mit den Adressen (Satznummer oder Kennung) von satzbez1 bis satzbez2 werden gelöscht. Soll nur ein Satz gelöscht werden, kann die Angabe -satzbez2 entfallen.

*REPLACE Ersetzen von Sätzen

Mit *REPLACE werden Datensätze oder Satzbereiche im Element durch angegebene Datensätze ersetzt.

| Operation | Operanden |
|---|---|
| *REP[LACE] | satzbez1[-satzbez2] |

*REPLACE Name der Korrekturanweisung.

satzbez1-satzbez2
Satzbereich. Alle Sätze mit den Adressen von satzbez1 bis einschließlich satzbez2 werden durch die auf diese Korrekturanweisung folgenden Datensätze ersetzt. Soll nur ein Satz ersetzt werden, kann die Angabe -satzbez2 entfallen.

Folgen *REPLACE mehr Dateisätze als durch satzbez1 - satzbez2 angegeben, werden sie trotzdem an der aktuellen Stelle eingefügt.

*CHANGE Ändern von Sätzen

Über *CHANGE werden Textstellen durch angegebenen Text ersetzt oder Text wird spaltengerecht eingefügt.

| Operation | Operanden |
|---|---|
| *CHA[NGE] | [satzbez1[-satzbez2]]'text1'[<spalte>][{:=\|=}'text2'] |

*CHANGE Name der Korrekturanweisung.

satzbez1-satzbez2
Satzbereich. In allen Sätzen, deren Adressen (Satznummer oder Kennungsfeld) größer oder gleich satzbez1 und kleiner oder gleich satzbez2 sind, wird jeder im angegebenen Spaltenbereich gefundene Prüftext durch den Korrekturtext ersetzt.

text1 Prüftext:
Die zu ersetzende Zeichenfolge ist eine beliebige Folge von Zeichen, die in Hochkommata ' ' oder Anführungszeichen " " eingeschlossen ist.

Dabei gilt für Anführungszeichen " ":

Nur wenn das im Satz unmittelbar links neben dem ersten Zeichen der Folge stehende und das im Satz unmittelbar rechts neben dem letzten Zeichen der Folge stehende Zeichen nicht alphanumerisch ist, gilt der Text als gefunden.

Wird als Prüftext die leere Zeichenfolge '' angegeben, wird der Korrekturtext beginnend in der durch spalte (nur ein Wert darf angegeben werden) definierten Position eingefügt. Bei Überschreitung der zulässigen Satzlänge (251 Zeichen) wird auf diese abgeschnitten. Falls dadurch Zeichen (außer Zwischenraum) verloren gehen, wird im Korrekturprotokoll eine entsprechende Warnmeldung ausgegeben.

spalte definiert einen Spaltenbereich spalte1 - spalte2 im Satz, innerhalb dessen der Prüftext beginnen muß. Ist nur spalte1 angegeben, muß der Prüftext in dieser Spalte im Satz beginnen.
Besteht der Prüftext aus der leeren Zeichenkette '', muß genau ein Spaltenwert angegeben werden. Dieser bezeichnet die Position, an der der Korrekturtext eingefügt wird.
Die Angabe des Spaltenbereichs darf nur dann weggelassen werden, wenn der Prüftext nicht aus der leeren Zeichenkette besteht. Der Prüftext wird dann im gesamten Satz gesucht.

| | |
|---|---|
| =:= | Prüftext und Korrekturtext müssen gleich lang sein. |
| = | Prüftext und Korrekturtext können unterschiedliche Länge haben. |
| text2 | Korrekturtext:<br>Eine beliebige Folge von Zeichen, die in Hochkomma oder Anführungszeichen eingeschlossen ist. Der angegebene Korrekturtext ersetzt den im Spaltenbereich gefundenen Prüftext oder wird (wenn Prüftext die leere Kette ist) an der Spaltenposition eingefügt. |

*CHANGE ohne Angabe eines Satzbereichs satzbez1 - satzbez2 muß stets am Anfang der Korrekturanweisungen für COR liegen.

Bei Angabe eines Satzbereichs muß satzbez1 größer oder gleich einem in einer vorangegangenen Korrekturanweisung (*REP, *DEL, *INS) angegebenen satzbez1 sein, satzbez2 dagegen kann beliebig gewählt werden.

*END Beenden der Korrektureingaben

*END beendet die Korrektureingaben für COR.

| Operation | Operanden |
|---|---|
| *END | |

*END Name der Korrekturanweisung

Diese Korrekturanweisung besitzt keine Operanden.

*Beispiel*

```
/SET-FILE-LINK FILE-NAME=SRC.LIB,LINK-NAME=LIB017
/START-PROGRAM $LMS
$LIB (17),BOTH
$CORS SRC82
**INSERT #3,SRC80:#3-#8
**DEL #21
**END
  .
  .
  .
$END
```

Das Element SRC82 der Bibliothek SRC.LIB wird korrigiert. Hinter dem Satz mit der Satznummer #3 werden die Sätze mit den Satznummern #3 bis #8 aus dem Element SRC80 eingefügt. Der Satz mit der Satzkennung #21 wird gelöscht.

# CTL Steuern der Anweisungseingabe

Mit CTL wird die Eingabequelle für LMS-Anweisungen definiert.

Die Eingabequellen sind:

- die Datensichtstation
- die Systemdatei SYSDTA
- ein Bibliothekselement

| Operation | Operanden |
|---|---|
| CTL | [(CMD) \| ([SYS]DTA) \| (RDR) \| elem[(lib)] \| ?] |

CTL — Name der Anweisung.

CMD — Anweisungen werden im Dialog von der Datensichtstation, im Stapelbetrieb von der Systemdatei SYSDTA gelesen.

SYSDTA — Anweisungen werden von der Systemdatei SYSDTA gelesen.

RDR — wirkt wie der Operand SYSDTA und wird nur noch aus Kompatibilitätsgründen unterstützt.

elem — bezeichnet das Element, aus dem die Anweisungen gelesen werden. Nur ein Element darf angegeben werden, eine Auswahlangabe ist nicht erlaubt.

Das Element muß bei Programmbibliotheken vom Typ J, bei Quellprogrammbibliotheken vom Typ S sein.
Eine Quellprogrammbibliothek darf dann in diesem LMS-Lauf nicht anderweitig verwendet werden.

lib — Kurzbezeichnung der Bibliothek.

? — Der aktuelle Wert wird protokolliert.

*Beispiel*

```
/START-PROGRAM $LMS
$LIB ANWS,BOTH
$CTL BFL
$END
```

Die Anweisungen aus dem Element BFL werden ausgeführt. Die letzte Anweisung in diesem Element ist CTL (CMD). Anschließend werden die Anweisungen, in diesem Fall END, wieder von der Datensichtstation gelesen.

# DEL Löschen von Elementen

DEL löscht die angegebenen Elemente der zugewiesenen Eingabebibliothek. Dabei werden die Inhaltsverzeichniseinträge gelöscht und der Speicherplatz freigegeben.

In Programmbibliotheken werden Elemente zusätzlich physikalisch gelöscht, d.h. die Daten mit binären Nullen überschrieben, wenn

- das Element ein Kennzeichen für physikalisches Löschen enthält.
- der Verarbeitungsoperand DESTROY=YES gesetzt ist.

Für sequentielle Bibliotheken ist DEL nicht erlaubt.

| Operation | Operanden |
|---|---|
| DEL[x] | {elem[(lib)] / (lib)}[,...] |

DELx Name der Anweisung mit Angabe des Elementtyps. Es sind alle Elementtypen erlaubt: S, M, R, J, P, C, D, X, H, L, F, U

* Stellvertretend für alle Elementtypen (nur bei Programmbibliotheken).

elem Elementbezeichnung oder Auswahlangabe.

lib Kurzbezeichnung der Eingabebibliothek.
Ist kein Element angegeben, gilt die Funktion für alle Elemente der jeweils höchsten Version der angesprochenen Bibliothek, d.h. die gesamte Bibliothek wird gelöscht.

**Verarbeitungsoperanden**

TYPE bestimmt den Elementtyp, falls bei der Anweisung selbst kein Typ angegeben wird.

DESTROY bestimmt, ob die Elemente physikalisch gelöscht werden sollen.
(Nur bei Programmbibliotheken möglich.)

REFERENCE bestimmt, welche Referenznamen die zu löschenden Elemente enthalten müssen. Ist die Referenzbedingung nicht erfüllt, wird das Element nicht gelöscht. REFERENCE ist nur beim Elementtyp R erlaubt. Nach der Vereinbarung mit REFERENCE muß R bei DEL angegeben werden.

*Hinweis*

Bei jedem UPDATE auf einen Delta-Baum wird die Delta-Struktur reorganisiert, d.h. nicht mehr benötigte Datensätze werden gelöscht und der nicht mehr benötigte Speicher freigegeben.

*Beispiel*

```
/START-PROGRAM $LMS
$LIB LS.LIB,BOTH
$DELD DATA
$DEL* TPROG
$END
```

Das Element DATA der Bibliothek LS.LIB wird gelöscht. Alle Elemente des Namens TPROG der Bibliothek LS.LIB werden, unabhängig von ihrem Elementtyp, gelöscht.

# DUP
# Duplizieren von Elementen und strukturerhaltendes Duplizieren

Mit DUP können Elemente oder ganze Delta-Bäume strukturerhaltend dupliziert werden. Dafür hat DUP 2 Formate:

### Format 1

Duplizieren von Elementen

| Operation | Operanden |
|---|---|
| DUP[x] | [elem,...[(lib)]] / (lib) [>elemu][,(...)[>...]] [,BASEVERSION=[*NONE / version / *HIGH]] |

### Format 2

Strukturerhaltendes Duplizieren

| Operation | Operanden |
|---|---|
| DUP[x] | name1[(lib)][>name2],STRUC=Y[ES] |

### Format 1: Duplizieren von Elementen

DUP dupliziert die angegebenen Elemente der zugewiesenen Eingabebibliothek oder eine ganze Bibliothek in die eröffnete Ausgabebibliothek. Dabei können die duplizierten Elemente neue Elementbezeichnungen erhalten.

Wenn die Ausgabebibliothek eine Programmbibliothek ist, können die duplizierten Elemente entweder als Voll-Element oder als Delta-Element gespeichert werden. Falls beim Duplizieren von Delta-Elementen die Eingabebibliothek mit der Ausgabebibliothek identisch ist, müssen die duplizierten Delta-Elemente neue Elementnamen erhalten.

Eine Ausgabebibliothek muß vorher mit LIB zugewiesen werden.

Bestimmte Satzarten können bei Elementen vom Typ R und C mit Hilfe des Verarbeitungsoperanden STRIP vom Duplizieren ausgeschlossen werden.

Elemente können mit Hilfe ihrer Referenznamen zum Duplizieren ausgewählt werden, d.h. wird mit dem Verarbeitungsoperanden REFERENCE eine Referenzbedingung festgelegt, werden nur Elemente dupliziert, die diese Bedingung erfüllen.

| Operation | Operanden |
|---|---|
| DUP[x] | {elem,...[(lib)] / (lib)} [>elemu][,{...}[>...]] [,BASEVERSION={*NONE / version / *HIGH}] |

DUPx Name der Anweisung mit Angabe des Elementtyps.

- Für Voll-Elemente sind alle Elementtypen zulässig:
  S, M, R, J, P, C, D, X, H, L, F, U
- Für Delta-Elemente sind nur die textartigen Elementtypen zulässig:
  S, M, J, P, D
- * Stellvertretend für alle Elementtypen (nur zulässig bei Programmbibliotheken).

Der Elementtyp braucht nicht angegeben zu werden, wenn im Verarbeitungsoperanden TYPE der entsprechende Elementtyp vereinbart ist.

elem Elementbezeichnung des zu duplizierenden Elementes oder Auswahlangabe.

Für elem sind auch Namen zugelassen, die nicht den LMS-Konventionen genügen, um die weitere Bearbeitung solcher Elemente zu ermöglichen.

lib Kurzbezeichnung der Eingabebibliothek.

elemu Elementbezeichnung des Ausgabeelementes oder Konstruktionsangabe.

BASEVERSION

bestimmt das Basis-Element bei der Delta-Speicherung. Er bezieht sich auf alle zu erzeugenden Elemente (elemu). Er muß der letzte Operand der Anweisung sein.
Fehlt dieser Operand wird aus einem Delta-Element ein Voll-Element erstellt.

=*NONE elemu wird als erstes Element (=Basis) eines Delta-Baums abgelegt (Ersterstellung).

=version Basis-Element ist das Element mit version als Versionsangabe; dieses Element muß existieren und als Delta-Element gespeichert sein.
Eine Konstruktionsangabe ist nicht erlaubt.

=*HIGH Basis-Element ist das Element mit der höchsten Versionsbezeichnung (zum Erstellzeitpunkt des Delta-Elementes). Das erzeugte Delta-Element wird an dieses Basis-Element angehängt.

Mit DUPx */* werden alle Elemente eines Typs unter Beibehaltung der Elementbezeichnung dupliziert.

**Verarbeitungsoperanden**

TYPE bestimmt den Elementtyp, falls bei der Anweisung selbst kein Typ angegeben wird.

OVERWRITE steuert das Überschreiben von gleichnamigen Voll-Elementen in der Ausgabebibliothek.
Dieser Verarbeitungsoperand ist für Delta-Elemente unzulässig.

STRIP nur für Elemente vom Typ R und C.
bestimmt, welche Satzarten beim Duplizieren ausgeschlossen werden sollen.

DESTROY bestimmt, ob in Ausgabeelementen ein Kennzeichen für physikalisches Löschen eingetragen wird.
(Nur bei Programmbibliotheken möglich.)

REFERENCE bestimmt, welche Referenznamen die zu duplizierenden Elemente enthalten müssen. Ist die Referenzbedingung nicht erfüllt, wird das Element nicht dupliziert. REFERENCE ist nur beim Elementtyp R erlaubt. Nach der Vereinbarung mit REFERENCE muß R bei DUP angegeben werden.

*Beispiel*

```
/SET-FILE-LINK FILE-NAME=DUP.LIB,LINK-NAME=LIB002
/START-PROGRAM $LMS
$LIB OLD.LIB,IN
$LIB (2),NEW,OUT
$DUPS OLD1>DUP1/001/1983-02-07,OLD2>DUP2
$END
```

Elemente der Bibliothek OLD.LIB werden in eine zu erstellende Programmbibliothek DUP.LIB dupliziert. Der zu erstellenden Ausgabebibliothek DUP.LIB wird der Dateikettungsname LIB002 zugewiesen. Die Eingabebibliothek wird mit LIB explizit zugewiesen. Das Element OLD1 wird als DUP1, Version 1 mit neuem Datum dupliziert. OLD2 wird als DUP2 mit unveränderter Version und unverändertem Datum dupliziert.

### Format 2: Strukturerhaltendes Duplizieren

LMS erkennt bei diesem Format die Form der Elementabspeicherung in den PLAM-Bibliotheken. Dementsprechend werden Delta-Bäume als Delta-Bäume und alle andere Elemente als Vollelemente in die Ausgabedatei dupliziert.

| Operation | Operanden |
|---|---|
| DUP[x] | name1[(lib)][>name2],STRUC=Y[ES] |

DUPx — Name der Anweisung mit Angabe des Elementtyps. Es sind alle Elementtypen erlaubt: S, M, R, J, P, C, D, X, H, L, F, U

* Stellvertretend für alle Elementtypen (nur zulässig bei Programmbibliotheken).

name1 — Name (ohne Version und Datum) des Eingabeelementes oder Auswahlangabe. Es darf nur ein Eingabeelement angegeben werden.

lib — Kurzbezeichnung der Eingabebibliothek.

name2 — Name (ohne Version und Datum) des Ausgabeelementes oder Konstruktionsangabe.

STRUC=YES — Wird mit name1 der Namensraum eines Delta-Baumes angegeben, wird der Delta-Baum strukturerhaltend dupliziert. name2 darf in der Ausgabebibliothek noch nicht existieren (der Verarbeitungsoperand OVERWRITE wirkt nicht).

Wird mit name1 der Namensraum eines oder mehrerer Voll-Elemente angegeben, werden alle Voll-Elemente mit dem Namen name1 (d.h. wie name1/*) als Voll-Elemente dupliziert. Sind name1 und name2 gleich, wirkt der Verarbeitungsoperand OVERWRITE.

*Hinweis*

- Wird der Dupliziervorgang vorzeitig abgebrochen, bleibt der schon kopierte Teil eines Delta-Baumes erhalten.
- Angaben zu Version und Datum dürfen bei name1 und name2 nicht gemacht werden.

**Verarbeitungsoperanden**

| | |
|---|---|
| TYPE | bestimmt den Elementtyp, falls bei der Anweisung selbst kein Typ angegeben wird. |
| DESTROY | bestimmt, ob in Ausgabeelementen ein Kennzeichen für physikalisches Löschen eingetragen wird.<br>(Nur bei Programmbibliotheken möglich.) |

# EDT/EDR
# Erstellen, Korrigieren und Anschauen von Textelementen und Dateien

EDT/EDR ruft die Dateibearbeitungsprogramme EDT bzw. EDOR auf, um entweder Textelemente oder Dateien zu erstellen, zu korrigieren oder anzuschauen. Dafür hat EDT/EDR drei verschiedene Formate.

### Format 1

Erstellen und Korrigieren von Textelementen

| Operation | Operanden |
|---|---|
| EDT[x] | elem[(lib)][>elemu] |
| EDR[x] | elem[(lib)][>elemu] |

### Format 2

Erstellen und Korrigieren von Dateien

| Operation | Operanden |
|---|---|
| EDT | |
| EDR | |

### Format 3

Anschauen von Elementen

| Operation | Operanden |
|---|---|
| EDT[x] | elemu[(lib)]>*DUMMY |
| EDR[x] | elemu[(lib)]>*DUMMY |

*Hinweis*

In der Bibliothek gespeicherte PAM-Dateien sind mit EDT/EDR nicht bearbeitbar.

**Format 1: Erstellen und Korrigieren von Textelementen**

Dieses Format von EDT/EDR ruft den EDT bzw. EDOR auf und liest das angegebene Element aus der zugewiesenen Eingabebibliothek ein. Wenn EDT bzw. EDOR beendet werden, wird das korrigierte Element mit eventuell neuem Namen in die zugewiesene Ausgabebibliothek geschrieben.

Editoren, siehe EDT [5] und EDOR [4].

LMS unterstützt EDT-Versionen größer V16.2A.

Ein- und Ausgabebibliotheken dürfen bei sequentiellen Bibliotheken nicht identisch sein. Bei den anderen Bibliothekstypen dürfen Ein- und Ausgabebibliotheken identisch sein.

| Operation | Operanden |
|---|---|
| EDT[x] | elem[(lib)][>elemu] |
| EDR[x] | elem[(lib)][>elemu] |

EDTx]
EDRx] Name der Anweisung mit der Angabe des Elementtyps: S, M, J, P, D, X

Der Elementtyp braucht nicht angegeben zu werden, wenn im Verarbeitungsoperanden TYPE der entsprechende Elementtyp vereinbart ist.

elem Elementbezeichnung des zu korrigierenden bzw. des zu erzeugenden Elementes oder Auswahlangabe.

lib Kurzbezeichnung der Eingabebibliothek.

elemu Elementbezeichnung des Ausgabeelementes oder Konstruktionsangabe.

**Verarbeitungsoperanden**

TYPE bestimmt den Elementtyp, falls bei der Anweisung selbst kein Typ angegeben wird.

OVERWRITE bestimmt das Überschreiben gleichnamiger Elemente in der Ausgabebibliothek.

Dieser Verarbeitungsoperand wird jedoch nicht ausgewertet, wenn

```
Eingabebibliothek=Ausgabebibliothek und
                  elem=elemu
```

Das Eingabeelement wird dann überschrieben.

| | |
|---|---|
| DESTROY | bestimmt, ob im Ausgabeelement ein Kennzeichen für physikalisches Löschen eingetragen und die eventuell erstellte Hilfsdatei nach Gebrauch physikalisch gelöscht wird. (Nur bei Programmbibliotheken möglich.) |
| CHECK | definiert Lage und Länge des Kennungsfeldes in Eingabesätzen und veranlaßt Überprüfung auf aufsteigende Numerierung. |
| RANGE | definiert Lage und Länge des Kennungsfeldes in den Ausgabesätzen. |
| STRING | bestimmt, ob der ISAM-Schlüssel im Kennungsfeld abgelegt werden soll, bzw. gibt die Zeichenfolge an, die linksbündig in das Kennungsfeld der Ausgabesätze eingetragen wird. |
| VALUE | gibt Anfangswert und Schrittweite für die Numerierung im Kennungsfeld an. |

**Hilfsdatei**

LMS erzeugt beim Aufruf des EDT unter bestimmten Voraussetzungen und beim Aufruf des EDOR immer die Hilfsdatei:

S.LMS.TSNnnnn.datum.uhrzeit.elem

elem hat eine maximale Länge von 9 Zeichen. Längere Elemente werden bis auf die ersten 9 Zeichen abgekürzt. Ergeben sich durch das Anhängen des Elementnamens unzulässige BS2000-Dateinamen (z.B. das 9. Zeichen ist ein '.'), bildet LMS den Hilfsnamen ohne Elementnamen:

S.LMS.TSNnnnn.datum.uhrzeit

Ist im Eingabeelement RECORD-FORMAT=FIXED gespeichert, wird die Funktion mit Fehler abgebrochen, dasselbe gilt für KEY-POSITION>5.

Sind im Eingabeelement ISAM-Schlüssel gespeichert, werden diese in die Hilfsdatei und nach dem Korrigieren in das Ausgabeelement übernommen.

**Übernahme der ISAM-Schlüssel aus dem Kennungsfeld**

Mit dem Verarbeitungsoperanden CHECK=anf/lge ist ein Kennungsfeld in den Eingabesätzen zu vereinbaren. Der Inhalt dieses Kennungsfeldes wird linksbündig im ISAM-Schlüssel des Dateisatzes abgelegt. Haben Kennungsfeld (lge) und ISAM-Schlüsselfeld (KEY-LENGTH) unterschiedliche Länge, richtet LMS aus:

KEY-LENGTH<lge: das Kennungsfeld wird rechts abgeschnitten.
KEY-LENGTH>lge: das ISAM-Schlüsselfeld wird rechts mit Nullen aufgefüllt.

Sind im Eingabeelement keine ISAM-Schlüssel gespeichert, wird eine ISAM-Datei mit KEY-POSITION=5 und KEY-LENGTH=8 erstellt. Standardmäßig erzeugt LMS dann ISAM-Schlüssel mit einem Anfangswert von 1000 und einer Schrittweite von 1000. Ist das Element zu groß für diese Schrittweite (mehr als 100000 Sätze), so wird die Schrittweite aus der Anzahl der Sätze berechnet.

Das Übertragen erfolgt durch einen internen SEL-Aufruf. Der interne SEL-Aufruf ermöglicht hier die gleichen Funktionen, die mit SEL bezüglich der ISAM-Schlüssel geboten werden (siehe Verarbeitungsoperanden CHECK, STRING, VALUE, RANGE).

**Ablauf des Editierens:**

- EDTx elementname:
  Die Elementsätze werden von LMS an den EDT weitergereicht. Das Element steht dann im virtuellen Speicher zur Verfügung. Die Zeilennummernanzeige des EDT zeigt die ersten sechs Stellen der ISAM-Schlüssel.

  Eine Hilfsdatei mit linknamen=EDTISAM wird angelegt, wenn entweder die Vearbeitungsoperanden STRING und CHECK verwendet werden oder die gespeicherten ISAM-Schlüssel eine KEY-POSITION <5 oder KEY-LENGTH <8 haben.
- Wurde der Dateikettungsname EDRPRIMR außerhalb von LMS einer Datei zugeordnet und besteht diese Zuordnung noch, wenn der Benutzer mit EDRx ein Element bearbeiten will, muß er zuerst mit /REMOVE-FILE-LINK LINK-NAME=EDRPRIMR die Zuordnung löschen, da sonst die über EDRPRIMR zugewiesene Datei bearbeitet wird. Beim Beenden des EDOR wird jedoch nur die Hilfsdatei mit den Elementdaten zurückgeschrieben.
- Alte Bibliotheken (OSM) werden geschlossen. Andere Benutzer können auf die Bibliothek und auf das Element zugreifen.
  Programmbibliotheken werden nicht geschlossen, das bearbeitete Element ist für andere Benutzer gesperrt.
- Der entsprechende Editor wird nun als Unterprogramm aufgerufen. Falls eine Hilfsdatei erstellt wurde, wird sie eröffnet und kann bearbeitet werden.
- Beim Aufruf des Editors wird der LMS-STXIT abgemeldet und nach Rückkehr aus dem Editor wieder angemeldet.

– Beenden der Editoren:

EDT:

RETURN aus Arbeitsdatei 0:
Die aktuelle Arbeitsdatei wird als Element in die Ausgabebibliothek aufgenommen. Die EDT-Daten (Dateien im virtuellen Speicher, Variablen,...) bleiben erhalten. Nur bei einem schwerwiegenden EDT-Fehler werden diese Daten freigegeben.

HALT aus Arbeitsdatei 0:
Es erfolgt die Abfrage:

```
LMS0420: EDITED ELEMENT
(Typ)Elementname/Version[(Variantennummer)]/Datum TO BE ADDED?
REPLY (Y=YES, N=NO)?
```

Entsprechend der Antwort wird die aktuelle Arbeitsdatei als Element aufgenommen oder nicht. Die EDT-Daten bleiben erhalten. Nur bei einem schwerwiegenden EDT-Fehler werden diese Daten freigegeben.

HALT/RETURN aus Arbeitsdatei ≠ 0:
Es erfolgt die Abfrage:

```
LMS0420: EDITED ELEMENT
(Typ)Elementname/Version[(Variantennummer)]/Datum TO BE ADDED?
REPLY (Y=YES, N=NO)?
```

Wird 'N' eingegeben, wird das Element nicht aufgenommen. Bei der Antwort 'Y' wird als nächstes ein Dialog mit dem Anwender geführt:

```
LMS0421: WORKFILE TO BE ADDED (0 = WORKFILE(0),.., N = NONE)
```

Wird hier die Antwort 'N' gegeben, kehrt LMS in die gerade bearbeitete Arbeitsdatei des EDT zurück.

EDT im Stapelbetrieb:

@RETURN:
Das korrigierte Element wird aus der Arbeitsdatei 0 in die Ausgabebibliothek aufgenommen, wenn diese nicht leer ist.

@HALT:
Das korrigierte Element wird nicht in die Ausgabebibliothek aufgenommen.

EDOR:

H!H: Das korrigierte Element wird in die Ausgabebibliothek übernommen. Wurde eine Hilfsdatei erzeugt, erfolgt das Zurückschreiben mit einem internen ADD-Aufruf. Die bei ADD möglichen Verarbeitungsoperanden werden ausgewertet (z.B. die Ablage des ISAM-Schlüssels im Element über die Verarbeitungsoperanden RANGE und STRING).

**Aufnehmen von bearbeiteten Elementen in die Ausgabebibliothek**

Das bearbeitete Element (die bearbeitete Hilfsdatei) wird nur dann in die Ausgabedatei übernommen, wenn LMS nicht im TEST-Modus ist. Abhängig vom Dateibearbeitungsprogramm gilt folgendes zusätzlich:

- Das durch EDOR bearbeitete Element wird nur dann in die Ausgabebibliothek übernommen,
  - wenn die Hilfsdatei nicht leer ist,
  - wenn der EDOR mit H!H beendet wird.

- Das durch EDT bearbeitete Element wird nur dann in die Ausgabebibliothek übernommen,
  - wenn der EDT mit RETURN aus Arbeitsdatei 0 beendet wird und die Arbeitsdatei nicht leer ist,
  - wenn der EDT mit HALT/RETURN aus Arbeitsdatei ≠ 0 beendet wird und die darauf folgende Meldung "EDITED ELEMENT ..." mit "Y" beantwortet wird.
  - wenn im Stapelbetrieb der EDT mit RETURN beendet wird und die Arbeitsdatei 0 nicht leer ist.

### Format 2: Erstellen und Korrigieren von Dateien

Dieses Format von EDT/EDR ruft den EDT/EDOR nur auf. Bearbeitung von EDT/EDOR-Dateien, siehe EDT [5] und EDOR [4]. Wenn der EDT/EDOR beendet wird, wird der unterbrochene LMS-Lauf wieder aufgenommen. Die EDT-Daten (Dateien im virtuellen Speicher, Variablen,...) bleiben erhalten.

| Operation | Operanden |
|---|---|
| EDT | |
| EDR | |

### Format 3: Anschauen von Elementen

Dieses Format ermöglicht dem Benutzer, sich Elemente anzuschauen ohne eine Ausgabebibliothek zuzuweisen.

| Operation | Operanden |
|---|---|
| EDT[x] | elemu[(lib)]>*DUMMY |
| EDR[x] | elemu[(lib)]>*DUMMY |

elemu — Elementbezeichnung des Eingabeelementes. Eine Auswahlangabe ist in eingeschränkter Form (keine Listeneingabe) für Programmbibliotheken zulässig.

lib — Kurzbezeichnung der Eingabebibliothek.

*DUMMY — Das für den Editor bereitgestellte Element wird nicht zurückgeschrieben; zusätzlich gilt:
- eine Ausgabebibliothek muß nicht zugewiesen werden,
- Versions- und Datumsangabe werden syntaktisch geprüft,
- elemu muß existieren,
- eine evtl. angelegte Hilfsdatei wird immer gelöscht.

# END Beenden des LMS-Laufs

Mit END wird das Programm LMS beendet. Alle noch geöffneten Bibliotheken werden geschlossen.

Im Dialogbetrieb wird LMS immer normal beendet, im Stapelbetrieb bzw. in Prozeduren hängt die Beendigungsart vom Abbruchkennzeichen ab (siehe Verarbeitungsoperand TERMINATE). Wird ein LMS-internes Abbruchkennzeichen gesetzt, wird dieses bei der LMS-Ende-Meldung (LMS-TERM-MSG:) mitausgegeben.

| Operation | Operanden |
|---|---|
| END | |

Der Beendigungscode zeigt den schwerwiegendsten Fehler an, der aufgetreten ist. Dieser wird in der Monitor-Jobvariablen abgespeichert, die beim Aufruf

```
/START-PROGRAM LMS,MONJV=<name>
```

angegeben wurde.

| Beendigungscode | Bedeutung |
|---|---|
| 0 | kein Fehler aufgetreten |
| 1 | Warnungen wurden ausgegeben |
| 2 | Fehler mit Abbruch-Kennzeichen aufgetreten (siehe Verarbeitungsoperand TERMINATE) |
| 3 | interner LMS-Fehler (mit Speicherabzug) |

Die Zustandsanzeige der Jobvariablen für die Programmüberwachung enthält nach Beendigung des LMS im 1. Byte den Beendigungscode (siehe oben) und im 4. Byte das interne Abbruchkennzeichen (siehe PAR TERM).

# LIB Zuweisen und Schließen von Bibliotheken

Mit LIB können Ein- und Ausgabebibliotheken eingerichtet, eröffnet und auch wieder geschlossen werden. Aus der Eingabebibliothek liest LMS Elemente, in die Ausgabebibliothek gibt LMS Elemente aus.

Ausnahmen bilden jedoch DEL und NAM. Sie wirken auf die zugewiesene Eingabebibliothek.

Es gibt 3 Formate von LIB:

**Format 1**: Zuweisen von Bibliotheken

| Operation | Operanden |
|---|---|
| LIB | {FILE=bibliotheksname / LINK=dateikettungsname / [LIBRARY=]name / [LID=](lib)} [,[USAGE=]{IN / OUT / BOTH}] [,[FORMAT=]{PL / OML / OSM}] [,[STATE=]{O[LD] / N[EW] / A[NY]}] |

**Format 2**: Schließen von Bibliotheken

| Operation | Operanden |
|---|---|
| LIB | C[LOSE][,{FILE=bibliotheksname / LINK=dateikettungsname / [LIBRARY=]name / [LID=](lib)}] |

**Format 3**: Anzeigen der zugewiesenen Bibliotheken

| Operation | Operanden |
|---|---|
| LIB | ? |

**Format 1: Zuweisen von Bibliotheken**

LIB weist Bibliotheken zu. Dabei können folgende Angaben gemacht werden:

- Die Bibliothek wird als Ein- oder Ausgabebibliothek oder beides definiert.
- Der Bibliothekstyp wird definiert (Programm-, Quellprogramm-, Makro- oder Bindemodulbibliothek).
- Es wird festgelegt, ob die Bibliothek neu einzurichten ist, ob sie bereits existieren muß oder ob sie bei Bedarf neu eingerichtet werden soll.

LIB schließt durch Zuweisen einer neuen Eingabebibliothek die vorher mit LIB zugewiesene Eingabebibliothek. Das gleiche gilt für Ausgabebibliotheken.

Die neu zugewiesene Bibliothek wird eröffnet.

| Operation | Operanden |
|---|---|
| LIB | {FILE=bibliotheksname / LINK=dateikettungsname / [LIBRARY=]name / [LID=](lib)} [,[USAGE=]{IN / OUT / BOTH}] [,[FORMAT=]{PL / OML / OSM}] [,[STATE=]{O[LD] / N[EW] / A[NY]}] |

LIB Name der Anweisung.

FILE=bibliotheksname
vollqualifizierter Dateiname der Bibliothek.
Für bibliotheksname kann auch pfadname (siehe Benutzer-Kommandos [7]) angegeben werden.

LINK=dateikettungsname
gibt den Dateikettungsnamen an, mit dem die Bibliothek in einem /SET-FILE-LINK-Kommando zugewiesen wurde.

LIBRARY
=name LMS versucht zuerst name als Dateikettungsnamen zu interpretieren. Falls kein derartiger Dateikettungsname zuvor mit einem /SET-FILE-LINK-Kommando zugewiesen wurde, wird name als bibliotheksname interpretiert.

Das Schlüsselwort 'LIBRARY=' kann weggelassen werden. In diesem Fall darf der Bibliotheksname nicht 'CLOSE' sein, da sonst LIB, Format 2 angenommen wird.

LID

=(lib) Gibt die Bibliothekskurzbezeichnung, bestehend aus maximal 3 Ziffern, an.
Die Bibliothekskurzbezeichnung muß zuvor mit einem /SET-FILE-LINK-Kommando mit dem Dateikettungsnamen LIBlib vereinbart worden sein.

USAGE Dieser Operand legt fest, ob die Bibliothek zur Eingabe, zur Ausgabe oder zu beidem verwendet werden soll.

=IN Die Bibliothek ist im LMS-Lauf die Eingabebibliothek.
Standardwert bei STATE=OLD.

=OUT Die Bibliothek ist im LMS-Lauf die Ausgabebibliothek.
Standardwert bei STATE=NEW und STATE=ANY.

=BOTH Die Bibliothek ist im LMS-Lauf die Ein- und Ausgabebibliothek.

FORMAT Dieser Operand legt den Bibliothekstyp der zuzuweisenden Bibliothek fest.

Der Operand ist nur dann notwendig, wenn Bibliotheken neu angelegt werden sollen und der Standardwert PL nicht erwünscht ist. Den Bibliothekstyp existierender Bibliotheken erkennt LMS.

=PL Programmbibliothek (Program Library)

=OML Bindemodulbibliothek (Object Modul Library)

=OSM Quellprogramm- und Makrobibliotheken (Old Source/Macro Library)

STATE bestimmt, ob die Bibliothek neu eingerichtet werden soll, ob sie bereits existieren muß oder ob sie bei Bedarf neu eingerichtet werden soll.

=O[LD] Die Bibliothek muß bereits existieren.

=N[EW] Die Bibliothek soll neu eingerichtet werden.
Ist die Bibliothek bereits eingerichtet, wird die Anweisung mit einer Fehlermeldung abgewiesen.

=A[NY] Die Bibliothek wird neu eingerichtet, falls sie noch nicht existiert.

Bei DEL, NAM, NUM, COR, UPD, EDT, EDR, DUP, COM, LST und TOC kann durch die Bibliothekskurzbezeichnung eine andere als die standardmäßige Eingabebibliothek zugewiesen werden. Diese Eingabebibliothek ist nur während der Ausführung der entsprechenden Anweisung gültig, dann wird wieder die standardmäßige Eingabebibliothek zugewiesen.

*Beispiel*

```
/SET-FILE-LINK FILE-NAME=LMS.BEI,LINK-NAME=BEILINK
/SET-FILE-LINK FILE-NAME=LMS.EINB,LINK-NAME=LIB001
/START-PROGRAM $LMS
$LIB FILE=LMS.AUS,OUT,NEW
$LIB LINK=BEILINK,IN
  .
  .
  .
$LIB LID=(001),USAGE=IN
  .
  .
  .
$END
```

Die Bibliothek LMS.BEI wird mit einem /SET-FILE-LINK-Kommando zugewiesen und über LIB mit dem Dateikettungsnamen BEILINK und dem Operandenwert IN als Eingabebibliothek mit LMS verknüpft.

Die neu einzurichtende Ausgabebibliothek (OUT,NEW) LMS.AUS wird mit dem Dateinamen bei LIB spezifiziert.

Bei Eingabe des dritten LIB, die über Bibliothekskurzbezeichnung die Bibliothek LMS.EINB als Eingabebibliothek zuweist, wird die Bibliothek LMS.BEI geschlossen.

### Zuweisen sequentieller Bibliotheken

Sequentielle Bibliotheken können mit LIB nur als Eingabebibliotheken zugewiesen werden. Zur Erstellung von sequentiellen Bibliotheken muß LIBOUT (lib),NEWLIB (siehe Seite 345) verwendet werden.

Bereits bestehende sequentielle Ausgabebibliotheken werden mit LIBOUT (lib) zugewiesen.

### Format 2: Schließen von Bibliotheken

Mit diesem Format von LIB werden Bibliotheken geschlossen. Die Zuordnung der Ein- und Ausgabebibliothek geht verloren.

| Operation | Operanden |
|---|---|
| LIB | C[LOSE][,{FILE=bibliotheksname / LINK=dateikettungsname / [LIBRARY=]name / [LID=](lib)}] |

LIB Name der Anweisung.

C[LOSE] bestimmt das Schließen der Bibliothek(en).

FILE=bibliotheksname
gibt den vollqualifizierten Dateinamen der zu schließenden Bibliothek an.
Für bibliotheksname kann auch pfadname (siehe Benutzer-Kommandos [7]) angegeben werden.

LINK=dateikettungsname
gibt den Dateikettungsnamen der zu schließenden Bibliothek an, die mit einem /SET-FILE-LINK-Kommando zugewiesen wurde und LMS bekannt sein muß.

LIBRARY
=name LMS versucht zuerst name als den zugewiesenen Dateikettungsnamen der zu schließenden Bibliothek zu interpretieren.
Falls kein derartiger Dateikettungsname gefunden wird, wird name als bibliotheksname interpretiert. Der Dateikettungsname oder der Bibliotheksname müssen LMS bekannt sein.

LID
=(lib) gibt die Kurzbezeichnung der zu schließenden Bibliothek an.
Die Bibliothek muß mit einem /SET-FILE-LINK-Kommando mit dem Dateikettungsnamen LIBlib zugewiesen worden sein und LMS bekannt sein.

*Beipiel*

```
/START-PROGRAM $LMS
$LIB LMS.TEST,BOTH
  .
  .
  .
$LIB C,LMS.TEST
  .
  .
  .
$END
```

Die Bibliothek LMS.TEST wird mit dem ersten LIB als Ein- und Ausgabebibliothek zugewiesen und mit dem zweiten LIB geschlossen.

### Format 3: Anzeigen der zugewiesenen Bibliotheken

Dieses Format von LIB gibt Auskunft über die im LMS-Lauf verwendeten Bibliotheken.

Folgende Informationen werden ausgegeben:

- Verwendung der Bibliothek (Ein- oder Ausgabebibliothek oder beides)
- Zustand der Bibliothek (geöffnet oder geschlossen)
- Bibliotheksformat
- evtl. zugeordnete Kurzbezeichnung
- evtl. zugeordneter Dateikettungsname
- Dateiname der Bibliotheken

| Operation | Operanden |
|---|---|
| LIB | ? |

# LST Auflisten von Elementen

LST listet die angegebenen Elemente der zugewiesenen Eingabebibliothek auf.

Das Ausgabemedium ist je nach Wert von PRT

- die Datensichtstation und/oder
- die Systemdatei SYSLST oder
- ein Bibliothekselement.

Umfang und Gestaltung der Auflistung steuern die Verarbeitungsoperanden CSECT, FORMAT, LINE, PATH, SLICE, INFO und NEWFORM.

Für das Auflisten von Elementen vom Typ S, M, J, D, P, X kann der Benutzer auch in Benutzerroutinen verzweigen (siehe Anweisung USE).

| Operation | Operanden |
|---|---|
| LST[x] | $\left\{\begin{matrix}[elem[(lib)]] \\ [(lib)]\end{matrix}\right\}[,...]$ |

LSTx Name der Anweisung mit Angabe des Elementtyps. Es sind alle Elementtypen erlaubt: S, M, R, J, P, C, D, X, H, L, F, U

* Stellvertretend für alle Elementtypen (nur zulässig bei Programmbibliotheken).

Der Elementtyp braucht nicht angegeben zu werden, wenn im Verarbeitungsoperanden TYPE der entsprechende Elementtyp vereinbart ist.

elem Elementbezeichnung des aufzulistenden Elementes oder Auswahlangabe.

lib Kurzbezeichnung der Eingabebibliothek.

Da Listenelemente (Typ P in Programmbibliotheken und Typ S in Quellprogrammbibliotheken) nur in die Systemdatei SYSLST ausgegeben werden können, muß zusätzlich PRT (LST) angegeben werden. Der Verarbeitungsoperand LINE wird dabei ignoriert. (Bei Listenelementen in Quellprogrammbibliotheken ist außerdem der Verarbeitungsoperand PAR FORMAT=P anzugeben.)

LMS-Fehlermeldungen werden immer nach SYSOUT ausgegeben, auch wenn PRT (LST) gesetzt ist.

**Verarbeitungsoperanden**

| | |
|---|---|
| CSECT | bestimmt bei Typ L, welcher Programmabschnitt aufgelistet werden soll. |
| TYPE | bestimmt den Elementtyp, falls bei der Anweisung selbst kein Typ angegeben wird. |
| FORMAT | bestimmt, ob die Sätze zeichenweise, sedezimal oder gemischt ausgegeben werden. Näheres zur Ausgabe siehe Systemkonventionen [12]. |
| INFO | bestimmt, ob alle Sätze, nur bestimmte Satzarten, bestimmte Bereiche oder nur die wichtigsten Elementdaten ausgegeben werden. |
| LINE | bestimmt die Zeilen- und Spaltenanzahl einer Protokollseite bei Ausgabe in die Systemdatei SYSLST oder ein Bibliothekselement. |
| PATH | bestimmt bei Typ L dasjenige Sub-LLM, das aufgelistet werden soll. |
| REFERENCE | bestimmt, welche Referenznamen die aufzulistenden Elemente enthalten müssen. Ist die Referenzbedingung nicht erfüllt, wird das Element nicht mit aufgelistet. REFERENCE ist nur beim Elementtyp R erlaubt. Nach der Vereinbarung mit REFERENCE muß R bei LST angegeben werden. |
| SLICE | bestimmt bei Typ L den aufzulistenden Slice des Elements. |
| BASE | bestimmt die Basisadresse zur Relativierung von Bereichsadressen (siehe Verarbeitungsoperand INFO).<br>(Nur für BS2000-Lademodule.) |
| SEGMENT | bestimmt die Segmente, die aufgelistet werden sollen.<br>(Nur für BS2000-Lademodule.) |
| NEWFORM | bestimmt die Vorschubsteuerung des LMS-Protokolls. |
| LST | wird nur noch aus Kompatibilitätsgründen weiter unterstützt. Die Operandenwerte werden jedoch intern durch die Operandenwerte der Verarbeitungsoperanden FORMAT und INFO ersetzt. |

*Beispiel*

```
/SET-FILE-LINK FILE-NAME=TEST.LIB,LINK-NAME=LIB003
/START-PROGRAM $LMS
$PAR INFO=SUMMARY
$LSTS (3)
$END
```

Die Daten aller Typ S Elemente der Programmbibliothek TEST.LIB werden aufgelistet.

# NAM Umbenennen von Elementen

NAM benennt die angegebenen Elemente der zugeordneten Eingabebibliothek um. Der Name wird nur im Inhaltsverzeichnis der Eingabebibliothek geändert.

NAM ist für Delta-Elemente und auf sequentiellen Bibliotheken nicht zugelassen.

| Operation | Operanden |
|---|---|
| NAM[x] | {elem,...[(lib)] / (lib)} >elemu[,{...}>...] |

NAMx — Name der Anweisung mit Angabe des Elementtyps. Es sind alle Elementtypen erlaubt: S, M, R, J, P, C, D, X, H, L, F, U

* Stellvertretend für alle Elementtypen (nur zulässig bei Programmbibliotheken).

Der Elementtyp braucht nicht angegeben zu werden, wenn im Verarbeitungsoperanden TYPE der entsprechende Elementtyp vereinbart ist.

elem — Elementbezeichnung, die zu ändern ist, oder Auswahlangabe.

Für elem sind auch Namen zugelassen, die nicht den LMS-Konventionen genügen, um die weitere Bearbeitung solcher Elemente zu ermöglichen.

elemu — neue Elementbezeichnung; eine Konstruktionsangabe ist erlaubt.

lib — Kurzbezeichnung der Eingabebibliothek.

**Verarbeitungsoperanden**

TYPE — bestimmt den Elementtyp, falls bei der Anweisung selbst kein Typ angegeben wird.

DESTROY — bestimmt, ob in den umbenannten Elementen ein Kennzeichen für physikalisches Löschen gesetzt wird.
(Nur bei Programmbibliotheken möglich.)

OVERWRITE — steuert das Überschreiben gleichnamiger Elemente.

*Beispiel*

```
/START-PROGRAM $LMS
$LIB PROD.LIB,BOTH
$NAMM MAXOUT>MACOUT
$END
```

Der Makro MAXOUT wird in MACOUT umbenannt.

# NOP Leerfunktion

NOP löst in LMS keine Funktion aus. Sie kann in Prozeduren als Platzhalter verwendet werden.

| Operation | Operanden |
|---|---|
| NOP | [zeichenfolge] |

NOP Name der Anweisung.

zeichenfolge beliebiger Text.

*Hinweis*

Die Zeichen "!" und X'15' (NEW-LINE) dürfen nicht im Text vorkommen, da sie als Anweisungstrenner interpretiert werden.

*Beispiel*

NOP dient in einer Prozedur als Platzhalter für eine Anweisung.

```
/PROC C,(&ANW=NOP,...),SUBDTA=&
  .
  .
  .
/MODIFY-JOB-SWITCHES ON=(1)
/START-PROGRAM $LMS
  .
  .
  .
&ANW ELEM(1)
  .
  .
  .
END
/MODIFY-JOB-SWITCHES OFF=(1)
  .
  .
  .
/END-PROCEDURE
```

Die Prozedur wird aufgerufen mit dem Kommando /CALL-PROCEDURE ..., ANW=LSTS. Die Prozedur bewirkt dann, daß das Element ELEM der Bibliothek mit der Kurzbezeichnung (1) ausgegeben wird.

# NUM Numerieren von Elementsätzen

NUM numeriert das Kennungsfeld der angegebenen Elemente. Das erzeugte Element wird in die zugewiesene Ausgabebibliothek übertragen.

Die Lage, die Länge und der Inhalt des Kennungsfeldes wird durch Verarbeitungsoperanden definiert.
Die Satzkennungen dienen zur Identifikation eines Satzes und können in weiteren Anweisungen auf aufsteigende Reihenfolge überprüft werden.
Weitere Informationen über Kennungsfelder finden Sie auf Seite 48.

Ein- und Ausgabebibliotheken dürfen bei sequentiellen Bibliotheken nicht identisch sein. Bei den anderen Bibliothekstypen dürfen Ein- und Ausgabebibliothek identisch sein.

| Operation | Operanden |
|---|---|
| NUM[x] | {elem[(lib)] / (lib)} [>elemu] |

NUMx — Name der Anweisung mit Angabe des Elementtyps:
S, M, J, P, D, X

Der Elementtyp braucht nicht angegeben zu werden, wenn im Verarbeitungsoperanden TYPE der entsprechende Elementtyp vereinbart ist.

elem — Elementbezeichnung oder Auswahlangabe.

lib — Kurzbezeichnung der Eingabebibliothek.

elemu — Elementbezeichnung des Ausgabeelementes oder Konstruktionsangabe.

**Verarbeitungsoperanden**

TYPE — bestimmt den Elementtyp, falls bei der Anweisung selbst kein Typ angegeben wird.

DESTROY — bestimmt, ob im Ausgabeelement ein Kennzeichen für physikalisches Löschen gesetzt wird.
(Nur bei Programmbibliotheken möglich.)

RANGE — definiert Lage und Länge des Kennungsfeldes in den Ausgabesätzen.

VALUE — gibt Anfangswert und Schrittweite für die Numerierung im Kennungsfeld an.

STRING — gibt die Zeichenfolge an, die linksbündig in das Kennungsfeld der Ausgabesätze eingetragen wird.

OVERWRITE steuert das Überschreiben von gleichnamigen Elementen in der Ausgabebibliothek.

Der Verarbeitungsoperand wird jedoch nicht ausgewertet, wenn

```
Eingabebibliothek=Ausgabebibliothek und
                  elem=elemu
```

Das Eingabeelement wird dann überschrieben.

Das Element wird entsprechend den aktuellen Werten der Verarbeitungsoperanden RANGE, VALUE und STRING numeriert.

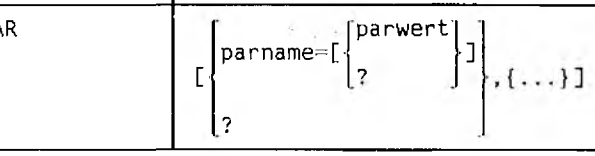
*Beispiel*

```
/START-PROGRAM $LMS
$LIB PROD1.LIB,BOTH
$PAR RANGE=1/8,VALUE=10000000/1000
$NUMS ELM2*
$END
```

Die Elemente, deren Elementname mit ELM2 beginnt, werden numeriert. Das Kennungsfeld des ersten Satzes erhält den Wert 10000000, des zweiten Satzes 10001000, usw.

# PAR Setzen von Verarbeitungsoperanden

PAR setzt die Verarbeitungsoperanden. Ein Verarbeitungsoperand kann sowohl einen Verarbeitungsmodus festlegen als auch für die Verarbeitung erforderliche Werte definieren.

| Operation | Operanden |
|---|---|
| PAR |  |



| | |
|---|---|
| PAR | Name der Anweisung. |
| parname | Name des Verarbeitungsoperanden (siehe ab Seite 205). |
| parwert | Wert des Verarbeitungsoperanden (siehe ab Seite 205). |
| ? | Für alle Verarbeitungsoperanden bzw. für die als parname angegebenen Verarbeitungsoperanden werden die aktuellen Werte protokolliert. |

Für alle Verarbeitungsoperanden sind Standardwerte definiert. Bei Beginn eines LMS-Laufs werden alle Verarbeitungsoperanden auf diese Standardwerte gesetzt. Die Standardwerte sind bei der Beschreibung jedes einzelnen Operanden (ab Seite 211) angegeben. Werden bei PAR die Werte nicht oder fehlerhaft angegeben, werden die entsprechenden Standardwerte eingesetzt. Die Verarbeitungsoperanden können in beliebiger Reihenfolge angegeben werden. Bei mehrmaliger Angabe des gleichen Verarbeitungsoperanden gilt immer die letzte Angabe. Die Angaben gelten bis zur expliziten Neufestlegung oder bis zum Rücksetzen auf die Standardwerte durch PAR ohne weitere Angaben. Bei den einzelnen Funktionen ist angegeben, welche Verarbeitungsoperanden die Funktion beeinflussen.

Einen Überblick über die Verarbeitungsoperanden finden Sie auf Seite 207.

# PRT Steuern der Protokollausgabe

PRT definiert das Ausgabemedium für LMS-Protokolle.

Die Ausgabemedien sind:

- die Datensichtstation
- die Systemdatei SYSLST
- ein Bibliothekselement

Wird das Protokoll in ein Element geschrieben, erzeugt LMS bei Programmbibliotheken ein Element mit dem Typ P, bei Quellprogrammbibliotheken ein Element mit dem Typ S.
Falls die Bibliothek, in die das Element geschrieben wird, eine Quellprogrammbibliothek ist, darf sie nicht die standardmäßige Ein- bzw. Ausgabebibliothek des aktuellen LMS-Laufes sein.

| Operation | Operanden |
|---|---|
| PRT | [(LST)<br>([SYS]OUT)<br>(CON)]<br>{<br>[(BOTH)<br>elem[(lib)]<br>?] |

| | |
|---|---|
| PRT | Name der Anweisung. |
| LST | gibt das Protokoll in die Systemdatei SYSLST aus. |
| SYSOUT | gibt das Protokoll in die Systemdatei SYSOUT aus (d.h. im Dialog auf die Datensichtstation). |
| CON | wirkt wie der Operand SYSOUT und wird nur noch aus Kompatibilitätsgründen unterstützt. |
| BOTH | gibt das Protokoll sowohl auf die Datensichtstation als auch in die Systemdatei SYSLST aus. |
| elem | gibt das Protokoll in das Bibliothekselement elem aus. |
| lib | Kurzbezeichnung der Bibliothek, in die das Element geschrieben wird. |
| ? | Der aktuelle Wert wird protokolliert. |

*Hinweis*

LMS-Fehlermeldungen werden immer nach SYSOUT ausgegeben, auch wenn PRT(LST) gesetzt ist.

**Verarbeitungsoperanden**

| | |
|---|---|
| LINE | steuert die Zeilen- und Spaltenanzahl der Protokollseite (nur bei SYSLST und elem sinnvoll). |
| LOG | steuert den Umfang des Protokolls; um ein Protokoll zu erhalten, muß PAR LOG=MED oder PAR LOG=MAX gesetzt sein (Standardwert LOG=MIN). |
| NEWFORM | definiert die Vorschubsteuerung für Protokolle. |
| OVERWRITE | steuert das Überschreiben von gleichnamigen Elementen in der Ausgabebibliothek. |
| DESTROY | bestimmt, ob im Protokollelement ein Kennzeichen für physikalisches Löschen gesetzt werden soll.<br>(Nur bei Programmbibliotheken möglich.) |

*Beispiel*

```
/SET-FILE-LINK FILE-NAME=PROT.LIB,LINK-NAME=LIB002
/START-PROGRAM $LMS
$PRT PROELEM(2)
$LIB EINAUS.LIB,BOTH
  :
  :
$END
```

Das Protokoll dieses LMS-Laufs wird in das Element PROELEM der Bibliothek PROT.LIB geschrieben.

# RST Verlassen des TEST-Modus

RST schaltet wieder in den RUN-Modus um, nachdem wegen eines Fehlers der TEST-Modus eingeschaltet wurde (siehe Seite 70). Die bisherigen Ein- und Ausgabebibliotheken werden geschlossen. Sie sind danach "undefiniert" und müssen erneut festgelegt werden.

| Operation | Operanden |
|---|---|
| RST | [STOP] |

RST Name der Anweisung.

STOP Der RUN-Modus wird wieder aufgenommen.
Das LMS-intern gesetzte Abbruchkennzeichen wird jedoch nicht gelöscht.

Ist das Abbruchbit bei der Beendigung von LMS gesetzt, beendet sich LMS mit TERMJ statt mit TERM. LMS gibt die Schlußmeldung 'TERMINATED' aus.

Fehlt dieser Operand, wird das evtl. gesetzte Abbruchkennzeichen gelöscht und der RUN-Modus wieder aufgenommen.

**Verarbeitungsoperand**

TEST schaltet den TEST-Modus ein.

Ist TEST=YES gesetzt, dann hat RST keine Wirkung.

# SEL Ausgeben von Elementen in Dateien und FMS-Bibliotheken

SEL gibt Elemente der zugewiesenen Eingabebibliotheken in Dateien oder FMS-Bibliotheken aus. Für diese verschiedenen Funktionen hat SEL zwei Formate:

### Format 1

Ausgeben von Elementen in Dateien

| Operation | Operanden |
|---|---|
| SEL[x] | elem,...[>{[prefix.](name)[.suffix] / dateiname / LINK=dateikettungsname}] |

### Format 2

Ausgeben von Elementen in FMS-Bibliotheken

| Operation | Operanden |
|---|---|
| SEL[x] | elem>FMS=fmslib(fmselem) |

### Format 1: Ausgeben von Elementen in Dateien

Dieses Format von SEL gibt Bibliothekselemente in Dateien aus.

LMS erzeugt die Dateien entsprechend

- den gespeicherten Dateimerkmalen (PAR KEY=YES) und dem Verarbeitungsoperanden FCBTYPE.
- dem Eintrag in der TASK-FILE-TABLE (TFT), wenn die Datei mit dem Dateikettungsnamen zugeordnet ist. Diese Angabe hat Vorrang vor den gespeicherten Dateimerkmalen.
- dem Katalogeintrag.

Die Dateien können RECORD-FORMAT=UNDEFINED und beliebige Angaben zu BUFFER-LENGTH und RECORD-SIZE haben. Die maximale Satzlänge von 32 KByte (inkl. Satzkopf) darf allerdings nicht überschritten werden.

Wurden die ISAM-Schlüssel einer ISAM-Datei mit in das Element aufgenommen (PAR KEY=YES), werden die ISAM-Schlüssel mit SEL auch wieder mit ausgegeben. Die Verarbeitungsoperanden CHECK, STRING und VALUE werden in diesem Fall ignoriert.

Für Elemente vom Typ C und für PAM-Dateien unter dem Typ X werden beim Selektieren wieder PAM-Dateien erzeugt.

| Operation | Operanden |
|---|---|
| SEL[x] | elem,...[>{[prefix.](name)[.suffix] \| dateiname \| LINK=dateikettungsname}] |

SELx — Name der Anweisung mit der Angabe des Elementtyps:
S, M, R, J, P, D, X

C nur für BS2000-Lademodule.

Der Elementtyp braucht nicht angegeben zu werden, wenn im Verarbeitungsoperanden TYPE der entsprechende Elementtyp vereinbart ist.

elem — Elementbezeichnung des auszugebenden Elementes.
Die Auswahlangabe ist zulässig, wenn

- dateiname nicht angegeben wird,
- beim Ausdruck [prefix.](name)[.suffix] für name eine Konstruktionsangabe vergeben wurde.

dateiname vollqualifizierter Dateiname der zu erzeugenden Datei.
Eine Auswahlangabe ist nicht erlaubt.
Für dateiname kann auch pfadname (siehe Benutzer-Kommandos [7]) angegeben werden.

LINK=dateikettungsname
Dateikettungsname, der auf eine mit dem /SET-FILE-LINK-Kommando zugewiesene Datei verweist. Die zugewiesene Datei darf keine Banddatei, sondern muß eine Plattendatei sein.

[prefix.](name)[.suffix]
Wenn für die Elementbezeichnung elem eine Auswahlangabe vereinbart wurde, können mit diesem Ausdruck mehrere Elemente in verschiedene Dateien ausgegeben werden. name muß in diesem Fall eine Konstruktionsangabe sein.

prefix. gibt den gemeinsamen vorderen Teilnamen der zu erzeugenden Dateien an.
'prefix' muß mit einem Punkt aufhören.

.suffix gibt den gemeinsamen hinteren Teilnamen der zu erzeugenden Dateien an.
suffix muß mit einem Punkt beginnen.

name gibt den Teilnamen an, mit dem prefix und/oder suffix zu einem oder mehreren vollqualifizierten Dateinamen ergänzt werden.
Wird für name eine Konstruktionsangabe vergeben, bildet sich dieser Teil der Dateinamen aus den Elementnamen.

*Hinweis*

Zulässige Elementnamen sind nicht in jedem Fall als Dateinamen zulässig.

**Verarbeitungsoperanden**

TYPE bestimmt den Elementtyp, falls bei der Anweisung selbst kein Typ angegeben wird.

FCBTYPE bestimmt für textartige Elemente den FCB-Typ der Ausgabedatei, entsprechend den mit PAR KEY gespeicherten Dateimerkmalen.

OVERWRITE bestimmt, ob eine gleichnamige Datei mit den Datensätzen des Eingabeelementes überschrieben, nicht überschrieben oder erweitert wird.

CHECK definiert ein Kennungsfeld in den Eingabesätzen.

STRING gibt eine Zeichenfolge an, die linksbündig im ISAM-Schlüsselfeld abgelegt wird.

VALUE definiert Anfangswert und Schrittweite des numerischen Wertes, der im ISAM-Schlüssel abgelegt wird.

PHASE bestimmt, ob eine NK- oder PK-Phase erzeugt wird.

Die Verarbeitungsoperanden CHECK, STRING und VALUE werden ignoriert, wenn im Element ISAM-Schlüssel gespeichert sind.

**Erzeugen von ISAM-Dateien**

Beim Ausgeben von Elementen in ISAM-Dateien erzeugt LMS die ISAM-Schlüssel wie folgt:

- Wurden bei der Aufnahme einer ISAM-Datei als Bibliothekselement die ISAM-Schlüssel mit aufgenommen (PAR KEY=YES), erzeugt LMS die ISAM-Datei mit diesen gespeicherten ISAM-Schlüsseln.
- Sind im Element keine ISAM-Schlüssel gespeichert, erzeugt LMS die ISAM-Schlüssel:

  1. Erzeugen der ISAM-Schlüssel mit den Verarbeitungsoperanden VALUE und STRING

     VALUE definiert Anfangswert und Schrittweite des rechtsbündigen numerischen Wertes im ISAM-Schlüssel. Eine evtl. mit STRING vereinbarte Zeichenfolge wird linksbündig im ISAM-Schlüsselfeld abgelegt.
     Der Verarbeitungsoperand CHECK muß in diesem Fall den Wert NO haben, d.h., es darf kein Kennungsfeld in den Eingabesätzen definiert sein.

  2. Erzeugen der ISAM-Schlüssel mit Standardwerten

     Haben die Verarbeitungsoperanden CHECK und VALUE den Wert NO, erzeugt LMS standardmäßig die ISAM-Schlüssel mit einem Anfangswert 1000 und der Schrittweite 1000. Die erzeugten Werte werden rechtsbündig im ISAM-Schlüssel abgelegt.
     Falls mit dem Verarbeitungsoperanden STRING eine Zeichenfolge vereinbart wurde, wird sie linksbündig im Schlüsselfeld abgelegt.

  3. Übernahme der ISAM-Schlüssel aus dem Kennungsfeld

     Mit dem Verarbeitungsoperanden CHECK=anf/lge ist ein Kennungsfeld in den Eingabesätzen zu vereinbaren. Der Inhalt dieses Kennungsfeldes wird linksbündig im ISAM-Schlüssel des Dateisatzes abgelegt. Haben Kennungsfeld (lge) und ISAM-Schlüsselfeld (KEY-LENGTH) unterschiedliche Länge, richtet LMS aus:

     KEY-LENGTH<lge: das Kennungsfeld wird rechts abgeschnitten.
     KEY-LENGTH>lge: das ISAM-Schlüsselfeld wird rechts mit Nullen aufgefüllt.

     Die Verarbeitungsoperanden STRING und VALUE werden in diesem Fall nicht ausgewertet.

*Hinweis*

- Elemente vom Typ R werden bis zum END-Satz ausgegeben.
  Eventuell nachfolgende Sätze werden ignoriert.
- Korrekturjournalsätze (TXTP) werden bei Elementen vom Typ C nicht mit ausgegeben.
- Die RECORD-SIZE wird nur bei RECORD-FORMAT=FIXED versorgt; sie hat bei RECORD-FORMAT=VARIABLE den Wert 0.
- Ist bei SEL und PAR OVERWRITE=YES bereits eine Datei mit dem Namen der zu erzeugenden Datei vorhanden, muß die bereits katalogisierte Datei dieselben Dateieigenschaften wie die zu erzeugende Datei besitzen.

*Beispiel 1*

```
/SET-FILE-LINK FILE-NAME=AUSDAT,LINK-NAME=AUSD, -
/              ACCESS-METHOD=SAM,RECORD-FORMAT=VARIABLE
/START-PROGRAM $LMS
$LIB EIN.BIB,IN
$SELS ELEM1>LINK=AUSD
$END
```

Das Element ELEM1 wird mit SEL in die Datei AUSDAT mit den spezifizierten Dateieigenschaften ausgegeben.

*Beispiel 2*

Sollen alle Elemente einer Bibliothek unter ihrem Namen ausgegeben werden, ist folgende Anweisung anzugeben:

$SEL[x] *

### Format 2: Ausgeben von Elementen in FMS-Bibliotheken

Dieses Format von SEL gibt ein Element einer LMS-Bibliothek in eine FMS-Bibliothek aus. Das Element wird mit den Eigenschaften in die FMS-Bibliothek geschrieben, die in der FMS-Bibliothek dafür definiert wurden. Standardmäßig wird eine SAM-Datei in der FMS-Bibliothek erzeugt. Falls ein Element überschrieben wird, erhält es die bestehenden Eigenschaften.

Bei dieser Anweisung wird das Programm FMS (siehe FMS [10]) als Unterprogramm aufgerufen.

| Operation | Operanden |
|---|---|
| SEL[x] | elem>FMS=fmslib(fmselem) |

SELx — Name der Anweisung mit der Angabe des Elementtyps:
S, M, R, J, P, D, X

Der Elementtyp braucht nicht angegeben zu werden, wenn im Verarbeitungsoperanden TYPE der entsprechende Elementtyp vereinbart ist.

elem — Elementbezeichnung des auszugebenden Elementes.
Eine Auswahlangabe ist nicht erlaubt.

fmslib — vollqualifizierter Dateiname der FMS-Bibliothek.
Für fmslib kann auch pfadname (siehe Benutzer-Kommandos [7]) angegeben werden.

fmselem — vollständige Elementbezeichnung des erzeugten Elementes in der FMS-Bibliothek.

**Verarbeitungsoperanden**

CHECK — definiert ein Kennungsfeld in den Eingabesätzen.

OVERWRITE — steuert das Überschreiben von gleichnamigen FMS-Elementen. Es dürfen nur die Werte YES, NO oder ONLY vereinbart werden.

STRING — gibt eine Zeichenfolge an, die linksbündig im ISAM-Schlüsselfeld abgelegt wird.

VALUE — definiert Anfangswert und Schrittweite des numerischen Wertes, der im ISAM-Schlüssel abgelegt wird.

*Beispiel*

```
/START-PROGRAM $LMS
$LIB ERT.LIB,BOTH
$SELS ERTEL>FMS=FMSERT(ERT3)
$DELS ERTEL
$END
```

Das Element ERTEL wird in die FMS-Bibliothek als Element ERT3 geschrieben, anschließend wird es in der Bibliothek ERT.LIB gelöscht.

# SUM Speichern von Vergleichsstatistiken

Beim Vergleichen von Elementen mit COM werden Vergleichsstatistiken erzeugt. Diese Vergleichsstatistiken speichert COM in einem Summenfeld mit der Bezeichnung S1, falls vor COM der Verarbeitungsoperand SUM gesetzt wurde.

SUM führt folgende Aktionen aus:

SUMPRT S1 gibt die aufsummierte Vergleichsstatistik im Summenfeld S1 mit einer Überschrift aus.

SUMADD S1>S2 addiert die Vergleichsstatistik im Summenfeld S1 auf das Summenfeld S2.

SUMDEL S1 das Summenfeld S1 wird gelöscht.

| Operation | Operanden |
|---|---|
| SUM | ['text'] |

SUM Name der Anweisung.

text Text, der als Überschrift vor der Vergleichsstatistik ausgegeben wird.

Hochkommata innerhalb des Textes müssen verdoppelt werden.
Standardwert: AREA S1

**Verarbeitungsoperand**

SUM wird vor COM vereinbart und steuert das Speichern der Vergleichsstatistik.

# SUMPRT Ausgeben der Vergleichsstatistik

SUMPRT gibt eines der verwendeten Summenfelder S1 oder S2 aus. Der in Hochkommata angegebene Text wird als Überschrift ausgegeben.

| Operation | Operanden |
|---|---|
| SUMPRT | [S1 / S2] [,'text'] |

SUMPRT Name der Anweisung.

S1
S2 Bezeichnung des auszugebenden Summenfeldes.

text Überschriftszeile beim Ausgeben des Summenfeldes.
Hochkommata innerhalb des Textes sind zu verdoppeln.

# SUMADD Addieren von Vergleichsstatistiken

SUMADD addiert die im Summenfeld S1 bzw. S2 gespeicherte Vergleichsstatistik in das Summenfeld S2 bzw. S1.

Summenfeld S2 = Summenfeld S2 + Summenfeld S1 (SUMADD S1>S2)

| Operation | Operanden |
|---|---|
| SUMADD | [S1 / S2] > {S1 / S2} |

SUMADD Name der Anweisung.

# SUMDEL Löschen der Vergleichsstatistik

SUMDEL löscht die Vergleichsstatistik im angegebenen Summenfeld.

| Operation | Operanden |
|---|---|
| SUMDEL | [S1 \| S2] |

SUMDEL Name der Anweisung.

S1
S2 Bezeichnung des zu löschenden Summenfeldes.

# SYS Absetzen von Systemkommandos

Mit SYS werden die Systemkommandos gegeben, die mit dem CMD-Makro verarbeitet werden. Der Programmzustand wird nicht verlassen. Wird kein Operand angegeben, wirkt SYS wie ein /BREAK-Kommando. Die Rückkehr in den Programmodus erfolgt mit dem /RESUME-PROGRAM-Kommando.

| Operation | Operanden |
|---|---|
| SYS | [['systemkommando' \| systemkommando]] |

SYS Name der Anweisung.

systemkommando
Name eines Kommandos mit notwendigen oder gewünschten Operanden. Das Systemkommando wird unverändert an den Kommandoprozessor weitergereicht.

# TCH Steuern des Bildschirmwechsels

TCH kontrolliert den Bildschirmüberlauf und bestimmt, ob die Ausgabe auf einem neuen Bildschirm oder im Roll-Up-Modus erfolgen soll.

| Operation | Operanden |
|---|---|
| TCH | [O[FLOW]=(N[O] / A[CK] / T[IMER] / ?)][,N[EWSCREEN]=(Y[ES] / N[O] / ?)] / ? |

OFLOW Dieser Operand ist an den Operanden OVERFLOW-CONTROL des /MODIFY-TERMINAL-OPTIONS-Kommandos angelehnt (siehe Benutzer-Kommandos [7]). Er steuert das Überschreiben eines vollen Bildschirms (Überlaufkontrolle).

Standardwert für OFLOW ist der vom Benutzer oder vom Systemverwalter festgelegte Wert im /MODIFY-TERMINAL-OPTIONS-Kommando.

=NO Es wird keine Überlaufkontrolle durchgeführt, d.h. falls der Bildschirm voll ist und weitere Daten zur Ausgabe anstehen, wird der Bildschirm überschrieben.

=ACK Ist der Bildschirm voll und stehen weitere Daten zur Ausgabe an, gibt LMS folgende Meldung aus:

```
PLEASE ACKNOWLEDGE (NO/TIMER/<ANY INPUT>) /
INTERRUPT (NE/NS/NI):
```

Der Bildschirm wird erst überschrieben, wenn der Benutzer nach dieser Meldung eine Eingabe macht, die in der PLEASE ACKNOWLEDGE-Meldung vorgeschlagen wurde. Die Bedeutung der Eingabemöglichkeiten NE/NS/NI ist im Abschnitt "Bildschirmwechsel steuern" (Seite 69) beschrieben.

<ANY INPUT> bedeutet, daß jede andere als eine der vorgeschlagenen Möglichkeiten die Ausgabe des nächsten Bildschirms bewirkt.

=TIMER Das System wartet 6 Sekunden, ehe der nächste Bildschirm ausgegeben wird.

| | |
|---|---|
| NEWSCREEN | Dieser Operand steuert den Bildschirmwechsel. |
| =YES | Nach Eingabe der Anweisung wird der Bildschirm gelöscht, bevor die Funktionsausgabe erfolgt. |
| =NO | Nach Eingabe der Anweisung wird der Bildschirm nicht gelöscht. |
| ? | Der aktuelle Wert wird protokolliert. |

# TOC Auflisten eines Inhaltsverzeichnis einer Bibliothek

TOC gibt die Inhaltsverzeichniseinträge der angegebenen Elemente oder der gesamten Bibliothek aus. Mit Hilfe des Verarbeitungsoperanden REFERENCE kann das Inhaltsverzeichnis auf die Elemente beschränkt werden, die einen bestimmten Referenznamen enthalten.

| Operation | Operanden |
|---|---|
| TOC[x] | [{elem[(lib)] / (lib)},...] |

TOCx Name der Anweisung mit Angabe des Elementtyps. Es sind alle Elementtypen erlaubt: S, M, R, P, J, C, D, X, H, L, F, U

* Stellvertretend für alle Elementtypen (nicht zulässig bei Bandbibliotheken).

Der Elementtyp braucht nicht angegeben zu werden, wenn im Verarbeitungsoperanden TYPE der entsprechende Elementtyp vereinbart ist.

elem Elementbezeichnung oder Auswahlangabe.

lib Kurzbezeichnung der Eingabebibliothek.

**Verarbeitungsoperanden**

TYPE gibt den Elementtyp an, falls bei der Anweisung selbst kein Typ angegeben wird.

REFERENCE bestimmt, welche Referenznamen die auszugebenden Elemente enthalten müssen. Ist die Referenzbedingung nicht erfüllt, wird das Element nicht mit ausgegeben. REFERENCE ist nur beim Elementtyp R erlaubt. Nach der Vereinbarung mit REFERENCE muß R bei TOC angegeben werden.

SORT definiert die Sortierkriterien bei der Ausgabe des Inhaltsverzeichnisses.

Standardwert: Das Inhaltsverzeichnis wird nach Namen, Versionsnummer und Datum sortiert ausgegeben.

TOC steuert das Format des Inhaltsverzeichnisprotokolls von Programmbibliotheken.

Standardwert: Das Format wird je nach Länge der Elementnamen für jeden Elementtyp neu festgelegt.

LINE bestimmt die Zeilen- und Spaltenanzahl einer Protokollseite bei Ausgabe in die Systemdatei SYSLST oder ein Bibliothekselement.

Inhaltsverzeichnisse, die aufgrund ihrer Größe nicht mehr sortiert werden können, können durch Setzen des Auftragsschalters 9 zusammenhängend sortiert werden.

*Beispiel*

```
/SET-FILE-LINK FILE-NAME=A.LIB,LINK-NAME=LIB001
/START-PROGRAM $LMS
$LIB EINAUS.LIB,BOTH
$TOC* *
$PAR SORT=V
$TOCS (1)
$END
```

Mit /SET-FILE-LINK-Kommando zugewiesen wird die Quellprogrammbibliothek A.LIB. Die Programmbibliothek EINAUS.LIB wird als standardmäßige Ein-/Ausgabebibliothek zugewiesen. Da sie eine Programmbibliothek ist, kann bei TOC der Elementtyp * angegeben werden. Bei anderen Bibliotheken kann der Elementtyp * nicht bei TOC angegeben werden. Durch die zweite TOC-Anweisung werden alle Elemente von Typ S aus der Bibliothek A.LIB nach Versionen sortiert ausgegeben.

*Hinweis*

Um den gesamten Inhalt einer Bibliothek zu erhalten (alle Elemente mit allen Versionen) muß TOC oder TOC* */* angegeben werden. Diese Funktion läuft auf allen Bibliothekstypen, auf denen der Typ '*' zugelassen ist.

# UPD Korrigieren von Binde- und Lademodulen und LLMs

UPD korrigiert das angegebene Element der zugewiesenen Eingabebibliothek. Das korrigierte Element wird dann in die zugewiesene Ausgabebibliothek geschrieben. Es kann dabei eine neue Elementbezeichnung erhalten.

Für die Korrekturen von Binde- und Lademodulen und LLMs hat UPD verschiedene Subanweisungen. Die Subanweisungen werden unmittelbar hinter UPD bis zu *END aus dem Anweisungsstrom gelesen.

In Subanweisungen sind *keine* Folgezeilen möglich.

Ein- und Ausgabebibliotheken dürfen bei sequentiellen Bibliotheken nicht identisch sein. Bei den anderen Bibliothekstypen dürfen Ein- und Ausgabebibliotheken identisch sein.

3 Formate stehen UPD zur Verfügung:

**Format 1**: Korrigieren von Bindemodulen

| Operation | Operanden |
|---|---|
| UPD[R] | elem[(lib)][>elemu] |

**Format 2**: Korrigieren von Lademodulen (BS2000-Phasen)

| Operation | Operanden |
|---|---|
| UPD[C] | elem[(lib)][>elemu] |

**Format 3**: Korrigieren von LLMs

| Operation | Operanden |
|---|---|
| UPD[L] | elem[(lib)][>elemu] |

**Format 1: Korrigieren von Bindemodulen**

| Operation | Operanden |
|---|---|
| UPD[R] | elem[(lib)][>elemu] |

UPDR — Name der Anweisung mit dem Elementtyp R.
R darf weggelassen werden, wenn im Verarbeitungsoperanden TYPE der Elementtyp R vereinbart ist.

elem — vollständige Elementbezeichnung des zu korrigierenden Elementes oder eine Auswahlangabe (keine Listenangabe).

lib — Kurzbezeichnung der Eingabebibliothek.

elemu — Elementbezeichnung des Ausgabeelementes oder Konstruktionsangabe.

**Verarbeitungsoperanden**

TYPE — bestimmt den Elementtyp, falls bei der Anweisung selbst kein Typ angegeben wird.

OVERWRITE — bestimmt das Überschreiben gleichnamiger Elemente in der Ausgabebibliothek.
Dieser Verarbeitungsoperand wird jedoch nicht ausgewertet, wenn

```
Eingabebibliothek=Ausgabebibliothek und
                  elem=elemu
```

Das Eingabeelement wird dann überschrieben.

DESTROY — bestimmt, ob im Ausgabeelement ein Kennzeichen für physikalisches Löschen eingetragen wird.
(Nur bei Programmbibliotheken möglich.)

STRIP — bestimmt, welche Satzarten beim Korrigieren ausgeschlossen werden sollen.

LMS sammelt zunächst die UPD-Subanweisungen und prüft dabei

- die Syntax der Subanweisungen,
- die Eindeutigkeit der Korrekturen (Überlappung),
- die Eindeutigkeit der Symbole beim Umbenennen.

Nach *END werden die eingegebenen Subanweisungen auf ihre Durchführbarkeit geprüft und dann ausgeführt.

## Korrekturanweisungen für UPDR

### Übersicht

| Korrekturanweisung | Funktion |
|---|---|
| *BAS basisadr | Festlegen einer Basisadresse |
| *CON kontrollzahl | Festlegen der CROSS-Kontrollzahl |
| *COR [csectname,][basisadr+]adresse,<br>[[{C\|X\|B}]'prüftext'=[:=]][{C\|X\|B}]'korrtext'<br>[,ID='ident'][,CONTROL=zahl] | Korrigieren von Textsätzen |
| *DEL {satzart\|(satzart,...)\|TXTP[,ID='ident']} | Löschen von Bindemodulteilen |
| *END | Beenden der Korrektureingaben |
| *ID 'ident' | Festlegen der Identifikation |
| *INS INCLUDE (modul,...)[,bibliothek] | Eintragen eines INCLUDE-Satzes |
| *INV {REP\|COR[,ID='ident']} | Umwandeln von Korrekturen |
| *NAM {CSECT:\|ENTRY:\|EXTRN:\|COMMON:}namealt,nameneu | Umbenennen von Symbolen |
| *REM [ID='ident'] | Zurücknehmen von Korrekturen |

| Korrekturanweisung | Funktion |
|---|---|
| *REP [csectname,][basisadr+]adresse,<br>[[{C\|X\|B}]'prüftext'=[:=]][{C\|X\|B}]'korrtext'<br>[,CONTROL=zahl] | Einfügen eines REP-Satzes |
| *SET {csectname\|*}[,PRIV={Y\|N}][,PUBLIC={Y\|N}]<br>[,VISIBLE={Y\|N}][,READONLY={Y\|N}][,PAGE={Y\|N}]<br>[,RESIDENT={Y\|N}][,RMODE={24\|ANY}][,AMODE={24\|31\|ANY}] | Verändern von Programmabschnitts-merkmalen |

## Beschreibung der Korrekturanweisungen für Bindemodule

*BAS Festlegen einer Basisadresse

*BAS legt eine Basisadresse fest. Die Basisadresse wird dann mit der Adresse in einem folgenden *COR zur absoluten Adresse im Bindemodul addiert, falls nicht explizit eine Basisadresse bei *COR angegeben ist. Standardwert ist 0.

| Operation | Operanden |
|---|---|
| *BAS | basisadr |

basisadr legt die Basisadresse (sedezimal) fest.
0 ≤ basisadr ≤ 7FFFFFFF

*CON Festlegen der CROSS-Kontrollzahl

*CON legt die CROSS-Kontrollzahl für den kompletten Korrekturlauf fest. Wird *CON mehrfach angegeben, gilt stets die letzte Angabe.

Über Kontrollzahlen ist es möglich, mehr Sicherheit bei der Durchführung und Weitergabe von Korrekturen zu erreichen. LMS errechnet beim Korrekturlauf aus der Zeichenfolge jeder Korrekturanweisung eine Kontrollzahl und aus diesen Zahlen wiederum die Cross-Kontrollzahl. Diese Werte werden auch im TEST-Modus ermittelt. Die Kontrollzahlen werden beim Protokollieren der jeweiligen Korrekturanweisung, die Cross-Kontrollzahl am Ende des Korrekturprotokolls ausgegeben. Bei einer Weitergabe der Korrektur oder einer endgültigen Durchführung im RUN-Modus sollten auch die Kontrollzahlen und die Cross-Kontrollzahl mitangegeben werden. Sind Korrekturen auszuführen, vergleicht LMS die angegebenen Zahlen mit den erneut berechneten. Stimmen sie nicht überein, wird keine Korrektur ausgeführt.

| Operation | Operanden |
|---|---|
| *CON | kontrollzahl |

kontrollzahl legt die CROSS-Kontrollzahl (sedezimal) fest.
0 ≤ kontrollzahl ≤ 7FFFFFFF

*COR Korrigieren von Textsätzen

*COR korrigiert Textsätze eines Bindemoduls und erzeugt einen Korrekturjournalsatz (TXTP-Satz), der den ursprünglichen Inhalt des Textbereiches enthält.

| Operation | Operanden |
|---|---|
| *COR | [csectname,][basisadr+]adresse,<br>[[{C\|X\|B}]'prüftext'=[:=]][{C\|X\|B}]'korrtext'<br>[,ID='ident'][,CONTROL=zahl] |

csectname gibt den Namen einer CSECT an.
Wird csectname angegeben, wird nur innerhalb dieser CSECT korrigiert.
Bei Großmodulen neuen Formats (mit komplettem ESD) muß immer über csectname korrigiert werden.

basisadr legt die Basisadresse (sedezimal) fest. Diese Basisadresse gilt nur für dieses *COR. Wird basisadr nicht angegeben, gilt der Wert, der bei *BAS angegeben ist.

0 ≤ basisadr ≤ 7FFFFFFF

adresse legt die relative Adresse fest.
basisadr + adresse ergeben die absolute Adresse im Bindemodul bzw. die CSECT-relative Adresse, falls csectname angegeben ist.

0 ≤ basisadr+adresse ≤ 7FFFFFFF

C'prüftext' gibt den Prüftext zeichenweise an. Ein Hochkomma im Text muß doppelt angegeben werden.
prüftext darf höchstens 50 Zeichen lang sein.

X'prüftext' gibt den Prüftext sedezimal an.
prüftext darf höchstens 50 Byte lang sein.

B'prüftext' gibt den Prüftext binär an.
prüftext darf höchstens 50 Zeichen lang sein.

Der Originaltext zum Vergleich mit dem Prüftext wird aus den für diesen Bereich existierenden TXT-Sätzen gebildet. Wenn es mehrere Texte für dieselbe Adresse gibt, gilt der letzte Text.

=:= Prüftext und Korrekturtext müssen gleich lang sein.

= Prüftext und Korrekturtext dürfen verschieden lang sein.

C'korrtext' gibt den Korrekturtext zeichenweise an. Ein Hochkomma im Text muß doppelt angegeben werden.
korrtext darf höchstens 50 Zeichen lang sein.

X'korrtext' gibt den Korrekturtext sedezimal an.
korrtext darf höchstens 50 Byte lang sein.

B'korrtext' gibt den Korrekturtext binär an.
korrtext darf höchstens 50 Zeichen lang sein.

ID='ident' gibt eine Identifizierung zeichenweise an.
ident darf maximal 8 Zeichen lang sein.
Diese Identifizierung gilt nur für dieses *COR. Wird dieser Operand nicht angegeben, gilt die Angabe bei *ID.

CONTROL = zahl
legt eine lokale Kontrollzahl (sedezimal) fest.

$0 \leq$ zahl $\leq$ FFFF

Existiert kein TXT-Satz bzw. nur für einen Teilbereich der Korrektur, erzeugt LMS einen TXT-Satz und fügt ihn in den Modul ein.

Je Adresse können mehrere Textsätze bestehen (z.B. durch ORG-Anweisungen). In diesen Fällen sind eventuell mehrere Textsätze zu ändern.

Innerhalb aller Korrekturanweisungen darf kein *REP angegeben werden, da die Textkorrekturen sonst später beim Binden wieder überschrieben würden.

*DEL Löschen von Bindemodulteilen

*DEL schließt aus dem Eingabeelement folgende Satzarten aus:

- ISD-Sätze
- LSD-Sätze
- REP-Sätze
- INCLUDE-Sätze
- TXTP-Sätze
- DSDD-Sätze

| Operation | Operanden |
|---|---|
| *DEL | [satzart<br>{(satzart,...)<br>{TXTP[,ID='ident']} |

satzart legt die Satzart fest, die nicht vom Eingabeelement ins Ausgabeelement übernommen werden soll.
Zulässig für satzart ist:

- ISD
- LSD
- REP
- INCLUDE
- TXTP
- DSDD

ID='ident' gibt eine Identifizierung zeichenweise an. Diese Identifizierung gilt nur für dieses *DEL.
ident darf höchstens 8 Zeichen lang sein.

Wird dieser Operand nicht angegeben, gilt die Angabe bei *ID.

*Hinweis*

REP-, INCLUDE- und TXTP-Sätze sollten nur nach gründlicher Planung gelöscht werden!

*END Beenden der Korrektureingaben

*END schließt die Folge der Korrekturanweisungen ab. Anschließend prüft LMS alle Anweisungen auf Durchführbarkeit und führt die Anweisungsfolge aus.

| Operation | Operanden |
|---|---|
| *END | |

*ID Festlegen der Identifikation

*ID legt eine globale Identifikation fest. Sie gilt für alle Anweisungen, bei denen die lokale Identifikation nicht angegeben ist.

| Operation | Operanden |
|---|---|
| *ID | ['ident'] |

'ident' gibt die globale Identifikation zeichenweise an.
ident darf höchstens 8 Zeichen lang sein.

Wird der Operand nicht angegeben, werden als Standardwert 8 Leerzeichen angenommen.

*INS Eintragen eines INCLUDE-Satzes

*INS trägt einen INCLUDE-Satz in einen Bindemodul ein. Der INCLUDE-Satz wird vom Dynamischen Bindelader (DBL) ausgewertet.

| Operation | Operanden |
|---|---|
| *INS | INCLUDE (modul,...)[,bibliothek] |

modul Name des Bindemoduls, der eingebunden werden soll. Es dürfen höchstens 10 Bindemodule angegeben werden. Wird nur ein Bindemodul angegeben, darf die Klammer entfallen.
modul darf höchstens 8 Zeichen lang sein.

bibliothek Name der Programm- oder Bindemodulbibliothek, die die angegebenen Bindemodule enthält. Wird dieser Operand nicht angegeben, nimmt der DBL die TASKLIB.

*Hinweis*

- Die Angabe "(modul,...)[,bibliothek]" darf höchstens 71 Zeichen lang sein.
- LMS prüft weder die Existenz der angegebenen Bindemodule noch die Angabe der Bibliothek.

*INV Umwandeln von Korrekturen

*INV wandelt entweder REP-Sätze in Textkorrekturen oder Textkorrekturen in REP-Sätze um.

**Format 1**

**Umwandeln von REP-Sätzen in Textkorrekturen**

Alle REP-Sätze des Bindemoduls werden in Textkorrekturen umgewandelt. Dadurch entfällt die Bearbeitung der REP-Sätze beim Binden und Laden. Die umgewandelten REP-Sätze werden aus dem Bindemodul entfernt. Für korrigierte Textsätze erstellt LMS Korrekturjournalsätze. Für Großmodule neuen Formats wird diese Anweisung mit einer Fehlermeldung abgewiesen.

| Operation | Operanden |
|---|---|
| *INV | REP |

**Format 2**

**Umwandeln von Textkorrekturen in REP-Sätze**

Alle Textkorrekturen bzw. Textkorrekturen einer bestimmten Identifikation in einem Bindemodul, für die ein Korrekturjournalsatz existiert, werden in REP-Sätze umgewandelt (evtl. Probleme bei Großmodulen, siehe *REP). Die Korrekturjournalsätze werden anschließend gelöscht. Für Großmodule neuen Formats wird diese Anweisung mit einer Fehlermeldung abgewiesen.

| Operation | Operanden |
|---|---|
| *INV | COR[,ID='ident'] |

ID='ident' gibt eine Identifizierung an.
ident darf maximal 8 Zeichen lang sein.

Wird dieser Operand nicht angegeben, gilt die Angabe bei *ID. Wenn bei *ID ebenfalls nichts angegeben wurde, werden alle Textkorrekturen umgewandelt.

*NAM Umbenennen von Symbolen

*NAM ändert den Namen von einer CSECT, einem ENTRY, EXTRN oder COMMON. Jede Umbenennung verursacht eine Änderung von ESD-Sätzen. LMS prüft die Eindeutigkeit der Namen innerhalb aller ESD-Sätze und weist eine Umbenennung zurück, wenn der neue Name bereits existiert (im Gegensatz zu LMR).

| Operation | Operanden |
|---|---|
| *NAM | [CSECT: / ENTRY: / EXTRN: / COMMON:] namealt,nameneu |

namealt alter Name des Symbols. Der Name muß vollständig angegeben werden.

nameneu neuer Name des Symbols. Der Name muß vollständig angegeben werden.
nameneu darf höchstens 8 Zeichen lang sein

*Hinweis*

Auch maskierte CSECT-Namen können umbenannt werden.

*REM Zurücknehmen von Korrekturen

*REM macht alle Textkorrekturen bzw. Textkorrekturen einer bestimmten Identifikation, für die ein Korrekturjournalsatz existiert, wieder rückgängig. Die Korrekturjournalsätze werden anschließend gelöscht.

| Operation | Operanden |
|---|---|
| *REM | [ID='ident'] |

ID='ident' gibt die lokale Identifizierung zeichenweise an.
ident darf maximal 8 Zeichen lang sein.

Wird dieser Operand nicht angegeben, gilt die Angabe bei *ID.

*REP Einfügen eines REP-Satzes

***REP fügt REP-Sätze in den Bindemodul ein. Diese REP-Sätze wertet der Dynamische Bindelader (DBL) aus.**

| Operation | Operanden |
|---|---|
| *REP | [csectname,][basisadr+]adresse,<br>[[{C / X / B}]'prüftext'=[:=]][{C / X / B}]'korrtext'<br>[,CONTROL=zahl] |

csectname gibt den Namen einer CSECT an.
Für Großmodule neuen Formats ist diese Angabe nicht zulässig. Sie können nur über absolute Adressen korrigiert werden.

basisadr legt die Basisadresse (sedezimal) fest. Diese Basisadresse gilt nur für dieses *REP. Wird basisadr nicht angegeben, gilt der Wert, der bei *BAS angegeben ist.

$0 \leq$ basisadr $\leq$ FFFFF

Ist basisadr > FFFFF angegeben, wird mit Fehlermeldung abgewiesen.

adresse legt die relative Adresse fest.
basisadr + adresse ergeben die absolute Adresse im Bindemodul.

$0 \leq$ basisadr + adresse $\leq$ FFFFF

Die Adresse muß innerhalb des Modultextbereichs liegen.

C'prüftext' gibt den Prüftext zeichenweise an. Ein Hochkomma im Text muß doppelt angegeben werden.
prüftext darf höchstens 50 Zeichen lang sein.

X'prüftext' gibt den Prüftext sedezimal an.
prüftext darf höchstens 50 Byte lang sein.
Für Großmodule neuen Formats wird 'prüftext' ignoriert, d.h. es erfolgt keine Überprüfung auf alten Inhalt.

B'prüftext' gibt den Prüftext binär an.
prüftext darf höchstens 50 Zeichen lang sein.
Für Großmodule neuen Formats wird 'prüftext' ignoriert, d.h. es erfolgt keine Überprüfung auf alten Inhalt.

Der Originaltext zum Vergleich mit dem Prüftext wird aus den für diesen Bereich existierenden TXT-Sätzen gebildet. Wenn es mehrere Texte für dieselbe Adresse gibt, gilt der letzte Text.

=:= Prüftext und Korrekturtext müssen gleich lang sein.

= Prüftext und Korrekturtext dürfen verschieden lang sein.

C'korrtext' gibt den Korrekturtext zeichenweise an. Ein Hochkomma im Text muß doppelt angegeben werden.
korrtext darf höchstens 50 Zeichen lang sein.

X'korrtext' gibt den Korrekturtext sedezimal an.
korrtext darf höchstens 50 Byte lang sein.

B'korrtext' gibt den Korrekturtext binär an.
korrtext darf höchstens 50 Zeichen lang sein.

CONTROL=zahl
legt die lokale Kontrollzahl (sedezimal) fest.

$0 \leq$ zahl $\leq$ FFFF

Der REP-Satz wird nur eingefügt, falls die von LMS ermittelte Kontrollzahl mit der hier angegebenen Kontrollzahl übereinstimmt.

*Hinweis*

Im Gegenzatz zu *COR prüft LMS nicht, ob für den Korrekturbereich bereits REP-Sätze existieren. Die Korrektur eines Großmoduls muß stets ohne CSECT-Angabe und Prüftext erfolgen, d.h. über relative Adressen innerhalb des Großmoduls. Der volle Funktionsumfang kann nur bei Bindemodulen genutzt werden, die aus einem Übersetzerlauf resultieren.

*SET Verändern von Programmabschnittsmerkmalen

*SET verändert Programmabschnittsmerkmale.

| Operation | Operanden |
|---|---|
| *SET | {csectname / *} [,PRIV={Y / N}][,PUBLIC={Y / N}] [,VISIBLE={Y / N}][,READONLY={Y / N}][,PAGE={Y / N}] [,RESIDENT={Y / N}][,RMODE={24 / ANY}][,AMODE={24 / 31 / ANY}] |

csectname Name eines Programmabschnitts, dessen Merkmale geändert werden sollen.

* legt fest, daß in allen Programmabschnitten die Merkmale geändert werden sollen.

PRIV

=Y legt fest, daß nur privilegierte Systemroutinen auf die angegebenen Programmabschnitte zugreifen dürfen.

=N keine Zugriffsbeschränkung.

PUBLIC

=Y Die angegebenen Programmabschnitte sind mehrbenutzbar.

=N Die angegebenen Programmabschnitte sind nicht mehrbenutzbar.

VISIBLE

=Y Die angegebenen Programmabschnitte werden nicht maskiert (siehe Binder-Lader-Starter [2]). Für diese Abschnitte wird ein Sekundärnamenssatz angelegt und die Namen werden im Sekundärnamensverzeichnis eingetragen.

=N Die angegebenen Programmabschnitte werden maskiert. Für sie wird weder ein Sekundärnamenssatz angelegt noch werden die Namen im Sekundärnamensverzeichnis eingetragen. Ein eventuell vorhandener Sekundärnamenssatz wird gelöscht.

Werden alle Programmabschnitte eines Bindemoduls maskiert, wird ein Bibliothekselement ohne Sekundärnamenseintrag erzeugt. Dieser Bindemodul ist nur über Primärnamen auffindbar. LMS gibt eine Warnung aus.

Der Modulname kann jedoch aus dem ersten Programmabschnittsnamen mit Hilfe aller ESD-Sätze abgeleitet werden, da dazu auch maskierte Programmabschnitte verwendet werden.

*Hinweis*

Bindemodule, die nur maskierte Programmabschnitte besitzen, kann der Binder nicht bearbeiten, z.B. Ausschließen eines Bindemoduls bei der Autolink-Funktion.

READONLY

=Y legt fest, daß die angegebenen Programmabschnitte zur Ablaufzeit des Programms nur gelesen werden dürfen.

=N erlaubt, daß während des Programmlaufs auch in die angegebenen Programmabschnitte geschrieben werden darf.

PAGE

=Y legt fest, daß die angegebenen Programmabschnitte auf Seitengrenze ausgerichtet werden sollen, d.h. die Ladeadresse soll ein Vielfaches von dezimal 4096 bzw. sedezimal 1000 sein.

=N läßt Seitengrenzen unberücksichtigt. Die Programmabschnitte beginnen jeweils bei der nächsten Doppelwortadresse, die sich beim Binden ergibt.

RESIDENT

=Y legt fest, daß die angegebenen Programmabschnitte in den Klasse-3-Speicher geladen und dort gesichert gehalten werden.

=N legt fest, daß die angegebenen Programmabschnitte nicht in den Klasse-3-Speicher geladen werden.

RMODE

=24 legt fest, daß die angegebenen Programmabschnitte in den Adreßraum unterhalb der 16 MB Grenze zu laden sind.

=ANY Es existiert keine Einschränkung.

AMODE

=24 legt fest, daß die angegebenen Programmabschnitte im 24-Bit-Modus ablauffähig sein sollen.

=31 legt fest, daß die angegebenen Programmabschnitte im 31-Bit-Modus ablauffähig sein sollen.

=ANY beliebig.

*Hinweis*

Mindestens ein Merkmal muß angegeben werden, andernfalls wird eine Fehlermeldung ausgegeben.

### Altes Format von UPDR

Aus Kompatibilitätsgründen wird noch das alte Format von UPDR unterstützt.

| Operation | Operanden |
|---|---|
| UPD[R] | elem[(lib)][;zahl] |

Beschreibung von UPDR, elem und lib siehe UPDR neues Format.

zahl gibt die CROSS-Kontrollzahl sedezimal an.

0 ≤ zahl ≤ FFFFFF

Wirkung der Verarbeitungsoperanden siehe UPDR neues Format.

### Korrekturjournal

LMS schreibt für jeden bestehenden Satz, der geändert wird, einen Korrekturjournalsatz (TXTP-Satz). Dieser Korrekturjournalsatz enthält den ursprünglichen Text des korrigierten Satzes.

### Beschreibung der Korrekturanweisung im alten Format

```
L(adresse)[bitnummer][,[[v]'text1'=[:=]][v]'text2'][,zahl]

oder die Kurzform:

L(adresse)bitnummer[.zahl]
```

Wenn nur ein Bit auf 1 gesetzt werden soll, kann die Kurzform verwendet werden. Korrekturanweisungen, die nur aus der Angabe einer Adresse bestehen, sind nicht erlaubt.

Adreßtyp: Moduladresse.

adresse Sedezimale Adresse, max. 8 Stellen.
Sie kann dem Übersetzungsprotokoll oder der Modulauflistung ohne Umrechnung entnommen werden. Führende Nullen dürfen entfallen.

bitnummer Dezimalzahl, max. 3 Stellen.
Sie gibt die Nummer eines Bits in dem durch adresse definierten Feld an. Die Bits werden von links mit 1 beginnend gezählt.

| | | |
|---|---|---|
| v | X: | Text ist sedezimal angegeben. |
| | B: | Text ist binär angegeben. |
| | Angabe v fehlt: | Text ist alphanumerisch angegeben oder besteht aus Sonderzeichen. |
| text1 | Prüftext, max. 50 Zeichen.<br>Ein Apostroph in einem alphanumerischen Text muß doppelt angegeben werden. Bei sedezimalem Text muß eine gerade Anzahl von Zeichen angegeben werden. Der Prüftext wird immer gegen den Originaltext im Bindemodul verglichen. | |
| =:= | Prüftext und Korrekturtext müssen gleich lang sein. | |
| = | Prüftext und Korrekturtext dürfen verschieden lang sein. | |
| text2 | Korrekturtext, wie text1. | |
| zahl | Kontrollzahl, sedezimal, 4-stellig. | |

*Hinweis*

Handelt es sich um einen Großmodul neuen Formats, der mit dieser Korrekturanweisung korrigiert werden soll, wird dies mit einer Fehlermeldung abgewiesen.

Ist eine Bitnummer angegeben, muß v=B gesetzt sein und umgekehrt.

Der Prüf(Korrektur-)textbereich wird aus der angegebenen Adresse und der Länge des jeweiligen in der Korrekturanweisung angegebenen Textes gebildet. Ist ein Prüftext angegeben, wird er mit dem Prüftextbereich im Element verglichen; bei Übereinstimmung wird die Korrektur ausgeführt und in Abhängigkeit vom Verarbeitungsoperanden STRIP ein Korrekturjournalsatz angelegt. Kommt eine Adresse im Element mehrfach vor, dann wird der letzte Text zur Prüfung herangezogen; bei Übereinstimmung werden alle in den Korrekturtextbereich fallenden Textstellen des Elementes verändert.

**Format 2: Korrigieren von Lademodulen (BS2000-Phasen)**

| Operation | Operanden |
|---|---|
| UPD[C] | elem[(lib)][>elemu] |

UPDC — Name der Anweisung mit dem Elementtyp C.
C darf weggelassen werden, wenn im Verarbeitungsoperanden TYPE der Elementtyp C vereinbart ist.

elem — vollständige Elementbezeichnung des zu korrigierenden Elementes. oder eine Auswahlangabe (keine Listenangabe).

lib — Kurzbezeichnung der Eingabebibliothek.

elemu — Elementbezeichnung des Ausgabeelementes oder Konstruktionsangabe.

**Verarbeitungsoperanden**

TYPE — bestimmt den Elementtyp, falls bei der Anweisung selbst kein Typ angegeben wird.

OVERWRITE — bestimmt das Überschreiben gleichnamiger Elemente in der Ausgabebibliothek.
Dieser Verarbeitungsoperand wird jedoch nicht ausgewertet, wenn

```
Eingabebibliothek=Ausgabebibliothek und
                  elem=elemu
```

Das Eingabeelement wird dann überschrieben.

DESTROY — bestimmt, ob im Ausgabeelement ein Kennzeichen für physikalisches Löschen eingetragen wird.
(Nur bei Programmbibliotheken möglich.)

STRIP — bestimmt, welche Satzarten beim Korrigieren ausgeschlossen werden sollen.

## Korrekturanweisungen für UPDC

### Übersicht

| Korrekturanweisung | Funktion |
|---|---|
| *BAS basisadr | Festlegen einer Basisadresse |
| *CON kontrollzahl | Festlegen der CROSS-Kontrollzahl |
| *COR [segment,][basisadr+]adresse,<br>[[{C / X}]'prüftext'=[:=]][{C / X}]'korrtext'<br>[,ID='ident'][,CONTROL=zahl] | Korrigieren von Textsätzen |
| *DEL TXTP[,ID='ident'] | Löschen von Korrekturjournalsätzen |
| *END | Beenden der Korrektureingaben |
| *ID 'ident' | Festlegen der Identifikation |
| *REM [ID='ident'] | Zurücknehmen von Korrekturen |
| *SEG {segment / %ROOT} | Festlegen eines Segments |

## Beschreibung der Korrekturanweisungen für Lademodule

*BAS Festlegen einer Basisadresse

*BAS legt eine Basisadresse fest. Die Basisadresse wird dann mit der Adresse in einem folgenden *COR zur absoluten Adresse im Lademodul addiert, falls nicht explizit eine Basisadresse bei *COR angegeben ist. Standardwert ist 0.

| Operation | Operanden |
|---|---|
| *BAS | basisadr |

basisadr legt die Basisadresse (sedezimal) fest.

$0 \leq$ basisadr $\leq$ 7FFFFFFF

*CON Festlegen der CROSS-Kontrollzahl

*CON legt die CROSS-Kontrollzahl für den kompletten Korrekturlauf fest. Wird *CON mehrfach angegeben, gilt stets die letzte Angabe.

| Operation | Operanden |
|---|---|
| *CON | kontrollzahl |

kontrollzahl legt die CROSS-Kontrollzahl (sedezimal) fest.

$0 \leq$ kontrollzahl $\leq$ 7FFFFFFF

*COR Korrigieren von Textsätzen

*COR korrigiert Textsätze innerhalb eines Segments und erzeugt einen Korrekturjournalsatz, der den ursprünglichen Inhalt des Textbereiches enthält.

| Operation | Operanden |
|---|---|
| *COR | [segment,][basisadr+]adresse, [[{C / X}]'prüftext'=[:=]][{C / X}]'korrtext' [,ID='ident'][,CONTROL=zahl] |

segment gibt den Namen des Segmentes an, das korrigiert werden soll. Korrigiert werden kann nur innerhalb eines Segments, nicht aber über Segmentgrenzen hinweg. segment darf höchstens 8 Zeichen lang sein.

Der hier angegeben Segmentname hat Vorrang vor dem Namen, der bei *SEG vereinbart ist. Mit %ROOT wird das Grundsegment ausgewählt. Wird dieser Operand nicht angegeben, gilt die Angabe bei *SEG.

basisadr legt die Basisadresse (sedezimal) fest. Diese Basisadresse gilt nur für dieses *COR. Wird basisadr nicht angegeben, gilt der Wert, der bei *BAS angegeben ist.

0 ≤ basisadr ≤ 7FFFFFFF

adresse legt die relative Adresse fest.
basisadr + adresse ergeben die absolute Adresse im Lademodul.

0 ≤ basisadr+adresse ≤ 7FFFFFFF

C'prüftext' gibt den Prüftext zeichenweise an. Ein Hochkomma im Text muß doppelt angegeben werden.
prüftext darf höchstens 50 Zeichen lang sein.

X'prüftext' gibt den Prüftext sedezimal an.
prüftext darf höchstens 50 Byte lang sein.

=:= Prüftext und Korrekturtext müssen gleich lang sein.

= Prüftext und Korrekturtext dürfen verschieden lang sein.

C'korrtext' gibt den Korrekturtext zeichenweise an. Ein Hochkomma im Text muß doppelt angegeben werden.
korrtext darf höchstens 50 Zeichen lang sein.

X'korrtext' gibt den Korrekturtext sedezimal an.
korrtext darf höchstens 50 Byte lang sein.

ID='ident' gibt eine Identifizierung zeichenweise an.
ident darf maximal 8 Zeichen lang sein.

Diese Identifizierung gilt nur für dieses *COR. Wird dieser Operand nicht angegeben, gilt die Angabe bei *ID.

CONTROL=zahl
legt eine lokale Kontrollzahl (sedezimal) fest.

0 ≤ zahl ≤ FFFF

*DEL Löschen von Korrekturjournalsätzen

*DEL schließt die Korrekturjournalsätze (TXTP) aus dem Eingabeelement aus.

| Operation | Operanden |
|---|---|
| *DEL | TXTP[,ID='ident'] |

ID='ident' gibt eine Identifizierung zeichenweise an.
ident darf maximal 8 Zeichen lang sein.

Diese Identifizierung gilt nur für dieses *DEL. Wird dieser Operand nicht angegeben, gilt die Angabe bei *ID.

*END Beenden der Korrektureingaben

*END schließt die Folge der Korrekturanweisungen ab. Anschließend prüft LMS alle Anweisungen auf Durchführbarkeit und führt die Anweisungsfolge aus.

| Operation | Operanden |
|---|---|
| *END | |

*ID Festlegen der Identifikation

*ID legt globale Identifikation fest. Sie gilt für alle Anweisungen, bei denen die lokale Identifikation nicht angegeben ist.

| Operation | Operanden |
|---|---|
| *ID | ['ident'] |

'ident' gibt die globale Identifikation zeichenweise an.
ident darf höchstens 8 Zeichen lang sein.

Wird der Operand nicht angegeben, werden als Standardwert 8 Leerzeichen angenommen.

*REM Zurücknehmen von Korrekturen

*REM macht alle Textkorrekturen bzw. Textkorrekturen einer bestimmten Identifikation, für die ein Korrekturjournalsatz existiert, wieder rückgängig. Die Korrekturjournalsätze werden anschließend gelöscht.

| Operation | Operanden |
|---|---|
| *REM | [ID='ident'] |

ID='ident' gibt die lokale Identifizierung zeichenweise an.
ident darf maximal 8 Zeichen lang sein.

Wird dieser Operand nicht angegeben, gilt die Angabe bei *ID.

*SEG Festlegen eines Segments

*SEG legt ein Segment eines Lademoduls fest, das mit nachfolgendem *COR korrigiert werden soll.

| Operation | Operanden |
|---|---|
| *SEG | {segment / %ROOT} |

segment gibt den Namen des Segments an, das korrigiert werden soll.
segment darf höchstens 8 Zeichen lang sein.

%ROOT legt fest, daß das Grundsegment korrigiert werden soll.

**Format 3: Korrigieren von LLMs**

| Operation | Operanden |
|---|---|
| UPDL | elem[(lib)][>elemu] |

UPDL Name der Anweisung mit dem Elementtyp L.

elem vollständige Elementbezeichnung des zu korrigierenden Elementes oder eine Auswahlangabe.

lib Kurzbezeichnung der Eingabebibliothek

elemu Elementbezeichnung des Ausgabeelementes oder Konstruktionsangabe.

**Verarbeitungsoperanden**

CSECT bestimmt die Basis für das Displacement in der *COR-Subanweisung

OVERWRITE bestimmt das Überschreiben gleichnamiger Elemente in der Ausgabebibliothek.
Dieser Verarbeitungsoperand wird jedoch nicht ausgewertet, wenn

```
Eingabebibliothek=Ausgabebibliothek und
                  elem=elemu.
```

Das Eingabeelement wird dann überschrieben.

PATH bestimmt die Basis für das Displacement in der *COR-Subanweisung

SLICE bestimmt die Basis für das Displacement in der *COR-Subanweisung

STRIP bestimmt, welche Satzarten beim Korrigieren ausgeschlossen werden.

## Korrekturanweisungen für UPDL

### Übersicht

| Korrekturanweisung | Funktion |
|---|---|
| *COR [csect,]displ,<br>[[{C / X / B}]'prüftext'=[:=]][{C / X / B}]'korrtext'<br>[,ID='ident'][,CONTROL=zahl] | Korrigieren von Textsätzen |
| *DEL TXTP,[ID='ident'] | Löschen von Korrekturjournalsätzen |
| *END | Beenden der Korrektureingaben |
| *ID ['ident'] | Festlegen der Identifikation |
| *REM [ID='ident'] | Zurücknehmen von Korrekturen |

## Beschreibung der Korrekturanweisungen für LLMs

*COR Korrigieren von Textsätzen

*COR korrigiert Textsätze eines LLM

| Operation | Operanden |
|---|---|
| *COR | [csect,]displ,<br>[[{C / X / B}]'prüftext'=[:=]][{C / X / B}]'korrtext'<br>[,ID='ident'][,CONTROL=zahl] |

csect 32 Zeichen langer CSECT-Name.
Wird CSECT angegeben, werden alle CSECT's mit dem angegebenen Namen korrigiert.

Die Priorität in der Auswertung der Namen ist wie folgt:

1. in *COR wurde csect angegeben: der Verarbeitungsoperand CSECT wird ignoriert
2. PAR CSECT=xxx: es darf nur der Verarbeitungsoperand PATH oder SLICE gesetzt sein
3. PAR PATH=xxx: der Verarbeitungsoperand SLICE darf nicht gesetzt sein
4. PAR SLICE=xxx: der Verarbeitungsoperand PATH darf nicht gesetzt sein

displ legt die relative Adresse fest.
displ ergibt die absolute Adresse im LLM bzw. die CSECT-relative Adresse, falls csect angegeben ist.

0 ≤ displ ≤ 7FFFFFFF

C'prüftext' gibt den Prüftext zeichenweise an. Ein Hochkomma im Text muß doppelt angegeben werden.
prüftext darf höchstens 50 Zeichen lang sein.

X'prüftext' gibt den Prüftext sedezimal an. prüftext darf höchstens 50 Byte lang sein.

B'prüftext' gibt den Prüftext binär an. prüftext darf höchstens 50 Zeichen lang sein.

Der Originaltext zum Vergleich mit dem Prüftext wird aus den für diesen Bereich existierenden TXT-Sätzen gebildet. Wenn es mehrere Texte für dieselbe Adresse gibt, gilt der letzte Text.

=:= Prüftext und Korrekturtext müssen gleich lang sein.

= Prüftext und Korrekturtext dürfen verschieden lang sein.

C'korrtext' gibt den Korrekturtext zeichenweise an. Ein Hochkomma im Text muß doppelt angegeben werden. korrtext darf höchstens 50 Zeichen lang sein.

X'korrtext' gibt den Korrekturtext sedezimal an. korrtext darf höchsens 50 Byte lang sein.

B'korrtext' gibt den Korrekturtext binär an. korrtext darf höchstens 50 Zeichen lang sein.

ID='ident' gibt eine Identifizierung an. ident darf maximal 8 Zeichen lang sein.

Diese Identifizierung gilt nur für dieses *COR. Wird dieser Operand nicht angegeben, gilt die Angabe bei *ID.

CONTROL = zahl

legt eine lokale Kontrollzahl (sedezimal) fest.

Die Kontrollzahl wird von LMS für jede *COR-Anweisung errechnet.

$0 \leq$ zahl $\leq$ FFFF

*DEL Löschen von Korrekturjournalsätzen

| Operation | Operanden |
|---|---|
| *DEL | TXTP[,ID='ident'] |

TXTP Es sollen Korrekturjournalsätze aus dem Eingabeelement ausgeschlossen werden.

ID='ident' gibt eine Identifizierung an.
ident darf maximal 8 Zeichen lang sein.

Diese Identifikation gilt nur für dieses *DEL. Wird dieser Operand nicht angegeben, gilt die Angabe *ID.

*END Beenden der Korrektureingaben

*END schließt die Folge der Korrekturanweisungen ab. Anschließend prüft LMS alle Anweisungen auf Durchführbarkeit und führt die Anweisungsfolge aus.

| Operation | Operanden |
|---|---|
| *END | |

*ID Festlegen der Identifikation

*ID legt die globale Identifikation fest. Sie gilt für alle Anweisungen, bei denen die lokale Identifikation nicht angegeben ist.

| Operation | Operanden |
|---|---|
| *ID | ['ident'] |

'ident' gibt die globale Identifikation an.
ident darf höchstens 8 Zeichen lang sein.

Wird der Operand nicht angegeben, werden als Standardwert 8 Leerzeichen angenommen.

*REM Zurücknehmen von Korrekturen

*REM macht alle Textkorrekturen bzw. Textkorrekturen einer bestimmten Identifikation, für die ein Korrekturjournalsatz existiert, wieder rückgängig. Die Korrekturjournalsätze werden anschließend gelöscht.

| Operation | Operanden |
|---|---|
| *REM | [ID='ident'] |

ID='ident' gibt die lokale Identifizierung an.
ident darf maximal 8 Zeichen lang sein.

Wird dieser Operand nicht angegeben, gilt die Angabe bei *ID.

# USE Verzweigen in Benutzerprogramme

Mit USE verzweigt LMS vor der Bearbeitung eines Eingabesatzes in eine Benutzerroutine.

Die Funktion ist erlaubt

- beim Auflisten von Elementen vom Typ R, S, M, P, J, D und X mit LST und
- beim Vergleichen von Elementen vom Typ S, M, P, J, D, X mit COM.

Folgende Aktionen sind vor der Bearbeitung des Elementsatzes durch LMS möglich:

- den aktuellen Elementsatz zu verändern
- Sätze durch die Benutzerroutine einzufügen
- den aktuellen Elementsatz von der Bearbeitung auszuschließen

Der Benutzerroutine werden Beginn und Ende des Elementes mitgeteilt, somit können auch vor dem ersten und nach dem letzten Elementsatz vom Benutzer Sätze eingefügt werden.

Für COM werden 2 Benutzerausgänge angeboten, einen für das Primärelement, einen für das Sekundärelement. Die Benutzerausgänge für LST und COM können gleichzeitig definiert werden.

Werden mehrere USE mit dem gleichen Benutzerausgang angegeben, gilt immer die zuletzt angegebene.

Zusätzlich können die Bibliotheken definiert werden, die LMS für den Anschluß der Unterprogramme EDT, EDOR und FMS benutzen soll.

| Operation | Operanden |
|---|---|
| USE | [ [LST / COMP / COMS]=[ [entry / [* / bibliothekb] (entry)] ] ] [ [EDTLIB / EDORLIB / FMSLIB]=bibliotheku ] [ ? ] |

| | |
|---|---|
| USE | Name der Anweisung. |
| LST | Benutzerausgang für die LST-Funktion. |
| COMP | Benutzerausgang für die COM-Funktion bei der Verarbeitung des Primärelementes. |
| COMS | Benutzerausgang für die COM-Funktion bei der Verarbeitung des Sekundärelementes. |
| entry | maximal 8 Zeichen langer Name des Einsprungpunktes der Benutzerroutine.<br>entry darf nicht mit der Zeichenfolge 'LMS' beginnen. |
| * | Die Benutzerroutine wird aus dem EAM-Bereich nachgeladen. |
| bibliothekb | maximal 54 Zeichen langer Name der Bibliothek, in der die Benutzerroutine abgelegt ist. |
| EDTLIB | Benutzerausgang für das Unterprogramm EDT. |
| EDORLIB | Benutzerausgang für das Unterprogramm EDOR. |
| FMSLIB | Benutzerausgang für das Unterprogramm FMS. |
| bibliotheku | maximal 54 Zeichen langer Name der Bibliothek, in der das entsprechende Unterprogramm zu suchen ist. |
| ? | Der aktuelle Wert wird protokolliert. |

**Nachladen des Benutzerprogramms**

Das Benutzerprogramm wird erst bei der ersten Benutzung aus der Bibliothek oder dem EAM-Bereich nachgeladen. Es wird immer in den benutzereigenen Klasse-6-Speicher geladen. Fehlt die Angabe * oder bibliothek, wird das Benutzerprogramm zuerst in einer eventuell vorhandenen privaten Tasklib gesucht (mit /SET-TASKLIB LIBRARY=bibliothek zugewiesen) und dann in der Tasklib des Systems $TASKLIB. Das gleiche gilt, wenn * oder bibliothek angegeben wurde, jedoch das Benutzerprogramm dort nicht gefunden wird.

**Deaktivieren des Benutzerausgangs**

Fehlen die Angaben zum Einsprungspunkt und zur Bibliothek, deaktiviert LMS den Benutzerausgang. Er wird bei dem nächsten entsprechenden LST oder COM nicht mehr benutzt. Die Benutzerroutine wird jedoch noch nicht entladen, da sie eventuell noch anderweitig benötigt wird. Das bedeutet, daß bei dem nächsten USE mit demselben Einsprungpunkt das Benutzerprogramm nicht neu dazugebunden werden muß.

**Schnittstelle des Benutzerausgangs**

Registerkonventionen

Beim Aufruf des Benutzerprogramms werden von LMS folgende Registerkonventionen berücksichtigt:

Register 1: Adresse einer Parameterleiste
Register 13: Adresse des Sicherstellungsbereiches (18 Worte)
Register 14: Rücksprungadresse
Register 15: Adresse des Einsprungpunktes

Aufbau der Parameterleiste

Die Parameterleiste besteht aus 5 Worten:

```
DC A(Auftragsbeschreibung)
DC A(Antwortbeschreibung)
DC A(zu übergebender Satz)
DC A(Bibliotheksname)
DC A(Elementbezeichnung)
```

Die ersten beiden Adressen werden von LMS versorgt, d.h. insbesondere, daß der Benutzer die Antwortbeschreibung nur an der von LMS vorgegebenen Adresse abliefern darf. Die Satzadresse kann sowohl von LMS als auch vom Benutzer versorgt werden.

1.Wort: Auftragsbeschreibung

Die Auftragsbeschreibung an das Benutzerunterprogramm besteht aus 3 Byte und kann folgenden Inhalt haben:

C'BOE' Beginn des Elementes
C'REC' Satz steht zur Verarbeitung an
C'EOE' Ende des Elementes

2.Wort: Antwortbeschreibung

Die Antwortbeschreibung vom Benutzerunterprogramm besteht aus 3 Byte und kann folgenden Inhalt haben:

C'CON' LMS verarbeitet den Satz, dessen Adresse nach Rückkehr vom Benutzerprogramm in der Parameterleiste steht und bietet danach dem Benutzerprogramm den nächsten Elementsatz oder EOE an.

C'DEL' LMS übergeht den zuletzt angebotenen Elementsatz und springt mit dem nächsten Elementsatz oder EOE erneut zum Benutzerunterprogramm.

C'INS' LMS verarbeitet zunächst den vom Benutzer zur Verfügung gestellten Satz und kehrt danach mit dem vorher angebotenen Elementsatz oder EOE an das Benutzerunterprogramm zurück; dies wird solange wiederholt, bis die Antwort DEL oder CON erfolgt; d.h. wenn LMS den Satz i anbietet, werden die mit INS zurückgesandten Sätze vor dem Satz i bearbeitet.

Auf die Aufträge kann wie folgt geantwortet werden:

| Auftrag | Antwort |
|---|---|
| BOE | nicht relevant |
| REC | CON, DEL, INS |
| EOE | INS, CON |

Bei falscher Antwort wird die LMS-Funktion mit einer Fehlermeldung abgebrochen.

Zur Verdeutlichung der Kommunikation zwischen LMS und dem Benutzerunterprogramm siehe Diagramm Seite 203.

3.Wort: Satz

Es enthält die Adresse des Satzes, der von LMS an das Benutzerunterprogramm übergeben wird. Diese Satzadresse wird von LMS erst beim Auftrag 'REC' versorgt. Davor ist die Satzadresse nicht versorgt, d.h. ihr Inhalt ist X'00000000'.

Kehrt das Benutzerunterprogramm mit der Antwort CON oder DEL zurück, kann die gleiche Satzadresse verwendet werden. Möchte der Benutzer Sätze einfügen, muß er einen eigenen Satzpuffer verwenden.
Die zwischen LMS und dem Benutzerunterprogramm ausgetauschten Sätze beginnen mit einem 4 Byte langen Satzlängenfeld.
Ist der Verarbeitungsoperand FORMAT=P gesetzt, muß zum Auflisten mit LST das 5. Byte ein gültiges Vorschubsteuerzeichen sein.

4.Wort: Bibliotheksname

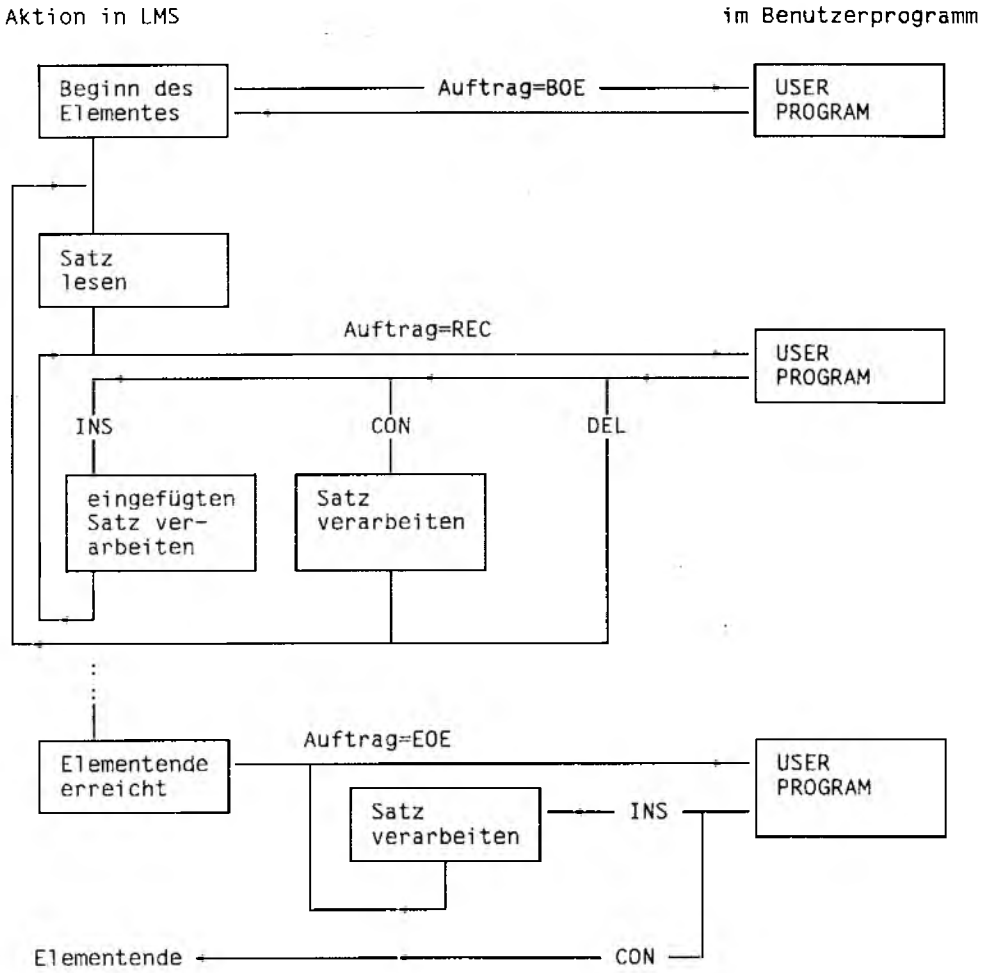
Dieses Wort enthält die Adresse des Bibliotheksnamens. Der Bibliotheksname beginnt mit einem 2 Byte langen Satzlängenfeld.

5.Wort: Elementbezeichnung

Dieses Wort enthält die Adresse der Elementbezeichnung. Die Elementbezeichnung hat die Form: 2 Byte langes Satzlängenfeld, gefolgt von

```
(Typ)Elementname/Version[(Variantennummer)]/Datum
```

Diagramm:




Ein Beispiel zum Benutzerausgang siehe Seite 287.

# $ Ausgeben des Anweisungspuffers

Die LMS-Anweisungen können im Block-Modus eingegeben werden. Das bedeutet, daß sie an der Datensichtstation nicht einzeln zur Verarbeitung abgeschickt werden müssen, sondern, daß für mehrere Anweisungen gemeinsam die Datenübertragung gestartet werden kann (siehe Seite 84, Anweisungen geblockt eingeben).

Im Block-Modus werden die Anweisungen in einen Anweisungspuffer geschrieben. Jede Folge von Anweisungen (Ausnahme: $) führt zur Veränderung des zwischengespeicherten Anweisungspuffers.

$ protokolliert den letzten Anweisungspuffer, er kann an der Datensichtstation verändert und wieder eingegeben werden.

| Operation | Operanden |
|---|---|
| $ | |

$ Name der Anweisung.

Nach einem Syntax-Fehler oder einem anderen schwerwiegenden Fehler wird im Dialogbetrieb die Bearbeitung des aktuellen Eingabepuffers beendet. LMS bleibt im RUN-Modus und erwartet neue Anweisungen von der Datensichtstation.

Mit $ kann sich der Benutzer nun den gesamten Anweisungspuffer ausgeben lassen, bereits ausgeführte Anweisungen entfernen, die fehlerhafte Anweisung korrigieren und den Anweisungspuffer neu eingeben.

# Verarbeitungsoperanden

Verarbeitungsoperanden beeinflussen den Ablauf von LMS.

Soll sich ein Verarbeitungsoperand auf eine Funktion auswirken, ist er vor der entsprechenden Anweisung zu setzen.

Verarbeitungsoperanden werden mit PAR gesetzt:

| Operation | Operanden |
|---|---|
| PAR | [ [ parname=[ { parwert } ] ] [ ? ] ] ,[...] |

Auf der Seite 207ff finden Sie eine tabellarische Übersicht über die Verarbeitungsoperanden. Die Verarbeitungsoperanden sind ab Seite 211ff alphabetisch geordnet.

Die Namen der Verarbeitungsoperanden und Operandenwerte dürfen gekürzt werden, solange die Abkürzung innerhalb aller Verarbeitungsoperanden eindeutig ist. BASE darf z.B. bis auf B gekürzt werden, STRING nur bis auf STRIN (STRIP!).

Folgende Verarbeitungsoperanden beeinflussen den Ablauf von LMS, jedoch keine einzelnen Funktionen:

| | |
|---|---|
| ERRCONS | Fehlermeldungen auf der Datensichtstation |
| TERMINATE | Abbruchverhalten im Fehlerfall |
| NEWFORM | Seitensteuerung für Protokolle |
| LCASE | Klein-, Großschreibung |
| LINE | Zeilenzahl pro Protokollseite |

Folgende Verarbeitungsoperanden beeinflussen sowohl den gesamten LMS-Lauf als auch einzelne Funktionen:

TEST — Testlauf, ohne Bibliotheksänderung; beeinflußt zusätzlich zum gesamten LMS-Lauf die Funktion RST.

LOG — Protokollierung der Anweisungen; beeinflußt zusätzlich das Protokoll der Funktionen COR und NUM.

Die übrigen Verarbeitungsoperanden beeinflussen nur einzelne Funktionen. Den Funktionsbeschreibungen ist zu entnehmen, welche Operanden die jeweilige Funktion beeinflussen. Einen tabellarischen Überblick über die Wirkung der Verarbeitungsoperanden finden Sie auf Seite 63.

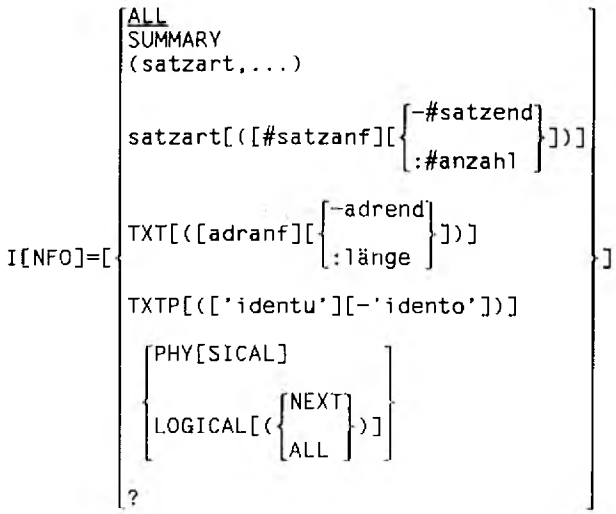
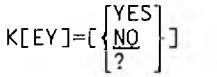
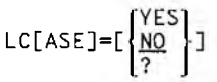
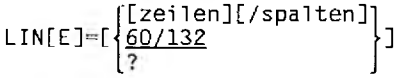
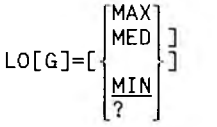
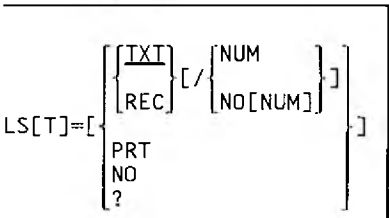
Wird PAR ohne Verarbeitungsoperand angegeben, werden alle Verarbeitungsoperanden auf ihre Standardwerte gesetzt.

Wird ein Verarbeitungsoperand ohne Operandenwert angegeben (z.B. PAR B=), wird der jeweilige Verarbeitungsoperand auf seinen Standardwert zurückgesetzt.

# Tabelle aller Verarbeitungsoperanden

Die nachfolgende Tabelle zeigt in alphabetischer Reihenfolge eine Übersicht aller Verarbeitungsoperanden.

| Verarbeitungsoperand | Anwendungsbereich |
|---|---|
| B[ASE]=[{[basisadr] / 0 / ?}] | Festlegen einer Basisadresse |
| CH[ECK]=[{[[anf][/lge]] / NO / ?}] | Definieren des Kennungsfeldes in Eingabesätzen |
| COM[PARE]=[{[[[anf]/lge][/vergleich[zahl]][/liste][/COR]] / 1/72/L/MED / ?}] | Steuern der Vergleichsfunktion |
| CS[ECT]=[{[name] / ?}] | Festlegen der CSECT |
| DES[TROY]=[{[YES] / NO / ?}] | Steuern des physikalischen Löschens |
| E[RRCONS]=[{[NO / 2 / 4] / ALL / YES / ?}] | Ausgeben der Meldungen nach SYSOUT |
| FC[BTYPE]=[{[ISAM / SAM / STD / CAT] / ?}] | Festlegen des FCB-Typs der Ausgabedatei |
| FO[RMAT]=[{[C / X / SYMBOLIC / XC / REC / P] / ?}] | Festlegen der Satzdarstellung |

| Verarbeitungsoperand | Anwendungsbereich |
|---|---|
|  | Festlegen des Ausgabeumfangs |
|  | Übernehmen des ISAM-Schlüssels und anderer Dateieigenschaften |
|  | Umsetzen Klein-/ Großbuchstaben |
|  | Festlegen der Zeilen- und Spaltenanzahl pro Protokollseite |
|  | Protokollieren der Anweisungen |
|  | Festlegen des Umfangs und der Art der Auflistung von Elementen |

| Verarbeitungsoperand | Anwendungsbereich |
|---|---|
| N[EWFORM]=[{YES / NO / 3 / zahl / ?}] | Steuern des Formularvorschubs |
| O[VERWRITE]=[{YES / ONLY / NO / V / D / EXTEND / ?}] | Überschreiben gleichnamiger Elemente |
| PA[TH]=[{name / ?}] | Festlegung des Pfadnamens |
| PH[ASE]=[{PK / NK / ?}] | Festlegung des Phasenformats |
| RA[NGE]=[{[anf][/lge] / NO / ?}] | Definieren des Kennungsfeldes in Ausgabesätzen |
| RE[FERENCE]=[{name / ([name],[{CSECT / ENTRY / ALL}]) / ?}] | Vereinbaren von Referenzbedingungen |
| SE[GMENT]=[{overlay / %ROOT / %ALL / ?}] | Festlegen von Segmenten eines Lademoduls |
| SL[ICE]=[{name / ?}] | Festlegung des Slices |
| SO[RT]=[{U / [N][V][D] / R / ?}] | Sortieren des Inhaltsverzeichnisses |

| Verarbeitungsoperand | Anwendungsbereich |
|---|---|
| STRIN[G]=[{'zeichenfolge' / NAME / KEY / <u>NO</u> / ?}] | Definieren einer Zeichenfolge im Kennungsfeld der Ausgabesätze |
| STRIP=[{satzart / (satzart,...) / YES / <u>NO</u> / ?}] | Unterdrücken von Sätzen |
| SU[M]={YES / <u>NO</u> / ?} | Erzeugen der Vergleichsstatistik |
| TER[MINATE]=[{1 / <u>2</u> / 3 / 4 / 5 / 6 / 7 / ?}] | Abbruchverhalten im Fehlerfall |
| TES[T]=[{YES / <u>NO</u> / ABORT / ?}] | Ein-/Ausschalten und Beenden des TEST-Modus |
| TO[C]=[{F / D / L / <u>I</u> / ?}] | Steuern des Ausgabeformats für Inhaltsverzeichnisse von Programmbibliotheken |
| TY[PE]=[{typ / ?}] | Vorbesetzen des Elementtyps |
| V[ALUE]=[{[anf][/stp] / <u>NO</u> / ?}] | Numerieren im Kennungsfeld der Ausgabesätze |

# PAR BASE Festlegen einer Basisadresse

Der Verarbeitungsoperand BASE legt eine Basisadresse fest. Auf diese Basisadresse beziehen sich die Bereichsadressen, die im Verarbeitungsoperanden INFO angegeben werden. Die Basisadresse wird zur angegebenen Bereichsadresse addiert.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | B[ASE]=[{basisadr / 0 / ?}] |

basisadr legt die Basisadresse sedezimal fest:

$0 \leq$ basisadr $\leq$ 7FFFFFFF

Werden im Verarbeitungsoperanden INFO keine Bereichsadressen angegeben, wird der Verarbeitungsoperand BASE ignoriert.

? Der aktuelle Wert wird protokolliert.

Der Verarbeitungsoperand BASE beeinflußt LST.

# PAR CHECK Definieren des Kennungsfeldes in Eingabesätzen

Der Verarbeitungsoperand CHECK definiert Lage und Länge des Kennungsfeldes in Eingabesätzen. Dieser Bereich wird auf aufsteigenden Inhalt überprüft. Bei der Ausgabe von Elementen in ISAM-Dateien können die Satzkennungen als ISAM-Schlüssel abgelegt werden.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | CH[ECK]=[{[anf][/lge] / NO / ?}] |

anf — Anfangsposition des Kennungsfeldes im Eingabesatz.
anf kann folgende Werte haben:
$1 \leq anf \leq 251$

lge — Länge des Kennungsfeldes.
lge kann folgende Werte haben:
$1 \leq lge \leq 16$

Falls nur anf angegeben wird, muß anf aus dem Bereich $65 \leq anf \leq 80$ sein, da sich lge aus lge = 81 - anf errechnet.

Falls nur lge angegeben wird, errechnet sich anf aus:
anf = 81 - lge

Falls anf und lge zusammen angegeben werden, muß gelten:
$anf + lge \leq 252$

NO — In den Eingabebereichen wird kein Kennungsfeld definiert.

? — Der aktuelle Wert wird protokolliert.

Der Verabeitungsoperand CHECK beeinflußt folgende Anweisungen:
ADD, COM, COR, EDR, EDT, NUM, SEL

*Hinweis*

Der Verarbeitungsoperand CHECK wirkt nur beim Kreuzvergleich auf COM.

# PAR CSECT Vereinbaren eines CSECT-Namens

Dieser Verarbeitungsoperand dient zum Festlegen eines CSECT-Namens für UPDL und LSTL.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | CS[ECT]=[{name / ?}] |

name max. 32 Zeichen langer CSECT Name

Bei LSTL wird dann nur die CSECT mit diesem Namen aufgelistet.

Bei UPDL dient der Name als Basis für das Displacement in der *COR-Subanweisung.

? Der aktuelle Wert wird protokolliert.

# PAR COMPARE Steuern der Vergleichsfunktion

Dieser Verarbeitungsoperand bestimmt den Vergleichsbereich, die Art des Vergleichs und das Protokollformat bei COM. Für den Vergleich stehen zwei Algorithmen zur Verfügung, zum einen der Heckel-Algorithmus (Standard ab der LMS-Version V1.3A) und wahlweise auch der Kreuzvergleich.
Beim Heckel-Algorithmus findet erst nach dem Einlesen aller Sätze ein Vergleich statt und nicht ein stückweiser Vergleich. Außerdem wird nur der Vergleichsbereich der Sätze protokolliert und nicht der komplette Satz wie beim Kreuzvergleich. Durch die unterschiedlichen Verfahrensweisen können die Vergleichsergebnisse verschieden sein.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | COM[PARE]=[{[[anf]/lge][/vergleich[zahl]][/liste][/COR] / 1/72/L/MED / ?}] |

anf — Anfangsposition des Vergleichsfeldes.
anf kann folgende Werte haben:

Kreuzvergleich: $1 \leq anf \leq 251$
Heckel-Algorithmus: $1 \leq anf \leq 32764$

lge — Länge des Vergleichsfeldes.
lge kann folgende Werte haben:

Kreuzvergleich: $1 \leq lge \leq 251$
Heckel-Algorithmus: $1 \leq lge \leq 32764$

Die Werte anf und lge dürfen jedoch zusammen nicht mehr als 252 (Kreuzvergleich) und 32765 (Heckel-Algorithmus) ergeben.

vergleich — Art des Vergleichs:

L logischer Vergleich
Die Vergleichsfelder werden Zeichen für Zeichen verglichen, Zwischenräume werden übergangen.

F formaler Vergleich
Die Vergleichsfelder werden zuerst auf gleiche Länge überprüft, sind die Längen unterschiedlich, werden die Sätze als ungleich protokolliert.
Haben sie gleiche Länge, dann werden die Felder vollständig verglichen.

zahl

Synchronisierungszähler.
zahl kann folgende Werte haben:
$1 \leq$ zahl $\leq 9$

zahl gibt die Anzahl der Sätze an, die mindestens übereinstimmen müssen, damit ein Synchronisierungsversuch als erfolgreich gilt (vgl. Seite 107). Werden nach ungleichen Sätzen weniger als zahl gleiche Sätze erkannt, werden diese als ungleich angenommen und als ungleich protokolliert.

*Hinweis*

Wird für zahl ein Wert angegeben, schaltet LMS auf den Kreuzvergleich um. Mit PAR COM= kann man wieder auf den Heckel-Algorithmus zurückschalten (siehe Seite 50).

liste

Umfang der Protokollierung.

MAX Ausführliches Vergleichsprotokoll.
Alle Sätze werden protokolliert.
Die Vergleichsstatistik wird ausgegeben.

MED Standard-Vergleichsprotokoll. Ungleiche Sätze werden vollständig protokolliert. Bei gleichen Sätzen werden nur Bereichsangaben (Satznummern und evtl. Satzkennungen) protokolliert.
Die Vergleichsstatistik wird ausgegeben.

MIN Minimales Vergleichsprotokoll.
Für gleiche und ungleiche Sätze werden nur Bereichsangaben (Satznummern und evtl. Satzkennungen) protokolliert.
Die Vergleichsstatistik wird ausgegeben.

SUM Kein Vergleichsprotokoll.
Nur die Vergleichsstatistik wird ausgegeben.

NO Keine Protokollierung, weder Vergleichsprotokoll noch Vergleichsstatistik.
NO ist nur sinnvoll bei Verwendung des Verarbeitungsoperanden SUM.

COR

Beim Vergleich von Textelementen erzeugt LMS aus den Sätzen INS und DEL des Vergleichsprotokolls Korrekturanweisungen für COR.

Die Korrekturanweisungen werden nach SYSOPT ausgegeben. Sollen die Korrekturanweisungen nicht auf Lochkarten ausgegeben werden, ist folgendes Kommando abzusetzen:

```
/ASSIGN-SYSOPT TO-FILE=datei
```

datei kann in diesem Fall wieder als Prozedurelement in eine Bibliothek aufgenommen und als Prozedur gestartet werden.

Ein Beispiel dieser Funktion siehe Seite 271.

? Der aktuelle Wert wird protokolliert.

Der Verarbeitungsoperand COMPARE beeinflußt COM.

# PAR DESTROY Steuern des physikalischen Löschens

Der Verarbeitungsoperand DESTROY steuert, ob beim Löschen zusätzlich zur Freigabe des Speicherplatzes physikalisch gelöscht werden soll, d.h. die zu löschenden Daten werden mit binären Nullen überschrieben.

Nur Elemente in Programmbibliotheken und die von EDT bzw. EDOR erzeugten Hilfsdateien können physikalisch gelöscht werden.

Für die volle Wirksamkeit dieses Löschens muß der Operand bereits beim Aufnehmen des Elementes gesetzt werden. Wird er später gesetzt, werden erst die ab diesem Zeitpunkt erzeugten Varianten physikalisch gelöscht.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | DES[TROY]=[{YES / NO / ?}] |

YES Alle zu löschenden Daten werden physikalisch gelöscht, d.h. mit binären Nullen überschrieben, bevor der Speicherplatz freigegeben wird.

Dies bedeutet, daß

- beim Löschen von Voll-Elementen in Programmbibliotheken physikalisch gelöscht wird; bei Delta-Elementen jedoch erst, wenn das letzte Delta-Element eines Baums, d.h. der komplette Delta-Baum gelöscht wird.
- beim Schreiben von Elementen in Programmbibliotheken ein Kennzeichen für physikalisches Löschen eingetragen wird, d.h. wenn solche Elemente später gelöscht werden, werden sie physikalisch gelöscht.
- bei Rückkehr aus dem EDT/EDOR dabei eventuell erzeugte Hilfsdateien physikalisch gelöscht werden.

Dieser Operandenwert wirkt nur auf Elemente von Programmbibliotheken und auf die von EDT bzw. EDOR erzeugten Hilfsdateien. Bei den übrigen Bibliotheken wirkt er wie NO.

NO Bei allen zu löschenden Daten wird nur der Speicherplatz freigegeben.

? Der aktuelle Wert wird protokolliert.

Der Verarbeitungsoperand DESTROY beeinflußt:
ADD, COR, DEL, DUP, EDR, EDT, NAM, NUM, PRT, UPD

# PAR ERRCONS Ausgeben der Meldungen nach SYSOUT

Der Verarbeitungsoperand ERRCONS bestimmt, welche LMS-Meldungen zusätzlich in die Systemdatei SYSOUT ausgegeben werden sollen, wenn das Ablaufprotokoll nicht in die Systemdatei SYSOUT ausgegeben wird (d.h. bei PRT (LST) oder PRT elem(lib)).

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | E[RRCONS]=[{NO / 2 / 4 / ALL / YES / ?}] |

NO gibt keine Meldungen zusätzlich in die Systemdatei SYSOUT aus.

2 gibt Fehlermeldungen aus.

4 gibt Fehlermeldungen und NOT FOUND-Meldungen aus.

ALL gibt Fehlermeldungen, NOT FOUND-, EXISTING- und NO OLD-Meldungen aus.

YES wirkt wie ALL und wird nur noch aus Kompatibilitätsgründen unterstützt.

? Der aktuelle Wert wird protokolliert.

Der Verarbeitungsoperand ERRCONS beeinflußt den gesamten LMS-Lauf.

# PAR FCBTYPE Festlegen des FCB-Typs der Ausgabedatei

Der Verarbeitungsoperand FCBTYPE legt fest, mit welchem FCB-Typ die Datei erzeugt werden soll, in die ein Element ausgegeben wird. Der Verarbeitungsoperand wird nur für textartige Elemente ausgewertet, d.h. nicht bei den Elementtypen R und C.

Im FILE-Kommando vergebene Dateimerkmale haben Vorrang vor den Angaben im Verarbeitungsoperanden FCBTYPE.
Dateimerkmale werden durch PAR KEY=YES im Element mitabgespeichert.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | FC[BTYPE]=[{ISAM / SAM / STD / CAT / ?}] |

ISAM erzeugt eine ISAM-Datei mit folgenden Eigenschaften:

- Ist in den gespeicherten Dateimerkmalen der FCB-Typ ISAM eingetragen, wird die Datei entsprechend den gespeicherten Eigenschaften erzeugt.
- Ist in den gespeicherten Dateimerkmalen der FCB-Typ SAM eingetragen, wird eine ISAM-Datei mit KEY-POSITION=5 und KEY-LENGTH=8 erzeugt.
- Sind keine Dateieigenschaften gespeichert, wird eine ISAM-Datei mit KEY-POSITION=5 und KEY-LENGTH=8 erzeugt.

SAM erzeugt eine SAM-Datei. Ist in den gespeicherten Dateieigenschaften der FCB-Typ ISAM eingetragen, wird eine SAM-Datei mit bzw. ohne KEY erzeugt.

STD erzeugt eine SAM- bzw. ISAM-Datei entsprechend den gespeicherten Dateimerkmalen.

Sind keine Dateimerkmale gespeichert, wird die Datei entsprechend dem Katalogeintrag erzeugt.

Fehlt auch der Katalogeintrag, wird eine ISAM-Datei mit KEY-POSITION=5 und KEY-LENGTH=8 erzeugt.

CAT erzeugt eine SAM- bzw. ISAM-Datei entsprechend dem Katalogeintrag.

Fehlt der Katalogeintrag, wird die Datei entsprechend den gespeicherten Dateimerkmalen erzeugt. Sind keine Dateimerkmale gespeichert, wird eine ISAM-Datei mit KEY-POSITION=5 und KEY-LENGTH=8 erzeugt.

? Der aktuelle Wert wird protokolliert.

Der Verarbeitungsoperand FCBTYPE beeinflußt SEL.

# PAR FORMAT Festlegen der Satzdarstellung

Der Verarbeitungsoperand FORMAT legt die Satzdarstellung beim Auflisten von Elementen mit LST fest. Möglich ist die Satzdarstellung

- zeichenweise,
- sedezimal,
- zeichenweise und sedezimal nebeneinander,
- zeichenweise und sedezimal übereinander,
- zeichenweise, wobei das erste Zeichen des Satzinhalts als Vorschubsteuerzeichen ausgewertet wird.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | FO[RMAT]=[{C \| X \| SYMBOLIC \| XC \| REC \| P \| ?}] |

C zeichenweise Darstellung.

X sedezimale Darstellung, nur für Elemente vom Typ

- R und C
- X, falls in diesen Elementen PAM-Dateien archiviert sind.

  In die Systemdatei SYSLST werden 2*4 4 Byte-Blöcke und in die Systemdatei SYSOUT 2*3 4 Byte-Blöcke ausgegeben.

Bei allen anderen Elementen wirkt dieser Operandenwert wie der Operandenwert REC.

SYMBOLIC stellt die Sätze für die verschiedenen Elementtypen jeweils verschieden dar:

| *Elementtyp* | *Satzdarstellung* |
|---|---|
| S, M, J, D, X | zeichenweise |
| P | zeichenweise, wobei das erste Zeichen jedes Satzes als Vorschubsteuerzeichen ausgewertet wird. |

R, L — Die LMS bekannten ESD-, ISD-, RLD-, TXT-, TXTP-, REP- und END-Informationen werden in aufbereiteter Form ausgegeben. Sonstige Informationen wie z.B. LSD und DSDD werden unaufbereitet ausgegeben, d.h. es wird das Satzlängenfeld und ggf. die Satznummer mit ausgegeben. Fortlaufende Textinformationen werden nicht aufgeteilt.

C — zeichenweise und sedezimal nebeneinander.

XC — zeichenweise und sedezimale Darstellung nebeneinander, nur für Elemente vom Typ

- R, L und C
- X, falls in diesen Elementen PAM-Dateien archiviert sind.

Bei allen anderen Elementen wirkt dieser Operandenwert wie der Operandenwert REC.

REC — zeichenweise und sedezimale Darstellung übereinander, d.h. für jeden Elementssatz werden zwei Zeilen ausgegeben, mit der zeichenweisen Darstellung in der ersten Zeile und der sedezimalen Darstellung in der zweiten Zeile.

P — zeichenweise Darstellung, wobei das erste Zeichen des Inhalts jedes Satzes als Vorschubsteuerzeichen ausgewertet wird.

Diese Art der Ausgabe ist nur für druckaufbereitete Textelemente oder Protokollelemente sinnvoll.

Elemente vom Typ R, L und C werden entsprechend dem Operandenwert XC dargestellt.

? — Der aktuelle Wert wird protokolliert.

Der Verarbeitungsoperand FORMAT beeinflußt LST.

# PAR INFO Festlegen des Ausgabeumfangs

Der Verarbeitungsoperand INFO legt den Satzumfang beim Auflisten von Elementen mit LST fest. Möglich ist die Ausgabe

- aller Sätze,
- der wichtigsten Elementdaten,
- bestimmter Satzarten,
- eines bestimmten Bereichs innerhalb einer Satzart.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | I[NFO]=[ ALL \| SUMMARY \| (satzart,...) \| satzart[([#satzanf][-#satzend \| :#anzahl])] \| TXT[([adranf][-adrend \| :länge])] \| TXTP[(['identu'][-'idento'])] \| [PHY[SICAL] \| LOGICAL[([NEXT \| ALL])]] \| ? ] |

ALL alle Sätze des Elementes werden ausgegeben.

SUMMARY die wichtigsten Elementdaten werden ausgegeben, d.h.

- für textartige Elemente (Elementtyp S, M, J, P, D, X) werden tabellarisch die vorhandenen Benutzersatzarten, die durch PAR KEY=YES aufgenommenen Dateieigenschaften und die Anzahl der Sätze ausgegeben.
- für Elemente vom Typ R werden die Länge des Bindemoduls sowie Namen, Längen und Adressen der CSECT's ausgegeben.
- für Elemente vom Typ C werden die Länge des Lademoduls sowie Namen, Längen und Adressen der Segmente ausgegeben.
- für Elemente vom Typ L wird die komplette logische Struktur ausgegeben.

satzart — die angegebene Satzart (siehe Handbuch BLS [...]) eines Elementes vom Typ R, C oder L wird ausgegeben. Bei textartigen Elementen (Elementtyp S, M, J, P, D, X) wird die Angabe einer Satzart ignoriert, d.h. alle Sätze des Elementes werden ausgegeben.

Mögliche Satzarten sind

- für Elemente vom Typ R:
  ESD, ISD, LSD, TXT, RLD, TXTP, REP, INCLUDE, DSDD, REF, END

  Mit REF werden alle zum Bindemodul gehörigen Referenznamen ausgegeben.
- für Elemente vom Typ C:
  ESD, ISD, LSD, TXT, RLD, TXTP
- für Elemente vom Typ L:
  ESVD, ESVR, TXT, LRLD, TXTP

satzanf — gibt den ersten Satz an, ab dem die angegebene Satzart ausgegeben wird.
Nur für Elemente vom Typ R.

1 ≤ satzanf ≤ 2147483647

satzend — gibt den letzten Satz an, bis zu dem die angegebene Satzart ausgegeben wird.
Nur für Elemente vom Typ R.

1 ≤ satzend ≤ 2147483647

anzahl — gibt die Anzahl der Sätze der angegebenen Satzarten an, die ausgegeben werden.
Nur für Elemente vom Typ R.

1 ≤ anzahl ≤ 2147483647, wobei satzanf+anzahl≤2147483647 sein muß.

adranf — gibt die Anfangsadresse des Bereichs sedezimal an, ab dem der angegebene TXT ausgegeben wird.
Nur für Elemente vom Typ R, L und C.

0 ≤ adranf ≤ 7FFFFFFF

adrend — gibt die Endeadresse des Bereichs sedezimal an, bis zu der der angegebene TXT ausgegeben wird.
Nur für Elemente vom Typ R, L und C.

0 ≤ adrend ≤ 7FFFFFFF

| | |
|---|---|
| länge | gibt die Länge des Adreßbereichs sedezimal an, die von dem angegebenen TXT ausgegeben wird.<br>Nur für Elemente vom Typ R, L und C.<br>1 ≤ länge ≤ 7FFFFFFF |
| identu | gibt die untere Identifikationsgrenze des TXTP an. |
| identu | gibt die obere Identifikationsgrenze des TXTP an. |
| PHYSICAL | die physikalische LLM Struktur wird aufgelistet. |
| LOGICAL | die logische LLM Struktur wird aufgelistet<br>NEXT es wird nur die nächsttiefere Ebene aufgelistet.<br>ALL es wird die komplette Struktur aufgelistet. |
| ? | Der aktuelle Wert wird protokolliert. |

Der Verarbeitungsoperand INFO beeinflußt LST.

# PAR KEY
# Übernehmen von Dateieigenschaften und des ISAM-Schlüssels

Der Verarbeitungsoperand KEY legt fest, ob die Dateieigenschaften und der ISAM-Schlüssel in das Ausgabeelement mit übernommen werden sollen.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | K[EY]=[{YES / NO / ?}] |

YES — Ist PAR KEY = YES gesetzt, werden bei ADD die Dateieigenschaften wie Dateiname, ACCESS-METHOD, RECORD-FORMAT, RECORD-SIZE BUFFER-LENGTH, PADDING-FACTOR, LOGICAl-FLAG-LENGTH, VALUE-FLAG-LENGTH, PROPAGATE-VALUE-FLAG, USER-ACCESS und die ISAM-Schlüssel jeder Datei unverändert in das Ausgabeelement übernommen.

Dieser Operandenwert ist nur bei der Bearbeitung von Programmbibliotheken erlaubt, nicht dagegen bei der Bearbeitung von Quellprogrammbibliotheken.

NO — übernimmt nicht die Dateieigenschaften und die ISAM-Schlüssel. In dem Fall ist es nur möglich, ISAM-Dateien mit KEY-POSITION=5, KEY-LENGTH≤16 und RECORD-FORMAT=VARIABLE in das Ausgabeelement aufzunehmen.

? — Der aktuelle Wert wird protokolliert.

*Hinweis*

- Der Verarbeitungsoperand KEY=YES ist bei der Bearbeitung von Quellprogrammbibliotheken nicht erlaubt.
- Wird mit LST ein Element aufgelistet, das ISAM-Schlüssel gespeichert hat, werden die Werte von KEY-POSITION und KEY-LENGTH mit ausgegeben.
- Wird eine Datei mit KEY-POSITION>5 oder mit RECORD-FORMAT=FIXED aufgenommen, kann das Element, das diese Datei enthält, nicht editiert oder numeriert werden.

Der Verarbeitungsoperand KEY beeinflußt ADD.

# PAR LCASE Umsetzen Klein-/Großbuchstaben

Standardmäßig werden alle Eingaben des Benutzers von der Datensichtstation von LMS in Großbuchstaben umgesetzt. Bei COR und UPDR kann durch den Verarbeitungsoperanden LCASE das Umsetzen von Kleinbuchstaben in Großbuchstaben unterdrückt werden.

Dieser Operand wirkt nur bei Eingaben von der Datensichtstation, jedoch nicht bei Eingaben aus einem Element oder aus Prozeduren.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | LC[ASE]=[{YES / NO / ?}] |

YES Kleinbuchstaben werden nicht in Großbuchstaben umgesetzt.

NO Alle Kleinbuchstaben werden in Großbuchstaben umgesetzt.

? Der aktuelle Wert wird protokolliert.

LCASE=YES ist sinnvoll bei Operanden, für die als Operandenwert eine beliebige Zeichenfolge (auch Kleinbuchstaben) in Hochkommata zulässig ist.

Dies sind:

- bei COR die Subanweisung *CHANGE,
- bei UPDR die Subanweisungen *COR, *DEL, *ID, *INV und *REP.

Während LCASE=YES wirksam ist, müssen alle LMS-Anweisungen, Operanden und Operandenwerte (mit Ausnahme der oben genannten) in Großbuchstaben eingegeben werden.
Es empfiehlt sich daher, LCASE=YES erst direkt vor einer Korrektur einzuschalten und nach der Korrektur wieder auf LCASE=NO zurückzuschalten.

*Beispiel*

```
/START-PROGRAM $LMS
$LIB BIBLIOTHEK,BOTH
$PAR LCASE=YES
$ADDD >BRIEF.A
*Sehr geehrte ...
.
.
.
**END
$PAR LCASE=NO
$END
```

Der Text 'Sehr geehrte...' ist genau in dieser Form im Element BRIEF.A gespeichert (vergleichen Sie auch ADD, Format 4).

Der Verarbeitungsoperand LCASE beeinflußt den gesamten LMS-Lauf.

# PAR LINE
## Festlegen der Zeilen- und Spaltenanzahl pro Protokollseite

Der Verarbeitungsoperand LINE bestimmt die Länge und Breite einer von LMS erzeugten Protokollseite. Die Spaltenangabe zur Steuerung der Breite wird bei TOC, COM und LST ausgewertet. Der Verarbeitungsoperand LINE wird nur ausgewertet, wenn das Ausgabemedium SYSLST oder ein Element ist.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | LIN[E]=[{[zeilen][/spalten] / 60/132 / ?}] |

zeilen — Zeilenanzahl pro Seite
21 ≤ zeilen ≤ 255

Die Protokollseite wird spätestens nach zeilen Zeilen gewechselt (siehe PAR NEWFORM), immer jedoch wenn ein Funktionswechsel stattfindet.

spalten — Anzahl der Spalten pro Zeile
21 ≤ spalten ≤ 255

? — Der aktuelle Wert wird protokolliert.

Der Verarbeitungsoperand LINE beeinflußt:
TOC, COM, LST

# PAR LOG Protokollieren der Anweisungen

Der Verarbeitungsoperand LOG steuert den Umfang des LMS-Protokolls.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | LO[G]=[{MAX / MED / MIN / ?}] |

MAX vollständiges Protokoll.

MED Anweisungen werden nur im Fehlerfall protokolliert. Erfolgsmeldungen werden protokolliert.

MIN Es werden nur Fehlermeldungen, Schlußmeldungen und Nichterfolgsmeldungen protokolliert.

? Der aktuelle Wert wird protokolliert.

Der Verarbeitungsoperand LOG beeinflußt den gesamten LMS-Lauf und COR, NUM und UPD.

# PAR LST
## Festlegen des Umfangs und der Art der Auflistung von Elementen

Der Verarbeitungsoperand LST wird nur noch aus Kompatiblitätsgründen unterstützt. Seine Funktionen haben die Verarbeitungsoperanden FORMAT und INFO übernommen.

Beim Verarbeitungsoperanden LST wird zwischen 2 Formaten unterschieden:

### Format 1

Auflisten von Textelementen

| Operation | Operanden |
|---|---|
| PAR | LS[T]=[{[{TXT / REC}[/{NU[M] / NO[NUM]}]] / PRT / NO / ?}] |

### Format 2

Auflisten von Bindemodulen

| Operation | Operanden |
|---|---|
| PAR | LS[T]=[{[REC[/{NU[M] / NO[NUM]}]] / ALL / TXT / UPD / SYM / NO / ?}] |

Die Funktionen des Verarbeitungsoperanden LST werden folgendermaßen von den Verarbeitungsoperanden FORMAT und INFO übernommen:

| LST-Wert | FORMAT-Wert | INFO-Wert |
|---|---|---|
| REC | REC | ALL |
| ALL | SYM | ALL |
| TXT | SYM | (ESD,TXT,RLD) |
| UPD | SYM | (REP,TXTP) |
| SYM | SYM | (ESD,ISD,LSD |
| NO | SYM | SUMMARY |
| PRT | P | ALL |

**Format 1: Auflisten von Textelementen**

Dieses Format gilt für die Elementtypen S, M, J, P, D, X. Es legt den Umfang und die Art der Auflistung fest.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | LS[T]=[{[TXT] / REC}[/{NU[M] / NO[NUM]}]<br>PRT<br>NO<br>?] |

TXT — alphanumerische Ausgabe.

REC — Für jeden Elementsatz werden zwei Zeilen ausgegeben:

1. alphanumerische Ausgabe
2. sedezimale Ausgabe mit vorangestellter Ausgabe des 4 Byte langen Satzlängenfeldes

PRT — Diese Angabe gilt nur für Listenelemente (Typ P in Programmbibliotheken, Typ S in Quellprogrammbibliotheken). Sie wirkt wie der Operand LAYOUT-CONTROL(CONTROL-CHARACTERS=EBCDIC) im PRINT-FILE-Kommando (siehe Benutzer-Kommandos [7]), d.h. das erste Zeichen des Satzinhalts wird als Vorschubsteuerzeichen interpretiert. LMS prüft dieses Zeichen und ersetzt es durch X'40', falls es kein gültiges Vorschubsteuerzeichen ist.
Als gültig akzeptiert werden alle zulässigen Vorschubsteuerzeichen (siehe Laserdrucker [9]). Der Benutzer muß darauf achten, daß im VFB (Vertical Format Buffer) (siehe Laserdrucker [9]) der verwendeten NDFILEs alle im Element verwendeten Steuerzeichen enthalten sind. Bei der Ausgabe entspricht ein Elementsatz einer Listenzeile.

NO — Nur die Anzahl der Elementsätze wird ausgegeben.

? — Der aktuelle Wert wird protokolliert.

NUM — Die Satznummer des Elementsatzes wird bei der Ausgabeart TXT vor dem Satz ausgegeben, bei der Ausgabeart REC vor dem Satzlängenfeld protokolliert.

Bei den Angaben PRT und NO hat dieser Wert keine Wirkung.

NONUM — Die Satznummer wird nicht ausgegeben.
Wirkungslos bei Ausgaben nach SYSLST oder in ein Element; hier wird die Satznummer immer mit ausgegeben.

### Format 2: Auflisten von Bindemodulen

Dieses Format gilt nur für den Elementtyp R. Es legt den Umfang und die Art der Auflistung fest.

Bei der Ausgabe von Bindemodulen wird zwischen unaufbereiteter Ausgabe (REC) und aufbereiteter Ausgabe (ALL, TXT, UPD, SYM) unterschieden. Bei der unaufbereiteten Ausgabe wird das Satzlängenfeld und, falls angegeben, die Satznummer mit ausgegeben. Bei der aufbereiteten Ausgabe wird weder Satzlängenfeld noch Satznummer ausgegeben.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | LS[T]=[{REC[/{NU[M] / NO[NUM]}] / ALL / TXT / UPD / SYM / NO / ?}] |

REC — Alle Sätze des Moduls werden in alphanumerischer und sedezimaler Form ausgegeben.
Das 4 Byte lange Satzlängenfeld wird vor der sedezimalen Zeile protokolliert.
Die Satznummer kann ausgegeben werden (siehe NUM).

ALL — Die LMS bekannten ESD-, ISD-, TXT-, RLD-, TXTP-, REP- und END-Informationen werden in aufbereiteter Form ausgegeben. Sonstige Informationen wie z.B. LSD und DSDD werden unaufbereitet ausgegeben. Fortlaufende Textinformationen werden nicht aufgeteilt.

TXT — Auflisten der TXT-, ESD- und RLD-Informationen in aufbereiteter Form. Fortlaufender Text wird nicht in Sätze aufgeteilt.

UPD — Das Korrekturjournal (TXTP) und REP-Informationen werden in aufbereiteter Form ausgegeben.

SYM — ESD- und ISD-Informationen werden in aufbereiteter Form ausgegeben.

NO — Die Anzahl der Elementsätze und die Modullänge werden ausgegeben.

? — Der aktuelle Wert wird protokolliert.

NUM Die Satznummer wird bei Angabe REC vor dem Satzlängenfeld ausgegeben. Dieser Wert hat nur Wirkung bei der Ausgabe an die Datensichtstation.

NONUM Die Satznummer wird nicht protokolliert.

*Beispiel*

```
/START-PROGRAM $LMS
$LIB LMS.TEST,BOTH
$PAR LST=REC/NUM
$LSTR ERFAS

                 E S D                              E R F A S       0
#1   0054000B  02C5E2C4404040404040001040400001C5D9C6C1E2404040F0000000000003

               BA4040404040404040404040404040404040404040404040404040404040

               4040404040404040404040404040404040
                 T X T                              &     &                    &
#2   00540014  02E3E7E340000000404000104040000105504110506641000004 0A9C451050

               1A4040404040404040404040404040404040404040404040404040404040

               4040404040404040404040404040404040
                                    .
                                    .
                                    .
```

Das Element ERFAS wird in alphanumerischer und sedezimaler Form mit vorangestellter Satznummer aufgelistet.

# PAR NEWFORM Steuern des Formularvorschubs

Der Verarbeitungsoperand NEWFORM bestimmt den Formularvorschub des LMS-Protokolls, falls das Ausgabemedium die Systemdatei SYSLST oder ein Element ist.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | N[EWFORM]=[{YES / NO / 3 / zahl / ?}] |

YES — Formularvorschub auf eine neue Seite, wenn entweder durch den Verarbeitungsoperanden LINE eine neue Seite errechnet wurde (siehe Verarbeitungsoperand LINE zeilen) oder ein Anweisungswechsel oder Elementwechsel stattfindet.

NO — Nur Vorschub auf eine neue Seite, wenn durch den Verarbeitungsoperanden LINE eine neue Seite errechnet wurde.

zahl — Anstelle eines Seitenvorschubes bei Beginn der Ausgabe für ein neues Element sollen zahl Leerzeilen ausgegeben werden. zahl kann Werte von 1 bis 15 annehmen.
Standardwert: 3.

? — Der aktuelle Wert wird protokolliert.

*Hinweis*

Soll nach jedem Element ein Seitenvorschub erfolgen, ist PAR NEWFORM=YES anzugeben. Der Standardwert 3 bewirkt, daß anstelle eines Seitenvorschubs 3 Leerzeilen ausgegeben werden.

Der Verarbeitungsoperand NEWFORM beeinflußt den gesamten LMS-Lauf.

# PAR OVERWRITE Überschreiben gleichnamiger Elemente

Dieser Verarbeitungsoperand steuert das Überschreiben gleichnamiger Elemente in der Ausgabebibliothek bzw. gleichnamiger Dateien oder FMS-Elemente bei SEL.
Für Delta-Elemente ist dieser Operand nicht zugelassen.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | O[VERWRITE]=[YES / ONLY / NO / V / D / EXTEND / ?] |

YES — Ein Element oder eine Datei mit gleicher Elementbezeichnung wird überschrieben oder neuangelegt, falls es noch nicht existiert.

ONLY — Ein Element wird nur dann geschrieben, wenn schon ein Element oder eine Datei mit gleicher Elementbezeichnung vorhanden ist.

NO — Ein Element oder eine Datei mit gleicher Elementbezeichnung wird nicht überschrieben, die Anweisung wird nicht ausgeführt.

V — Überschreiben eines Elementes mit gleicher Elementbezeichnung nur dann, wenn Versionsnummer neu > Versionsnummer alt.
Wirkt bei Programmbibliotheken wie NO.

D — Überschreiben eines Elementes mit gleicher Elementbezeichnung nur dann, wenn Datum neu > Datum alt.

EXTEND — Das Element oder die Datei soll erweitert werden. Dieser Operandenwert wird nur bei ADD und SEL berücksichtigt. Bei den übrigen Anweisungen wirkt EXTEND wie der Operandenwert NO.

Ein Element oder eine Datei wird jedoch nur dann erweitert, wenn im Element keine ISAM-Schlüssel gespeichert sind und die im Element gespeicherten Dateimerkmale mit den Merkmalen der Datei bis auf den Dateinamen übereinstimmen. Sonst wird ADD bzw. SEL mit einer Fehlermeldung zurückgewiesen.

? — Der aktuelle Wert wird protokolliert.

Der Verarbeitungsoperand OVERWRITE beeinflußt:
ADD, COR, DUP, EDR, EDT, NAM, NUM, PRT, SEL, UPD

# PAR PATH Vereinbaren eines Pfadnamens

Dieser Verarbeitungsoperand dient zum Festlegen eines Pfadnamens für UPDL und LSTL.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | PA[TH]=[{name / ?}] |

name max. 255 Zeichen langer Pfadname

Bei LSTL wird dann nur der Sub-LLM mit diesem Pfadnamen aufgelistet.

Bei UPDL dient der Name als Basis für das Displacement in der *COR-Subanweisung.

? Der aktuelle Wert wird protokolliert.

Der Verarbeitungsoperand PATH beeinflußt UPDL und LSTL.

*Hinweis*

Ist der Verarbeitungsoperand SLICE gesetzt, so wird er durch Setzen des Verarbeitungsoperanden PATH auf 'UNDEFINED' zurückgesetzt.

# PAR PHASE Festlegung des Phasenformats

Dieser Verarbeitungsoperand legt das zu erzeugende Phasenformat fest. Standardmäßig werden in der PAMKEY(PK)-Welt PK-Phasen erzeugt und in der NONPAMKEY(NK)-Welt NK-Phasen. Durch diesen Verarbeitungsoperanden können auch in der PK-Welt NK-Phasen erzeugt werden.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | PH[ASE]=[{PK / NK / ?}] |

PK — Die Phase wird im PK-Format erzeugt, falls sie auf eine PK-Platte geschrieben wird. Die Phase wird im NK-Format erzeugt, falls sie auf eine NK-Platte geschrieben wird.

NK — Die Phase wird immer im NK-Format erzeugt.

? — Der aktuelle Wert wird protokolliert.

Der Defaultwert des Operanden PHASE wird durch die CLASS2-OPTION bestimmt.

Der Verarbeitungsoperand PHASE beeinflußt SELC.

# PAR RANGE Definieren des Kennungsfeldes in Ausgabesätzen

Dieser Verarbeitungsoperand definiert Lage und Länge des Kennungsfeldes in Ausgabesätzen. Das Kennungfeld kann mit einem dynamisch ermittelten Wert (Verarbeitungsoperanden STRING und VALUE) besetzt werden. Besteht die Eingabe aus einer ISAM-Datei, kann der ISAM-Schlüssel im Kennungsfeld abgelegt werden (Verarbeitungsoperand STRING=KEY).

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | RA[NGE]=[{[anf][/lge] / NO / ?}] |

anf — Anfangsposition des Kennungsfeldes.
anf kann folgende Werte haben:
$1 \leq anf \leq 251$

lge — Länge des Kennungsfeldes.
lge kann folgende Werte haben:
$1 \leq lge \leq 16$

Falls nur anf angegeben wird, muß anf aus dem Bereich $65 \leq anf \leq 80$ sein, da sich lge aus lge = 81 - anf errechnet.

Falls nur lge angegeben wird, errechnet sich anf aus:
anf = 81 - lge

Falls anf und lge zusammen angegeben werden, muß gelten:
$anf + lge \leq 252$

NO — Es wird kein Kennungsfeld definiert.

? — Der aktuelle Wert wird protokolliert.

Der Verarbeitungsoperand RANGE beeinflußt:
ADD, COR, EDR, EDT, NUM

# PAR REFERENCE Vereinbaren von Referenzbedingungen

Der Verarbeitungsoperand REFERENCE vereinbart Bedingungen (Referenzbedingungen), unter denen Elemente zur Bearbeitung ausgewählt werden. Die Referenzbedingung besteht aus dem Paar Referenzname und Referenzattribut.
Dieser Verarbeitungsoperand wird nur bei Elementen vom Typ R in Programmbibliotheken ausgewertet.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | RE[FERENCE]=[ name / ([name],[ CSECT / ENTRY / ALL ]) / ? ] |

| | |
|---|---|
| name | gibt den Referenznamen an. Eine Auswahlangabe ist erlaubt. name darf höchstens 32 Zeichen lang sein. |
| CSECT | nur Referenznamen mit dem Attribut CSECT sollen bearbeitet werden. |
| ENTRY | nur Referenznamen mit dem Attribut ENTRY sollen bearbeitet werden. |
| ALL | Referenznamen mit beliebigem Attribut sollen bearbeitet werden. |
| ? | Der aktuelle Wert wird protokolliert. |

Die Angabe `PAR REFERENCE=` setzt eventuell bestehende Referenzbedingungen auf undefiniert zurück.

Die Angabe `PAR REFERENCE=name` wirkt wie `PAR REFERENCE=(name, ALL)`.

Die Angabe `PAR REFERENCE=(,{...})` wirkt wie `PAR REFERENCE=(*,{...})`.

Der Verarbeitungsoperand REFERENCE beeinflußt:
LST, TOC, DUP, DEL

*Hinweis*

Der Elementtyp R muß explizit angegeben werden, da sonst die Referenzbedingung ignoriert wird.

# PAR SEGMENT Festlegen von Segmenten eines Lademoduls

Der Verarbeitungsoperand SEGMENT legt die Segmente von Lademodulen fest, die aufgelistet werden sollen.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | SE[GMENT]=[{overlay / %ROOT / %ALL / ?}] |

| | |
|---|---|
| overlay | Name eines Overlays. overlay darf höchstens 8 Zeichen lang sein. |
| %ROOT | Name des Grundsegments. |
| %ALL | alle Segmente werden aufgelistet. |
| ? | Der aktuelle Wert wird protokolliert. |

Der Verarbeitungsoperand SEGMENT beeinflußt LST.

# PAR SLICE Vereinbaren eines Slices

Dieser Verarbeitungsoperand dient zum Festlegen eines Segments für UPDL und LSTL.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | SL[ICE]=[{name / ?}] |

name max. 32 Zeichen langer Slice-Name

Bei LSTL wird dann nur der Slice mit diesem Namen aufgelistet.

Bei UPDL dient der Name als Basis für das Displacement in der *COR-Subanweisung

? Der aktuelle Wert wird protokolliert.

*Hinweis*

Ist der Verarbeitungsoperand PATH gesetzt, so wird er durch Setzen des Verarbeitungsoperanden SLICE auf 'UNDEFINED' zurückgesetzt.

# PAR SORT Sortieren des Inhaltsverzeichnisses

Der Verarbeitungsoperand SORT definiert die Sortierkriterien für die Ausgabe der Inhaltsverzeichniseinträge (vgl. TOC, Seite 165).

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | SO[RT]=[{U / [N][V][D] / R / ?}] |

U — unsortierte Ausgabe:
Dieser Parameter ist nur bei sequentiellen Bibliotheken von Interesse.
Die Ausgabe erfolgt in der Reihenfolge der Elemente in der Bibliothek.

N — Ausgabe nach Namen sortiert.

V — Ausgabe nach Versionsnummern sortiert.

D — Ausgabe nach Datum sortiert.

R — Ausgabe nach Referenznamen sortiert, die mit dem Verarbeitungsoperand REFERENCE vereinbart sind.

Wurde mit dem Verarbeitungsoperanden REFERENCE keine Referenzbedingung vereinbart, gibt LMS das Primärnamensverzeichnis aus.

? — Der aktuelle Wert wird protokolliert.

N, V und D sind die Standardwerte, sie können in beliebiger Reihenfolge und Kombination angegeben werden.

Wenn mehrere Sortierkriterien angegeben sind, wird in der Reihenfolge ihrer Angabe sortiert.

Der Verarbeitungsoperand SORT beeinflußt TOC.

# PAR STRING
# Definieren einer Zeichenfolge im Kennungsfeld der Ausgabesätze

Der Verarbeitungsoperand STRING definiert eine Zeichenfolge, die bei Neunumerierung linksbündig im Kennungsfeld der Ausgabesätze (bestimmt durch RANGE) eingetragen wird.

Der Operandenwert KEY wird nur noch aus Kompatibilitätsgründen unterstützt. ISAM-Schlüssel werden mit dem neuen Verarbeitungsoperanden PAR KEY=YES in das Ausgabeelement aufgenommen.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | STRIN[G]=[{'zeichenfolge' \| NAME \| KEY \| NO \| ?}] |

zeichenfolge — alphanumerische- und Sonderzeichen (maximal 16 Stellen). Der Apostroph ist in der Zeichenfolge nicht erlaubt.

NAME — Die ersten 3 Zeichen des Elementnamens werden als Zeichenfolge übernommen.

KEY — Der Operandenwert KEY wird nur noch aus Kompatibilitätsgründen unterstützt. Er legt die ISAM-Schlüssel im Kennungsfeld ab, wird jedoch ignoriert, wenn die ISAM-Schlüssel mit dem Verarbeitungsoperanden PAR KEY=YES im Ausgabeelement aufgenommen werden.

Sind die Längen des Kennungsfeldes (siehe Wert lge im Verarbeitungsoperanden RANGE) und des ISAM-Schlüssels (KEY-LENGTH) unterschiedlich, gilt:

lge>KEY-LENGTH: Kennungsfeld wird rechts mit Nullen aufgefüllt.
lge<KEY-LENGTH: ISAM-Schlüssel wird rechts abgeschnitten.

NO — Im Kennungsfeld werden durch STRING keine Eintragungen gemacht.

? — Der aktuelle Wert wird protokolliert.

Der Verarbeitungsoperand STRING beeinflußt:
ADD, COR, EDR, EDT, NUM, SEL

# PAR STRIP Unterdrücken von Sätzen

Der Verarbeitungsoperand STRIP legt fest, welche Satzarten nicht vom Eingabeelement in das Ausgabeelement übernommen werden sollen. STRIP wird nur bei Elementen vom Typ R, L und C ausgewertet.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | STRIP=[{satzart / (satzart,...) / YES / NO / ?}] |

satzart — schließt die angegebene Satzart von der Übernahme in das Ausgabeelement aus.

Mögliche Satzarten sind

- für Elemente vom Typ R:
  ISD, LSD, TXTP, REP, INCLUDE, DSDD
- für Elemente vom Typ C und L:
  TXTP

Andere Angaben werden ignoriert.

YES — TXTP-Sätze werden nicht in das Ausgabeelement übernommen, d.h. das Korrekturjournal wird nicht übernommen.

NO — alle Sätze werden in das Ausgabeelement übernommen.

? — Der aktuelle Wert wird protokolliert.

Der Verarbeitungsoperand STRIP beeinflußt UPD für Elemente vom Typ R, L und C und DUP für Elemente des Types R und C.

# PAR SUM Erzeugen der Vergleichsstatistik

Der Verarbeitungsoperand SUM steuert das Speichern von Vergleichsstatistiken aus einem Vergleich mit COM. Diese Vergleichsstatistiken werden in benannten Summenfeldern abgelegt und können über SUM, SUMPRT, SUMDEL und SUMADD weiterverarbeitet werden.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | SU[M]=[YES / NO / ?] |

YES Die Vergleichstatistik nachfolgender Vergleiche wird auf das Summenfeld S1 addiert.

END bewirkt dann:

1. Summenfeld S1 wird auf Summenfeld S2 addiert.
2. Summenfeld S2 wird ausgegeben.
3. Der LMS-Lauf wird beendet.

NO Die Vergleichsstatistik nachfolgender Vergleiche wird nicht gespeichert.

Die Summenfelder werden bei der Zuweisung nicht gelöscht. Ein Löschen der Summenfelder erfolgt mit SUMDEL (siehe Seite 162).

Der Verarbeitungsoperand SUM beeinflußt COM.

# PAR TERMINATE Abbruchverhalten im Fehlerfall

Der Verarbeitungsoperand TERMINATE bestimmt, welche Fälle als Fehler behandelt werden und das LMS-interne Abbruchkennzeichen setzen.

Ist bei LMS-Ende das Abbruchkennzeichen gesetzt, führt das im Dialogbetrieb zur normalen Programmbeendigung, im Stapelbetrieb bzw. in Prozeduren zur Verzweigung auf STEP oder ABEND oder LOGOFF. Im Stapelbetrieb wird unabhängig vom aktuellen Wert bei schwerwiegenden Fehlern in den Testmodus übergegangen.

Zusätzlich steuert dieser Verarbeitungsoperand das Verhalten von LMS nach Fehlern, d.h. er bestimmt, bei welchen Fehlern LMS in den TEST-Modus wechselt.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | TER[MINATE]=[{1 / 2 / 3 / 4 / 5 / 6 / 7 / ?}] |

1 TEST-Modus:
Setzen des Abbruchbits bei schwerwiegenden Fehlern, d.h. Fehler bei denen eine Fortsetzung nicht sinnvoll wäre (z.B. Gerätefehler).

2 RUN-Modus, 3 TEST-Modus:
Setzen des Abbruchbits bei schwerwiegenden Fehlern und zusätzlich bei sonstigen Fehlern (z.B. syntaktischer Fehler in einer Anweisung, Element konnte nicht korrigiert werden).

4 RUN-Modus, 5 TEST-Modus:
Wie bei 2 und 3 und zusätzlich dann, wenn eine Funktion nicht ausführbar ist, weil ein Element nicht gefunden wurde.

6 RUN-Modus, 7 TEST-Modus:
Wie bei 4 und 5 und zusätzlich dann, wenn eine Funktion nicht ausführbar ist, weil das Überschreiben eines vorhandenen Elementes nicht erlaubt war (OVERWRITE=NO) oder das Element nicht geschrieben werden durfte, weil noch keines mit demselben Namen vorhanden war (OVERWRITE=ONLY).

? Der aktuelle Wert wird protokolliert.

Bei den geraden Werten (2, 4, 6) wird nach Auftreten der TERMINATE-Bedingung der LMS-Lauf im RUN-Modus fortgesetzt, bei den ungeraden Werten (1, 3, 5, 7) wird außer, wenn LMS die Anweisungen im Dialog von der Datensichtstation liest, in den TEST-Modus umgeschaltet.

Jede aufgetretene TERMINATE-Bedingung wird protokolliert.

Der Verarbeitungsoperand TERMINATE beeinflußt den gesamten LMS-Lauf.

# PAR TEST Ein-/Ausschalten und Beenden des TEST-Modus

Der Verarbeitungsoperand TEST schaltet unabhängig von Fehlerfällen den TEST-Modus ein. Er erlaubt eine Rückkehr in den RUN-Modus, wenn der TEST-Modus nicht durch Fehler verursacht worden ist. Weiter legt er fest, daß LMS beim Wechsel vom RUN- in den TEST-Modus seinen Lauf beendet.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | TES[T]=[{YES / NO / ABORT / ?}] |

YES — LMS schaltet in den TEST-Modus. Mit Ausnahme von END, LIB, PAR, SYS, CTL und PRT werden keine weiteren Anweisungen akzeptiert.

Die Anweisungen werden auf formale Richtigkeit und auf Zuweisung der erforderlichen Bibliotheken geprüft.
Bei den Korrekturfunktionen werden die Korrekturanweisungen formal überprüft; zusätzlich werden alle Prüfungen im Element durchgeführt. Datensätze werden überlesen. Fehler beim Korrigieren werden protokolliert.

NO — LMS schaltet in den RUN-Modus, d.h. sämtliche aufgerufenen Funktionen werden durchgeführt.

ABORT — Beim Wechsel vom RUN- in den TEST-Modus aufgrund eines Fehlers (siehe Verarbeitungsoperand TERMINATE) wird LMS beendet.

? — Der aktuelle Wert wird protokolliert.

Ist PAR TEST=YES gesetzt, dann hat RST keine Wirkung.

Der Verarbeitungsoperand TEST beeinflußt den gesamten LMS-Lauf und RST.

# PAR TOC
## Steuern des Ausgabeformats für Inhaltsverzeichnisse von Programmbibliotheken

Der Verarbeitungsoperand TOC bestimmt die Anzahl der Elementeinträge pro Zeile und ihre Darstellung bei der Ausgabe eines Inhaltsverzeichnisses einer Programmbibliothek. Der Operand wird zusammen mit dem Verarbeitungsoperanden LINE ausgewertet.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | TO[C]=[{F / D / L / I / ?}] |

F — Pro Protokollzeile wird ein Element eingetragen. Für jede Eintragart ist die maximale Länge vorgesehen (Typ=8, Name=64, Version=24, Variante=4, Datum=10, Flag=1). Befindet sich ein 'D' in der Flagspalte, ist das Element als Delta-Element gespeichert (siehe Beispiel 1). Mit dem Verarbeitungsoperanden LINE kann die Zeile bis auf maximal 80 Zeichen verkürzt werden. Wenn ein Elementeintrag wegen eines zu langen Elementnamens nicht in die Protokollzeile paßt, wird die Zeile zwischen Elementnamen und Version in zwei Zeilen aufgeteilt.

D — Bei Delta-Elementen wird immer ein kompletter Baum aufgelistet, unabhängig davon welches Element des Baumes bei TOC angegeben wurde. Durch die Elementangabe bei TOC wird nur der Baum ausgewählt. Zusätzlich zur Elementbezeichnung werden die interne Delta-Nummer (DELTA#, spiegelt die chronologische Reihenfolge wieder) und die entsprechende Nummer der Basis (BASE#) ausgegeben. Diese internen Delta-Nummern sind innerhalb eines Baums eindeutig und beschreiben die Verkettung der Elemente im Baum (unabhängig von der externen benutzereigenen Versionsbezeichnung). Die Ausgabe eines Baums ist immer nach DELTA# sortiert, d.h. der Verarbeitungsoperand SORT wirkt nicht innerhalb eines Baums. Verschiedene Bäume sind durch einen durchgehenden Strich voneinander getrennt.

Bei Voll-Elementen sind die Ausgabefelder DELTA# und BASE# leer (siehe Beispiel 2).

L — Der längste Elementeintrag des gesamten Inhaltsverzeichnisses wird festgestellt und daraus die Anzahl der möglichen Elementeinträge pro Protokollzeile berechnet. Mit dem Verarbeitungsoperanden LINE kann zusätzlich die Anzahl der Zeichen pro Zeile reduziert werden. Bei zu langen Elementnamen wird der Elementeintrag auf 80 Zeichen pro Zeile vergrößert und bei Bedarf die Zeile zwischen Elementnamen und Version geteilt.

T — Die Arbeitsweise ist ähnlich wie bei dem Verarbeitungsoperanden L. Der längste Elementeintrag wird pro Elementtyp gesondert festgestellt. Dadurch können in Abhängigkeit von der Länge des Elementnamens unterschiedlich viele Elementeinträge pro Protokollzeile für die verschiedenen Elementtypen erzeugt werden.

? — Der aktuelle Wert wird protokolliert.

Der Verarbeitungsoperand TOC beeinflußt TOC.

*Beispiel 1*

```
TYP       NAME        VERSION   (VAR#)    DATE         FLAG

(S    )   CTL                   (0002)    1991-05-28
(S    )   DELTA-42    001       (0003)    1991-06-24   D
(S    )   DELTA-42    002       (0003)    1991-06-23   D
(S    )   DELTA-42    003       (0003)    1991-05-28   D
```

*Beispiel 2*

```
TYP       NAME        VERSION   (VAR#)    DATE         DELTA#   BASE#

(S    )   CTL                   (0002)    1991-05-28
-------------------------------------------------------------------
(S    )   DELTA-42    001       (0003)    1991-06-24   00001    00000
(S    )   DELTA-42    002       (0003)    1991-06-23   00002    00001
(S    )   DELTA-42    003       (0003)    1991-05-28   00003    00002
```

# PAR TYPE Vorbesetzen des Elementtyps

Der Verarbeitungsoperand TYPE besetzt den Elementtyp vor, so daß der Elementtyp bei den Anweisungen nicht mehr angegeben zu werden braucht.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | TY[PE]=[{typ / ?}] |

typ gibt den Elementtyp an, der bearbeitet werden soll.

Zulässige Elementtypen sind:

- S Quellprogramme
- M Makros
- R Bindemodule (nicht für Delta-Elemente)
- C Lademodule (nicht für Delta-Elemente)
- H Compiler-Ergebnisinformationen
- J Prozeduren
- P druckaufbereitete Daten
- D Textdaten
- X Daten beliebigen Formates (nicht für Delta-Elemente)
- * Stellvertretend für alle Elementtypen (nur bei Programmbibliotheken und bei DEL, DUP, LST, NAM, SEL, TOC).

? Der aktuelle Wert wird protokolliert.

Die Angabe PAR TYPE= nimmt eine eventuell vorhandene Vorbesetzung wieder zurück. Der Elementtyp ist dann undefiniert.

Der Verarbeitungsoperand TYPE besetzt für folgende Anweisungen den Elementtyp vor: ADD, COM, COR, DEL, DUP, EDR, EDT, LST, NAM, NUM, SEL, TOC, UPD

# PAR VALUE Numerieren im Kennungsfeld der Ausgabesätze

Der Verarbeitungsoperand VALUE bestimmt den Anfangswert und die Schrittweite der Nummern, die beim Numerieren rechtsbündig im Kennungsfeld der Ausgabesätze bzw. im ISAM-Schlüssel (bei Ausgabe in eine ISAM-Datei) eingetragen werden. Linksbündig wird die - durch STRING definierte - Zeichenfolge eingetragen.

Bei STRING=KEY und Aufnahme aus einer ISAM-Datei (siehe ADD) wird der Verarbeitungsoperand VALUE ignoriert.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | V[ALUE]=[{[anf][/stp] / NO / ?}] |

anf — Anfangswert der Numerierung (max. 16 Stellen)
Fehlt diese Angabe, beginnt die Numerierung mit 0.

stp — Schrittweite (max. 16 Stellen)
Fehlt diese Angabe, wird mit Schrittweite 10 numeriert.

NO — Es soll keine Numerierung erfolgen.

? — Der aktuelle Wert wird protokolliert.

Findet bei der Numerierung ein Überlauf statt (in den STRING-Teil hinein, falls ein solcher vorhanden ist, oder aus dem Kennungsfeld heraus), wird die überlaufende Ziffer abgeschnitten und, nach Ausgabe einer Warnmeldung, die Numerierung mit dem Rest fortgesetzt.

Tritt dieser Überlauf bei der Ausgabe in eine ISAM-Datei auf, wird die Verarbeitung abgebrochen, da die Reihenfolge des ISAM-Schlüssels somit nicht aufsteigend ist (KEY OUT OF ORDER).

Der Verarbeitungsoperand VALUE beeinflußt:
ADD, COR, EDR, EDT, NUM, SEL

# Beispiele

## Einfache Beispiele

### Aufnehmen, Korrigieren und Übersetzen von Quellprogrammen in Bibliotheken

Ein Quellprogramm wird als Element vom Typ S in einer Programmbibliothek abgelegt und übersetzt. Da beim Übersetzen Fehler gefunden wurden, wird das Element mit dem EDT korrigiert und anschließend wieder übersetzt. Der Modul aus dem EAM-Bereich wird in dieselbe Programmbibliothek mit dem Elementtyp R aufgenommen.

```
/START-PROGRAM $LMS ——————————————————————————————————————— (01)
%  BLS0500 PROGRAM 'LMS', VERSION 'V02.0A10' OF '91-05-29' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG. 1990.
%  ALL RIGHTS RESERVED
%  LMS0310 LMS VERSION V02.0A10 LOADED
$PAR LOG=MAX ————————————————————————————————————————————— (02)
PAR LOG=MAX
$LIB UEB.BIB,NEW,BOTH ———————————————————————————————————— (03)
LIB UEB.BIB,NEW,BOTH
LIBRARY IS CLEARED AND PREPARED ——————————————————————————— (04)
$ADDS QUELL.ERFASS>ERFASS ————————————————————————————————— (05)
ADDS QUELL.ERFASS>ERFASS
INPUT  FILE
OUTPUT LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
         ADD QUELL.ERFASS AS (S)ERFASS/@(0001)/1991-07-25 ——————— (06)
$TOC* * ——————————————————————————————————————————————— (07)
TOC* *
INPUT  LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
TYP NAME    VER (VAR#) DATE
(S) ERFASS      (0001) 1991-07-25
     1 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS ——————————————— (08)
```

```
$LSTS ERFASS ──────────────────────────────────────────────────────── (09)
LSTS ERFASS
INPUT  LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
INPUT  ELEMENT= (S)ERFASS/@(0001)/1991-07-25
               TITLE 'ERFASSEN VON DATEN'
               PRINT NOGEN
ERFAS          START
               BALR 5,0
               USING *,5
               OPEN  DATEI,OUTPUT
LESEN          RDATA SATZ,ENDPGM
               PUT   DATEI,SATZ                    ──────────────── (10)
               B     LESEN
ENDPGM         TERM
*
DATEI          FXB   FCBTYPE=SAM, LINK=DATEN
SATZ           DS    CL84
               END
NUMBER OF PROCESSED RECORDS IS      14
$END ──────────────────────────────────────────────────────────────── (11)
END
%  LMS0311 LMS V02.0A10   ENDED NORMALLY
/START-PROGRAM $ASSEMB ─────────────────────────────────────────────── (12)
%  BLS0500 PROGRAM 'ASSEMB', VERSION '300' OF '89-11-03' LOADED
V30.0A20 OF SIEMENS BS 2000 ASSEMBLER READY
GIVE ASSEMBLER OPTIONS !
**COMOPT SOURCE=UEB.BIB(ERFASS) ────────────────────────────────────── (13)
GIVE ASSEMBLER OPTIONS!
**END HALT
FLAGS IN 00003 STATEMENTS, 000 PRIVILEGED FLAGS, 000 MNOTES ──────── (14)
HIGHEST ERROR-WEIGHT : 1
SYSTEM MACROLIBRARY :  :M:$TSOS.MACROLIB
SOURCE LIBRARY  :      :N:$USER.UEB.BIB
SOURCE PROGRAM  :      ERFASS
SOURCE VERS/DATE:        @/910725
ASSEMBLY TIME :        1.5383  SEC.
/START-PROGRAM $LMS ────────────────────────────────────────────────── (15)
%  BLS0500 PROGRAM 'LMS', VERSION 'V02.0A10' OF '91-05-29' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG. 1990.
%  ALL RIGHTS RESERVED
%  LMS0310 LMS VERSION V02.0A10 LOADED
$PAR LOG=MAX
PAR LOG=MAX
$LIB UEB.BIB,BOTH ──────────────────────────────────────────────────── (16)
LIB UEB.BIB,BOTH
```

```
$EDTS ERFASS                                                          (17)
EDTS ERFASS

  0.10          TITLE 'ERFASSEN VON DATEN'
  0.20          PRINT NOGEN
  0.30 ERFAS    START
  0.40          BALR 5,0
  0.50          USING *,5
  0.60          OPEN  DATEI,OUTPUT
  0.70 LESEN    RDATA SATZ,ENDPGM
  0.80          PUT   DATEI,SATZ
  0.90          B     LESEN
  1.00 ENDPGM   TERM
  1.10 *
x 1.20 DATEI    FCB   FCBTYPE=SAM, LINK=DATEN
  1.30 SATZ           DS    CL84                                      (18)
  1.40 END
  2.40
  3.40
  4.40
  5.40
  6.40
  7.40
  8.40
  9.40
          OUTPUT ELEMENT= (S)ERFASS/@(0002)/1991-07-25
halt                                                  0000.10:001(0)

EDT0905 EDITED MEMBER TO BE ADDED? REPLY (Y=YES; N=NO) y              (19)
INPUT  LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
OUTPUT LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
INPUT  ELEMENT= (S)ERFASS/@(0001)/1991-07-25
OUTPUT ELEMENT= (S)ERFASS/@(0002)/1991-07-25                          (20)
CORRECT (S)ERFASS/@(0001)/1991-07-25 AS (S)ERFASS/@(0002)/1991-07-25
                , OUTPUT REPLACED
$END
END
%  LMS0311 LMS V02.0A10   ENDED NORMALLY
/DELETE-SYSTEM-FILE FILE-NAME=OMF
/ASSIGN-SYSDTA TO-FILE=*LIBRARY-ELEMENT(LIBRARY=UEB.BIB,
                                        ELEMENT=ERFASS)
/START-PROGRAM $ASSEMB
%  BLS0500 PROGRAM 'ASSEMB', VERSION '300' OF '89-11-03' LOADED
V30.0A20 OF SIEMENS BS 2000 ASSEMBLER READY                           (21)
GIVE ASSEMBLER OPTIONS !
FLAGS IN 00000 STATEMENTS, 000 PRIVILEGED FLAGS, 000 MNOTES
HIGHEST ERROR-WEIGHT : -
SYSTEM MACROLIBRARY :  :M:$TSOS.MACROLIB
ASSEMBLY TIME :        2.1016  SEC.
/START-PROGRAM $LMS                                                   (22)
%  BLS0500 PROGRAM 'LMS', VERSION 'V02.0A10' OF '91-05-29' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG. 1990.
%  ALL RIGHTS RESERVED
%  LMS0310 LMS VERSION V02.0A10 LOADED
$PAR LOG=MAX
PAR LOG=MAX
```

```
$LIB UEB.BIB,BOTH ——————————————————————————————————— (23)
LIB UEB.BIB,BOTH
$ADDR *OMF ——————————————————————————————————————— (24)
ADDR *OMF
INPUT  OMF
OUTPUT LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
         ADD ERFAS AS (R)ERFAS/@(0001)/1991-07-25
$END ————————————————————————————————————————————— (25)
END
%  LMS0311 LMS V02.0A10   ENDED NORMALLY
/ASSIGN-SYSDTA TO-FILE=*PRIMARY ——————————————————— (26)
/
```

(01) LMS wird aufgerufen.

(02) Alle Meldungen und Anweisungen werden protokolliert.

(03) Die Bibliothek UEB.BIB wird als Programmbibliothek neu eingerichtet und als Ein- und Ausgabebibliothek zugewiesen.

(04) Die Bibliothek UEB.BIB ist eingerichtet.

(05) Die Datei QUELL.ERFASS wird als Element ERFASS vom Typ S in die Bibliothek aufgenommen.

(06) Erfolgsmeldung: Das Element ERFASS wurde mit der höchsten Versionsbezeichnung
und der Variantennummer 0001 in die Bibliothek
geschrieben.

(07) Das Inhaltsverzeichnis der Bibliothek UEB.BIB soll aufgelistet werden.

(08) Inhaltsverzeichniseintrag der Bibliothek UEB.BIB.

(09) Das Element ERFASS soll aufgelistet werden.

(10) Inhalt des Elementes ERFASS.

(11) LMS wird beendet.

(12) Der ASSEMBLER wird aufgerufen.

(13) Das Quellprogramm im Element ERFASS der Programmbibliothek UEB.BIB soll übersetzt werden.

(14) Das Programm ist fehlerhaft.

(15) LMS wird wieder aufgerufen.

(16) Die Programmbibliothek UEB.BIB wird als Ein- und Ausgabebibliothek zugewiesen.

(17) Das Element ERFASS soll mit dem EDT bearbeitet werden.

(18) Der Fehler wird behoben:

- mit "x" in der Anweisungsspalte Zeile 1.20 wird die Zeile auf überschreibbar gestellt;
- in der Zeile 1.20 wird "FXB" in "FCB" korrigiert und der EDT durch HALT in der Anweisungszeile beendet.

(19) Mit "Y" wird das korrigierte Element in die Ausgabebibliothek aufgenommen. Da Ein- und Ausgabebibliothek identisch sind und für das Ausgabeelement kein neuer Name angegeben wurde, wird das fehlerhafte Eingabeelement mit dem korrigierten Ausgabeelement überschrieben.

(20) Erfolgsmeldung: Das Eingabeelement ERFASS wurde korrigiert. Das Ausgabeelement erhält denselben Namen und dieselbe Versionsbezeichnung, die Variantennummer wurde um 1 auf "0002" erhöht.

(21) Der Übersetzungslauf wird fehlerfrei ausgeführt.

(22) LMS wird aufgerufen.

(23) Die Programmbibliothek UEB.BIB wird wiederum als Ein- und Ausgabebibliothek zugewiesen.

(24) Der Modul ERFAS wird aus dem EAM-Bereich als Element ERFAS abgelegt.

(25) LMS wird beendet.

(26) Umweisen der Systemeingabedatei SYSDTA.

## Duplizieren von Elementen

Elemente einer Modulbibliothek, Makrobibliothek, Quellprogrammbibliothek und einer Programmbibliothek werden in eine Programmbibliothek dupliziert.

```
/SET-FILE-LINK LINK-NAME=LIB001,FILE-NAME=MODUL.LIB
/SET-FILE-LINK LINK-NAME=LIB002,FILE-NAME=MACRO.LIB
/SET-FILE-LINK LINK-NAME=LIB003,FILE-NAME=QUELL.LIB ------------------------ (01)
/SET-FILE-LINK LINK-NAME=LIB004,FILE-NAME=TEST.LIB
/START-PROGRAM $LMS ---------------------------------------------------------- (02)
%  BLS0500 PROGRAM 'LMS', VERSION 'V02.0A10' OF '91-05-29' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG. 1990.
%  ALL RIGHTS RESERVED
%  LMS0310 LMS VERSION V02.0A10 LOADED
$PAR LOG=MAX
PAR LOG=MAX
$LIB LMSPL.LIB,NEW,BOTH ------------------------------------------------------ (03)
LIB LMSPL.LIB,NEW,BOTH
LIBRARY IS CLEARED AND PREPARED
$TOCR (1) -------------------------------------------------------------------- (04)
TOCR (1)
INPUT  LIBRARY= :N:$USER.MODUL.LIB,LINK=LIB001,DEV=DISK
TYP NAME    VER (VAR#) DATE
(R) MODERF      (0001) 1991-07-26
     1 (R)-ELEMENT(S) IN THIS TABLE OF CONTENTS
$DUPR MODERF(1)>MOD.ERF ------------------------------------------------------ (05)
DUPR MODERF(1)>MOD.ERF
INPUT  LIBRARY= :N:$USER.MODUL.LIB,LINK=LIB001,DEV=DISK
OUTPUT LIBRARY= :N:$USER.LMSPL.LIB,DEV=DISK
  DUPLICATE (R)MODERF/@(0001)/1991-07-26 AS (R)MOD.ERF/@(0001)/1991-07-2
$TOCM (2) -------------------------------------------------------------------- (06)
TOCM (2)
INPUT  LIBRARY= :N:$USER.MACRO.LIB,LINK=LIB002,DEV=DISK
TYP NAME VER (VAR#) DATE          NAME VER (VAR#) DATE
(M) MAC1     (0001) 1991-07-26    MAC2 @   (0001) 1991-07-26
     2 (M)-ELEMENT(S) IN THIS TABLE OF CONTENTS
$DUPM MAC*(2)>MU* ------------------------------------------------------------ (07)
DUPM MAC*(2)>MU*
INPUT  LIBRARY= :N:$USER.MACRO.LIB,LINK=LIB002,DEV=DISK
OUTPUT LIBRARY= :N:$USER.LMSPL.LIB,DEV=DISK
  DUPLICATE (M)MAC1/@(0001)/1991-07-26 AS (M)MUC1/@(0001)/1991-07-26
  DUPLICATE (M)MAC2/@(0001)/1991-07-26 AS (M)MUC2/@(0001)/1991-07-26 ---- (08)
$TOCS (3) -------------------------------------------------------------------- (09)
TOCS (3)
INPUT  LIBRARY= :N:$USER.QUELL.LIB,LINK=LIB003,DEV=DISK
TYP NAME     VER (VAR#) DATE          NAME     VER (VAR#) DATE
(S) EDTB         (0001) 1991-07-26    PROT     @   (0001) 1991-07-26
(S) SEINAUS      (0001) 1991-07-26    SERFAS   @   (0001) 1991-07-26
     4 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS
$DUPS *,-EDTB(3) ------------------------------------------------------------- (10)
DUPS *,-EDTB(3)
INPUT  LIBRARY= :N:$USER.QUELL.LIB,LINK=LIB003,DEV=DISK
OUTPUT LIBRARY= :N:$USER.LMSPL.LIB,DEV=DISK
  DUPLICATE (S)PROT/@(0001)/1991-07-26 AS (S)PROT/@(0001)/1991-07-26
  DUPLICATE (S)SEINAUS/@(0001)/1991-07-26 AS
            (S)SEINAUS/@(0001)/1991-07-26
  DUPLICATE (S)SERFAS/@(0001)/1991-07-26 AS (S)SERFAS/@(0001)/1991-07-26
```

```
$TOC* (4) ------------------------------------------------------------ (11)
TOC* (4)
INPUT  LIBRARY= :N:$USER.TEST.LIB,LINK=LIB004,DEV=DISK
TYP NAME    VER (VAR#) DATE
(S) SERFAS      (0001) 1991-07-26
     1 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS
$DUP* (4)>*/007 ------------------------------------------------------ (12)
DUP* (4)>*/007
INPUT  LIBRARY= :N:$USER.TEST.LIB,LINK=LIB004,DEV=DISK
OUTPUT LIBRARY= :N:$USER.LMSPL.LIB,DEV=DISK
  DUPLICATE (S)SERFAS/@(0001)/1991-07-26
            (S)SERFAS/007(0001)/1991-07-26
$TOC* * -------------------------------------------------------------- (13)
TOC* *
INPUT  LIBRARY= :N:$USER.LMSPL.LIB,DEV=DISK
TYP NAME VER (VAR#) DATE          NAME VER (VAR#) DATE
(M) MUC1     (0001) 1991-07-26    MUC2 @   (0001) 1991-07-26
     2 (M)-ELEMENT(S) IN THIS TABLE OF CONTENTS
TYP NAME     VER (VAR#) DATE
(R) MOD.ERF      (0001) 1991-07-26
     1 (R)-ELEMENT(S) IN THIS TABLE OF CONTENTS
TYP NAME     VER (VAR#) DATE          NAME     VER (VAR#) DATE
(S) PROT         (0001) 1991-07-26    SEINAUS @    (0001) 1991-07-26
(S) SERFAS       (0001) 1991-07-26
     3 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS
$TOC* */* ------------------------------------------------------------ (14)
TOC* */*
INPUT  LIBRARY= :N:$USER.LMSPL.LIB,DEV=DISK
TYP NAME VER (VAR#) DATE          NAME VER (VAR#) DATE
(M) MUC1     (0001) 1991-07-26    MUC2 @   (0001) 1991-07-26
     2 (M)-ELEMENT(S) IN THIS TABLE OF CONTENTS
TYP NAME     VER (VAR#) DATE
(R) MOD.ERF      (0001) 1991-07-26
     1 (R)-ELEMENT(S) IN THIS TABLE OF CONTENTS
TYP NAME     VER (VAR#) DATE          NAME     VER (VAR#) DATE
(S) PROT         (0001) 1991-07-26    SEINAUS @    (0001) 1991-07-26
(S) SERFAS   007 (0001) 1991-07-26    SERFAS       (0001) 1991-07-26
     4 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS
$LIB ? --------------------------------------------------------------- (15)
LIB ?
USAGE  STATUS FORMAT LID LINKNAME FILENAME
IN     OPEN   PL                  :N:$USER.LMSPL.LIB
OUT    OPEN   PL                  :N:$USER.LMSPL.LIB
       CLOSED PL     001 LIB001   :N:$USER.MODUL.LIB
       CLOSED PL     002 LIB002   :N:$USER.MACRO.LIB
       CLOSED PL     003 LIB003   :N:$USER.QUELL.LIB
       CLOSED PL     004 LIB004   :N:$USER.TEST.LIB
$LIB C --------------------------------------------------------------- (16)
LIB C
$LIB ? --------------------------------------------------------------- (17)
LIB ?
USAGE  STATUS FORMAT LID LINKNAME FILENAME
       CLOSED PL                  :N:$USER.LMSPL.LIB
       CLOSED PL     001 LIB001   :N:$USER.MODUL.LIB
       CLOSED PL     002 LIB002   :N:$USER.MACRO.LIB
       CLOSED PL     003 LIB003   :N:$USER.QUELL.LIB
       CLOSED PL     004 LIB004   :N:$USER.TEST.LIB
```

```
$END ——————————————————————————————————————————————————— (18)
END
%  LMS0311 LMS V02.0A10   ENDED NORMALLY
/
```

(01) Die so zugewiesenen Bibliotheken können im LMS-Lauf mit der Bibliothekskurzbezeichnung angesprochen werden. Im Beispiel werden sie als temporäre Eingabebibliotheken benötigt.

(02) LMS wird aufgerufen.

(03) Die Programmbibliothek LMSPL.LIB wird neu eingerichtet und als Ein- und Ausgabebibliothek zugewiesen.

(04) Das Inhaltsverzeichnis der Bindemodulbibliothek MODUL.LIB, die über die Bibliothekskurzbezeichnung (1) zugewiesen wird, soll aufgelistet werden.

(05) Der Modul MODERF aus der Bibliothek MODUL.LIB wird in die Bibliothek LMSPL.LIB mit dem Elementnamen MOD.ERF dupliziert.

(06) Das Inhaltsverzeichnis der Makrobibliothek LIB.MAC, die über die Bibliothekskurzbezeichnung (2) zugewiesen wird, soll aufgelistet werden.

(07) Diejenigen Elemente der Bibliothek LIB.MAC werden in die Bibliothek LMSPL.LIB dupliziert, deren Elementbezeichnung mit "MAC" beginnt. Die neuen Elementbezeichnungen beginnen mit "MU". Ab der dritten Stelle werden die Elementbezeichnungen der Eingabeelemente übernommen.

(08) Erfolgsmeldung: Die ausgewählten Elemente wurden mit den neuen Elementbezeichnungen aus der Modulbibliothek in die Programmbibliothek dupliziert.

(09) Das Inhaltsverzeichnis der Quellprogrammbibliothek QUELL.LIB, die über die Bibliothekskurzbezeichnung (3) zugewiesen wird, soll aufgelistet werden.

(10) Alle Elemente der Quellprogrammbibliothek QUELL.LIB mit Ausnahme des Elementes EDTB werden in die Ausgabebibliothek dupliziert.

(11) Das Inhaltsverzeichnis der Programmbibliothek TEST.LIB, die über die Bibliothekskurzbezeichnung (4) zugewiesen wird, soll aufgelistet werden.

(12) Alle Elemente der Programmbibliothek TEST.LIB werden in die Ausgabebibliothek dupliziert und mit gleichen Namen und der Versionsnummer 007 abgelegt.

(13) Das Inhaltsverzeichnis der aktuellen Eingabebibliothek LMSPL.LIB soll aufgelistet werden.

(14) Das Inhaltsverzeichns der aktuellen Eingabebibliothek LMSPL.LIB soll mit allen Elementen aufgelistet werden. Es existieren 2 Elemente mit dem Elementnamen SERFAS mit unterschiedlicher Versionsnummer. Mit TOC* * wurde nur das Element mit der höchsten Versionsnummer "SERFAS/007" angezeigt.

(15) Der Zustand der im LMS-Lauf verwendeten Bibliotheken wird abgefragt.

(16) Die Bibliothek LMSPL.LIB wird geschlossen.

(17) Der Zustand der im LMS-Lauf verwendeten Bibliotheken wird abgefragt.

(18) Der LMS-Lauf wird beendet.

## Vergleichen von Elementen

Das Element ERFASS (aufgelistet im Beispiel auf Seite 255) und das Element EINAUS werden miteinander verglichen. Ein Vergleichsprotokoll wird erstellt.

```
/START-PROGRAM $LMS ------------------------------------------------------------ (01)
%  BLS0500 PROGRAM 'LMS', VERSION 'V02.0A10' OF '91-05-29' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG. 1990.
%  ALL RIGHTS RESERVED
%  LMS0310 LMS VERSION V02.0A10 LOADED
$PAR LOG=MAX
PAR LOG=MAX
$LIB UEB.BIB,BOTH -------------------------------------------------------------- (02)
LIB UEB.BIB,BOTH
$ADDS QUELL.EINAUS>EINAUS ------------------------------------------------------ (03)
ADDS QUELL.EINAUS>EINAUS
INPUT  FILE
OUTPUT LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
         ADD QUELL.EINAUS AS (S)EINAUS/@(0001)/1991-07-29
$LSTS EINAUS ------------------------------------------------------------------- (04)
LSTS EINAUS
INPUT  LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
INPUT  ELEMENT= (S)EINAUS/@(0001)/1991-07-29
         TITLE 'ERFASSEN VON DATEN'
         PRINT NOGEN
ERFAS    START
         BALR  5,0
         USING *,5
         OPEN  DATEI,OUTPUT
LESEN    RDATA SATZ,ENDPGM
         CLC   TEXT(4),=C'/EOF'
         BE    ENDPGM
         MVC   ATEXT,TEXT
         LH    9,SL
         AH    9,=H'1'
         STH   9,ASL
         WROUT ASATZ,ENDPGM
         PUT   DATEI,SATZ
         B     LESEN
ENDPGM   TERM
*
DATEI    FCB   FCBTYPE=SAM,LINK=DATEN
         DS    0H
SATZ     DS    CL84
SL       DS    CL2
         DS    CL2
TEXT     DS    CL80
ASATZ    DS    0CL85
ASL      DS    CL2
         DC    X'000001'
ATEXT    DS    CL80
         END
NUMBER OF PROCESSED RECORDS IS      29
$PAR COMPARE=/MAX -------------------------------------------------------------- (05)
PAR COMPARE=/MAX
$COMS EINAUS=ERFASS ------------------------------------------------------------ (06)
COMS EINAUS=ERFASS
FUNCTION           = C O M P A R E
PAR          COMPARE= 00001/00072/L/MAX
PRIMARY      LIBRARY= :N:$USER.UEB.BIB
PRIMARY      ELEMENT= (S)EINAUS/@(0001)/1991-07-29  ---------------------------- (07)
SECONDARY    LIBRARY= :N:$USER.UEB.BIB
SECONDARY    ELEMENT= (S)ERFASS/@(0002)/1991-07-26
```

```
SAME FROM       #1 TO       #7 AS FROM       #1 TO       #7 ------------------ (08)

     #1 >              TITLE 'ERFASSEN VON DATEN'<
     #2 >              PRINT NOGEN<
     #3 >ERFAS         START<
     #4 >              BALR 5,0<                          ------------------ (09)
     #5 >              USING *,5<
     #6 >              OPEN  DATEI,OUTPUT<
     #7 >LESEN         RDATA SATZ,ENDPGM<
-------------------------------------------------------------------------------
INS. FROM       #8 TO       #14 ----------------------------------------------- (10)

     #8 >          CLC   TEXT(4),=C'/EOF'<
     #9 >          BE    ENDPGM<
     #10 >         MVC   ATEXT,TEXT<
     #11 >         LH    9,SL<                          ------------------ (11)
     #12 >         AH    9,=H'1'<
     #13 >         STH   9,ASL<
     #14 >         WROUT ASATZ,ENDPGM<
-------------------------------------------------------------------------------
SAME FROM       #15 TO       #19 AS FROM       #8 TO       #12 ------------------ (12)

     #15 >             PUT   DATEI,SATZ<
     #16 >             B     LESEN<
     #17 >ENDPGM       TERM<                              ------------------ (13)
     #18 >*<
     #19 >DATEI        FCB   FCBTYPE=SAM, LINK=DATEN<
-------------------------------------------------------------------------------
INS.            #20 ----------------------------------------------------------- (14)

     #20 >         DS    0H< ----------------------------------------------- (15)
-------------------------------------------------------------------------------
SAME            #21            AS          #13 ------------------------------- (16)

     #21 >SATZ         DS    CL84< ---------------------------------------- (17)
-------------------------------------------------------------------------------
INS. FROM       #22 TO       #28 ----------------------------------------------- (18)

     #22 >SL           DS    CL2<
     #23 >             DS    CL2<
     #24 >TEXT         DS    CL80<
     #25 >ASATZ        DS    OCL85<                     ------------------ (19)
     #26 >ASL          DS    CL2<
     #27 >             DC    X'000001'<
     #28 >ATEXT        DS    CL80<
-------------------------------------------------------------------------------
SAME            #29            AS          #14 ------------------------------- (20)

     #29 >             END<  ---------------------------------------------- (21)
-------------------------------------------------------------------------------
PRIMARY    ELEMENT= (S)EINAUS/@(0001)/1991-07-29
SECONDARY  ELEMENT= (S)ERFASS/@(0002)/1991-07-26
RESULT: C   PRIMARY=  29 INSERTED=  15 (    3)  DELETED=   0 (    0)
          SECONDARY=  14     SAME=  14 (    4)                          ------- (22)
$END ------------------------------------------------------------------------- (23)
END
%  LMS0311 LMS V02.0A10   ENDED NORMALLY
/
```

(01) LMS wird aufgerufen.

(02) Die Programmbibliothek UEB.BIB wird als Ein- und Ausgabebibliothek zugewiesen.

(03) Die Datei QUELL.EINAUS wird als Element EINAUS vom Typ S in die Bibliothek aufgenommen.

(04) Das Element EINAUS wird aufgelistet.

(05) Mit dem Verarbeitungsoperanden COMPARE wird festgelegt, daß das Vergleichsprotokoll in vollem Umfang ausgegeben wird.

(06) Die Elemente EINAUS und ERFASS werden miteinander verglichen.

(07) Beginn des Vergleichsprotokolls:
Protokolliert werden die eingestellten Werte des Verarbeitungsoperanden COMPARE, die Namen von Primär- und Sekundärbibliothek und von Primär- und Sekundärelement.

(08) – (21) Vergleichsprotokoll

(08) Die Sätze mit der Satzkennung #1 bis #7 sind in beiden Elementen gleich.

(09) Die gleichen Sätze werden ausgegeben.

(10) Die Sätze #8 bis #14 sind nur im Primärelement vorhanden und werden als INS(erted) dargestellt.

(11) Die eingefügten Sätze werden ausgegeben.

(12) Die Sätze #15 bis #19 des Primärelementes sind gleich den Sätzen #8 bis #12 des Sekundärelementes.

(13) Die gleichen Sätze werden ausgegeben.

(14) Der Satz #20 ist nur im Primärelement vorhanden und wird als INS(erted) dargestellt.

(15) Der eingefügte Satz wird ausgegeben.

(16) Der Satz #21 des Primärelementes ist gleich dem Satz #13 des Sekundärelementes.

(17) Der gleiche Satz wird ausgegeben.

(18) Die Sätze #20 bis #28 sind nur im Primärelement vorhanden und werden als INS(erted) dargestellt.

(19) Die eingefügten Sätze werden ausgegeben.

(20) Der Satz #29 des Primärelementes ist gleich dem Satz #14 des Sekundärelementes.

(21) Der gleiche Satz wird ausgegeben.

(22) Ergebnis des Vergleichs; ausgegeben wird die Anzahl der Sätze des Primärelementes, des Sekundärelementes, der eingefügten, gleichen und gelöschten Sätze.

Die Zahl in Klammern gibt an, wieviel zusammenhängende Teile eingefügt, gleich oder gelöscht sind.

(23) LMS wird beendet.

## Arbeiten mit Delta-Elementen

```
/START-PROGRAM $LMS ──────────────────────────────────────────────────── (01)
%  BLS0500 PROGRAM 'LMS', VERSION 'V02.0A10' OF '91-05-29' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG. 1990.
%  ALL RIGHTS RESERVED
%  LMS0310 LMS VERSION V02.0A10 LOADED
$PAR LOG=MAX
PAR LOG=MAX
$LIB LIB.DELTA,NEW,BOTH ──────────────────────────────────────────────── (02)
LIB LIB.DELTA,NEW,BOTH
LIBRARY IS CLEARED AND PREPARED
$ADDS WORKELEM>DELTA/V00,BASEVERSION=*NONE ────────────────────────────── (03)
ADDS WORKELEM>DELTA/V00,BASEVERSION=*NONE
INPUT  FILE
OUTPUT LIBRARY= :N:$USER.LIB.DELTA,DEV=DISK
         ADD WORKELEM AS (S)DELTA/V00(0001)/1991-07-29 , FIRST DELTA VERSION
$EDTS DELTA/V00>VOLLELEM ──────────────────────────────────────────────── (04)
EDTS DELTA/V00>VOLLELEM
```

```
  0.10  MINI      START
  0.20            BALR  3,0
  0.30            USING *,3
  0.40            OPEN  SAMFCB,OUTPUT
  0.50            PUT   SAMFCB,EINGABE
  0.60            TERM
  0.70  SAMFCB    FCB   FCBTYPE=SAM,LINK=MINI
  0.80  EINGABE   DC    C'DA IST ER JA'
  0.90            END
  1.90
  2.90
  3.90
  4.90
  5.90
  6.90
  7.90
  8.90
  9.90
 10.90
 11.90
 12.90
 13.90
                OUTPUT ELEMENT= (S)VOLLELEM/V00(0001)/1991-07-29
halt                                                    0000.10:001(0)
```

```
EDT0905 EDITED MEMBER TO BE ADDED? REPLY (Y=YES; N=NO) y
INPUT  LIBRARY= :N:$USER.LIB.DELTA,DEV=DISK
OUTPUT LIBRARY= :N:$USER.LIB.DELTA,DEV=DISK
INPUT  ELEMENT= (S)DELTA/V00(0001)/1991-07-29
OUTPUT ELEMENT= (S)VOLLELEM/V00(0001)/1991-07-29
         CORRECT (S)DELTA/V00(0001)/1991-07-29 AS
                 (S)VOLLELEM/V00(0001)/1991-07-29
```

```
$DUPS VOLLELEM>DELTA/V01,BASEVERSION=*HIGH ------------------------------ (05)
DUPS VOLLELEM>DELTA/V01,BASEVERSION=*HIGH
INPUT  LIBRARY= :N:$USER.LIB.DELTA,DEV=DISK
OUTPUT LIBRARY= :N:$USER.LIB.DELTA,DEV=DISK
         DUPLICATE (S)VOLLELEM/V00(0001)/1991-07-29 AS
                   (S)DELTA/V01(0002)/1991-07-29 ON BASE
                   (S)DELTA/V00(0002)/1991-07-29
$LIB LIB.ARBEIT,IN ------------------------------------------------------ (06)
LIB LIB.ARBEIT,IN
$DUPS INPUT>DELTA/V02,BASEVERSION=V00 ----------------------------------- (07)
DUPS INPUT>DELTA/V02,BASEVERSION=V00
INPUT  LIBRARY= :N:$USER.LIB.ARBEIT,DEV=DISK
OUTPUT LIBRARY= :N:$USER.LIB.DELTA,DEV=DISK
      DUPLICATE (S)INPUT/@(0001)/1991-07-29 AS (S)DELTA/V02(0003)/1991-07-29
                ON BASE (S)DELTA/V00(0003)/1991-07-29
$LIB LIB.DELTA,IN ------------------------------------------------------- (08)
LIB LIB.DELTA,IN
$PAR TOC=D -------------------------------------------------------------- (09)
PAR TOC=D
$TOC -------------------------------------------------------------------- (10)
$TOC* */* GENERATED BY LMS
INPUT  LIBRARY= :N:$USER.LIB.DELTA,DEV=DISK
TYP      NAME                      VERSION    (VAR#) DATE       DELTA# BASE#
(S     ) DELTA . . . . . . . . . . V00  . . . (0003) 1991-07-29 00001  00000
(S     ) DELTA . . . . . . . . . . V01  . . . (0003) 1991-07-29 00002  00001
(S     ) DELTA . . . . . . . . . . V02  . . . (0003) 1991-07-29 00003  00001
-------------------------------------------------------------------------------
(S     ) VOLLELEM  . . . . . . . . V00  . . . (0001) 1991-07-29
-------------------------------------------------------------------------------
     4 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS
$ADDS WORKELEM>DELTA/V11,BASEVERSION=V01 --------------------------------- (11)
ADDS WORKELEM>DELTA/V11,BASEVERSION=V01
INPUT  FILE
OUTPUT LIBRARY= :N:$USER.LIB.DELTA,DEV=DISK
         ADD WORKELEM AS (S)DELTA/V11(0004)/1991-07-29 ON BASE
             (S)DELTA/V01(0004)/1991-07-29
$TOC -------------------------------------------------------------------- (12)
$TOC* */* GENERATED BY LMS
INPUT  LIBRARY= :N:$USER.LIB.DELTA,DEV=DISK
TYP      NAME                      VERSION    (VAR#) DATE       DELTA# BASE#
(S     ) DELTA . . . . . . . . . . V00  . . . (0004) 1991-07-29 00001  00000
(S     ) DELTA . . . . . . . . . . V01  . . . (0004) 1991-07-29 00002  00001
(S     ) DELTA . . . . . . . . . . V02  . . . (0004) 1991-07-29 00003  00001
(S     ) DELTA . . . . . . . . . . V11  . . . (0004) 1991-07-29 00004  00002
-------------------------------------------------------------------------------
(S     ) VOLLELEM  . . . . . . . . V00  . . . (0001) 1991-07-29
-------------------------------------------------------------------------------
     5 (S)-ELEMENT(S) IN THIS TABLE OF CONTENTS
$END -------------------------------------------------------------------- (13)
END
%  LMS0311 LMS V02.0A1D   ENDED NORMALLY
/
```

(01) LMS wird aufgerufen.

(02) Die Programmbibliothek LIB.DELTA wird neu eingerichtet und als Ein- und Ausgabebibliothek zugewiesen.

(03) Die Datei WORKELEM wird als neues Delta-Element DELTA/V00 vom Typ S in die Bibliothek aufgenommen.

(04) Das Delta-Element DELTA/V00 soll mit dem EDT bearbeitet werden. Dazu wird aus dem Delta-Element DELTA/V00 ein neues Voll-Element VOLLELEM erzeugt.

(05) Das bearbeitete Voll-Element VOLLELEM wird als Delta-Element DELTA/V01 vom Typ S zum Delta-Element DELTA/V00 in die Ausgabebibliothek dupliziert.

(06) Die Programmbibliothek LIB.ARBEIT wird als Eingabebibliothek zugewiesen.

(07) Das Voll-Element INPUT wird als Delta-Element DELTA/V02 vom Typ S zum Delta-Element DELTA/V00 in die Ausgabebibliothek dupliziert.

(08) Die Programmbibliothek LIB.DELTA wird wieder als Eingabebibliothek zugewiesen.

(09) Der Verarbeitungsoperand PAR TOC=D muß angegeben werden, damit beim Aufruf von TOC der komplette Delta-Baum aufgelistet wird.

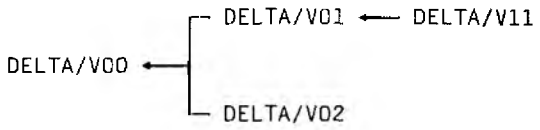
(10) Das Inhaltsverzeichnis der Bibliothek LIB.DELTA soll aufgelistet werden.

(11) Die Datei WORKELEM wird als Delta-Element DELTA/V11 vom Typ S zum Delta-Element DELTA/V01 aufgenommen.

(12) Das Inhaltsverzeichnis der Bibliothek LIB.DELTA soll aufgelistet werden.

(13) LMS wird beendet.

Die Delta-Elemente haben nun folgende Zuordnung:



# Komplexe Beispiele

## Korrigieren eines Quellprogramms mit COR

Das Element DAT wird mit Korrekturanweisungen von COR korrigiert.

```
/START-PROGRAM $LMS ------------------------------------------------------ (01)
%  BLS0500 PROGRAM 'LMS', VERSION 'V02.0A10' OF '91-05-29' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG. 1990.
%  ALL RIGHTS RESERVED
%  LMS0310 LMS VERSION V02.0A10 LOADED
$PAR LOG=MAX!LIB UEB.BIB,BOTH -------------------------------------------- (02)
PAR LOG=MAX
LIB UEB.BIB,BOTH
$ADDS QUELL.DAT>DAT ------------------------------------------------------ (03)
ADDS QUELL.DAT>DAT
INPUT  FILE
OUTPUT LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
         ADD QUELL.DAT AS (S)DAT/@(0001)/1991-07-30
$PAR LST=TXT/NUM --------------------------------------------------------- (04)
PAR LST=TXT/NUM
$LSTS DAT ---------------------------------------------------------------- (05)
LSTS DAT
INPUT  LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
INPUT  ELEMENT= (S)DAT/@(0001)/1991-07-30
    #1  >TEST     START                    |
    #2  >TEST     START                    |
    #3  >         BALR  3.0                |
    #4  >         USING *,3                |
    #5  >         GDATE DATUM,FORMAT=ISO   |
    #6  >         WROUT SATZ1,ENDE         |
    #7  >ENDE     TERM                     | ------------------------------ (06)
    #8  >*                                 |
    #9  >         DS    OF                 |
   #10  >SATZ1    DC    AL2(17)            |
   #11  >         DC    X'000001'          |
   #12  >DATUM    DS    CL12               |
   #13  >         END   TEST               |
NUMBER OF PROCESSED RECORDS IS      13
$DUPS DAT>SDAT ----------------------------------------------------------- (07)
DUPS DAT>SDAT
INPUT  LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
OUTPUT LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
         DUPLICATE (S)DAT/@(0001)/1991-07-30 AS (S)SDAT/@(0001)/1991-07-30
$CORS DAT ---------------------------------------------------------------- (08)
CORS DAT
**DEL #1 ----------------------------------------------------------------- (09)
INPUT  LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
OUTPUT LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
INPUT  ELEMENT= (S)DAT/@(0001)/1991-07-30
OUTPUT ELEMENT= (S)DAT/@(0002)/1991-07-30
*DEL #1
*DEL*        #1 >TEST     START
```

```
**CHA #3'.0'<17>=:=',0' ------------------------------------------------ (10)
*CHA #3'.0'<17>=:=',0'
          #1 >TEST     START
**REP #5 -------------------------------------------------------------- (11)
*REP #5
*HIT*     #2 >         BALR  3,0
          #3 >         USING *,3
*DEL*     #5 >         GDATE DATUM,FORMAT=ISO
        GDATE DATUM,FORMAT=ISO,TOD=ZEIT ---------------------------------- (12)
*ADD*     #4 >         GDATE DATUM,FORMAT=ISO,TOD=ZEIT
**INS #6 -------------------------------------------------------------- (13)
*INS #6
          #5 >         WROUT SATZ1,ENDE
*       WROUT SATZ2,ENDE
*ADD*     #6 >         WROUT SATZ2,ENDE
**INS #12 ------------------------------------------------------------- (14)
*INS #12
          #7 >ENDE     TERM
          #8 >*
          #9 >         DS    0F
         #10 >SATZ1    DC    AL2(17)
         #11 >         DC    X'000001'
         #12 >DATUM    DS    CL12
*SATZ2   DC    A(13)
*ADD*     #13 >SATZ2   DC    A(13)
*       DC    X'000001'
*ADD*     #14 >        DC    X'000001'
*ZEIT    DS    CL8
*ADD*     #15 >ZEIT    DS    CL8
**END ----------------------------------------------------------------- (15)
*END
          #16 >        END   TEST
       CORRECT (S)DAT/@(0001)/1991-07-30 AS (S)DAT/@(0002)/1991-07-30
             , OUTPUT REPLACED
$LSTS DAT ------------------------------------------------------------- (16)
LSTS DAT
INPUT  LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
INPUT  ELEMENT= (S)DAT/@(0002)/1991-07-30
    #1  >TEST     START
    #2  >         BALR  3,0
    #3  >         USING *,3
    #4  >         GDATE DATUM,FORMAT=ISO,TOD=ZEIT
    #5  >         WROUT SATZ1,ENDE
    #6  >         WROUT SATZ2,ENDE
    #7  >ENDE     TERM
    #8  >*
    #9  >         DS    0F
   #10  >SATZ1    DC    AL2(17)
   #11  >         DC    X'000001'
   #12  >DATUM    DS    CL12
   #13  >SATZ2    DC    A(13)
   #14  >         DC    X'000001'
   #15  >ZEIT     DS    CL8
   #16  >         END   TEST
NUMBER OF PROCESSED RECORDS IS      16
```

```
$END ——————————————————————————————————————————————— (17)
END
%  LMS0311 LMS V02.0A10   ENDED NORMALLY
/
```

(01) LMS wird aufgerufen.

(02) Zwei durch Ausrufezeichen getrennte Anweisungen werden eingegeben:

a) Alle Meldungen und Anweisungen werden protokolliert.
b) Die Programmbibliothek UEB.BIB wird als Ein- und Ausgabebibliothek zugewiesen.

(03) Die Datei QUELL.DAT wird als Element DAT vom Typ S in die Bibliothek aufgenommen.

(04) Mit dem Verarbeitungsoperanden LST wird festgelegt, daß beim Auflisten die Satznummer mit ausgegeben werden soll.

(05) Das Element DAT wird aufgelistet.

(06) Der Inhalt des Elementes DAT mit vorangestellter Satznummer.

(07) Das Element DAT wird in das Element SDAT dupliziert.

(08) Das Element DAT soll mit COR korrigiert werden.

(09) Der Satz mit der Satznummer #1 wird gelöscht.

(10) Die Zeichenfolge '.0' in Spalte 17 des Satzes mit der Nummer #3 wird in ',0' geändert.

(11), (12)
Der Satz mit der Nummer #5 wird ersetzt durch den mit (12) gekennzeichneten Satz.

(13), (14)
Anschließend an die Sätze mit den Nummern #6 und #12 werden Sätze eingefügt.

(15) Die Korrekturanweisungen werden beendet.

(16) Das korrigierte Element wird noch einmal aufgelistet.

(17) LMS wird beendet.

## Korrigieren eines Bindemoduls mit UPD

Das Element USELST wird mit Korrekturanweisungen von UPDR korrigiert. Zuerst wird im TEST-Modus die CROSS-Kontrollzahl errechnet und anschließend bei der Korrektur zur Überprüfung mit angegeben.

```
/START-PROGRAM $LMS ------------------------------------------------------ (01)
%  BLS0500 PROGRAM 'LMS', VERSION 'V02.0A10' OF '91-05-29' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG. 1990.
%  ALL RIGHTS RESERVED
%  LMS0310 LMS VERSION V02.0A10 LOADED
$PAR LOG=MAX,TEST=YES ---------------------------------------------------- (02)
PAR LOG=MAX,TEST=YES
$LIB UEB.BIB,BOTH -------------------------------------------------------- (03)
LIB UEB.BIB,BOTH
$PAR LCASE=YES ----------------------------------------------------------- (04)
PAR LCASE=YES
$UPDR USELST ------------------------------------------------------------- (05)
UPDR USELST
**COR CO,'ER'=:='aa' ----------------------------------------------------- (06)
INPUT  LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
OUTPUT LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
INPUT  ELEMENT= (R)USELST/@(0001)/1991-07-31
*COR CO,'ER'=:='aa'
**END -------------------------------------------------------------------- (07)
*END
*COR CO,'ER'=:='aa'
CONTROL NUMBER:       009926
TEXT-ADR:             00000CC0
TEXT BEFORE CHANGE:   E R
                      C5D9        ------------------------------------------ (08)
TEXT AFTER  CHANGE:   a a
                      8181
CROSS CONTROL NUMBER:00009926
      NO CORRECT (R)USELST/@(0001)/1991-07-31 , LMS IN TESTMODE !!! ------- (09)
$PAR TEST=NO ------------------------------------------------------------- (10)
PAR TEST=NO
$UPDR USELST ------------------------------------------------------------- (11)
UPDR USELST
**CON 9926 --------------------------------------------------------------- (12)
INPUT  LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
OUTPUT LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
INPUT  ELEMENT= (R)USELST/@(0001)/1991-07-31
OUTPUT ELEMENT= (R)USELST/@(0002)/1991-07-31
*CON 9926
**COR CO,'ER'=:='aa' ----------------------------------------------------- (13)
*COR CO,'ER'=:='aa'
**END
*END
*CON 9926
*COR CO,'ER'=:='aa'
CONTROL NUMBER:       009926
TEXT-ADR:             00000CC0
TEXT BEFORE CHANGE:   E R
                      C5D9
TEXT AFTER  CHANGE:   a a
                      8181
```

```
CROSS CONTROL NUMBER:00009926
        CORRECT (R)USELST/@(0001)/1991-07-31 AS (R)USELST/@(0002)/1991-07-31
                , OUTPUT REPLACED ———————————————————————————— (14)
$PAR LCASE=NO ———————————————————————————————————————————————— (15)
PAR LCASE=NO
$END ————————————————————————————————————————————————————————— (16)
END
%  LMS0311 LMS V02.0A10   ENDED NORMALLY
/
```

(01) LMS wird aufgerufen.

(02) Die Verarbeitungsoperanden LOG und TEST werden bei PAR eingegeben:

a) Alle Meldungen und Anweisungen werden protokolliert.
b) Der TEST-Modus wird eingeschaltet.

(03) Die Programmbibliothek UEB.BIB wird als Ein- und Ausgabebibliothek zugewiesen.

(04) Alle eingegebenen Kleinbuchstaben werden nicht in Großbuchstaben umgesetzt.

(05) Der Modul USELST soll korrigiert werden.

(06) Der Text der Adresse 0000C0 wird ersetzt.

(07) Die Korrekturanweisungen werden beendet.

(08) LMS überprüft die Korrekturangaben und berechnet die Kontrollzahl und die Cross-Kontrollzahl.

(09) Die Korrektur wird nicht durchgeführt, da der TEST-Modus eingeschaltet ist.

(10) Der TEST-Modus wird ausgeschaltet.

(11) Der Modul USELST soll korrigiert werden.

(12) Die CROSS-Kontrollzahl wird eingegeben.

(13) Der Text der Adresse 0000C0 wird ersetzt.

(14) Die Korrektur wurde ausgeführt.

(15) Alle Eingaben werden in Großbuchstaben umgesetzt.

(16) LMS wird beendet.

## Vergleichen von Elementen mit Erstellen von Korrekturanweisungen

Beim Vergleichen der Elemente DAT und SDAT werden Korrekturanweisungen erstellt. Diese werden in eine Datei geschrieben und wieder als Element aufgenommen. Die Elemente DAT und SDAT sind im Beispiel auf Seite 271 aufgelistet.

```
/START=PROGRAM $LMS ------------------------------------------------------------ (01)
%  BLS0500 PROGRAM 'LMS', VERSION 'V02.0A10' OF '91-05-29' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG. 1990.
%  ALL RIGHTS RESERVED
%  LMS0310 LMS VERSION V02.0A10 LOADED
$PAR LOG=MAX,COM=/MAX/COR,SUM=YES ---------------------------------------------- (02)
PAR LOG=MAX,COM=/MAX/COR,SUM=YES
$SYS ASSIGN-SYSOPT TO-FILE=COR.ELEM -------------------------------------------- (03)
SYS ASSIGN-SYSOPT TO-FILE=COR.ELEM
$LIB UEB.BIB,BOTH -------------------------------------------------------------- (04)
LIB UEB.BIB,BOTH
$COMS DAT=SDAT ----------------------------------------------------------------- (05)
COMS DAT=SDAT
FUNCTION          = C O M P A R E
PAR         COMPARE= 00001/00072/L/MAX/COR    COR-CARD OUTPUT=SYSOPT
PRIMARY     LIBRARY= :N:$USER.UEB.BIB
PRIMARY     ELEMENT= (S)DAT/@(0002)/1991-08-01
SECONDARY   LIBRARY= :N:$USER.UEB.BIB
SECONDARY   ELEMENT= (S)SDAT/@(0001)/1991-08-01
-------------------------------------------------------------------------------
SAME            #1              AS              #1

      #1 >TEST      START<
-------------------------------------------------------------------------------
DEL.                                  FROM      #2 TO       #3

      #2 >TEST      START<
      #3 >          BALR  3,0<
-------------------------------------------------------------------------------
INS.            #2

      #2 >          BALR  3,0<
-------------------------------------------------------------------------------
SAME            #3              AS              #4

      #3 >          USING *,3<
-------------------------------------------------------------------------------
DEL.                                            #5

      #5 >          GDATE DATUM,FORMAT=ISO<
-------------------------------------------------------------------------------
INS.            #4

      #4 >          GDATE DATUM,FORMAT=ISO,TOD=ZEIT<
-------------------------------------------------------------------------------
SAME            #5              AS              #6

      #5 >          WROUT SATZ1,ENDE<
-------------------------------------------------------------------------------
INS.            #6

      #6 >          WROUT SATZ2,ENDE<
-------------------------------------------------------------------------------
SAME FROM       #7 TO      #12 AS FROM       #7 TO       #12

      #7 >ENDE      TERM<
      #8 >*<
      #9 >          DS    0F<
     #10 >SATZ1     DC    AL2(17)<
     #11 >          DC    X'000001'<
     #12 >DATUM     DS    CL12<
-------------------------------------------------------------------------------
```

```
INS. FROM      #13 TO      #15

     #13 >SATZ2    DC    A(13)<
     #14 >         DC    X'000001'<
     #15 >ZEIT     DS    CL8<
```

```
SAME           #16            AS           #13

     #16 >         END   TEST<
```

```
PRIMARY    ELEMENT= (S)DAT/@(0002)/1991-08-01
SECONDARY  ELEMENT= (S)SDAT/@(0001)/1991-08-01
RESULT: C   PRIMARY=      16 INSERTED=      6 (    4)  DELETED=      3 (    2)
          SECONDARY=      13     SAME=     10 (    5)
```

**$SUM** (06)

```
SUM
AREA C1
                PRIM.    PRIM.     INS.     SAME     DEL.  INS+DEL     SEC.     SEC.
STATISTIC       ELEM.    LINES    LINES    LINES    LINES    LINES    LINES    ELEM.
S (SAME)            0        0        -        0        -        -        0        0
C (CHANGED)         1       16        6       10        3        9       13        1
I (INSERTED)        0        0        0        -        -        0        -        -
D (DELETED)         -        -        -        -        0        0        0        0
   TOTAL            1       16        6       10        3        9       13        1
```

**$SYS ASSIGN-SYSOPT TO-FILE=*PRIMARY** (07)

```
SYS ASSIGN-SYSOPT TO-FILE=*PRIMARY
```

**$ADDJ COR.ELEM** (08)

```
ADDJ COR.ELEM
INPUT  FILE
OUTPUT LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
         ADD COR.ELEM AS (J)COR.ELEM/@(0001)/1991-08-01
```

**$LSTJ COR.ELEM** (09)

```
LSTJ COR.ELEM
INPUT  LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
INPUT  ELEMENT= (J)COR.ELEM/@(0001)/1991-08-01
$PAR CHECK=NO
$CORS SDAT/@/1991-08-01
*DEL #2-#3
*INS #3
         BALR  3,0
*DEL #5
*INS #5
         GDATE DATUM,FORMAT=ISO,TOD=ZEIT
*INS #6
         WROUT SATZ2,ENDE
*INS #12
SATZ2    DC    A(13)
         DC    X'000001'
ZEIT     DS    CL8
*END
NUMBER OF PROCESSED RECORDS IS      15
```

**$END** (10)

```
END
AREA C2
                PRIM.    PRIM.     INS.     SAME     DEL.  INS+DEL     SEC.     SEC.
STATISTIC       ELEM.    LINES    LINES    LINES    LINES    LINES    LINES    ELEM.
S (SAME)            0        0        -        0        -        -        0        0
C (CHANGED)         1       16        6       10        3        9       13        1
I (INSERTED)        0        0        0        -        -        0        -        -
D (DELETED)         -        -        -        -        0        0        0        0
   TOTAL            1       16        6       10        3        9       13        1
% LMS0311 LMS V02.0A10  ENDED NORMALLY
/
```

(01) LMS wird aufgerufen.

(02) Die Verarbeitungsoperanden LOG, COM und SUM werden bei PAR eingegeben:

a) Alle Meldungen und Anweisungen werden protokolliert.
b) Das Vergleichsprotokoll wird in vollem Umfang ausgegeben und es werden Korrekturanweisungen erzeugt.
c) Die Vergleichsstatistik wird gespeichert.

(03) Die Systemdatei SYSOPT, in die LMS die Korrekturanweisungen ausgibt, wird auf die Datei COR.ELEM umgewiesen.

(04) Die Programmbibliothek UEB.BIB wird als Ein- und Ausgabebibliothek zugewiesen.

(05) Die Elemente DAT und SDAT werden verglichen. Anschließend wird das Vergleichsprotokoll ausgegeben.

(06) Die Vergleichsstatistik (Summenfeld S1) wird ausgegeben.

(07) Die Systemdatei SYSOPT wird wieder zurückgewiesen.

(08) Die Datei COR.ELEM, die die Korrekturanweisungen enthält, wird als gleichnamiges Element vom Typ J in die Bibliothek aufgenommen.

(09) Das Element COR.ELEM wird aufgelistet.

(10) LMS wird beendet.

Da SUM im LMS-Lauf gegeben wurde (06), wird nun das Summenfeld S2 ausgegeben.

### Ausgeben eines Elementes in eine Datei

Mit dem EDT wird eine SAM-Datei erzeugt. Diese wird als Element aufgenommen und auf drei verschiedene Arten als Datei ausgegeben:

- als SAM-Datei, auf Grund der im Element gespeicherten Dateieigenschaften;
- als ISAM-Datei mit standardmäßig erzeugten ISAM-Schlüsseln, indem der Verarbeitungsoperand FCBTYPE=ISAM vor der Ausgabe gesetzt wird;
- als ISAM-Datei mit ISAM-Schlüsseln, die mit dem Verarbeitungsoperanden VALUE festgelegt werden. Der Verarbeitungsoperand FCBTYPE muß dabei wie oben den Operandenwert ISAM haben.

```
/START-PROGRAM $LMS ------------------------------------------------------ (01)
%  BLS0500 PROGRAM 'LMS', VERSION 'V02.0A10' OF '91-05-29' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG. 1990.
%  ALL RIGHTS RESERVED
%  LMS0310 LMS VERSION V02.0A10 LOADED
$PAR LOG=MAX!LIB UEB.BIB,BOTH -------------------------------------------- (02)
PAR LOG=MAX
LIB UEB.BIB,BOTH
$EDT ---------------------------------------------------------------------- (03)
EDT
```

```
  1.00 BACH          SEBASTIAN      MUENCHEN       AUF DER HOEHE 7     AB 3
  2.00 BERGMANN      NORBERT        MUENCHEN       TORWEG 10           AB 5
  3.00 FINK          SUSANNE        NUERNBERG      RINGSTR. 23         AB 1
  4.00 MEYER         FRANZ          NUERNBERG      WASSERMUNGENWEG     AB 1
  5.00 GRUNDLER      WOLFGANG       BASEL          SONNENSTR. 11       AB 2
  6.00 KNOLL         MONIKA         FRANKFURT      BAUMALLEE 12        AB 3
  7.00 LIEDL         ERIKA          MUENCHEN       IN DER BREITE 1     AB 5
  8.00 WAGNER        JOHANN         AUGSBURG       AM SEE 45           AB 4
  9.00
 10.00
 11.00
 12.00
 13.00
 14.00
 15.00
 16.00
 17.00
 18.00
 19.00
 20.00
 21.00
 22.00
 23.00
write'pers.dat';halt                                  0001.00:001(0)
```

```
$PAR KEY=YES ------------------------------------------------------------ (04)
PAR KEY=YES
$ADDD PERS.DAT>PERDAT --------------------------------------------------- (05)
ADDD PERS.DAT>PERDAT
INPUT  FILE
OUTPUT LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
       ADD PERS.DAT AS (D)PERDAT/@(0001)/1991-08-05
$TOCD * ----------------------------------------------------------------- (06)
TOCD *
INPUT  LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
TYP NAME    VER (VAR#) DATE
(D) PERDAT      (0001) 1991-08-05
      1 (D)-ELEMENT(S) IN THIS TABLE OF CONTENTS
$SELD PERDAT ------------------------------------------------------------ (07)
SELD PERDAT
INPUT  LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
OUTPUT FILE
       SEL (D)PERDAT/@(0001)/1991-08-05 AS PERDAT
$SYS SHOW-FILE-ATTRIBUTES PERDAT,INFORMATION=ALL ------------------------ (08)
SYS SHOW-FILE-ATTRIBUTES PERDAT,INFORMATION=ALL
00000003 :N:$USER.PERDAT
  FCBTYPE = SAM              VSNTYPE = PUB
  LASTPG  = 00000001         2ND ALLO= 00003
  SHARE   = NO               ACCESS  = WRITE
  ACL     = NO               AUDIT   = NONE          DESTROY = NO
  CRDATE  = 1991-08-05       EXDATE  = 1991-08-05    LADATE  = 1991-08-05
  RDPASS  = NONE             WRPASS  = NONE          EXPASS  = NONE
  ACCESS# = 001              VERSION = 001
  LARGE   = NO               BACKUP  = A             MIGRATE = ALLOWED
  BLKTYPE = STD              BLKSIZE = 002048        BLKCTRL = PAMKEY
  RECFORM = (V,N)            RECSIZE = 000000
  VSN/DEV/EXT  =    PUBN02 / D3480    / 001
  EXTCNT  =   1
:N:    PUBLIC:     1  FILE  RES=       3  FREE=       2  REL=       0 PAGES
$PAR FCBTYPE=ISAM!SELD PERDAT>PERSDAT ----------------------------------- (09)
PAR FCBTYPE=ISAM
SELD PERDAT>PERSDAT
INPUT  LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
OUTPUT FILE
       SEL (D)PERDAT/@(0001)/1991-08-05 AS PERSDAT
$SYS SHOW-FILE-ATTRIBUTES PERSDAT,INFORMATION=ALL ------------------------ (10)
SYS SHOW-FILE-ATTRIBUTES PERSDAT,INFORMATION=ALL
00000003 :N:$USER.PERSDAT
  FCBTYPE = ISAM             VSNTYPE = PUB
  LASTPG  = 00000002         2ND ALLO= 00003
  SHARE   = NO               ACCESS  = WRITE
  ACL     = NO               AUDIT   = NONE          DESTROY = NO
  CRDATE  = 1991-08-05       EXDATE  = 1991-08-05    LADATE  = 1991-08-05
  RDPASS  = NONE             WRPASS  = NONE          EXPASS  = NONE
  ACCESS# = 001              VERSION = 001
  LARGE   = NO               BACKUP  = A             MIGRATE = ALLOWED
  BLKTYPE = STD              BLKSIZE = 002048        BLKCTRL = PAMKEY
  RECFORM = (V,N)            RECSIZE = 000000
  KEYLEN  = 008              KEYPOS  = 00005
  VSN/DEV/EXT  =    PUBN01 / D3480    / 001
  EXTCNT  =   1
:N:    PUBLIC:     1  FILE  RES=       3  FREE=       1  REL=       0 PAGES
```

```
$EDT ──────────────────────────────────────────────────────────────── (11)
EDT
```

```
  0.10 BACH          SEBASTIAN      MUENCHEN       AUF DER HOEHE 7      AB 3
  0.20 BERGMANN      NORBERT        MUENCHEN       TORWEG 10            AB 5
  0.30 FINK          SUSANNE        NUERNBERG      RINGSTR. 23          AB 1
  0.40 MEYER         FRANZ          NUERNBERG      WASSERMUNGENWEG      AB 1
  0.50 GRUNDLER      WOLFGANG       BASEL          SONNENSTR. 11        AB 2
  0.60 KNOLL         MONIKA         FRANKFURT      BAUMALLEE 12         AB 3
  0.70 LIEDL         ERIKA          MUENCHEN       IN DER BREITE 1      AB 5
  0.80 WAGNER        JOHANN         AUGSBURG       AM SEE 45            AB 4
  1.80
  2.80
  3.80
  4.80
  5.80
  6.80
  7.80
  8.80
  9.80
 10.80
 11.80
 12.80
 13.80
 14.80
 15.80
get'persdat'noreseq                                   0001.00:001(0)
halt
```

```
$PAR VALUE=10000000/200 ──────────────────────────────────────────── (12)
PAR VALUE=10000000/200
$SELD PERDAT>PERKEY3 ─────────────────────────────────────────────── (13)
SELD PERDAT>PERKEY3
INPUT  LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
OUTPUT FILE
         SEL (D)PERDAT/@(0001)/1991-08-05 AS PERKEY3
```

```
$EDT ──────────────────────────────────────────────────────────────── (14)
EDT
```

```
1000.00 BACH          SEBASTIAN      MUENCHEN       AUF DER HOEHE 7     AB 3
1000.02 BERGMANN      NORBERT        MUENCHEN       TORWEG 10           AB 5
1000.04 FINK          SUSANNE        NUERNBERG      RINGSTR. 23         AB 1
1000.06 MEYER         FRANZ          NUERNBERG      WASSERMUNGENWEG     AB 1
1000.08 GRUNDLER      WOLFGANG       BASEL          SONNENSTR. 11       AB 2
1000.10 KNOLL         MONIKA         FRANKFURT      BAUMALLEE 12        AB 3
1000.12 LIEDL         ERIKA          MUENCHEN       IN DER BREITE 1     AB 5
1000.14 WAGNER        JOHANN         AUGSBURG       AM SEE 45           AB 4
1001.14
1002.14
1003.14
1004.14
1005.14
1006.14
1007.14
1008.14
1009.14
1010.14
1011.14
1012.14
1013.14
1014.14
1015.14
get'perkey3'noreseq                                        0001.00:001(0)
halt
```

```
$END ──────────────────────────────────────────────────────────────── (15)
END
%  LMS0311 LMS V02.0A10   ENDED NORMALLY
/
```

(01) LMS wird aufgerufen.

(02) Zwei durch Ausrufezeichen getrennte Anweisungen werden eingegeben:

a) Alle Meldungen und Anweisungen werden protokolliert.
b) Die Programmbibliothek UEB.BIB wird als Ein- und Ausgabebibliothek zugewiesen.

(03) Mit diesem Format von EDT verzweigt LMS in den EDT, um eine Datei zu erstellen oder zu bearbeiten.

Anschließend werden die Daten eingegeben und mit WRITE als SAM-Datei PERS.DAT gespeichert. Mit HALT wird der EDT beendet und zu LMS zurückgekehrt.

(04) Der Verarbeitungsoperand PAR KEY wird gleich YES gesetzt, damit alle Dateieigenschaften übernommen werden können.

(05) Die Datei PERS.DAT wird als Element PERDAT vom Typ D in die Bibliothek aufgenommen.

(06) Das Inhaltsverzeichnis der Bibliothek UEB.BIB für den Elementtyp D soll aufgelistet werden.

(07) Das Element PERDAT wird als Datei PERDAT ausgegeben. Da für diese Datei keine Dateieigenschaften spezifiziert wurden, erzeugt LMS entsprechend den gespeicherten Dateieigenschaften eine SAM-Datei.

(08) Die Dateieigenschaften der erzeugten Datei werden aufgelistet.

(09) a) Mit dem Verarbeitungsoperanden FCBTYPE=ISAM wird nun festgelegt, daß textartige Elemente als ISAM-Dateien ausgegeben werden sollen.
b) Das Element PERDAT wird als ISAM-Datei PERSDAT ausgegeben.

(10) Die Dateieigenschaften der erzeugten Datei werden aufgelistet.

(11) LMS verzweigt in den EDT. In der Auflistung der ISAM-Datei mit dem EDT sind die ersten sechs Stellen der ISAM-Schlüssel in der Zeilennummernanzeige zu sehen.

LMS erzeugte 8-stellige ISAM-Schlüssel mit einem Anfangswert von 1000 und einer Schrittweite von 1000.

(12) Für eine erneute Ausgabe des Elementes PERDAT werden für die ISAM-Schlüssel der Anfangswert und die Schrittweite festgelegt.

(13) Das Element PERDAT wird als ISAM-Datei PERKEY3 ausgegeben. Beim Generieren der ISAM-Schlüssel wird der Verarbeitungsoperand VALUE ausgewertet, da im Beispiel noch der Verarbeitungsoperand FCBTYPE=ISAM gesetzt ist.

(14) LMS verzweigt in den EDT. In der Auflistung der ISAM-Datei mit dem EDT sind die ersten sechs Stellen der ISAM-Schlüssel in der Zeilennummernanzeige zu sehen.

(15) LMS wird beendet.

## Ausgeben von Vergleichsstatistiken

Alle Elemente mehrerer Bibliotheken werden miteinander verglichen, einmal mit dem bisherigen Kreuzvergleich und anschließend mit dem Heckel-Algorithmus. Bei beiden werden nur die Vergleichsstatistiken ausgegeben.

```
/SET-FILE-LINK LINK-NAME=LIB001,FILE-NAME=LIB.SOU.V1 ┐
/SET-FILE-LINK LINK-NAME=LIB002,FILE-NAME=LIB.MAC.V1 ├ ──────────── (01)
/SET-FILE-LINK LINK-NAME=LIB003,FILE-NAME=LIB.ALL.V2 ┘
/START-PROGRAM $LMS ──────────────────────────────────────────────── (02)
%  BLS0500 PROGRAM 'LMS', VERSION 'V02.0A10' OF '91-05-29' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG. 1990.
%  ALL RIGHTS RESERVED
%  LMS0310 LMS VERSION V02.0A10 LOADED
$PAR LOG=MAX,COM=/L2/SUM,SUM=YES ─────────────────────────────────── (03)
PAR LOG=MAX,COM=/L2/SUM,SUM=YES
$COMS *(3)=*(1) ──────────────────────────────────────────────────── (04)
COMS *(3)=*(1)
┌──────────────────────────────────────────────────────────────────┐
PRIMARY    ELEMENT= (S)EINAUS/@(0001)/1991-08-09
SECONDARY  ELEMENT= (S)EINAUS/@(0003)/1991-08-09
RESULT: S   PRIMARY=   8 INSERTED=    - (   -)  DELETED=   - (   -)
          SECONDARY=   8     SAME=    8 (   1)
──────────────────────────────────────────────────────────────────
PRIMARY    ELEMENT= (S)ERFASS/@(0002)/1991-08-09
SECONDARY  ELEMENT= (S)ERFASS/@(0002)/1991-08-09                   ├ (05)
RESULT: S   PRIMARY=  14 INSERTED=    - (   -)  DELETED=   - (   -)
          SECONDARY=  14     SAME=   14 (   1)
──────────────────────────────────────────────────────────────────
PRIMARY    ELEMENT= (S)PROT/@(0001)/1991-08-09
SECONDARY  ELEMENT= (S)PROT/@(0003)/1991-08-09
RESULT: S   PRIMARY=   5 INSERTED=    - (   -)  DELETED=   - (   -)
          SECONDARY=   5     SAME=    5 (   1)
└──────────────────────────────────────────────────────────────────┘
$SUM ─────────────────────────────────────────────────────────────── (06)
SUM
AREA C1
                PRIM.    PRIM.     INS.     SAME     DEL.  INS+DEL     SEC.     SEC.
STATISTIC       ELEM.    LINES    LINES    LINES    LINES    LINES    LINES    ELEM.
S (SAME)            3       27        -       27        -        -       27        3
C (CHANGED)         0        0        0        0        0        0        0        0
I (INSERTED)        0        0        0        -        -        0        -        -
D (DELETED)         -        -        -        -        0        0        0        0
──────────────────────────────────────────────────────────────────────────────────────
   TOTAL            3       27        0       27        0        0       27        3
$PAR COM= ─────────────────────────────────────────────────────────── (07)
PAR COM=
$COMM *(3)=*(2) ──────────────────────────────────────────────────── (08)
COMM *(3)=*(2)
FUNCTION           = C O M P A R E
PAR          COMPARE= 00001/00072/L/MED
PRIMARY      LIBRARY= :N:$USER.LIB.ALL.V2
PRIMARY      ELEMENT= (M)MAC1/@(0002)/1991-08-09
SECONDARY    LIBRARY= :N:$USER.LIB.MAC.V1
SECONDARY    ELEMENT= (M)MAC1/@(0003)/1991-08-09
```

```
SAME FROM       #1 TO       #5 AS FROM       #1 TO       #5
PRIMARY    ELEMENT= (M)MAC1/@(0002)/1991-08-09                    - (09)
SECONDARY  ELEMENT= (M)MAC1/@(0003)/1991-08-09
RESULT: S    PRIMARY=   5 INSERTED=   - (   -)  DELETED=   - (   -)
           SECONDARY=   5     SAME=   5 (   1)
FUNCTION          = C O M P A R E
PAR        COMPARE= 00001/00072/L/MED
PRIMARY    LIBRARY= :N:$USER.LIB.ALL.V2
PRIMARY    ELEMENT= (M)MAC2/@(0001)/1991-08-09
SECONDARY  LIBRARY= :N:$USER.LIB.MAC.V1
SECONDARY  ELEMENT= (M)MAC2/@(0002)/1991-08-09

SAME FROM       #1 TO       #5 AS FROM       #1 TO       #5
PRIMARY    ELEMENT= (M)MAC2/@(0001)/1991-08-09                    - (09)
SECONDARY  ELEMENT= (M)MAC2/@(0002)/1991-08-09
RESULT: S    PRIMARY=   5 INSERTED=   - (   -)  DELETED=   - (   -)
           SECONDARY=   5     SAME=   5 (   1)
FUNCTION          = C O M P A R E
PAR        COMPARE= 00001/00072/L/MED
PRIMARY    LIBRARY= :N:$USER.LIB.ALL.V2
PRIMARY    ELEMENT= (M)MUC1/@(0001)/1991-08-09

PRIMARY    ELEMENT= (M)MUC1/@(0001)/1991-08-09
RESULT: I    PRIMARY=   5 INSERTED=   5 (   1)  DELETED=   - (   -)  - (09)
           SECONDARY=   -     SAME=   - (   -)
FUNCTION          = C O M P A R E
PAR        COMPARE= 00001/00072/L/MED
PRIMARY    LIBRARY= :N:$USER.LIB.ALL.V2
PRIMARY    ELEMENT= (M)MUC2/@(0001)/1991-08-09

PRIMARY    ELEMENT= (M)MUC2/@(0001)/1991-08-09
RESULT: I    PRIMARY=   5 INSERTED=   5 (   1)  DELETED=   - (   -)  - (09)
           SECONDARY=   -     SAME=   - (   -)
$SUM                                                               (10)
SUM
AREA C1
                PRIM.    PRIM.     INS.     SAME     DEL.  INS+DEL     SEC.     SEC.
STATISTIC       ELEM.    LINES    LINES    LINES    LINES    LINES    LINES    ELEM.
S (SAME)            2       10        -       10        -        -       10        2
C (CHANGED)         0        0        0        0        0        0        0        0
I (INSERTED)        2       10       10        -        -       10        -        -
D (DELETED)         -        -        -        -        0        0        0        0

   TOTAL            4       20       10       10        0       10       10        2
$END                                                               (11)
END
AREA C2
                PRIM.    PRIM.     INS.     SAME     DEL.  INS+DEL     SEC.     SEC.
STATISTIC       ELEM.    LINES    LINES    LINES    LINES    LINES    LINES    ELEM.
S (SAME)            5       37        -       37        -        -       37        5
C (CHANGED)         0        0        0        0        0        0        0        0
I (INSERTED)        2       10       10        -        -       10        -        -
D (DELETED)         -        -        -        -        0        0        0        0

   TOTAL            7       47       10       37        0       10       37        5
```

```
END
%  LMS0311 LMS V02.0A10   ENDED NORMALLY
/
```

(01) Die zu vergleichenden Bibliotheken werden zugewiesen.

(02) LMS wird aufgerufen.

(03) Die Verarbeitungsoperanden LOG, COM und SUM werden bei PAR eingegeben:

a) Alle Meldungen und Anweisungen werden protokolliert.
b) Es wird in den Kreuzvergleich verzweigt. Es wird kein Vergleichsprotokoll erstellt, sondern es werden nur die Ergebnisse ausgegeben.
c) Die Vergleichsstatistik wird gespeichert.

(04) Alle Elemente vom Typ S der Programmbibliothek LIB.ALL.V2 werden mit den Elementen der Quellprogrammbibliothek LIB.SOU.V1 verglichen.

(05) Das Ergebnis des Vergleichs wird ausgegeben.

(06) Durch SUM wird das Summenfeld S1 ausgegeben, das die Vergleichsstatistik des gesamten Vergleichs enthält. Der Vergleichsstatistik kann z.B. entnommen werden, daß insgesamt 56 Sätze verglichen wurden.

(07) Es wird zurück auf den Heckel-Algorithmus geschaltet (Standard).

(08) Alle Elemente vom Typ M der Programmbibliothek LIB.ALL.V2 werden mit den Elementen der Makrobibliothek LIB.MAC.V1 verglichen.

(09) Das Ergebnis des Vergleichs wird ausgegeben.

(10) Die Vergleichsstatistik des Gesamtvergleichs wird ausgegeben.

(11) LMS wird beendet.

Nun wird das Summenfeld S2 automatisch ausgegeben. In das Summenfeld S2 wurden die beiden Vergleichstatistiken addiert. Diese Statistik gibt also das Ergebnis sämtlicher Vergleiche dieses LMS-Laufs wieder.

## Verzweigen in ein Benutzerprogramm beim Auflisten eines Elementes

Das Benutzerprogramm listet nur die ersten 10 Eingabesätze eines Elementes auf. Besteht ein Element aus weniger als 10 Sätzen, füllt das Programm mit eigenen Sätzen auf 10 Sätze auf.

```
/START-PROGRAM $LMS ------------------------------------------------------ (01)
%  BLS0500 PROGRAM 'LMS', VERSION 'V02.0A10' OF '91-05-29' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG. 1990.
%  ALL RIGHTS RESERVED
%  LMS0310 LMS VERSION V02.0A10 LOADED
$PAR LOG=MAX!LIB TEST.LIB,BOTH ------------------------------------------- (02)
PAR LOG=MAX
LIB TEST.LIB,BOTH
$LSTS USELST ------------------------------------------------------------- (03)
LSTS USELST
INPUT  LIBRARY= :N:$USER.TEST.LIB,DEV=DISK
INPUT  ELEMENT= (S)USELST/@(0001)/1991-08-09
*        TITLE 'USEREXIT FUER DIE FUNKTION: LST'
*
*              1.) DURCH ANSCHLUSS DIESES UNTERPROGRAMMS WERDEN NUR NOCH
*                  DIE ERSTEN 10 SAETZE JE ELEMENT AUFGELISTET.
*
*              2.) HAT DAS ELEMENT WENIGER ALS 10 SAETZE, SO WERDEN
*                  WEITERE SAETZE EINGEFUEGT.
* INPUT FROM LMS: R1=A(PARAMETERLISTE)
*                 R13=A(SAVEAREA), 18 WORTE
*                 R14=RUECKSPRUNGADRESSE
*                 R15=A(BENUTZERPROGRAMM)
*
PARDSEC  DSECT
AUFTRAG  DS    A              A(AUFTRAG VOM LMS)
*                             - 'BOE':START OF ELEMENT
*                             - 'REC':RECORD ANGEBOTEN
*                             - 'EOE':END OF ELEMENT
ANTWORT  DS    A              A(ANTWORT VOM BENUTZERPROGRAMM)
*                             - 'CON':CONTINUE
*                             - 'DEL':DELETE RECORD
*                             - 'INS':INSERT NEW RECORD
SATZ     DS    A              A(SATZ, INKL. 4 BYTE HEADER)
*                             -
PARDSECL EQU   *-PARDSEC      L'DSECT
USELST   CSECT PAGE
         STM   0,15,0(13)     SAVE REGISTERS
         LR    10,15          BASE
         USING PARDSEC,1      LMS-PARAMETERLISTE
         USING USELST,10
         L     6,AUFTRAG      A(AUFTRAG)
         L     7,ANTWORT      A(ANTWORT)
         L     8,SATZ         A(SATZ)
         CLC   0(3,6),REC     SATZ ANGEBOTEN ?
         BE    DOSATZ         JA ---?
         CLC   0(3,6),BOE     START OF ELEMENT
         BE    DOBOE          JA ---?
         CLC   0(3,6),EOE     END OF ELEMENT
         BE    DOEOE          JA ---?
```

```
         B     RETURN
*
DOBOE    EQU   *
         ZAP   ANZAHL,P0        ZAEHLER := 0
         B     RETURN
*
DOSATZ   EQU   *
         CP    ANZAHL,P10       SCHON 10 SAETZE AUSGEGEBEN ?
         BNL   DODEL            JA (REST UEBERGEHEN) ---?
         AP    ANZAHL,P1        ZAEHLER := ZAEHLER +1
         B     DOCON
*
DOEOE    EQU   *
         CP    ANZAHL,P10       SCHON 10 SAETZE AUSGEGEBEN ?
         BNL   DOCON            JA (KEIN EINFUEGEN) ---?
         AP    ANZAHL,P1        ZAEHLER := ZAEHLER +1
         B     DOINS
*
DOINS    EQU   *
         MVC   0(3,7),INS       INSERT RECORD
         LA    9,INSSATZ
         ST    9,SATZ           A(RECORD TO BE INSERTED)
         B     RETURN
*
DODEL    EQU   *
         MVC   0(3,7),DEL       DELETE RECORD
         B     RETURN
*
DOCON    EQU   *
         MVC   0(3,7),CON       CONTINUE
*
RETURN   EQU   *
         LM    0,15,0(13)       RESTORE REGISTERS
         BR    14
         TITLE 'KONSTANTEN UND VARIABLE'
BOE      DC    'BOE'            START OF ELEMENT
REC      DC    'REC'            SATZ ANGEBOTEN
EOE      DC    'EOE'            END OF ELEMENT
CON      DC    'CON'            CONTINUE
DEL      DC    'DEL'            DELETE RECORD
INS      DC    'INS'            INSERT NEW RECORD
ANZAHL   DC    PL2'0'
P0       DC    PL2'0'
P1       DC    PL2'1'
P10      DC    PL2'10'
INSSATZ  DC    Y(INSSATZE-INSSATZ)
         DC    XL2'4040'
         DC    '**************** INSERT BY USER-PROGRAM  *************'
INSSATZE EQU   *
         LTORG
         END
NUMBER OF PROCESSED RECORDS IS      89
$USE LST=TEST.LIB(USELST) ------------------------------------------------ (04)
USE LST=TEST.LIB(USELST)
$LSTS EINAUS ------------------------------------------------------------- (05)
LSTS EINAUS
INPUT  LIBRARY= :N:$USER.TEST.LIB,DEV=DISK
INPUT  ELEMENT= (S)EINAUS/@(0001)/1991-08-09
```

```
USER EXIT TEST.LIB(USELST)  FOR  LSTE IS ACTIVE
         TITLE 'ERFASSEN VON DATEN'
         PRINT NOGEN
ERFAS    START
         BALR  5,0
         USING *,5
         OPEN  DATEI,OUTPUT
LESEN    RDATA SATZ,ENDPGM
         CLC   TEXT(4),=C'/EOF'
         BE    ENDPGM
         MVC   ATEXT,TEXT
NUMBER OF PROCESSED RECORDS IS      10
$LSTS PERSDAT ------------------------------------------------------------ (06)
LSTS PERSDAT
INPUT  LIBRARY= :N:$USER.TEST.LIB,DEV=DISK
INPUT  ELEMENT= (S)PERSDAT/@(0001)/1991-08-09
USER EXIT TEST.LIB(USELST)  FOR  LSTE IS ACTIVE
BACH          SEBASTIAN      MUENCHEN       AUF DER HOEHE 7     AB 3
BERGMANN      NORBERT        MUENCHEN       TORWEG 10           AB 5
FINK          SUSANNE        NUERNBERG      RINGSTR. 23         AB 1
MEYER         FRANZ          NUERNBERG      WASSERMUNGENWEG     AB 1
GRUNDLER      WOLFGANG       BASEL          SONNENSTR. 11       AB 2
KNOLL         MONIKA         FRANKFURT      BAUMALLEE 12        AB 3
LIEDL         ERIKA          MUENCHEN       IN DER BREITE 1     AB 5
WAGNER        JOHANN         AUGSBURG       AM SEE 45           AB 4
**************** INSERT BY USER-PROGRAM  *************
**************** INSERT BY USER-PROGRAM  *************
NUMBER OF PROCESSED RECORDS IS      10
$END ---------------------------------------------------------------------- (07)
END
%  LMS0311 LMS V02.0A10   ENDED NORMALLY
/
```

(01) LMS wird aufgerufen.

(02) Zwei durch Ausrufezeichen getrennte Anweisungen werden eingegeben:

a) Alle Meldungen und Anweisungen werden protokolliert.
b) Die Programmbibliothek TEST.LIB wird als Ein- und Ausgabebibliothek zugewiesen.

(03) Das Benutzer-Quellprogramm USELST wird aufgelistet.

(04) LMS verzweigt vor dem Auflisten eines Eingabesatzes in das Benutzerprogramm USELST, das in der Bibliothek TEST.LIB steht.

(05) Die ersten 10 Sätze des Elementes EINAUS der zugewiesenen Programmbibliothek TEST.LIB werden aufgelistet.

(06) Das Element PERSDAT wird aufgelistet. Da es kürzer ist als 10 Sätze, werden vom Benutzerprogramm Sätze angefügt.

(07) LMS wird beendet.

# Alte LMS-Unterprogramm-Schnittstelle

Aus Kompatibilätsgrunden wird noch die hier beschriebene alte Unterprogramm-Schnittstelle unterstützt.

Die neue Unterprogramm-Schnittstelle ist im Handbuch "LMS Unterprogramm-Schnittstelle" [15] beschrieben.

Wird LMS als Unterprogramm mit eigenständiger Dialogführung aufgerufen, kehrt es nach der Verarbeitung von END zum aufrufenden Programm zurück. LMS bleibt nach der Verarbeitung von END geladen. Ansonsten ist der Ablauf wie nach dem Laden mit dem Kommando /START-PROGRAM.

Bei jedem Aufruf von LMS meldet LMS eine eigene STXIT-Routine an. Beim Rücksprung in das Hauptprogramm wird die LMS-STXIT-Routine abgemeldet; das Hauptprogramm muß seine eigene STXIT-Routine wieder anmelden.

Beim Aufruf der Unterprogramm-Schnittstelle sind folgende Registerkonventionen zu beachten:

| | |
|---|---|
| Register 1 | muß Null sein. |
| Register 13 | enthält die Adresse eines Sicherstellungsbereichs von 18 Worten, der vom aufrufenden Programm zur Verfügung gestellt werden muß. Dieser Bereich wird von LMS für die Speicherung der Register des rufenden Programms verwendet. |
| Register 14 | enthält die Rücksprungadresse. |
| Register 15 | enthält die Einsprungadresse LMSUP. |

Vor dem Rücksprung in das rufende Programm hinterlegt LMS im Register 15 folgende Rücksprungcodes:

| | |
|---|---|
| X'00' | LMS wurde normal beendet (entspricht TERM-Beendigng). |
| X'04' | LMS wurde abnormal beendet (entspricht TERMJ-Beendigung; vgl. PAR TERMINATE Seite 248). |

*Beispiel*

Aus dem Programm UPROG wird LMS als Unterprogramm aufgerufen. In LMS wird ein Element aus einer Programmbibliothek in eine Datei ausgegeben. Nach Beenden des LMS-Laufs wird ins Benutzerprogramm zurückgesprungen.

```
/START-PROGRAM $LMS
%  BLS0500 PROGRAM 'LMS', VERSION 'V02.0A10' OF '91-05-29' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG. 1990.
%  ALL RIGHTS RESERVED
%  LMS0310 LMS VERSION V02.0A10 LOADED
$LIB UEB.BIB,BOTH!LSTS LMSCALL ------------------------------------------ (01)
INPUT  LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
INPUT  ELEMENT= (S)LMSCALL/@(0001)/1991-08-12
UPROG    START
         BALR  3,0
         USING *,3
         MVC   OUTPUT,ANMELD
AUFRUF   WROUT OUT,TERM
         LA    14,RUECK ------------------------------------------------ (02)
         LA    1,0 ----------------------------------------------------- (03)
         LA    13,SAVE ------------------------------------------------- (04)
         L     15,=V(LMSUP) -------------------------------------------- (05)
         BALR  14,15
RUECK    MVC   OUTPUT,ABMELD
         WROUT OUT,TERM
*
*
TERM     TERM
*
*
SAVE     DS    18F
*
ANMELD   DC    '********** *LMS WIRD JETZT AUFGERUFEN ************'
ABMELD   DC    '***** LMS WURDE BEENDET - WEITER IM PROGRAMM *****'
OUT      DC    Y(ENDE-OUT)
         DS    CL2
         DC    X'01'
OUTPUT   DS    CL50
ENDE     EQU   *
*
*
         END   UPROG
NUMBER OF PROCESSED RECORDS IS      29
$END
%  LMS0311 LMS V02.0A10 ENDED NORMALLY
/
.
.
.
/START-PROGRAM FROM-FILE=(LIB=UEB.BIB,ELEM=LMSCALL)
%  BLS0500 PROGRAM 'LMSCALL', VERSION '007' OF '91-08-12' LOADED
********** *LMS WIRD JETZT AUFGERUFEN ************
```

```
%  LMS0310 LMS VERSION V02.0A10 LOADED ————————————————————————— (06)
$PAR LOG=MAX!LIB UEB.BIB,BOTH
PAR LOG=MAX
LIB UEB.BIB,BOTH
$TOCC *
TOCC *
INPUT  LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
TYP NAME     VER (VAR#) DATE
(C) LMSCALL 007 (0001) 1991-08-12
     1 (C)-ELEMENT(S) IN THIS TABLE OF CONTENTS
$SELS LMSCALL>LMSCALL1
SELS LMSCALL>LMSCALL1
INPUT  LIBRARY= :N:$USER.UEB.BIB,DEV=DISK
OUTPUT FILE
         SEL (S)LMSCALL/@(0001)/1991-08-12 AS LMSCALL1
$END ————————————————————————————————————————————————————————— (07)
END
%  LMS0311 LMS V02.0A10 ENDED NORMALLY
***** LMS WURDE BEENDET - WEITER IM PROGRAMM *****
/
```

(01) Zwei durch Ausrufezeichen getrennte Anweisungen werden eingegeben:

a) Die Programmbibliothek UEB.BIB wird als Ein- und Ausgabebibliothek zugewiesen.

b) Das Quellprogramm LMSCALL wird aufgelistet.

(02) Die Rücksprungadresse wird nach Register 14 geladen.

(03) Register 1 wird auf 0 gesetzt.

(04) Die Adresse des Sicherstellungsbereiches wird nach Register 13 geladen.

(05) Die Einsprungadresse LMSUP wird nach Register 15 geladen.

(06) Aus dem Benutzerprogramm heraus wird LMS aufgerufen.

(07) Nach Beenden des LMS-Laufes wird ins Benutzerprogramm zurückgesprungen.

# Meldungen

Die Fehlermeldungen werden 7-stellig (LMSnnnn) mit ihrer Bedeutung und eventuell erforderlichen Maßnahmen in englischer und deutscher Sprache ausgegeben.

## Liste der Meldungen

LMS0001 START: NOT ENOUGH MEMORY FOR LMS
LMS0001 START: SPEICHER FUER LMS NICHT AUSREICHEND

**Maßnahme**
Systemverwalter verstaendigen.

LMS0002 LIB ASSIGNMENT LOST DURING LMS RUN
LMS0002 LIB-ZUWEISUNG GING WAEHREND DES LMS-LAUFS VERLOREN

LMS0003 INVALID STATEMENT
LMS0003 UNZULAESSIGE ANWEISUNG

**Bedeutung**
Moegliche Fehler:
- Zwischenraum zwischen Operation und Operand fehlt.
- Die Anweisung ist LMS nicht bekannt.

LMS0004 BLANK MISSING IN FRONT OF OPERAND
LMS0004 VOR DEM OPERANDEN FEHLT EIN ZWISCHENRAUM

LMS0005 MEMBER TYPE INVALID OR NOT ASSIGNED
LMS0005 ELEMENT-TYP UNZULAESSIG ODER NICHT ZUGEWIESEN

LMS0006 END OR CTL STATEMENT EXPECTED
LMS0006 END- ODER CTL-ANWEISUNG ERWARTET

LMS0007 ELEMENT CONTAINS FORMAT-B RECORDS
LMS0007 DAS ELEMENT ENTHAELT FORMAT-B SAETZE

LMS0008 SYNTAX ERROR IN OPERAND
LMS0008 SYNTAX-FEHLER IM OPERANDEN

LMS0009 SYNTAX ERROR IN STATEMENT
LMS0009 SYNTAX-FEHLER IN DER ANWEISUNG

LMS0010 SYNTAX ERROR IN DATE
LMS0010 SYNTAX-FEHLER IM DATUM

**Bedeutung**
Die Syntax fuer das Datum ist JJJJ-MM-TT oder JJMMTT.

LMS0011 SYNTAX ERROR IN VERSION NUMBER
LMS0011 SYNTAX-FEHLER IN DER VERSIONSNUMMER

**Bedeutung**
Die Syntax fuer die Versionsnummer der vorliegenden Bibliotheksart ist im LMS-Manual beschrieben.

LMS0012 SYNTAX ERROR IN NAME
LMS0012 SYNTAX-FEHLER IM NAMEN

LMS0013 SYNTAX ERROR IN (LIB)
LMS0013 SYNTAX-FEHLER IN (LIB)

LMS0014 LENGTH OF RECORDTYPE 163 MUST BE BETWEEN 36 AND 44
LMS0014 DIE SATZLAENGE DER SATZART 163 MUSS ZWISCHEN 36 UND 44 LIEGEN

**Bedeutung**
Die Satzart 163 enthaelt einen 1-32 Zeichen langen Sekundaernamen gefolgt von einem 0-8 Zeichen langen Sekundaerattribut

LMS0015 (LIB) WRONG OR NOT ASSIGNED
LMS0015 (LIB) UNZULAESSIG ODER NICHT ZUGEWIESEN

LMS0016 INPUT LIBRARY NOT ASSIGNED OR SPECIFICATION INVALID
LMS0016 EINGABE-BIBLIOTHEK NICHT ZUGEWIESEN ODER ANGABE UNZULAESSIG

LMS0017 FUNCTION NOT YET IMPLEMENTED
LMS0017 FUNKTION NOCH NICHT IMPLEMENTIERT

LMS0018 WARNING: USE CORRECT SYNTAX FOR STATEMENT
LMS0018 WARNUNG: RICHTIGES FORMAT FUER DIE ANWEISUNG VERWENDEN

LMS0019 REQUESTED MEMBER OF TYPE 'J' IN CTL STATEMENT DOES NOT EXIST
LMS0019 GESUCHTES ELEMENT VOM TYP 'J' DER CTL-ANWEISUNG EXISTIERT NICHT

LMS0020 REQUESTED MEMBER DOES NOT EXIST IN SPECIFIED LIBRARY
LMS0020 GESUCHTES ELEMENT EXISTIERT NICHT IN DER ANGEGEBENEN BIBLIOTHEK

LMS0021 LIBRARY IS DESTROYED
LMS0021 DIE BIBLIOTHEK IST ZERSTOERT

**Maßnahme**
Versuchen, Elemente mit DUP-Anweisung in eine neue Bibliothek zu duplizieren.

LMS0022 NOT ENOUGH SPACE IN LIBRARY
LMS0022 NICHT GENUEGEND PLATZ IN DER BIBLIOTHEK

**Bedeutung**
Grenzen der Bibliothek sind erreicht.

**Maßnahme**
Bibliotheks-Reorganisation ist notwendig.

LMS0023 DMS ERROR CODE '(&00)'. ERROR INFO IN SYSTEM MODE: /HELP DMS(&00),INF=D
LMS0023 DVS-FEHLERCODE '(&00)'. FEHLER-INFO IM SYSTEM-MODUS: /HELP DMS(&00),INF=D

**Bedeutung**
Wiederholte Versuche waren erfolglos.
Naehere Information ueber den DVS-Fehler kann ueber das /HELP-Kommando im Systemmodus erfragt bzw. den BS2000-Manualen 'Systemmeldungen', 'DVS Plattenverarbeitung' oder 'DVS Bandverarbeitung' entnommen werden.

**Maßnahme**
Siehe DVS-Fehlercode.

LMS0024 ACCESS ERROR ON '(&00)' LIBRARY ** (&01)
LMS0024 ZUGRIFFS-FEHLER AUF DER (&00)-BIBLIOTHEK ** (&01)

**Bedeutung**
AMCB0010: Adresse nicht im Element
AMCB0016: Ungueltiges Komprimierungskennzeichen
AMCB0017: Letztes Element geloescht
AMCB0018: Keine OSM-Bibliothek
AMCB0025: Kein DIR2-Eintrag erstellt
AMCB0027: Keine neue FP-Kette erstellt
AMCB0052: Element ersetzt
AMCB0054: Leere Datei ersetzt
AMCB0108/AMCB109:USER/OPEN-Fehler
AMCB0120: Dateiname ungueltig
AMCB0121: Kein FT-Eintrag fuer Datei
AMCB0122: Open-Status-Konflikt
AMCB0125: 2. Zugriff auf Ausgabebib.
AMCB0131: Bibliothek ist bereits fuer CTL oder PRT geoeffnet
AMCB0134: CTL-Element ist geoeffnet
AMCB0137: 2. Zugriff auf seq. Bibliothek
AMCB0255: SVC-PLAM fehlt

LMS0026 FILE IS NOT A LIBRARY
LMS0026 DATEI IST KEINE BIBLIOTHEK

**Bedeutung**
Die in der LIB-Anweisung angegebene Datei ist keine Bibliothek.

LMS0027 MEMBER TYPE NOT PERMITTED FOR '(&00)' LIBRARY
LMS0027 ELEMENT-TYP FUER (&00)-BIBLIOTHEK NICHT ERLAUBT

LMS0029 ORDER OF MEMBER TYPES FOR SEQUENTIAL LIBRARY INVALID. STATEMENT NOT PROCESSED
LMS0029 UNERLAUBTE ELEMENT-TYPEN-REIHENFOLGE FUER EINE SEQUENTIELLE BIBLIOTHEK. ANWEISUNG NICHT AUSGEFUEHRT

**Bedeutung**
Bei Ausgabe auf eine sequentielle Bibliothek ist die Reihenfolge R, M, S fuer die Element-Typen nicht eingehalten worden.

LMS0030 VSN OF VOLUME DOES NOT AGREE WITH SPECIFICATION IN LIBIN OR LIBOUT STATEMENT
LMS0030 VSN DES DATENTRAEGERS STIMMT MIT DER ANGABE IN DER LIBIN- BZW. LIBOUT-ANWEISUNG NICHT UEBEREIN

**Maßnahme**
VSN ueberpruefen.

LMS0031 INTERNAL ERROR. AMCB ERROR CODE '(&00)'
LMS0031 INTERNER FEHLER. AMCB-FEHLERCODE '(&00)'

**Bedeutung**
AMCB0002: Ungueltiger Op-Code
AMCB0003: Dateiname im Kontrollblock fehlt
AMCB0004: Keine/geaenderte FCB-Adresse
AMCB0007/AMCB0012: Widerspruechliche Angaben im Kontrollblock/FCB
AMCB0013/AMCB0014: Zusatzinformation fehlt/ungueltig
AMCB0108/AMCB0109: Benutzer/OPEN-Fehler
AMCB0127: LT-Eintrag fehlt
AMCB0136: Zugriffsfehler z.B. gesperrte Datei
AMCB0141: Unbekannte Zugriffsart

LMS0033 INTERNAL ERROR WHEN ASSIGNING '(&00)' FILE
LMS0033 INTERNER FEHLER BEI ZUWEISUNG VON '(&00)' DATEI

LMS0036 WARNING: LIBRARY TO BE CLOSED IS NOT ASSIGNED
LMS0036 WARNUNG: EINE NICHT ZUGEWIESENE BIBLIOTHEK SOLL GESCHLOSSEN WERDEN

**Maßnahme**
Bibliotheksnamen ueberpruefen.

LMS0038 READ ERROR '(&00)' WHEN READING MEMBER RECORD
LMS0038 BEIM LESEN EINES ELEMENT-SATZES TRAT LESE-FEHLER '(&00)' AUF

**Bedeutung**
Handelt es sich um einen System-Fehlercode, so kann im Systemmodus naehere Information ueber das Kommando /HELP erfragt bzw.
den BS2000-Manualen 'Systemmeldungen', 'DVS Plattenverarbeitung' oder 'DVS Bandverarbeitung' entnommen werden.

LMS0039 INPUT AND OUTPUT LIBRARY MAY NOT BE IDENTICAL FOR SEQUENTIAL LIBRARIES
LMS0039 EIN- UND AUSGABEBIBLIOTHEK DUERFEN BEI SEQUENTIELLEN BIBLIOTHEKEN NICHT IDENTISCH SEIN

**Maßnahme**
Siehe COR-Anweisung.

LMS0040 WARNING: INVALID VERSION. VERSION '001' ASSUMED
LMS0040 WARNUNG: UNGUELTIGE VERSION. VERSION '001' WIRD ANGENOMMEN

**Bedeutung**
Fehler in der Versionsangabe, z.B. enthaelt die 2. bzw. 3. Stelle keine Ziffer. Es wird die Version 001 eingetragen.

LMS0041 NOT ENOUGH MEMORY TO PROCESS STATEMENT
LMS0041 SPEICHERPLATZ REICHT FUER AUSZUFUEHRENDE ANWEISUNG NICHT AUS

LMS0043 CONTROL NUMBER DOES NOT MATCH THE ONE COMPUTED BY LMS
LMS0043 KONTROLLZAHL STIMMT NICHT MIT DER VON LMS ERMITTELTEN UEBEREIN

**Bedeutung**
Die LMS CROSS-Kontrollzahl kann ueber TEST-MODE ermittelt werden.

**Maßnahme**
UPD-Anweisung pruefen.

LMS0047 UPDATE NOT (COMPLETELY) IN TEXT AREA
LMS0047 KORREKTUR-BEREICH NICHT (VOLLSTAENDIG) IM TEXT-BEREICH ENTHALTEN

**Maßnahme**
UPD-Anweisung pruefen.

LMS0048 SPECIFIED CORRECTION ADDRESS DOES NOT EXIST
LMS0048 ANGEGEBENE KORREKTUR-ADRESSE NICHT GEFUNDEN

**Bedeutung**
Moeglicher Fehler: Adresse in der UPD-Anweisung zu klein oder zu gross angegeben.

**Maßnahme**
UPD-Anweisung pruefen.

LMS0049 CONTROL TEXT NOT AT SPECIFIED ADDRESS
LMS0049 AN ANGEGEBENER ADRESSE STEHT NICHT DER ERWARTETE KONTROLL-TEXT

**Maßnahme**
Ueber die LST-Anweisung die Textstellen des Moduls ueberpruefen und die UPD-Anweisung entsprechend korrigieren.

LMS0050 ASSIGNMENT FOR (&00) MISSING OR INVALID
LMS0050 KEINE ODER FALSCHE ZUWEISUNG FUER (&00)

LMS0051 OUTPUT LIBRARY WRONG OR NOT ASSIGNED
LMS0051 FALSCHE ODER NICHT ZUGEWIESENE AUSGABE-BIBLIOTHEK

**Maßnahme**
Mit LIB ...,OUT eine Ausgabe-Bibliothek zuordnen und Anweisung wiederholen.

LMS0052 SYNTAX ERROR IN SPECIFIED ABBREVIATION (LIB)
LMS0052 SYNTAX-FEHLER IN KURZBEZEICHNUNGS-ANGABE (LIB)

LMS0054 (LIB) WRONG OR NOT ASSIGNED - (&00)
LMS0054 (LIB) FALSCH ODER NICHT ZUGEORDNET - (&00)

**Bedeutung**
Bibliotheksdatei in TFT nicht enthalten, d.h. Bibliothek nicht zugeordnet.
Datei wurde angegeben, die keine Bibliothek ist.
Bibliothek ist nicht vorhanden.
AMCB0102: Unbekannter Dateityp
AMCB0109: OPEN Fehler
AMCB0150: Unbekannte Zugriffsart

**Maßnahme**
Ueber FILE-Kommando die Bibliothek zuordnen oder eine korrekte Bibliothek angeben.

LMS0055 INPUT MEMBERS SPECIFIED IN COM STATEMENT MAY NOT BE IN THE SAME SEQUENTIAL LIBRARY
LMS0055 DIE IN DER COM-ANWEISUNG ANGEGEBENEN EINGABE-ELEMENTE DUERFEN SICH NICHT AUF DERSELBEN SEQUENTIELLEN BIBLIOTHEK BEFINDEN

LMS0057 CROSS CONTROL NUMBER NOT DEFINED
LMS0057 CROSS-KONTROLLZAHL NICHT DEFINIERT

**Maßnahme**
Mit *CON die CROSS-Kontrollzahl zuweisen.

LMS0059 MEMBER TYPE NOT PERMITTED FOR THIS FUNCTION
LMS0059 DEFINIERTER ELEMENT-TYP FUER DIESE FUNKTION UNZULAESSIG

**Bedeutung**
Zugelassenen Element-Typen sind dem LMS-Manual zu entnehmen.

LMS0060 INPUT AND OUTPUT PERFORMED FOR THE SAME FILE
LMS0060 EINGABE UND AUSGABE ERFOLGEN AUF DIESELBE DATEI

**Bedeutung**
Fuer Ein- und Ausgabe-Bibliothek darf nicht dieselbe Datei zugewiesen sein.

LMS0061 INPUT AND OUTPUT MAY NOT BE PERFORMED FOR THE SAME SEQUENTIAL LIBRARY
LMS0061 EINGABE UND AUSGABE DUERFEN NICHT AUF DIESELBE SEQUENTIELLE BIBLIOTHEK ERFOLGEN

LMS0062 ILLEGAL STATEMENT FOR SEQUENTIAL INPUT LIBRARY. PROCESSING CONTINUES WITH NEXT STATEMENT
LMS0062 UNZULAESSIGE ANWEISUNG FUER SEQUENTIELLE EINGABE-BIBLIOTHEK. NAECHSTE ANWEISUNG WIRD VERARBEITET

**Bedeutung**
Z.B. ist eine NAM-, DEL-Anweisung unzulaessig.

LMS0063 LOG OUTPUT TO SEQUENTIAL INPUT/OUTPUT LIBRARY INVALID
LMS0063 PROTOKOLL-AUSGABE AUF SEQUENTIELLE EIN-/AUSGABE-BIBLIOTHEK IST UNZULAESSIG

**Bedeutung**
Protokollausgabe auf eine sequentielle Bibliothek, die als Ein-/Ausgabebibliothek dient, ist nicht erlaubt.

LMS0065 SYNTAX ERROR IN SPECIFIED VSN
LMS0065 SYNTAX-FEHLER BEI VSN-ANGABE

**Bedeutung**
Moegliche Fehler:
(1) Es wurden keine Klammern angegeben
(2) Angabe ist nicht 6-stellig
(3) (vsn) ist nicht der letzte Operand der LIBOUT-Anweisung.

LMS0066 WARNING: AT LEAST ONE NON-NUMERIC ISAM KEY GENERATED
LMS0066 WARNUNG: MINDESTENS EIN NICHT-NUMERISCHER ISAM-SCHLUESSEL WURDE GENERIERT

**Bedeutung**
Bei der Anweisung EDTx wurde mindestens 1 Satz mit nicht-numerischem ISAM-Schluessel ausgegeben, der von EDT nicht verarbeitet werden kann.

**Maßnahme**
Entweder die COR-Anweisung verwenden oder derartige Saetze in der Datei loeschen.

LMS0067 TYPE 'R' NOT ALLOWED FOR OSM LIBRARIES
LMS0067 TYP 'R' IST FUER OSM-BIBLIOTHEKEN NICHT ERLAUBT

LMS0068 FUNCTION NOT ALLOWED FOR THIS LIBRARY TYPE
LMS0068 FUNKTION FUER DIESEN BIBLIOTHEKS-TYP NICHT ERLAUBT

**Bedeutung**
z.B. ist PRT- oder CTL-Anweisung auf Band- oder OML-Bibliotheken unzulaessig.

LMS0072 WARNING: NO NAME TO BE PROCESSED
LMS0072 WARNUNG: KEIN GUELTIGER NAME VORHANDEN

LMS0075 WARNING: NO RANGE FIELD DEFINED FOR OUTPUT RECORDS. NUMBERING NOT PERFORMED
LMS0075 WARNUNG: KENNUNGSFELD FUER AUSGABE-SAETZE NICHT DEFINIERT. KEINE NUMERIERUNG DURCHGEFUEHRT

**Bedeutung**
Das Kennungsfeld fuer die Ausgabesaetze wurde nicht definiert, obwohl VALUE angegeben wurde.

**Maßnahme**
Verarbeitungs-Operand RANGE mit Werten versorgen.

LMS0077 WARNING: STRING LONGER THAN CHECK FIELD. NO STRING INSERTED.
LMS0077 WARNUNG: STRING LAENGER ALS KENNUNGSFELD. KEINEN STRING EINGEFUEGT.

**Bedeutung**
Laenge des Kennungsfeldes fuer die Ausgabesaetze kuerzer als die STRING-Angabe.
Anweisung wurde ausgefuehrt, jedoch die STRING-Angabe ignoriert.

**Maßnahme**
RANGE- und STRING-Werte aufeinander abstimmen.

LMS0078 WARNING: OVERFLOW WHILE RENUMBERING
LMS0078 WARNUNG: BEIM NUMERIEREN TRITT UEBERLAUF IM NUMMERNFELD AUF

**Bedeutung**
Die Numerierung wurde ausgefuehrt, jedoch tritt Ueberlauf auf.

LMS0079 WARNING: CHECK FIELDS IN INPUT RECORDS NOT ASCENDING
LMS0079 WARNUNG: NICHT-AUFSTEIGENDE KENNUNGSFELDER IN EINGABE-SAETZEN ERMITTELT

LMS0086 NAME OR OPERAND IS TOO LONG
LMS0086 NAMENS- BZW. OPERANDEN-ANGABE ZU LANG

**Bedeutung**
Die Namens- bzw. Operanden-Angabe darf maximal 8 Zeichen lang sein.

LMS0090 INPUT ALLOWED ONLY FROM IMPLICIT INPUT LIBRARY
LMS0090 EINGABE NUR UEBER IMPLIZITE EINGABE-BIBLIOTHEK ZULAESSIG

**Bedeutung**
Explizite Kurzbezeichnung ist verboten.

LMS0091 WARNING: NO RENUMBERING PERFORMED. NUMERIC VALUE FOR VALUE OPERAND TOO HIGH
LMS0091 WARNUNG: KEINE NUMERIERUNG ERFOLGT. NUMERISCHER WERT IN VALUE ZU GROSS

**Maßnahme**
Mit RANGE-Angabe vergleichen.

LMS0093 MEMBER FOR LOG OUTPUT ALREADY EXISTS
LMS0093 ELEMENT FUER DIE PROTOKOLL-AUSGABE EXISTIERT BEREITS

**Bedeutung**
Protokoll-Ausgabe (PRT-Anweisung) in ein schon vorhandenes Element ist mit OVERWRITE=NO nicht moeglich.

LMS0095 INPUT DATA RECORDS ARE MISSING
LMS0095 EINGABE-DATENSAETZE FEHLEN

LMS0097 VALUE OR STRING OPERAND MISSING. STATEMENT NOT PROCESSED
LMS0097 VERARBEITUNGS-OPERAND VALUE ODER STRING NICHT GESETZT.
ANWEISUNG NICHT AUSGEFUEHRT

**Maßnahme**
Entsprechende Verarbeitungs-Operanden versorgen.

LMS0098 INCONSISTENT OPERANDS. STATEMENT NOT PROCESSED
LMS0098 UNVERTRAEGLICHE VERARBEITUNGS-OPERANDEN. ANWEISUNG NICHT
AUSGEFUEHRT

LMS0099 OPERAND RANGE=NO SPECIFIED. STATEMENT NOT PROCESSED
LMS0099 VERARBEITUNGS-OPERAND RANGE=NO GESETZT. ANWEISUNG NICHT
AUSGEFUEHRT

LMS0100 INVALID DELIMITER
LMS0100 FALSCHER BEGRENZER

LMS0101 AT LEAST ONE OPERAND MISSING IN STATEMENT
LMS0101 IN DER ANWEISUNG FEHLT MINDESTENS EIN OPERAND

LMS0102 WARNING: AT LEAST ONE INCOMPLETE MODULE FOUND IN *OMF
LMS0102 WARNUNG: MINDESTENS EIN UNVOLLSTAENDIGER MODUL IN *OMF

**Bedeutung**
Beim Uebernehmen von Moduln aus der EAM-Bindemodul-Datei durch die 'ADDR *OMF' Anweisung wurde mindestens ein unvollstaendiger Modul gefunden (z.B. wegen Abbruch eines Assemblerlaufs).

LMS0103 RECORD(S) OF ILLEGAL TYPE WERE REMOVED
LMS0103 UNZULAESSIGE SATZARTEN WURDEN ENTFERNT

LMS0104 NAME IS TOO LONG
LMS0104 ZU LANGER NAME

LMS0105 COLUMN IN '*CHANGE...' OF COR STATEMENT INCOMPLETE
LMS0105 SPALTEN-ANGABE IN '*CHANGE...' DER COR-ANWEISUNG IST
UNVOLLSTAENDIG

LMS0106 COLUMN IN '*CHANGE...' OF COR STATEMENT TOO LONG
LMS0106 SPALTEN-ANGABE IN '*CHANGE...' DER COR-ANWEISUNG ZU LANG

LMS0107 LENGTH OF CONTROL TEXT NOT EQUAL TO LENGTH OF UPDATE TEXT
LMS0107 LAENGE VON PRUEFTEXT UNGLEICH LAENGE VON KORREKTURTEXT

**Bedeutung**
Ist bei einer Ersetzung '=:=' angegeben, so muessen der Prueftext und der Korrekturtext die gleiche Laenge haben.

LMS0108 INVALID REPLY FROM USER PROGRAM. CORRECT PROGRAM
LMS0108 UNERLAUBTE ANTWORT DES BENUTZER-PROGRAMMS. PROGRAMM KORRIGIEREN

**Bedeutung**
Die Antwort eines mit USE angeschlossenen Benutzer-Programms ist nicht CON, DEL oder INS.

LMS0109 WARNING: LIST MEMBER IN OLD FORMAT DESTROYED
LMS0109 WARNUNG: LISTEN-ELEMENT IM ALTEN FORMAT IST ZERSTOERT

**Maßnahme**
Evt. Listen-Element kopieren oder neu zuweisen und Vorgang wiederholen.

LMS0110 UPDATE TEXT MISSING
LMS0110 ANGABE DES KORREKTURTEXTES FEHLT

LMS0111 MODIFICATION MISSING
LMS0111 MODIFIKATION FEHLT

LMS0113 CONTROL TEXT IS PARTLY OR ENTIRELY IN RANGE FIELD
LMS0113 PRUEFTEXT LIEGT TEILWEISE ODER VOLLSTAENDIG IM KENNUNGSFELD-BEREICH

LMS0114 UPDATE TEXT IS PARTLY OR ENTIRELY IN RANGE FIELD
LMS0114 KORREKTURTEXT LIEGT TEILWEISE ODER VOLLSTAENDIG IM KENNUNGSFELD-BEREICH

LMS0116 ENTRY-NAME OF USER EXIT MAY NOT START WITH 'LMS'
LMS0116 ENTRY-NAME FUER BENUTZER-AUSGANG DARF NICHT MIT 'LMS' BEGINNEN

**Maßnahme**
ENTRY-Namen aendern.

LMS0117 COLUMN < 1 OR > 80 INVALID
LMS0117 SPALTE < 1 BZW. > 80 UNZULAESSIG

LMS0118 LMS TERMINATED ABNORMALLY
LMS0118 LMS ABNORMAL BEENDET

LMS0119 FIRST COLUMN CANNOT BE HIGHER THAN SECOND COLUMN
LMS0119 ERSTE SPALTEN-ANGABE DARF NICHT GROESSER ALS ZWEITE SPALTEN-ANGABE SEIN

LMS0121 RECORD NUMBER IN CORRECTION STATEMENT INCORRECT
LMS0121 SATZNUMMER IN KORREKTUR-ANWEISUNG FALSCH DEFINIERT

**Bedeutung**
Die Satznummer hat folgendes Format: #<zahl>.
<zahl>: Maximal 8-stellige, positive Zahl.

LMS0123 COLUMN IS IN RANGE FIELD
LMS0123 SPALTEN-ANGABE LIEGT IM KENNUNGSFELD-BEREICH

LMS0124 NO COLUMN SPECIFIED
LMS0124 SPALTEN-ANGABE FEHLT

LMS0125 RIGHT STRING DELIMITER MISSING
LMS0125 RECHTER STRING-BEGRENZER FEHLT

LMS0126 ILLEGAL /INTR COMMAND SPECIFIED
LMS0126 UNZULAESSIGES /INTR-KOMMANDO ANGEGEBEN

LMS0127 LENGTH OF CHECK FIELD DOES NOT MATCH CHECK LENGTH
LMS0127 LAENGE DES KENNUNGSFELDES STIMMT NICHT MIT CHECK-LAENGE UEBEREIN

LMS0128 NO CHECK FIELD DEFINED. SUBSEQUENT RECORD INSERTED AT CURRENT POSITION
LMS0128 KEIN KENNUNGSFELD DEFINIERT. DER FOLGENDE SATZ WIRD AN AKTUELLER POSITION EINGEFUEGT

LMS0129 WARNING: LMS STATEMENT ABORTED DUE TO INTERRUPT
LMS0129 WARNUNG: LMS-ANWEISUNG WEGEN EINER UNTERBRECHUNG BEENDET

LMS0131 MEMBER EMPTY AFTER PROCESSING COR STATEMENT. MEMBER NOT CORRECTED
LMS0131 ELEMENT NACH ANWEISUNGS-AUSFUEHRUNG LEER. ELEMENT NICHT KORRIGIERT

**Bedeutung**
Im Zuge einer Korrektur (COR-Anweisung) wurden alle Saetze des zu korrigierenden Elements geloescht.

LMS0132 PROCESSED MEMBER STILL IN FILE '(&00)'. USE ADD STATEMENT TO ADD FILE TO LIBRARY
LMS0132 BEARBEITETES ELEMENT NOCH IN HILFS-DATEI '(&00)'. HILFS-DATEI MIT ADD-ANWEISUNG IN BIBLIOTHEK AUFNEHMEN

**Bedeutung**
Rueckschreiben des Elements nach Rueckkehr vom Editor nicht moeglich.

LMS0133 LMS CANNOT CALL EDT. REDEFINE '@' ESCAPE CHARACTER
LMS0133 LMS KANN EDT NICHT AUFRUFEN. FLUCHTSYMBOL '@' REDEFINIEREN

LMS0134 EDITOR TERMINATED ABNORMALLY
LMS0134 EDITOR ABNORMAL BEENDET

**Bedeutung**
Elementsaetze mit einer Satzlaenge =256 vorhanden.
Elementsaetze mit einer Satzlaenge = 4 vorhanden.

LMS0135 (LIB) OF PRIMARY MEMBER WRONG OR NOT ASSIGNED
LMS0135 (LIB) DES PRIMAER-ELEMENTS FALSCH ODER NICHT ZUGEWIESEN

LMS0138 CHARACTER '(&00)' MISSING IN OPERAND
LMS0138 IM OPERANDEN FEHLT ZEICHEN '(&00)'

LMS0139 LIBIN ASSIGNMENT WRONG OR MISSING
LMS0139 FALSCHE ODER FEHLENDE LIBIN-ZUWEISUNG

LMS0140 WARNING: SPECIFIED STRING LONGER THAN ISAM KEY. NO STRING INSERTED
LMS0140 WARNUNG: ZEICHENFOLGE IM STRING-OPERANDEN LAENGER ALS ISAM-SCHLUESSEL. KEINE ZEICHENFOLGE EINGEFUEGT

LMS0141 CRT STATEMENT NO LONGER SUPPORTED. USE COR STATEMENT
LMS0141 CRT-ANWEISUNG WIRD NICHT MEHR UNTERSTUETZT. COR-ANWEISUNG VERWENDEN

LMS0142 WARNING: CHARACTER(S) NOT EQUAL TO 'BLANK' LOST IN FOLLOWING RECORD
LMS0142 WARNUNG: ZEICHEN UNGLEICH LEERZEICHEN GINGEN IM FOLGENDEN SATZ VERLOREN

**Bedeutung**
Im folgenden Satz gingen durch Einfuegungen oder Ersetzungen Zeichen ungleich Zwischenraum verloren.

LMS0143 WARNING: ISAM KEY TRUNCATED FROM RIGHT WHEN MOVED TO CHECK FIELD
LMS0143 WARNUNG: ISAM-SCHLUESSEL BEI UEBERNAHME INS KENNUNGSFELD RECHTS ABGESCHNITTEN

LMS0144 KEY NOT ASCENDING OR OVERFLOW DURING NUMBERING
LMS0144 SCHLUESSEL NICHT AUFSTEIGEND ODER UEBERLAUF BEI NUMERIERUNG

**Bedeutung**
Der Inhalt des Kennungsfeldes zur Bildung des ISAM-Schluessels ist nicht aufsteigend oder bei der Generierung des ISAM-Schluessels ueber den Verarbeitungs-Operanden VALUE trat ein Ueberlauf auf.

LMS0145 MEMBER NAME NOT PERMITTED FOR PLAM LIBRARY
LMS0145 ELEMENT-NAME FUER PLAM-BIBLIOTHEK UNZULAESSIG

**Bedeutung**
Z.B. unterschiedliche Element-Bezeichnung fuer PLAM und OSM-Bibliotheken.

LMS0146 LOG OUTPUT NOT POSSIBLE, AS OVERWRITE=ONLY SPECIFIED THOUGH NO MEMBER WITH SAME NAME EXISTS
LMS0146 PROTOKOLL-AUSGABE NICHT MOEGLICH, DA OVERWRITE=ONLY UND KEIN MEMBER WITH SAME NAME EXISTS
LMS0146 PROTOKOLL-AUSGABE NICHT MOEGLICH, DA OVERWRITE=ONLY UND KEIN ELEMENT MIT GLEICHEM NAMEN EXISTIERT

**Bedeutung**
Protokollausgabe (PRT- oder OUT-Anweisung) ist bei OVERWRITE=ONLY nicht moeglich, wenn kein gleichnamiges Element existiert.

LMS0147 TEXT AREA TO BE CORRECTED OVERLAPS REP AREA SPECIFIED IN REP STATEMENT
LMS0147 ZU KORRIGIERENDER TEXT-BEREICH UBERSCHNEIDET SICH MIT DEM IN DER REP-ANWEISUNG DEFINIERTEN REP-BEREICH

LMS0148 NEWLIB OPERAND MISSING IN LIBOUT STATEMENT FOR SEQUENTIAL LIBRARY
LMS0148 NEWLIB-OPERAND FEHLT IN LIBOUT-ANWEISUNG FUER SEQUENTIELLE BIBLIOTHEK

**Bedeutung**
Bei Ausgabe in sequentielle Bibliotheken muss stets der Operand NEWLIB in der LIBOUT-Anweisung angegeben werden.

LMS0149 EDT OR EDOR CANNOT BE LOADED
LMS0149 EDT BZW. EDOR KOENNEN NICHT NACHGELADEN WERDEN

LMS0150 USER ROUTINE CANNOT BE LOADED
LMS0150 BENUTZER-ROUTINE KONNTE NICHT NACHGELADEN WERDEN

**Bedeutung**
Benutzer-Routine existiert nicht.

LMS0151 WARNING: DEFAULT VALUES GENERATED FOR INCORRECT CTL OR PRT STATEMENT
LMS0151 WARNUNG: STANDARDWERTE FUER FEHLERHAFTE CTL- BZW. PRT-ANWEISUNG EINGESTELLT

LMS0152 USER EXIT FOR COM STATEMENT NOT ALLOWED WHEN GENERATING COR STATEMENTS FROM COM
LMS0152 BEI DER GENERIERUNG VON COR-ANWEISUNGEN AUS COM IST KEIN BENUTZER-AUSGANG FUER DIE COM-ANWEISUNG ERLAUBT

LMS0153 GENERATION OF 'COR' STATEMENT ABORTED PREMATURELY
LMS0153 GENERIERUNG VON COR-ANWEISUNGEN VORZEITIG ABGEBROCHEN

LMS0154 WARNING: A RECORD NUMBER WAS GENERATED IN AT LEAST ONE COR STATEMENT
LMS0154 WARNUNG: IN MINDESTENS EINER COR-ANWEISUNG WURDE EINE SATZNUMMER ERZEUGT

**Bedeutung**
Obwohl PAR CHECK ungleich NO gesetzt ist, wurde eine Satznummer erzeugt.

LMS0155 NO CORRECTION BY COR SUBSTATEMENT POSSIBLE
LMS0155 KEINE KORREKTUR DURCH COR-SUBANWEISUNG MOEGLICH

**Bedeutung**
Die Kennung bzw. Satznummer ist niedriger als die aktuelle Kennung bzw. Satznummer.

LMS0156 CHANGE CONTINUATION STATEMENT EXPECTED
LMS0156 CHANGE-FORTSETZUNGS-ANWEISUNG ERWARTET

LMS0157 MEMBER TYPE ONLY ALLOWED FOR PLAM-LIBRARIES
LMS0157 ELEMENT-TYP NUR FUER PLAM-BIBLIOTHEKEN ERLAUBT

LMS0158 SYS STATEMENT CANNOT BE PROCESSED BECAUSE OF INCORRECT '(&00)'
LMS0158 SYS-ANWEISUNG KANN WEGEN FALSCHEM '(&00)' NICHT AUSGEFUEHRT WERDEN

LMS0159 FMS OR $FMSLIB DOES NOT EXIST
LMS0159 FMS ODER $FMSLIB NICHT GEFUNDEN

**Bedeutung**
FMS kann in der vom DBL verwendeten Modulbibliothek nicht gefunden werden oder es gibt keine $FMSLIB.

LMS0160 THE OUTPUT OF THE ELEMENT AND THE LMS LOG ARE NOT PERMITTED TO SAME OSM LIBRARY
LMS0160 DIE AUSGABE EINES BIBLIOTHEKSELEMENTS UND DES LMS-PROTOKOLLS DUERFEN NICHT IN DIESELBE OSM BIBLIOTHEK ERFOLGEN

LMS0162 RANGE LIMITS IN LIBRARY WILL SOON BE REACHED (SATURATION). REORGANIZE LIBRARY
LMS0162 BEREICHSGRENZEN (SAETTIGUNG) IN BIBLIOTHEK IN KUERZE ERREICHT. BIBLIOTHEK REORGANISIEREN

LMS0163 WARNING: AT LEAST ONE RECORD TRUNCATED
LMS0163 WARNUNG: MINDESTENS EIN SATZ WURDE ABGESCHNITTEN

LMS0164 WARNING: TABLE OF CONTENTS (TOC) COULD NOT BE UPDATED DUE TO ABNORMAL PROGRAM TERMINATION
LMS0164 WARNUNG: INHALTSVERZEICHNIS (TOC) WEGEN PROGRAMM-ABBRUCH NICHT AKTUALISIERT

**Maßnahme**
Bibliothek erneut zum Schreiben eroeffnen, wobei automatisch das Inhaltsverzeichnis aktualisiert wird.

LMS0165 WARNING: TABLE OF CONTENTS (TOC) HAS BEEN UPDATED
LMS0165 WARNUNG: INHALTSVERZEICHNIS (TOC) WURDE AKTUALISIERT

**Bedeutung**
Inhaltsverzeichnis (TOC) wurde beim Eroeffnen der Bibliothek zum Schreiben aktualisiert.

LMS0166 OVERWRITE OPERAND HAS ILLEGAL VALUE
LMS0166 OVERWRITE-VERARBEITUNGS-OPERAND IST UNZULAESSIG

**Bedeutung**
Es wurde V, D oder EXTEND angegeben.

LMS0167 SPECIFIED LINK NAME CANNOT BE ASSIGNED TO ANY FILE
LMS0167 ANGEGEBENER LINK-NAME KANN KEINER DATEI ZUGEORDNET WERDEN

**Maßnahme**
Dem LINK-Namen eine Datei zuweisen.

LMS0168 FILE FORMAT OF '(&00)' FILE NOT SUPPORTED FOR THIS STATEMENT
LMS0168 DATEI-FORMAT DER (&00)-DATEI WIRD FUER DIESE ANWEISUNG NICHT UNTERSTUETZT

**Bedeutung**
Der FCB-Typ der Datei, die mit ADD aufgenommen werden soll, ist nicht erlaubt.

LMS0169 DMS ERROR '(&00)' ON PROCESSING '(&01)' FILE. ERROR INFO IN SYSTEM MODE: /HELP DMS(&00),INF=D
LMS0169 DVS-FEHLER '(&00)' BEI VERARBEITUNG DER (&01)-DATEI. FEHLER-INFO IM SYSTEM-MODUS: /HELP DMS(&00),INF=D

**Bedeutung**
Naehere Information ueber den DVS-Fehlercode kann ueber das /HELP-Kommando im System-Modus erfragt bzw. den BS2000-Manualen 'Systemmeldungen', 'DVS-Plattenverarbeitung' bzw. 'DVS-Bandverarbeitung' entnommen werden.

**Maßnahme**
Siehe entsprechenden DVS-Fehlercode.

LMS0170 NEXT RECORD IN INPUT MEMBER NOT IN CORRECT ORDER
LMS0170 NAECHSTER SATZ IM EINGABE-ELEMENT NICHT AUFSTEIGEND NUMERIERT

**Bedeutung**
Das Eingabeelement ist nicht aufsteigend numeriert.

LMS0171 RECORD NUMBER OR CHECK FIELD IN PRECEDING CORRECTION STATEMENT NOT IN CORRECT ORDER. CORRECTION NOT PERFORMED
LMS0171 SATZNUMMER ODER KENNUNGSFELD IN DER VORANGEHENDEN KORREKTUR-ANWEISUNG NICHT AUFSTEIGEND NUMERIERT. KORREKTUR NICHT AUSGEFUEHRT

**Bedeutung**
Die Korrektur-Anweisungen fuer COR sind nicht aufsteigend sortiert.
Der Operand in der vorangehenden Korrektur-Anweisung fuer COR ist nicht hoeher als die in frueheren Korrektur-Anweisungen angegebenen Operanden.

LMS0172 NO CHECK FIELD DEFINED. INPUT MEMBER NOT CORRECTED
LMS0172 KEIN KENNUNGS-FELD DEFINIERT. EINGABE-ELEMENT NICHT KORRIGIERT

**Bedeutung**
Obwohl CHECK=NO gesetzt ist, wurde in einer Korrektur-Anweisung eine Kennung angegeben. Es erfolgt keine Standard-Einstellung fuer CHECK.

LMS0173 RECORD NUMBER OR CHECK FIELD IN CORRECTION STATEMENT NOT IN CORRECT ORDER
LMS0173 SATZNUMMER ODER KENNUNGSFELD IN DER KORREKTUR-ANWEISUNG NICHT AUFSTEIGEND NUMERIERT. KORREKTUR NICHT AUSGEFUEHRT

**Bedeutung**
Die Satznummer oder das Kennungsfeld in dem folgenden Satz der Korrektur-Anweisung ist nicht hoeher als die in vorangehenden Anweisungen fuer COR angegebenen.

LMS0174 RECORD NUMBER OR CHECK FIELD IN NEXT DATA RECORD LOWER THAN CURRENT POSITION IN INPUT MEMBER. NO CORRECTION USING 'COR' POSSIBLE
LMS0174 SATZNUMMER ODER KENNUNGSFELD IM FOLGENDEN DATENSATZ IST NIEDRIGER ALS DIE AKTUELLE POSITION IM EINGABE-ELEMENT. KEINE KORREKTUR MIT 'COR' MOEGLICH

LMS0175 WARNING: EDITOR CALLED, BUT LMS IS IN TEST MODE. NO CORRECTIONS MADE.

LMS0175 WARNUNG: EDITOR AUFGERUFEN, LMS ABER IM TEST-MODUS. KORREKTUREN NICHT AUSGEFUEHRT.

**Bedeutung**
LMS befindet sich im Test-Modus; damit wird nicht in die Ausgabe-Bibliothek geschrieben.
Mit Hilfe der Editoren durchgefuehrte Korrekturen gehen verloren.

**Maßnahme**
Verarbeitungsoperand TEST=NO setzen oder fehlerhafte Prozedur korrigieren und nochmals editieren.

LMS0176 WARNING: ONLY INPUT MEMBER WITH HIGHEST VERSION IN FMS LIBRARY INCLUDED IN LIBRARY

LMS0176 WARNUNG: NUR DAS EINGABE-ELEMENT MIT HOECHSTER VERSION IN DER FMS-BIBLIOTHEK WURDE IN DIE BIBLIOTHEK AUFGENOMMEN

LMS0177 TOO MANY DELIMITERS SPECIFIED

LMS0177 ZU VIELE BEGRENZER ANGEGEBEN

LMS0178 INTERNAL ERROR '(&00)' WHEN WRITING A RECORD

LMS0178 INTERNER FEHLER '(&00)' BEIM SCHREIBEN EINES SATZES

LMS0179 INPUT MEMBER '(&00)' OF TYPE 'C' IS NOT A BS2000 PHASE. ERROR CODE '(&01)'

LMS0179 DAS EINGABE-ELEMENT '(&00)' VOM TYP 'C' IST KEINE BS2000-PHASE. FEHLERCODE '(&01)'

LMS0180 MEMBER TYPE 'C' ONLY PERMITTED WHEN INPUT AS WELL AS OUTPUT LIBRARY IS A TAPE OR A PLAM LIBRARY

LMS0180 ELEMENT-TYP 'C' NUR ERLAUBT, WENN EIN- UND AUSGABE-BIBLIOTHEK ENTWEDER BAND- ODER PLAM-BIBLIOTHEKEN SIND

LMS0181 MEMBER TYPE 'C' ONLY PERMITTED WHEN INPUT LIBRARY IS A TAPE OR PLAM LIBRARY

LMS0181 ELEMENT-TYP 'C' NUR ERLAUBT, WENN DIE EINGABE-BIBLIOTHEK EINE BAND- ODER PLAM-BIBLIOTHEK IST

LMS0182 MEMBER TYPE 'C' ONLY PERMITTED WHEN OUTPUT LIBRARY IS A TAPE OR PLAM LIBRARY

LMS0182 ELEMENT-TYP 'C' NUR ERLAUBT, WENN DIE AUSGABE-BIBLIOTHEK EINE BAND- ODER PLAM-BIBLIOTHEK IST

LMS0183 MISSING SUBSTATEMENTS WHEN CORRECTING USING UPD STATEMENT

LMS0183 BEI KORREKTUR MIT UPD-ANWEISUNG FEHLEN DIE SUBANWEISUNGEN

LMS0184 IDENTIFICATION SPECIFIED IN UPD SUBSTATEMENT DOES NOT EXIST. CHECK ID
LMS0184 DIE BEI DER UPD-SUBANWEISUNG ANGEGEBENE IDENTIFIKATION WURDE NICHT GEFUNDEN. ID PRUEFEN

LMS0185 SPECIFIED SEGMENT OF LOAD MODULE TO BE PROCESSED DOES NOT EXIST
LMS0185 DEFINIERTES SEGMENT DES ZU BEARBEITENDEN LADEMODULS EXISTIERT NICHT

LMS0186 UPDATE TEXT OF MEMBER TO BE CORRECTED NOT CONTAINED WITHIN MEMBER
LMS0186 KORREKTURTEXT DES ZU KORRIGIERENDEN ELEMENTS IST AUSSERHALB DES ELEMENTS

LMS0187 CONTROL TEXT OF MEMBER TO BE CORRECTED NOT CONTAINED WITHIN MEMBER
LMS0187 PRUEFTEXT DES ZU KORRIGIERENDEN ELEMENTS IST AUSSERHALB DES ELEMENTS

LMS0188 CROSS CONTROL NUMBER INCORRECT IN UPD SUBSTATEMENT. CHECK CONTROL NUMBER
LMS0188 FEHLER IN CROSS-KONTROLLZAHL BEI UPD-SUBANWEISUNG. KONTROLLZAHL PRUEFEN

**Bedeutung**
Z.B. CROSS-Kontrollzahl >FFFFFF.

LMS0189 RECORD TYPES NOT EQUAL TO '1' ONLY PERMITTED FOR PLAM LIBRARIES
LMS0189 SATZARTEN UNGLEICH '1' SIND NUR FUER PLAM-BIBLIOTHEKEN ZULAESSIG

LMS0190 NAME DEFINED IN *NAM SUBSTATEMENT ALREADY EXISTS
LMS0190 MIT *NAM-SUBANWEISUNG DEFINIERTER NAME EXISTIERT BEREITS

**Bedeutung**
Mit der *NAM-Subanweisung sollte ein Name definiert werden, der bereits im Element vorkommt bzw. in einer vorangehenden *NAM-Subanweisung bereits erzeugt wurde.

LMS0191 REF OPERAND ONLY SUPPORTED FOR PLAM LIBRARIES AND MEMBERS OF TYPE 'R'
LMS0191 VERARBEITUNGS-OPERAND REF WIRD NUR FUER PLAM-BIBLIOTHEKEN UND ELEMENT-TYP 'R' UNTERSTUETZT

LMS0192 CSECT DEFINED IN A SUBSTATEMENT DOES NOT EXIST IN MODULE. CHECK CSECT NAME

LMS0192 DIE IN EINER SUBANWEISUNG DEFINIERTE CSECT WURDE IM MODUL NICHT GEFUNDEN. CSECT-NAMEN PRUEFEN

LMS0193 NAME TO BE RENAMED USING *NAM SUBSTATEMENT DOES NOT EXIST IN MODULE

LMS0193 DER MIT *NAM-SUBANWEISUNG UMZUBENENNENDE NAME WURDE IM MODUL NICHT GEFUNDEN

LMS0194 BINARY DEFINITION OF CONTROL TEXT OR UPDATE TEXT NOT PERMITTED IN '*COR' SUBSTATEMENT OF UPDC STATEMENT

LMS0194 BEI DER UPDC-ANWEISUNG DARF IN DER *COR-SUBANWEISUNG DER PRUEF- BZW. KORREKTUR-TEXT NICHT BINAER DEFINIERT WERDEN

LMS0195 FILE CANNOT BE ADDED, BECAUSE RECSIZE IS LARGER THAN '(&00)'

LMS0195 DATEI KANN NICHT AUFGENOMMEN WERDEN, DA SATZLAENGE GROESSER '(&00)' IST

LMS0196 MEMBER CANNOT BE EDITED, NUMBERED OR CORRECTED DUE TO '(&00)'

LMS0196 BIBLIOTHEKS-ELEMENT KANN NICHT EDITIERT, NUMERIERT ODER KORRIGIERT WERDEN, WEIL '(&00)'

**Bedeutung**
(&00): RECFORM=F oder KEYPOS >5 oder KEYLEN >16.

LMS0197 ELEMENT CANNOT BE EXTENDED DUE TO '(&00)'

LMS0197 BIBLIOTHEKS-ELEMENT KANN NICHT ERWEITERT WERDEN, WEIL '(&00)'

**Bedeutung**
(&00): RECFORM=F oder KEYPOS <>5 oder KEYLEN > 16.

LMS0198 ABSOLUTE CORRECTION ADDRESS TOO HIGH IN UPD SUBSTATEMENT

LMS0198 BEI UPD-SUBANWEISUNG IST DIE ABSOLUTE KORREKTUR-ADRESSE ZU HOCH

**Bedeutung**
Bei einer UPD-Subanweisung ist die Summe von Basis- und Korrektur-Adresse > 7FFFFFFF.

LMS0199 WARNING:INVALID RECORD LENGTH
LMS0199 WARNUNG: SATZLAENGE UNGUELTIG

**Bedeutung**
Ein oder mehrere Saetze eines Bibliothekselements, das mit festem Satzformat aufgenommen worden ist, enthalten ein falsches Satzlaengenfeld, das nicht mit dem intern gespeicherten Wert uebereinstimmt.
Der Satz mit aktueller Satzlaenge wird bearbeitet.

LMS0200 A PHASE BOUNDED VIA COREIM=NO CANNOT BE UPDATED USING A UPDC STATEMENT
LMS0200 MIT COREIM=NO GEBUNDENER LADEMODUL DARF NICHT MIT UPDC-ANWEISUNG KORRIGIERT WERDEN

LMS0201 WARNING: ONLY COMPARE-AREA OF RECORDS WILL BE LOGGED
LMS0201 WARNUNG: NUR DER VERGLEICHS-BEREICH DER SAETZE WIRD PROTOKOLLIERT

LMS0202 FUNCTION NOT PERMITTED FOR STORED PAM FILE
LMS0202 FUNKTION IST FUER DIE GESPEICHERTE PAM-DATEI NICHT ERLAUBT

LMS0203 NO ENTRY EXISTS FOR SPECIFIED REFERENCE CONDITION
LMS0203 KEIN EINTRAG FUER ANGEGEBENE REFERENZ-BEDINGUNG VORHANDEN

**Bedeutung**
Fuer die angegebene Referenzbedingung (PAR REF=...) gibt es keinen Inhaltsverzeichniseintrag.

LMS0204 THIS STATEMENT IS ONLY ALLOWED FOR BS2000-PHASES IN PLAM-LIBRARIES
LMS0204 DIESE ANWEISUNG IST NUR FUER BS2000-PHASEN IN PLAM-BIBLIOTHEKEN ERLAUBT

LMS0205 RENAME NOT ALLOWED FOR DELTA MEMBERS
LMS0205 DELTA-ELEMENTE DUERFEN NICHT UMBENANNT WERDEN

LMS0206 WARNING: PROGRAM ALREADY LOADED
LMS0206 WARNUNG: PROGRAMM BEREITS GELADEN

**Bedeutung**
EDT / EDOR / FMS / PLAM ist bereits geladen.

LMS0207 LIBRARY MEMBERS THAT HAVE FORMAT-B RECORDS CANNOT BE STORED AS DELTA MEMBERS
LMS0207 BIBLIOTHEKS-ELEMENTE MIT FORMAT-B-SAETZEN KOENNEN NICHT ALS DELTA-ELEMENTE ABGESPEICHERT WERDEN

LMS0208 DELTA MEMBER AND BASE MEMBER MUST NOT BE THE SAME.
LMS02D8 DELTA-ELEMENT UND BASIS-ELEMENT DUERFEN NICHT GLEICH SEIN.

**Maßnahme**
Andere Version fuer das Delta-Element vergeben.

LMS0209 OVERWRITE=EXTEND NOT ALLOWED WHEN CREATING DELTA MEMBERS
LMS02D9 OVERWRITE=EXTEND BEIM ERZEUGEN VON DELTA-ELEMENTEN NICHT ERLAUBT

LMS0210 ERROR WHEN CLOSING LOG MEMBER
LMS0210 FEHLER BEIM SCHLIESSEN DES PROTOKOLL-ELEMENTS

LMS0211 LIBRARY ALREADY EXISTS
LMS0211 BIBLIOTHEK EXISTIERT BEREITS

LMS0212 *END STATEMENT MISSING
LMS0212 *END-ANWEISUNG FEHLT

LMS0213 NAME ALREADY EXISTS AS DELTA ELEMENT
LMS0213 NAME EXISTIERT BEREITS ALS DELTA-ELEMENT

LMS0214 NAME FOR DELTA MEMBER ALREADY EXISTS AS FULL MEMBER
LMS0214 NAME FUER DELTA-ELEMENT EXISTIERT BEREITS ALS VOLL-ELEMENT

LMS0215 STATEMENT NOT ALLOWED FOR PRELINKED MODULES WITH COMPLETE 'ESD'
LMS0215 ANWEISUNG IST BEI GROSSMODULN MIT KOMPLETTEM 'ESD' NICHT ERLAUBT

**Bedeutung**
Diese Anweisung ist bei Grossmoduln mit neuem Format nicht erlaubt.

LMS0216 USE *REP STATEMENT WITH ABSOLUTE ADDRESS FOR PRELINKED MODULES WITH COMPLETE 'ESD'
LMS0216 BEI GROSSMODULN MIT KOMPLETTEM 'ESD' *REP-ANWEISUNG MIT ABSOLUTER ADRESSE BENUTZEN

**Bedeutung**
Bei Grossmoduln im neuen Format muss fuer die *REP-Anweisung die absolute Adresse innerhalb des Grossmoduls verwendet werden.

LMS0217 USE *COR STATEMENT WITH CSECT NAME AND DISTANCE FOR PRELINKED MODULES WITH COMPLETE 'ESD'
LMS0217 BEI GROSSMODULN MIT KOMPLETTEM 'ESD' *COR-ANWEISUNG MIT CSECT-NAMEN UND RELATIVER ADRESSE BENUTZEN

**Bedeutung**
Bei Grossmoduln im neuen Format muss fuer die *COR-Anweisung der CSECT-Namen und die Distanz und nicht die absolute Adresse angegeben werden.

LMS0218 SYNTAX ERROR IN SPECIFIED BASE VERSION
LMS0218 SYNTAX-FEHLER IN BASIS-VERSIONS-ANGABE

LMS0219 DELTA MEMBER ONLY ALLOWED FOR PLAM LIBRARIES
LMS0219 DELTA-ELEMENT NUR BEI PLAM-BIBLIOTHEKEN ERLAUBT

LMS0220 VERSION NOT ALLOWED FOR THIS FUNCTION
LMS0220 VERSIONS-ANGABE FUER DIESE FUNKTION NICHT ERLAUBT

LMS0221 DATE NOT ALLOWED FOR THIS FUNCTION
LMS0221 DATUM FUER DIESE FUNKTION NICHT ERLAUBT

LMS0222 NAME LIST NOT ALLOWED FOR THIS FUNCTION
LMS0222 FUER DIESE FUNKTION IST KEINE NAMENS-LISTE ERLAUBT

LMS0223 NO PLAM-LIBRARY
LMS0223 KEINE PLAM-BIBLIOTHEK

LMS0224 DELTA AND BASE ELEMENT MUST HAVE SAME TYPE AND NAME
LMS0224 DELTA- UND BASIS-ELEMENT MUESSEN DEN GLEICHEN TYP UND NAMEN BESITZEN

LMS0225 BUFFER LENGTH < 4
LMS0225 PUFFER-LAENGE < 4

LMS0226 RECORD LENGTH < 4 OR > 32K
LMS0226 SATZLAENGE < 4 ODER > 32K

**Bedeutung**
Die Satzlaenge muss zwischen 4 und 32K liegen

LMS0227 ELEMENT NOT OPENED
LMS0227 ELEMENT NICHT GEOEFFNET

LMS0228 NAME ALREADY EXISTING
LMS0228 NAME EXISTIERT BEREITS

LMS0229 ILLEGAL TID
LMS0229 ILLEGALE TID

LMS0230 ILLEGAL ELEMENT MASK
LMS0230 ILLEGALE ELEMENTMASKE

LMS0231 NO TOCPRIM/TOCSEC ACTIVE
LMS0231 KEIN TOCPRIM / TOCSEC AKTIV

LMS0232 UNABLE TO LINK L M S U P
LMS0232 L M S U P KANN NICHT NACHGELADEN WERDEN

LMS0233 INVALID CONTROL BLOCK VERSION
LMS0233 UNGUELTIGE KONTROLLBLOCKVERSION

LMS0234 INVALID MAXIMUM LENGTH OF LIBRARY NAME
LMS0234 UNGUELTIGE MAXIMALLAENGE DES BIBLIOTHEKSNAMEN

LMS0235 MASK ITEM TOO LONG
LMS0235 MASKENFELD ZU LANG

LMS0236 INVALID SUBCODE
LMS0236 UNGUELTIGER SUBCODE

LMS0240 WARNING: CLOSING BRACKET WAS ADDED
LMS0240 WARNUNG: SCHLIESSENDE KLAMMER WURDE HINZUGEFUEGT

LMS0247 SYNTAX ERROR: ILLEGAL MEMBER TYPE DEFINED
LMS0247 SYNTAX-FEHLER: UNZULAESSIGER ELEMENT-TYP GESETZT

LMS0248 WARNING: OPERAND VALUE OVERWRITTEN BY LAST VALUE SPECIFIED
LMS0248 WARNUNG: OPERANDEN-WERT DURCH LETZTE ANGABE UEBERSCHRIEBEN

**Bedeutung**
In einem Anweisungsblock wurde ein Operand mehrmals gesetzt und somit mit dem letzten Wert ueberschrieben.

LMS0249 UPDATE NOT ALLOWED FOR DELTA-ELEMENTS
LMS0249 UPDATE FUER DELTA-ELEMENTE NICHT ERLAUBT

LMS0250 SYNTAX ERROR: ILLEGAL OPERAND VALUE
LMS0250 SYNTAX-FEHLER: UNZULAESSIGER OPERANDEN-WERT

LMS0251 PAR PHASE=PK IN NK-WORLD NOT ALLOWED
LMS0251 PAR PHASE=PK IN DER NK-WELT NICHT ERLAUBT

**Bedeutung**
Die Erzeugung von PAMKEY-behafteten Phasen in der NON-PAMKEY-Welt ist nicht moeglich

LMS0252 SYNTAX ERROR: "=" MISSING BEHIND OPERAND
LMS0252 SYNTAX-FEHLER: NACH OPERANDEN FEHLT '=' ZEICHEN

LMS0253 SYNTAX ERROR: KEYWORD OPERAND MISSING
LMS0253 SYNTAX-FEHLER: EIN SCHLUESSELWORT-OPERAND FEHLT

LMS0254 SYNTAX ERROR: ILLEGAL KEYWORD SPECIFIED
LMS0254 SYNTAX-FEHLER: UNZULAESSIGES SCHLUESSELWORT DEFINIERT

LMS0255 SYNTAX ERROR: ABBREVIATION OF OPERAND IS NOT UNIQUE
LMS0255 SYNTAX-FEHLER: ABKUERZUNG FUER VERARBEITUNGS-OPERANDEN NICHT EINDEUTIG

LMS0256 INVALID RECORD TYPE
LMS0256 UNGUELTIGE SATZART

**Bedeutung**
Satzart nicht 1 - 159 , 163 , 164

LMS0257 THE NAME OF DELTA AND FULL ELEMENTS MUST BE DIFFERENT
LMS0257 DELTA- UND VOLL-ELEMENTE MUESSEN UNTERSCHIEDLICHE NAMEN BESITZEN

LMS0258 INPUT ELEMENT IS EMPTY
LMS0258 EINGABEELEMENT IST LEER

LMS0259 PRIMARY AND SECONDARY ELEMENT ARE EMPTY
LMS0259 PRIMAER- UND SEKUNDAERELEMENT SIND LEER

LMS0260 OUTPUT MEMBER CANNOT BE EXTENDED
LMS0260 AUSGABE-ELEMENT KANN NICHT ERWEITERT WERDEN

LMS0261 MEMBER CANNOT BE EXTENDED DUE TO DIFFERENT '(&00)' SPECIFICATIONS
LMS0261 ELEMENT KANN WEGEN UNTERSCHIEDLICHER (&00)-ANGABEN NICHT ERWEITERT WERDEN

**Bedeutung**
Das Element kann nicht erweitert werden, da die im Element gespeicherten Dateimerkmale mit den Merkmalen der Datei nicht uebereinstimmen.
(&00): Datei-Merkmale (FILETYPE, RECFORM, RECSIZE, KEYPOS,KEYLEN, LOGLEN oder VALLEN).

LMS0262 MEMBER CANNOT BE EXTENDED, AS ISAM KEYS ARE STORED IN MEMBER
LMS0262 ELEMENT KANN NICHT ERWEITERT WERDEN, WEIL IM ELEMENT ISAM-SCHLUESSEL GESPEICHERT SIND

LMS0263 FILE CANNOT BE ADDED BECAUSE OUTPUT LIBRARY IS NOT A PLAM LIBRARY AND '(&00)'
LMS0263 DATEI KANN NICHT AUFGENOMMEN WERDEN. DIE BIBLIOTHEK IST KEINE PLAM-BIBLIOTHEK UND '(&00)'

**Bedeutung**
Datei kann nicht aufgenommen werden, da die Ausgabe-Bibliothek keine Programmmbibliothek ist und
(&00): PAR KEY=YES oder
(&00): die Datei die Merkmale RECFORM=F oder KEYPOS=5 oder KEYLEN>16 besitzt

LMS0264 FILE CANNOT BE ADDED BECAUSE '(&00)'
LMS0264 DATEI KANN WEGEN '(&00)' NICHT AUFGENOMMEN WERDEN

**Bedeutung**
Datei kann nicht aufgenommen werden, da die Datei entweder RECFORM=F oder KEYPOS>5 oder KEYLEN>16 besitzt und PAR KEY=NO gesetzt ist.

LMS0265 INTERNAL LMS ERROR WHEN WRITING A MEMBER RECORD
LMS0265 INTERNER LMS-FEHLER BEIM SCHREIBEN EINES ELEMENT-SATZES

LMSD266 INPUT MEMBER IS NOT A BS2000 PHASE
LMS0266 DAS EINGABE-ELEMENT IST KEINE BS2000-PHASE

LMS0302 NO '(&00)' (&01' '), INPUT DOES NOT EXIST
LMS0302 KEIN '(&00)' (&01' '), EINGABE NICHT GEFUNDEN

**Bedeutung**
(&00): Jeweilige Anweisung
(&01): Elementbezeichnung.

LMS0303 NO '(&00)' (&01' '), NOT IN RANGE OF REFERENCE CONDITION
LMS0303 KEIN '(&00)' (&01' '), NICHT IM BEREICH DER REFERENZ-BEDINGUNG

**Bedeutung**
(&00): Jeweilige Anweisung
(&01): Elementbezeichnung.

LMS0305 TERMINATION CODE '(&00)'. LMS SWITCHES INTO TEST MODE
LMS0305 BEENDIGUNGS-SCHLUESSEL '(&00)'. LMS WECHSELT IN TEST-MODUS

**Maßnahme**
Mit RST kann wieder in den RUN MODE zurueckgekehrt werden.

LMS0306 SWITCH TO RUN MODE. LIBRARY NO LONGER ASSIGNED
LMS0306 WECHSEL IN RUN MODE. BIBLIOTHEK NICHT MEHR ZUGEWIESEN

LMS0307 LMS TERMINATION MESSAGE: (&00)
LMS0307 LMS ENDE-MELDUNG: (&00)

**Bedeutung**
(&00): Wert der Jobvariablen.

LMS0310 LMS VERSION (&00) LOADED
LMS0310 LMS-VERSION (&00) GELADEN

LMS0311 LMS (&00) (&01' ') ENDED NORMALLY
LMS0311 LMS (&00) (&01' ') NORMAL BEENDET

**Bedeutung**
(&00): LMS-Version
(&01): PLAM-Version (wenn vorhanden).

LMS0312 LMS (&00) (&01' ') ENDED ABNORMALLY
LMS0312 LMS (&00) (&01' ') ABNORMAL BEENDET

**Bedeutung**
(&00): LMS-Version
(&01): PLAM-Version (wenn vorhanden).

LMS0401 FILE (&00' ') IS LOCKED. ATTEMPT TO BE REPEATED? REPLY (Y=YES; N=NO)
LMS0401 DATEI (&00' ') IST GESPERRT. VERSUCH WIEDERHOLEN ? ANTWORT (Y=JA; N=NEIN)

**Maßnahme**
Y: Ein weiterer Versuch wird gestartet.
N: Kein weiterer Versuch wird gestartet.

LMS0402 MEMBER (&00' ') IN LIBRARY (&01' ') IS LOCKED. ATTEMPT TO BE REPEATED? REPLY (Y=YES; N=NO)
LMS0402 ELEMENT (&00' ') IN BIBLIOTHEK (&01' ') GESPERRT. VERSUCH WIEDERHOLEN ? ANTWORT (Y=JA; N=NEIN)

**Bedeutung**
Das Element (&00) in der Bibliothek (&01) ist gegen Schreiben oder gegen Lesen und Schreiben gesperrt.

**Maßnahme**
Y: Ein weiterer Versuch wird gestartet.
N: Kein weiterer Versuch wird gestartet.

LMS0403 TYPE= (&00' ') IN LIBRARY: (&01' ') IS LOCKED. ATTEMPT TO BE REPEATED? REPLY (Y=YES; N=NO)

LMS0403 TYP (&00' ') IN BIBLIOTHEK (&01' ') IST GESPERRT. VERSUCH WIEDERHOLEN ? ANTWORT (Y=JA; N=NEIN)

**Maßnahme**

Y: Ein weiterer Versuch wird gestartet.
N: Kein weiterer Versuch wird gestartet.

LMS0411 FILE (&00' ') IS LOCKED. NEXT ATTEMPT STARTED AFTER 6 SECONDS

LMS0411 DATEI (&00' ') IST GESPERRT. NAECHSTER VERSUCH WIRD NACH 6 SEKUNDEN GESTARTET

LMS0412 MEMBER (&00' ') IN LIBRARY (&01' ') IS LOCKED. NEXT ATTEMPT STARTED AFTER 6 SECONDS.

LMS0412 ELEMENT (&00' ') IN BIBLIOTHEK (&01' ') GESPERRT. NAECHSTER VERSUCH WIRD NACH 6 SEKUNDEN GESTARTET

**Bedeutung**

Das Element (&00) in der Bibliothek (&01) ist gegen Schreiben oder gegen Lesen und Schreiben gesperrt.

LMS0413 TYPE (&00' ') IN LIBRARY (&01' ') IS LOCKED. NEXT ATTEMPT STARTED AFTER 6 SECONDS

LMS0413 TYP (&00' ') IN BIBLIOTHEK (&01' ') IST GESPERRT. NAECHSTER VERSUCH WIRD NACH 6 SEKUNDEN GESTARTET

LMS0500 INPUT FILE DOES NOT EXIST OR IS NOT ASSIGNED CORRECTLY. AMCB ERROR CODE '(&00)'

LMS0500 EINGABE-DATEI EXISTIERT NICHT ODER IST FALSCH ZUGEWIESEN. AMCB-FEHLERCODE '(&00)'

LMS0501 WRONG OP-CODE SEQUENCE

LMS0501 FALSCHE OPCODE-REIHENFOLGE

LMS0502 INCOMPLETE MODULE (E.G. MISSING END-RECORD)

LMS0502 MODUL NICHT KOMPLETT (Z.B. KEIN END-SATZ)

LMS0503 WRONG RECORD TYPE IN MODULE

LMS0503 FALSCHER KARTENTYP IM MODUL

LMS0504 OVERWRITE ERROR

LMS0504 OVERWRITE-FEHLER

LMS0505 THE INPUT FILE IS EMPTY

LMS0505 DIE EINGABEDATEI IST LEER

LMS0506 NO FILE AVAILABLE
LMS0506 KEINE DATEI VORHANDEN

LMS0507 BS2000-PHASE IS NOT CORRECT
LMS0507 BS2000-PHASE IST NICHT KORREKT

LMSD508 1ST RECORD IS NO ESD RECORD
LMS0508 1. SATZ KEIN ESD-SATZ

LMS0509 PARAMTER OVERWRITE=NO IS SET, BUT A ELEMENT IS EXISTING
LMS0509 PARAMETER OVERWRITE=NO IST GESETZT, ABER EIN ELEMENT IST VORHANDEN

LMS0510 ELEMENT OR BASE NOT FOUND
LMS0510 ELEMENT ODER BASIS NICHT GEFUNDEN

**Bedeutung**
Eingabeelement nicht gefunden
Ausgabeelement nicht gefunden, aber OVERWRITE=ONLY gesetzt
Basiselement nicht gefunden

LMS0511 BASE NOT DELTA-STORED
LMS0511 BASIS NICHT DELTA-GESPEICHERT

LMS0512 INPUT AND BASE IN THE SAME CONTAINER
LMS0512 EINGABE UND BASIS SIND IM GLEICHEN CONTAINER

LMS0513 NEITHER LINK NAME NOR FILE NAME EXISTING
LMS0513 WEDER LINK-NAME NOCH DATEINAME VORHANDEN

LMS0514 TOO MANY LIBRARIES HAVE BEEN OPENED
LMS0514 ZUVIELE BIBLIOTHEKEN WURDEN GEOEFFNET

LMS0515 ILLEGAL ELEMENT-NAME
LMS0515 UNGUELTIGE ELEMENTBEZEICHNUNG

LMS0516 ERROR ON SEQUENTIAL LIBRARY (AMCB: (&00))
LMS0516 FEHLER BEI DER BEARBEITUNG DER SEQUENTIELLEN BIBLIOTHEK (AMCB: (&00))

**Bedeutung**
Einer der folgenden Fehler trat auf:
AMCB0040: Falsche BLKSIZE (weder 2048 noch 320 byte)
AMCB0041: Folge der Ausgabe-Bibliotheksabteilungen falsch
AMCB0042: Ruecksetzen vor aktuelle Spule
AMCB0043: Band-Konventionen verletzt
AMCB0044: Band und keine Kurzbezeichnung
AMCB0045: Datei nicht korrekt geschlossen
AMCB0048: Bandmarke beim Rueckwaertslesen
AMCB0049: EOV beim Schreiben erreicht

LMS2000 * LLM WARNING * : FUNCTIONALITY IS ASSUMED HOWEVER
LMS2000 * LLM WARNUNG * : FUNKTION WURDE TROTZDEM AUSGEFUEHRT

LMS2001 INVALID PARAMETERS FOR LLM .
LMS2001 UNGUELTIGER PARAMETER FUER LLM

LMS2002 CORRECTION REJECTED BY LLM .
LMS2002 KORREKTUR VON LLM ABGEWIESEN

LMS2003 PLAM ERROR DETECTED BY LLM .
LMS2003 PLAM-FEHLER VON LLM ENTDECKT

LMS2004 LLM INTERNAL ERROR
LMS2004 INTERNER LLM-FEHLER

LMS2005 ILLEGAL LLM MAIN RETURNCODE
LMS2005 UNGUELTIGER PRIMAER-FEHLERCODE VON LLM

LMS2020 ERROR IN LLM OPEN FUNCTION
LMS2020 FEHLER IN DER OPEN-FUNKTION VON LLM

LMS2021 ERROR IN LLM CLOSE FUNCTION
LMS2021 FEHLER IN DER CLOSE-FUNKTION VON LLM

LMS2022 ERROR IN LLM LIST FUNCTION
LMS2022 FEHLER IN DER LIST-FUNKTION VON LLM

LMS2023 ERROR IN LLM UPDATE FUNCTION
LMS2023 FEHLER IN DER UPDATE-FUNKTION VON LLM

LMS2040 OCCURENCE CAN ONLY BE FIRST OR ALL
LMS2040 ZULAESSIGE WERTE SIND FIRST UND ALL

LMS2050 CHARACTER STRING NOT ALLOWED FOR TYPE L
LMS2050 ZEICHENFOLGE FUER TYP L NICHT ERLAUBT

LMS2100 ILLEGAL LLM SECONDARY RETURNCODE
LMS2100 UNGUELTIGER SEKUNDAER-FEHLERCODE VON LLM

LMS2101 LLM READING NOT COMPLETE: BUFFER SIZE PHYSICALLY LIMITED.
LMS2101 KEIN VOLLSTAENDIGES LESEN DURCH LLM ERFOLGT: PUFFER PHYSIKALISCH BEGRENZT.

LMS2102 SPECIFIED PATHNAME NOT FOUND
LMS2102 ANGEGEBENEN PFADNAMEN NICHT GEFUNDEN

LMS2103 SPECIFIED SLICE NAME NOT FOUND
LMS2103 ANGEGEBENEN SLICE-NAMEN NICHT GEFUNDEN

LMS2104 SPECIFIED CSECT NAME NOT FOUND
LMS2104 ANGEGEBENEN CSECT-NAMEN NICHT GEFUNDEN

LMS2105 LLM: INVALID IDENTIFICATOR
LMS2105 LLM: UNGUELTIGER IDENTIFIKATOR

**Bedeutung**
Ungueltige ID wurde an LLM uebergeben

LMS2106 INVALID DISPLACEMENT
LMS2106 UNGUELTIGER VERSATZ

LMS2107 LLM: INVALID CONTINUATION
LMS2107 LLM: UNGUELTIGER FOLGEAUFRUF

**Bedeutung**
Ungueltiges Fortsetzungszeichen fuer LLM

LMS2108 INVALID OLD CONTENT
LMS2108 UNGUELTIGER ALTER INHALT

LMS2109 INVALID LENGTH
LMS2109 UNGUELTIGE LAENGE

LMS2110 LLM: INVALID MODE
LMS2110 LLM: UNGUELTIGER MODUS

**Bedeutung**
Ungueltige OPEN-Art wurde an LLM uebergeben

LMS2111 LLM PARAMETERS HAVE INVALID VALUES
LMS2111 LLM-PARAMETER HABEN UNZULAESSIGE WERTE

LMS2112 INVALID PATHNAME SPECIFIED
LMS2112 UNGUELTIGER PFADNAME ANGEGEBEN

### Zusatzinformationen

Zu den eigentlichen Texten der LMS-Meldungen erscheinen in verschiedenen Kombinationen die folgenden Zusatztexte:

| zusatzinf | Bedeutung |
|---|---|
| (STATEMENT MEMBER INPUT) | Anweisungseingabe aus Element |
| (COP OR DUP) | COP- oder DUP-Funktion |
| (DATA CARD INPUT) | Fehler bei der Funktion CAT |
| (DATA CARD OUTPUT) | Fehler bei der Funktion PCH, LAP |
| (FIRST STATEMENT IS // ...) | Die erste Anweisung ist // ... |
| FUNCTION TERMINATED | Funktion wird abgebrochen |
| (LIBRARY INPUT) | Bibliothekseingabe |
| (LIBRARY OUTPUT) | Bibliotheksausgabe |
| (LISTING-MEMBER-OUTPUT) | Protokollausgabe in Element |
| (LAST STATEMENT IS NOT // MENM OR MEND | Die letzte Anweisung ist nicht // MENM oder // MEND |
| (SECOND STATEMENT IS NOT // JOM OR // JOB) | Die zweite Anweisung ist nicht // JOM oder // JOB |
| STATEMENT(S) IS (ARE) SKIPPED | Die ganze Anweisung wird ignoriert |
| SYSIPT (CATALOG- OR CORRECT-FUNCTION) | SYSIPT beim Lesen von Daten und Korrekturkarten |
| SYSLST (LISTING) | Protokollausgabe auf SYSLST |
| SYSOPT (PUNCH-FUNCTION) | SYSOPT beim Stanzen |
| ILLEGAL COMMAND | Unzulässiges Kommando |
| ILLEGAL OPERAND | Unzulässiger Operand |
| ERROR FROM COMMAND | Fehler ist auf unzulässiges Kommando zurückzuführen |

| zusatzinf | Bedeutung |
|---|---|
| WHOLE ITEM IS SKIPPED | Bis zum nächsten Komma werden Operanden übergangen |
| OUTPUT-LIBRARY LOCKED | Die Ausgabebibliothek ist gesperrt |
| OPENERROR ON LIBRARY/ELEMENT | Beim Öffnen der Bibliothek/des Elementes trat ein Fehler auf. |
| OUTPUT-LIBRARY MISSING | Die Ausgabebibliothek existiert nicht. |
| OUTPUT-ELEMENT CHANGED | Das zu erstellende Ausgabeelement hat eine andere Versionsnummer. |
| PAR KEY=NO AND RECFORM=F | Der Verarbeitungsoperand KEY ist gleich NO gesetzt und die Datei hat ein festes Satzformat. |
| PAR KEY=NO AND KEYPOS>5 | Der Verabeitungsoperand KEY ist gleich NO gesetzt und die KEYPOS der aufzunehmenden Datei ist größer als 5. |
| PAR KEY=NO AND KEYLEN>16 | Der Verarbeitungsoperand ist gleich NO gesetzt und die KEYLEN der aufzunehmenden Datei ist größer als 16. |
| PAR KEY=YES AND OSM-LIBRARY | Der Verarbeitungsoperand KEY ist gleich YES gesetzt und die Ausgabebibliothek ist eine alte Quellprogramm- oder Makrobibliothek. |
| RECFORM=F AND OSM-LIBRARY | Die aufzunehmende Datei hat Sätze fester Länge und die Ausgabebibliothek ist eine alte Quellprogramm- oder Makrobibliothek. |
| KEYPOS>5 AND OSM-LIBRARY | Die aufzunehmende Datei hat eine KEYPOS, die größer als 5 ist und die Ausgabebibliothek ist eine alte Quellprogramm- oder Makrobibliothek. |
| KEYLEN>16 AND OSM-LIBRARY | Die aufzunehmende Datei hat eine KEYLEN, die größer als 16 ist und die Ausgabebibliothek ist eine alte Quellprogramm- oder Makrobibliothek. |
| RECFORM=F | Das Bibliothekselement hat Sätze fester Länge. |
| KEYPOS>5 | Das Bibliothekselement ist mit PAR KEY=YES aufgenommen worden und die KEYPOS ist größer als 5. |
| KEYLEN>8 | Das Bibliothekselement ist mit PAR KEY=YES aufgenommen worden und die KEYLEN ist größer als 8. Dies gilt nur für den Aufruf des EDT. |
| KEYLEN>16 | Das Bibliothekselement ist mit PAR KEY=YES aufgenommen worden und die KEYLEN ist größer als 16. Dies gilt nur für den Aufruf des EDOR. |

| zusatzinf | Bedeutung |
|---|---|
| PAR KEY=YES | Der Verarbeitungsoperand KEY ist gleich YES gesetzt. |
| KEYS DO EXIST IN ELEMENT | Das zu erweiternde Bibliothekselement ist mit PAR KEY=YES aufgenommen worden und enthält die ISAM-Schlüssel. |
| DIFFERENT FILETYPE/ VALUE PROPAGATION (MIN/MAX) | Der Dateityp bzw. die "VALUE PROPAGATION" der aufzunehmenden Datei stimmen nicht mit denen des zu erweiternden Bibliothekselement überein. |
| DIFFERENT RECORD FORMAT | Das Satzformat der aufzunehmenden Datei stimmt nicht mit dem Satzformat des zu erweiternden Bibliothekselementes überein. |
| DIFFERENT RECORD SIZE | Die Satzlänge der aufzunehmenden Datei stimmt nicht mit der Satzlänge des zu erweiternden Bibliothekselementes überein. |
| DIFFERENT KEYPOSITION | Die Position der Schlüssel der aufzunehmenden Datei stimmt nicht mit der Position der Schlüssel des zu erweiternden Bibliothekselementes überein. |
| DIFFERENT KEYLENGTH | Die Länge der Schlüssel der aufzunehmenden Datei stimmt nicht mit der Länge der Schlüssel des zu erweiternden Bibliothekselementes überein. |
| DIFFERENT LOGLENGTH | Die LOGLENGTH der aufzunehmenden Datei stimmt nicht mit der LOGLENGTH des zu erweiternden Bibliothekselementes überein. |
| DIFFERENT VALUE LENGTH | Die VALLEN der aufzunehmenden Datei stimmt nicht mit der VALLEN des zu erweiternden Bibliothekselementes überein. |
| OUTPUT-LIBRARY IS NOT A PLAM-LIBRARY | Die Ausgabebibliothek ist keine Programmbibliothek. |
| FIXED RECORD FORMAT-ON INPUT-FILE | Das zu erweiternde Bibliothekselement enthält keine Dateiattribute und die Eingabe-Datei hat Sätze fester Länge. |
| KEYPOSITION≠5 ON INPUT-FILE | Das zu erweiternde Bibliothekselement enthält keine Dateiattribute und die Eingabe-Datei hat eine Schlüsselposition, die ungleich '5' ist. |
| KEYLENGTH>16 ON INPUT-FILE | Das zu erweiternde Bibliothekselement enthält keine Dateiattribute und die Eingabe-Datei hat eine Schlüssellänge, die größer als '16' ist. |
| RECORD SIZE>2032 ON INPUT-FILE | Die aufzunehmende Datei hat eine Satzlänge von mehr als 2032 Byte. |
| (COMMAND INPUT/ TYP J) | Anweisungseingabe aus einem Element vom Typ J. |

# Fragemeldungen

Die nachfolgenden Fragemeldungen werden ausgegeben. Sie müssen jeweils mit Y oder N beantwortet werden.

```
DO YOU WISH A BKPT (Y/N) ?
```

**Bedeutung**
Ist Auftragsschalter 31 gesetzt und tritt im Dialog ein Programmfehler auf, fragt LMS, ob ein BKPT-Makro abgesetzt werden soll.

```
LMS0403 TYPE ... IN LIBRARY ... IS LOCKED. ATTEMPT TO BE REPEATED?
REPLY (Y=YES; N=NO)
```

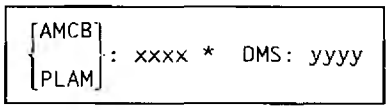
**Bedeutung**
Der Elementtyp ist gegen Schreiben gesperrt.

```
LMS0402 MEMBER ... IN LIBRARY ... IS LOCKED. ATTEMPT TO BE REPEATED?
REPLY (Y=YES; N=NO)
```

**Bedeutung**
Das Element ist gegen Schreiben oder gegen Lesen und Schreiben gesperrt.

# Meldungen der Zugriffsroutinen

Die Meldungen der internen LMS-Zugriffsroutinen haben folgenden Aufbau:




Bedeutung

xxxx AMCB-/PLAM-Fehlerschlüssel

yyyy DMS-Fehlercode (siehe Systemmeldungen)

Die Fehlercodes ab 200 sind PLAM-Fehlercodes. Diese PLAM-Fehlercodes können auch über das Kommando

```
/HELP PLAxxxx
```

abgefragt werden.

| xxxx | Bedeutung der AMCB-Fehlerschlüssel |
|---|---|
| 0000 | Kein Fehler |
| 0001 | DVS-Fehler (ERROR-CODE siehe RET3) |
| 0002 | Ungültiger OP-CODE |
| 0003 | Dateiname im Kontrollblock fehlt |
| 0004 | Kein / geänderte FCB-ADR oder FCB-ADR für FOP mit DMS-OPEN zeigt auf aktiven FCB |
| 0005 | Falsche OP-CODE-Reihenfolge |
| 0006 | Ungültiger Bibliothekstyp |
| 0007 | Widersprüchliche LIB-Typen im Kontrollblock |
| 0008 | Bibliothek wurde repariert |
| 0009 | Bibliothek muß repariert werden |
| 0010 | Adresse nicht innerhalb der Elementgrenzen |
| 0011 | Beim Schreiben: Bibliotheksgrenze erreicht |
| 0012 | Widersprüchliche Angaben im Kontrollblock und FCB |
| 0013 | Zusatzinformation fehlt |
| 0014 | Ungültige Zusatzinformation |
| 0015 | Satz zu lang |
| 0016 | Komprimierungskennzeichen in IHV ungültig |
| 0017 | Letztes Element in Bibliothek wurde gelöscht |
| 0019 | Es handelt sich um keine LMS-Bibliothek |
| 0020 | IHV-Bereich-Überlauf der R-Bibliothek DIR1 > 30 PAM-Seiten |
| 0021 | IHV-Bereich-Überlauf der R-Bibliothek DIR2 für diesen Modul > 4 PAM-Seiten (zuviele CSECT/COMMON/ENTRY) |
| 0022 | Modul nicht komplett (z.B. kein END-Satz) |
| 0023 | Falscher Satztyp im Modul |
| 0024 | Warnung ! Bibliothek kurz vor Überlauf |
| 0025 | Keinen DIR2-Eintrag erstellt |
| 0026 | OSM-Bibliothek muß mit /VERIFY-Kommando reorganisiert werden |
| 0040 | Falsche Blockgrösse auf dem Eingabe-Bibliotheksband, weder 2048 noch 320 |
| 0041 | Folge der Bibliotheksabteilungen bei Ausgabebandbibliothek falsch |
| 0042 | Bei Folgebandverarbeitung für Eingabeband-Bibliotheken ist ein Positionieren auf eine vorherige Spule notwendig, aber nicht erlaubt |
| 0043 | Auf der Eingabebandbibliothek folgt eine Abschnittsmarke beim *Start-Block |
| 0044 | Bandbibliothek und keine Kurzbezeichnung in der LIB-Anweisung |
| 0045 | Die Bandeingabebibliothek wurde nicht ordnungsgemäß geschlossen |
| 0046 | Keine Standardkennsätze bei Bandeingabebibliotheken |
| 0047 | Reserviert |
| 0048 | Beim Positionieren auf der Eingabebandbibliothek wurde die Anfangsmarke erkannt |
| 0049 | Beim Schreiben des Ausgabebibliotheksbandes wurde Bandende erreicht |
| 0050 | OVERWRITE-Fehler |
| 0051 | Speicher zu klein |
| 0052 | Element wird überschrieben |
| 0053 | Eingabedatei ist leer |
| 0054 | Leere Datei wird ersetzt |
| 0062 | Funktion nicht implementiert |
| 0063 | Keine Dateien vorhanden |

| xxxx | Bedeutung der AMCB-Fehlerschlüssel |
|---|---|
| 0064 | FMS kann nicht geladen werden |
| 0065 | XFR-/ PHASE-Satz nicht vorhanden |
| 0066 | 1. Satz kein ESD-Satz |
| 0100 | Illegale Programmdatei |
| 0101 | DVS-Fehler |
| 0102 | Unbekannter Dateityp |
| 0107 | Versorgung weder LINK-Name noch FILE-Name |
| 0108 | USER Fehler |
| 0109 | OPEN Fehler |
| 0111 | Kein freier Platz in FILE-Table |
| 0112 | FSTAT Fehler |
| 0118 | Kein leeres FILE bei CREATE |
| 0119 | Open auf leeres FILE |
| 0120 | Dateiname ungültig |
| 0121 | FILE-ID hat keinen Eintrag in FILE-Table |
| 0122 | Geforderter OPEN-Status ungleich IST-STATUS |
| 0123 | Kein Platz mehr für FCB |
| 0124 | Kein Platz mehr für Zugriffsvermerk in FT |
| 0125 | 2. Zugriff auf Ausgabebibliothek |
| 0126 | Kein Platz mehr für LINK-Table-Eintrag |
| 0127 | LINK-Table-Eintrag fehlt |
| 0129 | VSN-Prüfung ergibt Fehler |
| 0131 | Bibliothek ist noch für CTL oder PRT geöffnet |
| 0134 | CTL-Element ist geöffnet |
| 0135 | LINK-Name LIBxxx - xxx ist nicht numerisch |
| 0136 | Zugriffsfehler z.B. gesperrte Datei |
| 0137 | 2. Zugriff auf eine sequentielle Bibliothek |
| 0150 | Unbekannte Zugriffsart |

| xxxx | Bedeutung der PLAM-Fehlerschlüssel |
|---|---|
| | * 200 - 299 Warnungen ** |
| 0201 | Bibliothek existiert nicht |
| 0202 | Kein Bibliotheksname im PLCB und kein /FILE-Kommando |
| 0203 | Keine PLAM-Bibliothek |
| 0204 | Bibliothek nicht neu |
| 0205 | Element existiert nicht |
| 0206 | Element existiert schon |
| 0207 | Variante existiert nicht |
| 0208 | EOF |
| 0209 | Satz nicht gefunden |
| 0210 | Kein Satz übertragen |
| 0211 | Ende der Satzart |
| 0212 | Kein Eintrag mehr im Inhaltsverzeichnis |
| 0214 | Format-B Satz abgeschnitten |
| 0216 | Der eingelesene Satz wurde verkürzt |
| 0217 | Secondary Allocation der Bibliothek ist auf 0 gesetzt<br>Die Bibliothek kann nicht automatisch vergrößert werden |
| 0219 | Zugriff auf die Bibliothek über RFA mit SHAREUPD=YES nicht erlaubt |
| 0221 | Format-B Satz zu kurz |
| 0222 | Format-B Satz zu lang |
| 0255 | SVC PLAM nicht verfügbar |
| 0290 | Anforderung von RZ-EXIT Routine abgewiesen |
| | * Mehrfachzugriffsbeschränkungen ** |
| 0301 | Element eröffnet für anderen PLCB |
| 0302 | Schreibender Zugriff auf Elementtyp |
| 0303 | Elementtyp gesperrt |
| | * 401 - 599 Benutzerprogrammfehler ** |
| 0401 | Ungültige PLCB-Adresse |
| 0402 | Ungültige Operanden / PLCB |
| 0403 | Ungültige Funktion |
| 0404 | ATTACH ist für einen PLCB nicht möglich |
| 0405 | Bibliothek nicht zugewiesen für PLCB |
| 0406 | Element eröffnet für PLCB |
| 0407 | Weder Dateikettungsname noch Bibliotheksname im PLCB |
| 0408 | Ungültige P2-PLCB-Adresse |
| 0409 | Ungültiger P2-PLCB |
| 0410 | Ungültiger OPEN-Modus |
| 0411 | Adresse des Datenbereichs fehlt |
| 0412 | Element für PLCB nicht eröffnet |
| 0413 | Ungültige Variantennummer |
| 0414 | Überlauf der Variantennummer |
| 0415 | Element nicht eröffnet für WRITE/UPDATE/EXTEND |
| 0416 | Element nicht eröffnet für INPUT/UPDATE/EXTEND |
| 0417 | Inkonsistenz im Datenbereich |
| 0418 | Elementbezeichnung 1 gleich Elementbezeichnung 2 |
| 0419 | Ungültiger Datenbereich |
| 0420 | Ungültige Länge des Datenbereichs |
| 0421 | Ungültige Blocklänge |
| 0422 | Ungültige Satzart-Endebedingung |
| 0423 | Ungültiger ATTACH-Modus |
| 0424 | PLCB für INPUT zugewiesen |

| xxxx | Bedeutung der PLAM-Fehlerschlüssel |
|---|---|
| 0425 | INPUT-Zuweisung für zerstörte Bibliothek |
| 0426 | Ungültige ATTACH-Bedingung |
| 0427 | Ungültige OPEN-Bedingung |
| 0428 | Ungültige KEEP-Angabe |
| 0429 | Ungültige RENAME-Bedingung |
| 0430 | Ungültiger Elementtyp |
| 0431 | Ungültiger Elementname |
| 0432 | Ungültige Versionsangabe |
| 0433 | Ungültige INQUIRY-Angabe |
| 0434 | Ungültige SECURITY-ERASE-Angabe |
| 0435 | Elementtyp alt ungleich Elementtyp neu |
| 0436 | RENAME-Fehler |
| 0437 | Ungültige Satzlänge |
| 0438 | Ungültige Satznummer |
| 0439 | Ungültiger FCBTYPE im /FILE-Kommando |
| 0440 | SHARUPD=NO nicht erlaubt |
| 0441 | Ungültige OPEN-Angabe im /FILE-Kommando |
| 0442 | Ungültige Satzart |
| 0443 | Kein Elementtyp gesperrt für PLCB |
| 0444 | Elementtyp gesperrt für PLCB |
| 0445 | Ungültige Angabe zur Variantenauswahl |
| 0446 | Ungültige Angabe zur Versionsauswahl |
| 0447 | Ungültige Element-ACCESS-ID |
| 0448 | Ungültiges Benutzer-Datum-Kennzeichen |
| 0449 | Ungültiger Operand für GWRK |
| 0450 | Ungültiger Operand für RWRK |
| 0451 | Ungültige WORK-AREA-Adresse |
| 0452 | Ungültige WORK-AREA-Länge |
| 0453 | Ungültiger Sekundärname |
| 0454 | Ungültige Angabe zur Sekundärnamens-Auswahl |
| 0455 | Ungültige Format-B-Puffer-Adresse |
| 0456 | Ungültige Format-B-Puffer-Länge |
| 0457 | Ungültiger Format-B-Puffer |
| 0458 | Ungültige Angabe zur Format-B-Positionierung |
| 0459 | ungültiger Attributname |
| 0460 | ATTACH-Modus EXEC nicht für P1 |
| 0461 | ungültige Option zum weiteren Suchen unter $TSOS |
| 0462 | ungültige Option für Systembibliothek |
| 0463 | ungültige Katalogkennzeichnung |
| 0464 | ungültige Option zur Format-B Satz-Übernahme |
| 0465 | ungültige Option zur Format-B Satz-Maskierung |
| 0466 | ungültige Format-B Satz-Maske |
| 0467 | ungültiger Suchstandard |
| | * 601 - 699 Bibliotheksdaten ungültig/zerstört |
| 0601 | Inkonsistenz in der Elementbeschreibung |
| 0602 | Ungültiger Format-A-Kontrollbereich |
| 0603 | Inkonsistenz im Format-A-Kontrollbereich |
| 0604 | Inkonsistenz im Format-A-Kontrollbereich-Eintrag |
| 0605 | Inkonsistenz im Format-A-Datenbereich |
| 0606 | Inkonsistenz im Inhaltsverzeichnis |
| 0607 | Ungültiger Operationscode für Restart |
| 0608 | Ungültiger Restartblock |

| xxxx | Bedeutung der PLAM-Fehlerschlüssel |
|---|---|
| 0609 | Inkonsistente Plattenspeicherverwaltung |
| 0610 | Inkonsistenter Elementname |
| 0611 | Inkonsistenz im Variantenverzeichnis |
| 0612 | Inkonsistenter Format-B-Kontrollbereich |
| | * 701 - 799 Systemservice |
| 0701 | ENASI Makro-Fehler |
| 0702 | DISSI Makro-Fehler |
| 0703 | ENQAR Makro-Fehler |
| 0704 | DEQAR Makro-Fehler |
| 0705 | ENAMP Makro-Fehler |
| 0706 | REQMP Makro-Fehler |
| 0707 | RELMP Makro-Fehler |
| 0708 | OPEN Makro-Fehler |
| 0709 | PAM Makro-Fehler |
| 0710 | CLOSE Makro-Fehler |
| 0711 | FILE Makro-Fehler |
| 0712 | EXRTN Makro-Fehler |
| 0713 | FSTAT Makro-Fehler |
| 0714 | REQM Makro-Fehler |
| 0715 | RELM Makro-Fehler |
| 0716 | $REQM Makro-Fehler |
| 0717 | $RELM Makro-Fehler |
| 0718 | $NCREBO Makro-Fehler |
| 0719 | $NDESBO Makro-Fehler |
| 0720 | $NENQBO Makro-Fehler |
| 0721 | $NDEQBO Makro-Fehler |
| 0722 | Fehler beim STAM-Makro |
| 0723 | Fehler beim RDTFT-Makro |
| | * Interne PLAM-Fehler ** |
| 0801 | Ungültiger LINK P2-PLCB → P1-PLCB |
| 0802 | Sperre gesetzt bei Funktionsende |
| 0803 | Kontrollbereich zum Zugriff eröffnet |
| 0804 | Kontrollbereich für den Zugriff nicht eröffnet |
| 0805 | Suchfehler im Kontrollbereich |
| 0806 | Falscher Komprimierungsmodus |
| 0807 | Bibliothekstabelle nicht gesperrt |
| 0808 | Bibliothek nicht gesperrt |
| 0809 | Element nicht gesperrt |
| 0810 | Bereits eine Bibliothek für PLCB eröffnet |
| 0811 | Keine Bibliothek für PLCB eröffnet |
| 0812 | Schließen Bibliothek obwohl Element eröffnet |
| 0813 | Bereits ein Element für PLCB eröffnet |
| 0814 | Kein Element für PLCB eröffnet |
| 0815 | Ungültiger OPEN-Modus für Element |
| 0816 | ungültiges Index-Flag im TOC-Bereich |
| 0817 | Plattenspeicherverwaltung ist nicht gesperrt |
| 0818 | Wiederanlaufspeicher nicht gesperrt |
| 0819 | Bibliotheks-Tabellen-Kopf nicht zugewiesen |
| 0820 | Bibliotheks-Tabellen-Kopf zugewiesen |
| 0821 | Fehler in der Sperreihenfolge |
| 0822 | Ungültiges Namenskennzeichen |
| 0823 | Schlüssel existiert schon |

| xxxx | Bedeutung der PLAM-Fehlerschlüssel |
|---|---|
| 0824 | Fehler bei der Aufteilung des Datenbereichs |
| 0825 | Elementtyp zu lang |
| 0826 | Elementname zu lang |
| 0827 | Elementversion zu lang |
| 0828 | Schlüssel im Inhaltsverzeichnis nicht vorhanden |
| 0829 | Kein Pufferbereich für das Inhaltsverzeichnis vorhanden |
| 0830 | Überlauf des Inhaltsverzeichnisses |
| 0831 | Inkonsistenzen im I/O-Management |
| 0832 | Kein Pufferbereich verfügbar |
| 0833 | Pufferadresse nicht gefunden |
| 0834 | Pufferbereich nicht reserviert |
| 0835 | Pufferverwaltung nicht initialisiert |
| 0836 | Ungültiger PRIV-FLAG für $REQM |
| 0837 | Ungültiger ACCESS-FLAG für $VALD |
| 0838 | Falsche Anzahl von Byte |
| 0839 | Ungültige Speicherklasse |
| 0840 | Speicher nicht verfügbar |
| 0841 | Speicherbereich nicht freigegeben |
| 0842 | Ungültige Speicheradresse |
| 0843 | Teilseite nicht zugewiesen |
| 0844 | Eröffnen einer eröffneten Datei |
| 0845 | Schließen einer geschlossenen Datei |
| 0846 | Ungültiger Eröffnungsmodus |
| 0847 | Bibliotheksdatei nicht eröffnet |
| 0848 | Ungültige Nummer der E/A-Anforderung |
| 0849 | Ungültiges LOCK/WAIT-Kennzeichen |
| 0850 | Börse bereits erzeugt |
| 0851 | Börse nicht erzeugt |
| 0852 | Sperrstatusfehler der Bibliothekstabelle |
| 0853 | Sperrstatusfehler der Bibliothek |
| 0854 | Sperrstatusfehler des Plattenspeichers |
| 0855 | Sperrstatusfehler des Elementes |
| 0856 | Bibliothek existiert nicht |
| 0857 | Bibliothek ist leer |
| 0858 | Ein-Ausgabe nicht beendet |
| 0859 | Bibliotheks-ACCESS ID verändert |
| 0860 | Element nicht gefunden |
| 0861 | Kein gültiger Speicherbereich |
| 0862 | Ungültige Inhaltsverzeichnisnummer |
| 0863 | Sekundärname zu lang |
| 0864 | Variantennummer zu lang |
| 0865 | Suchfehler im Format-B-Kontrollbereich |
| 0866 | Format-B-Satz Zugriff nicht möglich |
| 0867 | Format-B-Satz Zugriff möglich |
| 0868 | Attribut-Name zu lang |
| 0869 | ungültige Option für paralleles Ändern |
| 0870 | ungültige FCB-Adresse |

| xxxx | Bedeutung der PLAM-Fehlerschlüssel |
|---|---|
| 1001 | Adresse des System-Versions-Eintrags ungültig |
| 1002 | TASK WORK AREA konnte nicht angelegt werden |
| 1003 | TASK WORK AREA zerstört |
| 1004 | Register 1 ungleich 0 |
| 1005 | inkompatible PLAM-Version |
| 1006 | Element war für WRITE, UPDATE oder EXTEND eröffnet |
| 1007 | Element existiert nicht mehr |
| 1008 | Variante wurde geändert |
| 1009 | Typsperre war gesetzt |
| 1010 | ungültige Verknüpfung vom P1-PLCB zum PLCB |
| 1011 | Zu viele Aufrufe durch RESTART |
| 9999 | Unbekannter Fehler |

# Anhang

## Umstellung von MLU, LMR, COBLUR auf LMS

LMS faßt die Funktionen von MLU, LMR und COBLUR in sich zusammen. Es kann auch auf Bibliotheken zugreifen, die von diesen Programmen erzeugt wurden. Folgendes ist zu beachten:

### MLU-Format

Von MLU erstellte Bibliotheken werden von LMS verarbeitet. Da von MLU bearbeitete Elemente keine Versionsnummer enthalten, unterstellt LMS für solche Elemente die Versionsnummer 0 (Null). Von LMS erzeugte Makro- und Quellprogrammbibliotheken haben das MLU-Format und können von MLU weiterverarbeitet werden. (Programmbibliotheken kann MLU nicht bearbeiten.)

### LMR-Format

Von LMR erstellte Bibliotheken werden von LMS verarbeitet. Da von LMR bearbeitete Elemente keine Versionsnummer enthalten, unterstellt LMS für solche Elemente die Versionsnummer 0 (Null). Von LMS erzeugte und bearbeitete (nicht leere) Bindemodulbibliotheken können von LMR weiterverarbeitet werden. (Programmbibliotheken kann LMR nicht bearbeiten.)

### COBLUR-Format

Von COBLUR erstellte Bibliotheken werden von LMS nur gelesen. Änderungen in solchen Bibliotheken werden nicht durchgeführt. Da die Elemente einer COBLUR-Bibliothek weder Versionsnummer noch Datum enthalten, wird für beide Angaben Null unterstellt. LMS ignoriert die Einteilung in Bibliotheksabschnitte. Dem Protokoll eines Inhaltsverzeichnisses läßt sich also nicht entnehmen, in welchen Abschnitten die Elemente gespeichert sind. Sind in mehreren Abschnitten Elemente gleichen Namens enthalten, werden diese Namen im Protokoll des Inhaltsverzeichnisses mehrfach aufgeführt. Wird bei LST ein Name angegeben, der in mehreren Abschnitten vorhanden ist, wird das zuerst gefundene Element aufgelistet.

Auch bei DUP wird nur das zuerst gefundene Element dupliziert. Es sollten zunächst mit COBLUR die gleichnamigen Elemente unbenannt werden. Anschließend kann die gesamte Bibliothek dupliziert werden.

# Kompatibilität BS1000-BS2000

Mit LMS können Bibliotheken auf Bändern vom BS1000 ins BS2000 transportiert werden und umgekehrt.

LMS kann BS1000-Bandbibliotheken mit alter (320 Byte) und neuer BS1000-Blockgröße (2 KB) bearbeiten. Die von LMS erzeugten sequentiellen Bibliotheken haben jedoch immer neue Blockgröße (2 K-Blöcke).

Soll eine im BS2000 erstellte Bandbibliothek später im BS1000 verarbeitet werden, muß im /SET-FILE-LINK-Kommando ACCESS-METHOD=BTAM angegeben werden.

Eine BS2000-Programmdatei, die auf ein BS1000-Bibliotheksband übertragen werden soll (ADDC), muß mit COREIM=N in der PROGRAM-Anweisung des TSOSLNK gebunden werden.

Banddateien haben im BS1000 standardmäßig den Dateinamen DOS-LIB, es sei denn, es wurde über VOL- und TPLAB-Karten bei der Banderstellung ein anderer Name vereinbart.

Bei der Vergabe von Elementnamen ist zu beachten, daß Namen von BS1000-Auftragsmakros und Modifikationssätzen maximal 7 Zeichen lang sein dürfen. Dabei ist Auftragsmakronamen das Zeichen . (Punkt) und Modifikationssätzen für Auftragsmakros das Zeichen / voranzustellen.

### Behandlung von Folgebändern

Folgebänder, die im BS1000 erstellt wurden, können auch im BS2000 verarbeitet werden. Sie müssen jedoch einzeln mit unterschiedlichen Kurzbezeichnungen im /SET-FILE-LINK-Kommando zugewiesen werden, z.B. Band 1 mit LIB001, Band 2 mit LIB002, usw. Da sie den gleichen Dateinamen haben, muß vor dem Zuweisen des 2. Bandes die erste Eingabebibliothek geschlossen und der Dateiname aus dem Katalog gelöscht werden.

### Besonderheiten beim Speichern von BS1000-Jobmakros

Mit der Funktion WRT können Jobmakros in Quellprogrammbibliotheken eingetragen werden. Die Eingabe ist nur von der Datensichtstation möglich; Eingaben über SYSDTA/SYSIPT sind im BS2000 nicht möglich. Jobmakros werden daran erkannt, daß das erste Zeichen des Namens ein Punkt ist.

LMS prüft Jobmakros auf die Einhaltung folgender Vorschriften:

1. Die erste Anweisung darf nicht mit // beginnen. Sie muß eine Ladeoperandenanweisung (siehe Monitor-Beschreibung, JMS1 [11]) oder leer sein.
2. Die zweite Anweisung muß mit // JOM oder // JOB beginnen.
3. Die letzte Anweisung muß mit // MENM oder // MEND beginnen.

Falls diese Regeln nicht eingehalten sind, wird ein Jobmakro von LMS nicht katalogisiert.

In BS1000-Jobmakros wird durch NOP innerhalb der LMS-Anweisungen Plätze für Modifikationen freigehalten.

# Anweisungen und Verarbeitungsoperanden

Diese Anweisungen werden nur noch aus Kompatibiltätsgründen unterstützt.

- CAT Aufnehmen von Elementen über SYSIPT
- COP Kopieren von Elementen
- LAP Auflisten und Ausgeben nach SYSOPT
- LIBIN Zuweisen der Eingabebibliothek
- LIBOUT Zuweisen der Ausgabebibliothek
- PCH Ausgeben von Elementen nach SYSOPT
- WRT Aufnehmen von Elementen in Bibliotheken über SYSOPT

## LIBIN Zuweisen der Eingabebibliothek

LIBIN wird aus Kompatibilitätsgründen noch unterstützt. Ihre Funktionen übernimmt LIB.

Festlegen der impliziten Eingabebibliothek und schließen der bisherigen Eingabebibliothek.

| Operation | Operanden |
|---|---|
| LIBIN | [{[(lib)[,(vsn)]] / libname / LINK=linkname}] |

| | |
|---|---|
| LIBIN | Name der Anweisung |
| lib | Kurzbezeichnung der Eingabebibliothek |
| vsn | Archivnummer des Datenträgers (6 Zeichen) |
| libname | Vollqualifizierter Dateiname der Eingabebibliothek |
| linkname | Dateikettungsname, der auf die Eingabebibliothek verweist |

LIBIN

- schließt die bisherige Eingabebibliothek
- hebt die Festlegung der impliziten Eingabebibliothek auf
- legt die in libname, lib bzw. linkname angegebene Bibliothek als Eingabebibliothek fest.

Die so festgelegte implizite Eingabebibliothek gilt bis zum nächsten LIBIN. Zu Beginn des LMS-Laufs und nach LIBIN ohne Operand ist die implizite Eingabebibliothek undefiniert.

Nach Fehlern ist die implizite Eingabebibliothek undefiniert und muß erneut zugewiesen werden.

Bei der Verwendung einer Bibliothekskurzbezeichung (lib) oder eines Dateikettungsnamens (LINK=...) bei LIBIN ist die Angabe eines /FILE-Kommandos notwendig (siehe Seite 38).

Mit dem Zusatz (vsn) kann eine Archivnummer angegeben werden. Diese muß mit der vsn des zugewiesenen Datenträgers übereinstimmen.

Eine sequentielle Bibliothek kann nicht gleichzeitig Ein- und Ausgabebibliothek sein.

## LIBOUT Zuweisen der Ausgabebibliothek

LIBOUT wird aus Kompatibilitätsgründen noch unterstützt.
Ihre Funktionen übernimmt LIB. Nur wenn eine sequentielle Bibliothek als Ausgabebibliothek festgelegt werden soll, muß LIBOUT noch verwendet werden.

Schließen der bisherigen Ausgabebibliothek und Festlegen der neuen Ausgabebibliothek.

| Operation | Operanden |
|---|---|
| LIBOUT | [{(lib)[,{(vsn) / NEWLIB / NEWLIB(vsn)}]} / {libname / LINK=linkname}] |

| | |
|---|---|
| LIBOUT | Name der Anweisung |
| lib | Kurzbezeichnung der Ausgabebibliothek |
| vsn | Archivnummer des Datenträgers (6 Zeichen) |
| NEWLIB | Neueinrichten der Bibliothek |
| libname | Vollqualifizierter Dateiname der Ausgabebibliothek |
| linkname | Dateikettungsname, der auf die Ausgabebibliothek verweist |

LIBOUT

- schließt die bisherige Ausgabebibliothek
- hebt die Festlegung der impliziten Ausgabebibliothek auf
- legt die in libname, lib bzw. linkname angegebene Bibliothek als Ausgabebibliothek fest.

Die so festgelegte implizite Ausgabebibliothek gilt bis zum nächsten LIBOUT. Zu Beginn des LMS-Laufs und nach LIBOUT ohne Operand ist die implizite Ausgabebibliothek undefiniert.

Nach Fehlern ist die implizite Ausgabebibliothek undefiniert und muß erneut zugewiesen werden.

Mit dem Zusatz (vsn) kann eine Archivnummer angegeben werden. Diese muß mit der vsn des zugewiesenen Datenträgers übereinstimmen.

Bei der Verwendung einer Bibliothekskurzbezeichnung (lib) oder eines Dateikettungsnamens (LINK=...) bei LIBOUT ist die Angabe eines /FILE-Kommandos notwendig (siehe Seite 38).

Eine sequentielle Bibliothek kann nicht gleichzeitig Ein- und Ausgabebibliothek sein. Mit LIBOUT wird die aktuelle Ausgabebibliothek geschlossen. Dann kann dieselbe Bibliothek als Eingabebibliothek zugewiesen werden.

Ist für lib ein Band zugewiesen, wird immer ein neues Bibliotheksband eröffnet, d.h. es werden neue Anfangskennsätze für die Bibliothek geschrieben.

Für sequentielle Bibliotheken ist die Angabe von NEWLIB bei LIBOUT immer notwendig.

Die Angabe NEWLIB ist nur bei einer neu zu erstellenden Datei erlaubt und bewirkt, daß die Ausgabebibliothek als leere Bibliothek eingerichtet wird. Für die Datei müssen folgende Attribute im FILE-Kommando gegeben werden:

| Bibl. - typ | FCBTYPE | KEYPOS | KEYLEN |
|---|---|---|---|
| Source - Bibl. | ISAM | 5 | 8 |
| Macro - Bibl. | ISAM | 5 | 8 |
| Modul - Bibl. | PAM | | |
| Sequent. - Bibl. | BTAM | | |

Bei ISAM-Dateien legt erst die erste Verwendung als Ausgabebibliothek den Typ als Makro- oder Quellprogrammbibliothek fest.

### Verarbeitungsoperanden

Diese Verarbeitungsoperanden werden nur noch aus Kompatibilitätsgründen unterstützt.

- PAR COLLECT: Sammeln von Anweisungen
- PAR SAVE: Sicherstellen von Elementen beim Korrigieren

PAR DECOMPRESSED
Steuern der Komprimierung für Makros und Quellprogramme

Der Verarbeitungsoperand DECOMPRESSED wird aus Kompatibilitätsgründen noch unterstützt. Er wirkt bei Programmbibliotheken wie NOP. Bei den anderen Bibliotheken steuert er die Komprimierung bei der Ablage von Makros und Quellprogrammen in Bibliotheken.

| Operation | Verarbeitungsoperand |
|---|---|
| PAR | DEC[OMPRESSED]=[{YES / NO / ?}] |

YES Alle Elemente werden entkomprimiert abgelegt.

NO Alle Elemente werden komprimiert abgelegt.

? Der aktuelle Wert wird protokolliert.

# Literatur


[ 1] BS2000
**Systemanwendung**
Taschenbuch

*Zielgruppe*
erfahrene BS2000 Anwender
*Inhalt*
Eine Zusammenstellung der
- Kommandos und Makros im BS2000
- Befehle und Assembleranweisungen
- Anweisungen der Softwareprodukte und Dienstprogramme
  - EDT, EDOR, SORT, LMS, ARCHIVE, PERCON, LEASY
  - TSOSLNK, DCAT, PASSWORD, FDEXIM, FDRIVE, DPAGE, SODUMP, TPCOMP2, PRSERVE
- wichtigsten Tabellen und Register des BS2000
- Code-Tabellen
- Systemkonventionen

*Einsatz*
BS2000 Dialog- und Stapelbetrieb

[ 2] BS2000
**Binder-Lader-Starter (BLS)**
Benutzerhandbuch

*Zielgruppe*
BS2000 Anwender
*Inhalt*
Beschreibung der Anweisungen zum Binden und Laden von Programmen mit BINDER, ELDE und DBL
*Einsatz*
BS2000 Dialog- und Stapelbetrieb

[ 3] BS2000
**Dienstprogramme**
Beschreibung

*Zielgruppe*
BS2000 Anwender (nicht privilegiert)
*Inhalt*
Dienstprogramme für den nichtprivilegierten Benutzer des BS2000.
*Einsatz*
BS2000 Teilnehmerbetrieb

[ 4] **EDOR** (BS2000)
Beschreibung

*Zielgruppe*
Datenerfasser, Programmierer
*Inhalt*
Beschreibung der Anweisungen an den Datenbearbeiter EDOR.
*Einsatz*
BS2000 Dialogbetrieb

[ 5] **EDT** (BS2000)
Beschreibung

*Zielgruppe*
Datenerfasser, Programmierer
*Inhalt*
Beschreibung der Anweisungen an den Dateibearbeiter EDT, EDT-Prozeduren, Unterprogrammschnittstelle des EDT
*Einsatz*
BS2000 Dialog- und Stapelbetrieb

[ 6] BS2000
**Einführung in die Systemanwendung**
Benutzerhandbuch

*Zielgruppe*
BS2000-Anwender
*Inhalt*
Einführung ins BS2000; Beschreibung der meistgebrauchten Benutzerkommandos des BS2000; Einführung in die Benutzung der Dienstprogramme und Softwareprodukte EDT, SORT, ARCHIVE, FDEXIM, TSOSLNK, LMS, PERCON; Hinweise für den programmierenden Benutzer.
*Einsatz*
BS2000 Dialog- und Stapelbetrieb

[ 7] BS2000
**Benutzer-Kommandos (SDF-Format)**
Beschreibung

*Zielgruppe*
BS2000 Anwender (nicht privilegiert)
*Inhalt*
Alle BS2000 Systemkommandos in lexikalischer Reihenfolge mit Hinweisen und Beispielen.
Folgende Liefereinheiten sind berücksichtigt:
BS2000-GA, MSCF, JV, FT, TIAM
*Einsatz*
BS2000 Dialogbetrieb, Prozeduren, Stapelbetrieb

[ 8] 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

[ 9] BS2000
**Laserdrucker**
Beschreibung

BS2000-Anwendungsprogramme

[10] BS2000
**Softwareprodukt FMS**
Benutzerhandbuch

[11] BS1000
**Systeme zur automatischen Auftragsabwicklung (JMS)**
Beschreibung Teil 1

[12] BS1000
BS2000
TRANSDATA PDN
**Systemkonventionen**
Beschreibung

*Zielgruppe*
Benutzer von SIEMENS-Großrechenanlagen
*Inhalt*
Betriebssystemkonventionen für BS1000, BS2000 und TRANSDATA PDN, Konventionen für Datenträger, Codes für die Zeichendarstellung.

[13] LMS(BS2000) Anweisungsformate
Taschenbuch

*Zielgruppe*
erfahrene LMS-Benutzer
*Inhalt*
Das Taschenbuch enthält eine alphabetische Zusammenstellung aller LMS-Anweisungen und Verarbeitungsoperanden.

[14] BS2000
**DVS Einführung und Kommandoschnittstelle**
Beschreibung

*Zielgruppe*
nicht privilegierte BS2000-Anwender
*Inhalt*
- Funktionen des Datenverwaltungssystems im BS2000
- Verarbeitung von Platten- und Banddateien
- Zugriffsmethoden UPAM, SAM, BTAM, EAM, ISAM
- DVS-Kommandos

[15] BS2000
**LMS Unterprogramm-Schnittstelle**
Benutzerhandbuch

*Zielgruppe*
BS2000-Anwender, die LMS als Unterprogramm benutzen.
*Inhalt*
- Aufrufvorbereitungen mit Aufbau der Parameterstrukturen
- Unterprogramm-Funktionen
- COBOL-, C- und ASSEMLER-Schnittstelle mit den jeweiligen Parameterstrukturen und Beispiel

**Bestellen von Handbüchern**

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

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

# Stichwörter