1 Einleitung

Der Assembler **ASSEMBH** kann in zwei separaten Liefereinheiten unterschiedlicher Leistung bezogen werden:

ASSEMBH

- Übersetzung von Assemblerquellprogrammen in Objektmodule oder Bindelademodule
- Bereitstellung des Stand Alone Listing Generators ASSLG zusätzlich zur Listenausgabe im Standardformat
- Unterstützung der strukturierten Programmierung, d.h. Erweiterung um die Makros zur strukturierten Programmierung inklusive der Listenaufbereitungsprogramme für Nassi/Shneiderman-Diagramme und Strukturlisten.
- ILCS-Anschluß für die strukturierte Programmierung
- Möglichkeit des symbolischen Testens von Assemblerprogrammen durch Bereitstellen von LSD-Sätzen für die Dialogtesthilfe AID
- Unterstützung des ASSEMBH-Diagnoseprogramms ASSDIAG
- Ausgabe von strukturierten Listen bei Verwendung der Makros zur strukturierten Programmierung über den ASSEMBH
- Unterstützung der ESA-Befehle

ASSEMBH-BC

Der Assembler ASSEMBH-BC (Basic Configuration) ist der ASSEMBH im Grundausbau mit verminderter Leistung.

- Übersetzung von Assemblerquellprogrammen in Objektmodule oder Bindelademodule
- Listenausgabe im Standardformat

1.1 Kurzbeschreibung des Produkts

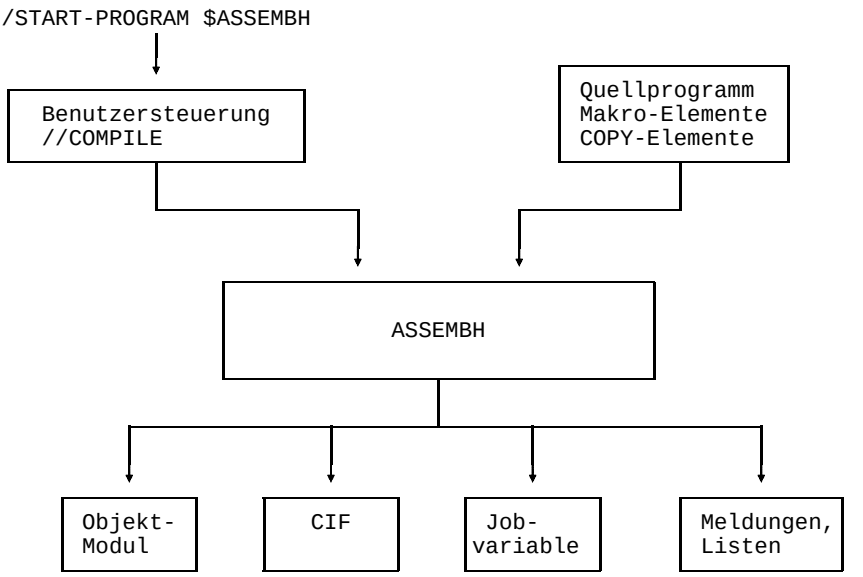
Der ASSEMBH ist ein 2 Pass-Assembler. Durch die Struktur des Assemblers ist die Bearbeitung eines Quellprogramms festgelegt. Hierdurch treten einige Inkompatibilitäten zum ASSEMB V30 auf (siehe 2.4.3, COMPILER-ACTION-Option).
Die Protokoll-Listen werden aus der internen Protokollinformation
(**C**ompiler **I**nformation **F**ile - **CIF**) erzeugt.

Funktionen des ersten Assembler Passes
Einlesen aller Instruktionen des Quellprogramms, inklusive eventueller COPY- und Makro-Elemente (siehe "ASSEMBH", Beschreibung [1]). Die Instruktionen werden syntaktisch analysiert und in eine Zwischensprache überführt, wobei die vollständige Makroverarbeitung durchgeführt wird. Es wird quellprogrammbezogene Protokollinformation abgelegt.

Funktionen des zweiten Assembler Passes
Aus der Zwischensprache wird der Objektmodul generiert und objektbezogene Protokollinformation abgelegt.

Standard Listing Generator
Der Standard Listing Generator erzeugt aus der internen Protokollinformation die Listen.

Überblick über den Datenfluß des ASSEMBH
Die Ein- und Ausgabemöglichkeiten des ASSEMBH sind in Kapitel 3 beschrieben.



Allgemeiner Datenfluß des ASSEMBH

1.2 Zielgruppe

Das Handbuch richtet sich an Anwender, die im BS2000 Programme in der Assembler- oder Makrosprache erstellen, benutzen oder warten. Betriebssystem-Grundkenntnisse werden vorausgesetzt.

1.3 Konzept des Handbuchs

Dieses Handbuch beschreibt die Handhabung des ASSEMBH im BS2000.
Kapitel 2 und 3 beschreiben das Übersetzen mit dem ASSEMBH,
Kapitel 4 und 10 die Unterstützung der strukturierten Programmierung,
Kapitel 5 und 6 das Binden, Laden, Starten und die zugehörigen Listen,
Kapitel 7 beschreibt die Verknüpfung von Programmen unterschiedlicher Programmiersprachen mit ASSEMBH-Programmen,
Kapitel 8 und 9 beschreiben Diagnose- und Testhilfen und
im Anhang sind die Meldungen des ASSEMBH, eine Übersicht über das Format der Assemblerbefehle und eine Gegenüberstellung von *COMOPT- und COMPILE-Anweisungen aufgeführt.

Die Assembler- und Makrosprache für den Übersetzer ASSEMBH ist im Handbuch "ASSEMBH (BS2000), Beschreibung" [1] beschrieben.

1.4 Änderungen gegenüber der vorigen Ausgabe

Die über das ganze Handbuch verteilten Korrekturen sind nicht eigens aufgeführt. Wesentliche fachliche Neuerungen und Änderungen sind folgende:

Mit dem Operanden SOURCE-FORMAT=STRUCTURED der LISTING-Option (siehe 2.4.4) werden strukturierte Listen erzeugt (Voraussetzung ist die Verwendung der Makros zur strukturierten Programmierung).
Auch der Stand-Alone-Listengenerator kann über die GENERATE-Anweisung (siehe 2.5) strukturierte Listen erzeugen.

Die über den ASSEMBH strukturierte Liste ist in Abschnitt 6.5 beschrieben.

Neuer Operand NOPRINT-PREFIX in der LISTING-Option (siehe 2.4.4).

Mit dem Operanden INSTRUCTION-SET=BS2000-ESA der Option SOURCE-PROPERTIES (siehe 2.4.1.4) wird der ESA-Befehlssatz generiert (ESA-Unterstützung siehe 5.8).

Mit dem Operanden MODULE-FORMAT=LLM der Option COMPILER-ACTION (siehe 2.4.2.1) wird ein Modul im LLM-Format erzeugt (siehe auch 3.2 und 6.6).

In der Option TEST-SUPPORT (siehe 2.4.5) wurde der Operand YES durch AID ersetzt.

Das Kapitel 5, ’Binden, Laden und Starten’ wurde überarbeitet und neu strukturiert sowie um einen neuen Abschnitt 5.2 (Binden mit dem BINDER) ergänzt.

Die Auflistung der Assemblerbefehle (siehe Anhang 11.3) wurde um die ESA-Befehle erweitert.
Das Kapitel "Handbuchergänzungen" enthält gesammelt weitere Neuerungen.

1.5 Verwendete Metasprache

Für die Formatdarstellung von BS2000-Kommandos und Programmanweisungen wird in diesem Benutzerhandbuch folgende Metasprache verwendet:

*STD	Großbuchstaben, Ziffern und Sonderzeichen, die nicht zu den metasprachlichen Zeichen gehören, bezeichnen Schlüsselwörter bzw. Konstanten, die in dieser Form angegeben werden müssen.
name	Kleinbuchstaben bezeichnen Variablen, die bei der Eingabe durch aktuelle Werte ersetzt werden müssen.
YES NO	Die Unterstreichung kennzeichnet den Standardwert, der automatisch eingesetzt wird, wenn keine Angabe gemacht wird.
{YES NO}	Geschweifte Klammern schließen Alternativen ein. Aus den angegebenen Größen muß eine ausgewählt werden. Wird der unterstrichene Standardwert gewünscht, ist keine Angabe erforderlich.
YES / NO	Ein Schrägstrich zwischen nebeneinander stehenden Angaben kennzeichnet ebenfalls Alternativen, von denen eine ausgewählt werden muß. Falls der angegebene Standardwert gewünscht wird, ist keine Angabe erforderlich.
[ ]	Eckige Klammern schließen Angaben ein, die weggelassen werden können.
( )	Runde Klammern sind Konstanten und müssen angegeben werden.
␣	Dieses Zeichen deutet an, daß mindestens ein Leerzeichen syntaktisch notwendig ist.
...	Drei Punkte bedeuten, daß die davorstehende Einheit mehrmals wiederholt werden kann.
[,...]	Komma und drei Punkte bedeuten, daß die davorstehende Einheit mehrmals wiederholt werden kann, aber jeweils durch ein Komma getrennt werden muß. Die eckigen Klammern zeigen die Wahlfreiheit an.

Hinweis

Zur SDF-Schnittstelle gibt es eine eigene Metasyntax (siehe 2.3.2).

2 Übersetzen

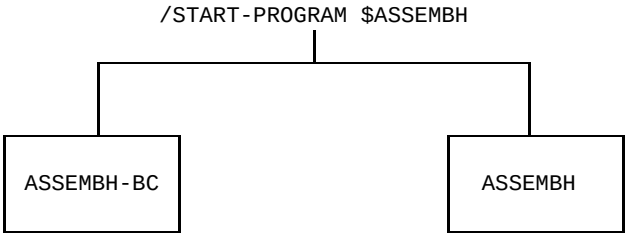
2.1 Aufruf des ASSEMBH

Der Assembler bearbeitet jeweils ein Quellprogramm. Ein Quellprogramm besteht aus einer Folge von Instruktionen (Assembleranweisungen, Assemblerbefehlen, Makroaufrufen und Makroanweisungen) und Kommentaren. Ein Quellprogramm kann aus einer oder mehreren Übersetzungseinheiten bestehen. Die einzelnen Übersetzungseinheiten sind im Quellprogramm durch END-Anweisungen voneinander getrennt. Der Assembler erzeugt für jede Übersetzungseinheit einen Objektmodul.

Die Steuerung des Assemblerlaufes erfolgt über Benutzersteuerungen (Optionen), sowie über Assembler-Anweisungen im Quellprogramm.

Nach dem Start werden die Optionen über die SDF-Schnittstelle (siehe "Einführung in die Dialogschnittstelle SDF" [5] und "BS2000/OSD-BC Kommandos" [6]) eingelesen und ausgewertet.
Die *COMOPT-Steuerung wird aus Kompatibilitätsgründen unterstützt (siehe 11.4 und 11.5).

Der Assembler **ASSEMBH-BC** und der Assembler **ASSEMBH** werden über die SDF-Kommandoschnittstelle wie folgt gestartet:



2.2 Steuerung des ASSEMBH

2.2.1 Einfachübersetzung

Einfachübersetzung heißt:
Ein Assemblerlauf mit einem Quellprogramm mit einer Übersetzungseinheit.

Der Assemblerlauf wird vom Start des Assemblers bis zur Beendigung durch die Optionen und den Inhalt des Quellprogramms gesteuert. Die Optioneneingabe erfolgt über SDF-Anweisungen (//COMPILE...) von SYSSTMT. Mit //END wird der Assemblerlauf beendet.

Für einen Assemblerlauf sind folgende Anweisungen notwendig:

```
/START-PROG $ASSEMBH
//COMPILE...
//END
```

2.2.2 Mehrfachübersetzung

Mehrfachübersetzung heißt:
ein Assemblerlauf mit einem Quellprogramm mit mehr als einer Übersetzungseinheit.

Bei der Mehrfachübersetzung werden zwischen der Bearbeitung der Übersetzungseinheiten keine Optionen gelesen, d.h. alle Übersetzungseinheiten des Quellprogramms werden vom Anfang bis zum Ende mit genau denselben Optionen übersetzt.

Für eine Mehrfachübersetzung sind folgende Anweisungen notwendig:

```
/START-PROG $ASSEMBH
//COMPILE...
//END
```

Hierzu muß das Quellprogramm, das über die COMPILE-Anweisung angegeben wurde, folgendes Format besitzen:

```
START
.
.              1. Übersetzungseinheit
.
END
START
.
.              2. Übersetzungseinheit
.
END
.
.
```

Hinweis

Falls sowohl das Quellprogramm als auch die Optionen von SYSDTA gelesen werden, wird keine Mehrfachübersetzung, sondern ein Assembler Restart durchgeführt. des Assemblers nach jeder Übersetzungseinheit ohne EOF-Erzeugung möglich.

2.2.3 Assembler Restart

Restart heißt:
Durch einen einzigen Aufruf des Assemblers können mehrere Quellprogramme hintereinander übersetzt werden.

Dabei bleiben alle Optionseinstellungen (außer für das Quellprogramm und die Objektmodulausgabe) der vorherigen Übersetzung erhalten, wenn sie nicht explizit überschrieben werden. Hierzu wird die Restart-Funktion des Assemblers verwendet, die durch folgende Steueranweisungen gestartet wird:

```
/START-PROG $ASSEMBH
//COMPILE SOURCE=...
//COMPILE SOURCE=...
   .
   .
   .
//COMPILE SOURCE=...
//COMPILE SOURCE=...
//END
```

2.3 SDF-Schnittstelle des ASSEMBH

Der ASSEMBH wird über SDF gesteuert, d.h. die Eingabe von Optionen muß im SDF-Format erfolgen. Auf Betriebssystemebene können die Kommandos sowohl im bisherigen ISP-Format als auch im SDF-Format angegeben werden.

Im Dialogbetrieb bietet SDF folgende Möglichkeiten:

- Eingabe von der Datensichtstation mit Benutzerführung in drei verschiedenen Stufen, nachfolgend "Menü-Modus" genannt.
- Eingabe von der Datensichtstation ohne Benutzerführung in zwei verschiedenen Formen, nachfolgend "Expert-Modus" genannt.
- Eingabe aus einer Prozedurdatei

Der Benutzer kann temporär vom Expert-Modus in den Menü-Modus wechseln. Arbeitet er im Menü-Modus mit mittlerer oder minimaler Führung, so kann er wiederum temporär in die nächsthöhere Führungsstufe wechseln.
Die Eingaben aus einer Prozedurdatei erfolgen im Expert-Modus.
Neben dem temporären Wechsel in einen anderen SDF-Modus gibt es auch die Möglichkeit, in einen Modus fest, d.h. permanent zu wechseln. Dies läßt sich mit folgendem SDF-Kommando (oder der SDF-Anweisung //MOD-SDF-OPT, siehe 2.3) erreichen.

/MODIFY-SDF-OPTIONS
GUIDANCE=UNCHANGED/EXPERT/NO/MAXIMUM/MEDIUM/MINIMUM

Dabei bedeutet

UNCHANGED
: Es gilt die bisherige Vereinbarung (default).

EXPERT
: Expert-Modus. Das System fordert mit "/" zur Kommandoeingabe bzw. mit "//" zur Anweisungeingabe auf; kein Syntaxfehlerdialog; detaillierte Fehlermeldungen; geblockte Kommandoeingabe. Dieser Modus ist standardmäßig nach dem LOGON-Kommando eingeschaltet.

NO
: Expert-Modus. Das System fordert mit "%CMD:" zur Kommandoeingabe bzw. mit "%STMT:" zur Anweisungseingabe auf; Syntaxfehlerdialog (Korrektur fehlerhafter Eingaben ohne Wiederholung des gesamten Kommandos); geblockte Kommandoeingabe (mehrere Kommandos, die durch das logische Zeilenendezeichen getrennt sind, können gleichzeitig abgeschickt werden).

MAXIMUM	Menü-Modus. Maximale Hilfestufe, d.h. sämtliche Operandenwerte mit Zusätzen, Hilfetexte für Kommandos und Operanden.
MEDIUM	Menü-Modus. Sämtliche Operandenwerte ohne Zusätze, Hilfetexte nur für Kommandos.
MINIMUM	Menü-Modus. Minimale Hilfestufe, d.h. nur Standardwerte der Operanden, keine Zusätze, keine Hilfetexte.

2.3.1 Bearbeitung des Operanden-Fragebogens

Nach dem Start des ASSEMBH gelangt man durch Eingabe von '?' auf die Anweisungsanfrage oder durch Angabe von //MOD-SDF-OPT GUIDANCE=MAX in den Operanden-Fragebogen der COMPILE-Anweisung, in dem jeder Operand einzeln abgefragt wird.

```
PROGRAM  : ASSEMBH                        STATEMENT: COMPILE


SOURCE                = *SYSDTA
MACRO-LIBRARY         = *NONE
COPY-LIBRARY          = *NONE
SOURCE-PROPERTIES     = STD
COMPILER-ACTION       = MODULE-GENERATION(MODE=STD,MODULE-FORMAT=OM)
MODULE-LIBRARY        = *OMF
COMPILATION-INFO      = NONE
LISTING               = STD
TEST-SUPPORT          = NO
COMPILER-TERMINATION = STD
CORRECTION-CYCLE      = NO
COMPILATION-SPACE     = STD

NEXT = *CONTINUE
*EXECUTE"F3" or + or Next-stmt or *CANCEL"K1"
```

Im folgenden sind wichtige Hinweise zur Bearbeitung des Operanden-Fragebogens zusammengefaßt. Die ausführliche Beschreibung der SDF-Bedienung finden Sie im Handbuch "Einführung in die Dialogschnittstelle (SDF)", [5].

Spezialeingaben

?	als Operandenwert liefert Hilfetext und Angabe des Wertebereichs für diesen Operanden. Hat SDF nach vorheriger fehlerhafter Eingabe die Meldung "CORRECT INCORRECT OPERANDS" gebracht, liefert das Fragezeichen zusätzliche detaillierte Fehlermeldungen. Der Zeilenrest muß nicht gelöscht werden.
!	als Operandenwert setzt für diesen Operanden den Standardwert wieder ein, wenn der abgebildete Standardwert vorher überschrieben wurde. Der Zeilenrest muß nicht gelöscht werden.
<operand>(	Geöffnete Klammer nach einem struktureinleitenden Operanden gibt den Unterfragebogen für die zugehörige Struktur aus. Nach der geöffneten Klammer angegebene Operanden werden im Unterfragebogen abgebildet.
–	als letztes Zeichen in einer Eingabezeile bewirkt die Ausgabe einer Fortsetzungszeile (siehe Beispiel in Abschnitt 5.6; bis zu 9 Fortsetzungszeilen pro Operand sind möglich).
LZF-Taste	löscht ab der Schreibmarke alle Zeichen der Eingabezeile.

Funktionstasten

K1 bricht den aktuellen Operanden-Fragebogen ab und wechselt zum übergeordneten Operanden-Fragebogen. Entspricht *CANCEL in der NEXT-Zeile.

K2 unterbricht ein laufendes Programm (z.B. den Assembler) oder eine laufende Prozedur.

K3 wiederholt den zuletzt ausgegebene Operanden-Fragebogen. Entspricht *RESTORE in der NEXT-Zeile.

F2 prüft Eingaben auf Syntaxfehler. Entspricht *TEST in der NEXT-Zeile.

F3 führt die aktuelle Operation aus. Entspricht *EXECUTE in der NEXT-Zeile.

NEXT-Zeile

Unter der NEXT-Zeile jeder Menü-Seite ist angegeben, welche Angaben gemacht werden dürfen. Die Begriffe sind im Handbuch "Einführung in die Dialogschnittstelle (SDF)", [5] erklärt bzw. sind selbsterklärend.

+, – blättert im Operanden-Fragebogen seitenweise vor und zurück.

++, -- schlägt die erste bzw. letzte Seite des Operanden-Fragebogens auf.

*EXECUTE führt die aktuelle Operation aus. Entspricht F3-Taste.

*CONTINUE blättert im Fragebogen vor, falls dessen Ende noch nicht erreicht ist. Andernfalls wird die aktuelle Operation ausgeführt.

*TEST prüft Eingaben auf Syntaxfehler. Entspricht F2-Taste.

*CANCEL bricht den aktuellen Fragebogen ab und wechselt zum übergeordneten Fragebogen. Entspricht K1-Taste.

*RESTORE wiederholt den zuletzt angezeigten Fragebogen. Entspricht K3-Taste.

<statement>?
führt die aktuelle Operation aus und gibt anschließend den Operanden-Fragebogen der Anweisung <statement> aus. Bereits angegebene Operandenwerte werden in den Fragebogen übernommen.

<statement>
führt die aktuelle Operation aus und anschließend auch die Anweisung <statement>. Ohne explizite Operandenangaben werden die voreingestellten Operandenwerte übernommen.

? schaltet für die aktuelle Eingabe in die nächsthöhere Hilfetext-Stufe.

*DOWN(<operand>)
gibt den Unterfragebogen für den angegebenen strukturfähigen Operanden <operand> aus.

*UP wechselt vom Unterfragebogen zurück in den übergeordneten Operanden-Fragebogen.

Beispiel

Es folgt ein Beispiel zur Bearbeitung des Operanden-Fragebogens der COMPILE-Anweisung. Wir wollen ein Quellprogramm namens test1, das als Element mit der Version 6 in der PLAM-Bibliothek plamlib steht, übersetzen.
Das übersetzte Programm, der Objektmodul, sowie das Übersetzungsprotokoll sollen in die PLAM-Bibliothek plamlib ausgegeben werden.

Name und Ort des Quellprogramms werden mit der SOURCE-Option angegeben.
Der Ort des Objektmoduls wird mit der Option MODULE-LIBRARY angegeben.
Der Ort des Übersetzungsprotokolls wird mit der LISTING-Option angegeben. Die Optionen der COMPILE-Anweisung sind in Abschnitt 2.4 beschrieben.

Der ASSEMBH wird gestartet und durch Eingabe eines Fragezeichens nach der Anweisungsanfrage (//) wird der Operanden-Fragebogen der COMPILE-Anweisung ausgegeben.

```
/START-PROG $ASSEMBH
%  BLS0500 PROGRAM 'ASSEMBH', VERSION 'V1.xxxx' OF 'yyyy-mm-dd' LOADED.
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1991. ALL
   RIGHTS RESERVED
%  ASS6010 V 1.xxxx OF BS2000 SIEMENS ASSEMBH READY
%//?
```

```
PROGRAM  : ASSEMBH                        STATEMENT: COMPILE


SOURCE                = *SYSDTA
MACRO-LIBRARY         = *NONE
COPY-LIBRARY          = *NONE
SOURCE-PROPERTIES     = STD
COMPILER-ACTION       = MODULE-GENERATION(MODE=STD,MODULE-FORMAT=OM)
MODULE-LIBRARY        = *OMF
COMPILATION-INFO      = NONE
LISTING               = STD
TEST-SUPPORT          = NO
COMPILER-TERMINATION = STD
CORRECTION-CYCLE      = NO
COMPILATION-SPACE     = STD

NEXT = *CONTINUE
       *EXECUTE"F3" or + or Next-stmt or *CANCEL"K1"
```

Die möglichen Operanden einer jeden Option können abgefragt werden. Wir geben z.B. bei den Optionen SOURCE und LISTING ein Fragezeichen ein.

```
PROGRAM  : ASSEMBH                        STATEMENT: COMPILE


SOURCE                = ?SYSDTA
MACRO-LIBRARY         = *NONE
COPY-LIBRARY          = *NONE
SOURCE-PROPERTIES     = STD
COMPILER-ACTION       = MODULE-GENERATION(MODE=STD,MODULE-FORMAT=OM)
MODULE-LIBRARY        = *OMF
COMPILATION-INFO      = NONE
LISTING               = ?TD
TEST-SUPPORT          = NO
COMPILER-TERMINATION  = STD
CORRECTION-CYCLE      = NO
COMPILATION-SPACE     = STD
-------------------------------------------------------------------------------
NEXT = *CONTINUE
       *EXECUTE"F3" or + or Next-stmt or *CANCEL"K1"
```

Die möglichen Operanden werden ausgegeben:

```
PROGRAM  : ASSEMBH                        STATEMENT: COMPILE
OPERANDS : SOURCE=*SYSDTA,LISTING=STD

-------------------------------------------------------------------------------
SOURCE                = *SYSDTA
                        *SYSDTA or full-filename_1..54 or *LIBRARY-ELEMENT(LIBRAR
                        Y=?,ELEMENT=?)
                        specification of the file containing the source
MACRO-LIBRARY         = *NONE
COPY-LIBRARY          = *NONE
SOURCE-PROPERTIES     = STD
COMPILER-ACTION       = MODULE-GENERATION(MODE=STD,MODULE-FORMAT=OM)
MODULE-LIBRARY        = *OMF
COMPILATION-INFO      = NONE
LISTING               = STD
                        STD or PARAMETERS()
                        selection of size and structure of the standard listing
TEST-SUPPORT          = NO
COMPILER-TERMINATION  = STD
CORRECTION-CYCLE      = NO
COMPILATION-SPACE     = STD
-------------------------------------------------------------------------------
NEXT = *CONTINUE
       *EXECUTE"F3" or + or Next-stmt or *CANCEL"K1"
```

Wir geben jetzt die Operandenwerte für folgende Optionen ein:
SOURCE: Bibliotheksname plamlib und Elementname test1 mit Version 6
MODULE-LIBRARY: Bibliotheksname plamlib
LISTING: Bibliotheksname plamlib

```
PROGRAM  : ASSEMBH                        STATEMENT: COMPILE
OPERANDS : SOURCE=*SYSDTA,LISTING=STD

-------------------------------------------------------------------------------
SOURCE               = (plamlib,test1(6))
                       *SYSDTA or full-filename_1..54 or *LIBRARY-ELEMENT(LIBRAR
                       Y=?,ELEMENT=?)
                       specification of the file containing the source
MACRO-LIBRARY        = *NONE
COPY-LIBRARY         = *NONE
SOURCE-PROPERTIES    = STD
COMPILER-ACTION      = MODULE-GENERATION(MODE=STD,MODULE-FORMAT=OM)
MODULE-LIBRARY       = plamlib
COMPILATION-INFO     = NONE
LISTING              = par(output=(plamlib))
                       STD or PARAMETERS()
                       selection of size and structure of the standard listing
TEST-SUPPORT         = NO
COMPILER-TERMINATION = STD
CORRECCTION-CYCLE    = NO
COMPILATION-SPACE    = STD
-------------------------------------------------------------------------------
NEXT = *CONTINUE
       *EXECUTE"F3" or + or Next-stmt or *CANCEL"K1"
```

```
%  ASS6011 ASSEMBLY TIME: 183 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT : NO ERRORS
%  ASS6006 LISTING-GENERATOR TIME : 531 MSEC
%//


Nach der Übersetzung meldet sich der Assembler wieder mit einer
Anweisungsanfrage. Der Assemblerlauf wird nun mit der Anweisung END beendet.

%//END
%  ASS6012 END OF ASSEMBH
```

2.3.2 Metasyntax zur SDF-Schnittstelle

Die Formatübersicht der COMPILE-Anweisung (siehe 2.4) ist in zwei Felder aufgeteilt. Das 1. Feld enthält die COMPILE-Anweisung (COMPILE), das 2. Feld enthält die möglichen Optionen mit den Operandenwerten.

In der Formatdarstellung werden bestimmte Zeichen verwendet (sog. Metazeichen), deren Bedeutung in der nachfolgenden Tabelle erläutert ist:

Kennzeichnung	Bedeutung	Beispiele
GROSSBUCHSTABEN	Großbuchstaben bezeichnen Schlüsselwörter. Einige Schlüsselwörter beginnen mit *	LITERAL = YES SOURCE = *SYSDTA
=	Das Gleichheitszeichen verbindet einen Operandennamen mit den dazugehörenden Operandenwerten.	MODULE-LIBRARY = *OMF
< >	Spitze Klammern kennzeichnen Variablen, deren Wertevorrat durch Datentypen und ihre Zusätze beschrieben wird (siehe Tabellen 2 und 3).	VERSION = <text 1..24>
Unterstreichung	Der Unterstrich kennzeichnet den Standardwert eines Operanden.	LISTING = STD
/	Der Schrägstrich trennt alternative Operandenwerte.	LASER-PRINTER = NO / ND2
(...)	Runde Klammern kennzeichnen Operandenwerte, die eine Struktur einleiten.	SYMBOL = NO / YES(...)
Einrückung	Die Einrückung kennzeichnet die Abhängigkeit zu dem jeweils übergeordneten Operanden.	SYMBOL = NO / YES(...) YES(...)

Kennzeichnung	Bedeutung	Beispiele
|	Der Strich kennzeichnet zusammengehörende Operanden einer Struktur. Sein Verlauf zeigt Anfang und Ende einer Struktur an. Innerhalb einer Struktur können weitere Strukturen auftreten. Die Anzahl senkrechter Striche vor einem Operanden entspricht der Strukturtiefe.	*LIBRARY-ELEMENT(...) | LIBRARY = | ,ELEMENT = | | VERSION =
,	Das Komma steht vor weiteren Operanden der gleichen Strukturstufe.	,LITERAL = NO / YES ,MACRO = NO / YES
list-poss(n):	Aus den list-poss folgenden Operandenwerten kann eine Liste gebildet werden. Ist (n) angegeben, können maximal n Elemente in der Liste vorkommen. Enthält die Liste mehr als ein Element, muß sie in runde Klammern eingeschlossen werden.	list-poss: <full-filename> / *LINK list-poss(256): <name 1..1>

Hinweis

Konstante Operandenwerte beginnen manchmal mit Stern (*). Dies gilt, wenn alternativ zum konstanten Wert ein Datentyp vorhanden ist, dessen Zeichenvorrat die Zeichenfolge des konstanten Wertes zuläßt.

Beispiel

ELEMENT = *ALL / <name>

Für den Datentyp name darf der Wert "ALL" eingesetzt werden. Zur Unterscheidung muß daher der gleichnamige konstante Operandenwert mit Stern (*) beginnen: *ALL.

2.3.2.1 Datentypen und Zusätze

Datentyp	Zeichenvorrat	Bedeutung
full-filename 1..54	A - Z, 0 - 9, $, #, @, Punkt, Bindestrich	Vollqualifizierter Name einer katalogisierten Datei, einer PLAM-Bibliothek oder eines Bibliothekselements. In Elementnamen ist ein Unterstrich, wie im LMS möglich, nicht zulässig. Das erste Zeichen muß eine Ziffer oder ein Buchstabe sein, das letzte Zeichen darf kein Bindestrich oder Punkt sein. Der Name darf nicht nur aus Ziffern oder Sonderzeichen bestehen. Die maximale Länge beträgt - einschließlich user-id und cat-id 54 Zeichen.
full-filename 1..8	A - Z, 0 - 9, $, #, @, Punkt, Bindestrich	Linkname einer katalogisierten Datei oder einer PLAM-Bibliothek. Das 1. Zeichen muß ein Buchstabe oder eine Ziffer sein, das letzte Zeichen darf kein Bindestrich oder Punkt sein. Der Name darf nicht nur aus Ziffern oder Sonderzeichen bestehen. Die maximale Länge beträgt 8 Zeichen.
composed-name 1..24	A - Z, 0 - 9, $, #, @, Punkt, Bindestrich	Versionsangabe eines PLAM-Bibliothekselements. Die maximale Länge beträgt 24 Zeichen. Der von LMS unterstützte Zeichenvorrat ist möglich.
composed-name 1..64	A - Z, 0 - 9, $, #, @, Punkt, Bindestrich	Name eines PLAM-Bibliothekselements
name 1..64	A - Z, 0 - 9, $, #, @	Präfix für Makro- und Adreßnamen Ab SDF V2.0 ist auch ein Unterstrich möglich.
integer 2..255	0 - 9	Gibt Intervall an (0-32767)
c-string (character-string)	EBCDIC-Zeichen	In Hochkommas eingeschlossene Folge von EBCDIC-Zeichen. Der Buchstabe C kann vorangestellt werden.

Die Datentypen können folgende Zusätze enthalten:

Zusatz	Bedeutung
1..n integer m..n	Erlaubte Anzahl von Zeichen. Gibt einen Intervallwert an.
without -gen(eration) -vers(ion) -cat-id	 Angabe einer Dateigeneration oder Dateigenerations-gruppe nicht erlaubt. Angabe einer Elementversion nicht erlaubt. Angabe einer Katalogkennung nicht erlaubt.

Hinweis

– Zu ’@’
  Ab PLAM V1.4 ist ’@’ als Version bei Objektmodulausgabe nicht mehr möglich.

2.4 COMPILE-Anweisung

Diese Anweisung steuert die Übersetzung eines Assembler-Quellprogramms. Sie besitzt folgende Operanden der obersten Strukturstufe:

```
COMPILE
```

```
zur Eingabeunterstützung:

     SOURCE =
    ,MACRO-LIBRARY =
    ,COPY-LIBRARY =
    ,SOURCE-PROPERTIES =

zur Modulerzeugung:

    ,COMPILER-ACTION =
    ,MODULE-LIBRARY =

zur CIF-Unterstützung:

    ,COMPILATION-INFO =

zur Protokollunterstützung:

    ,LISTING =

zur Testunterstützung:

    ,TEST-SUPPORT =

zum Übersetzungsabbruch:

    ,COMPILER-TERMINATION =

zum Aktivieren des Korrekturzyklus:

    ,CORRECTION-CYCLE =

zur Maintenanceunterstützung:

    ,MAINTENANCE-OPTIONS =

zur Verringerung des virtuellen Adreßraumbedarfs

    ,COMPILATION-SPACE =
```

2.4.1 Optionen zur Eingabeunterstützung

Diese Optionen nennen das zu übersetzende Quellprogramm, die Makrobibliotheken des Benutzers, Bibliotheken für COPY-Elemente, sowie die Formate des Quellprogramms, den Befehlssatz und einen Wert für den globalen variablen Systemparameter &SYSPARM.

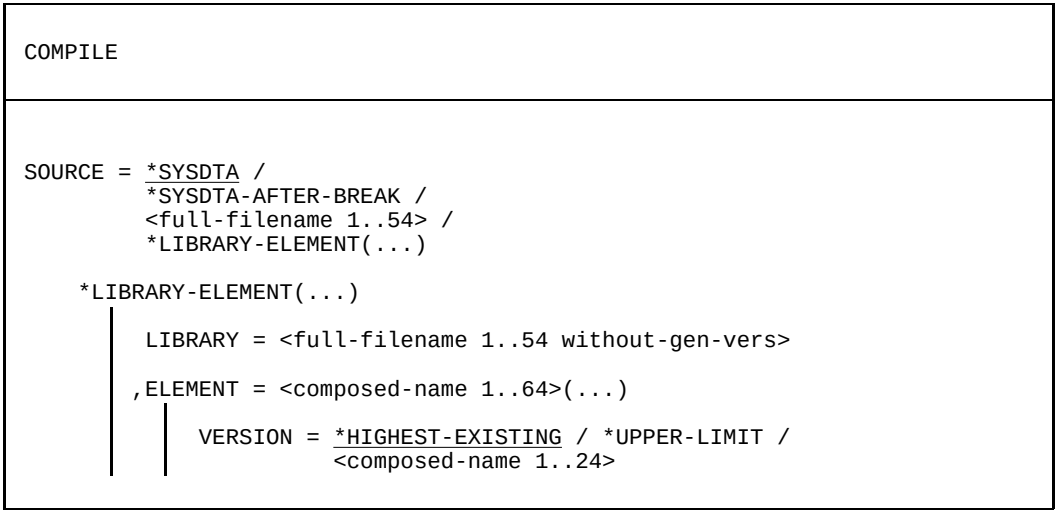
```
COMPILE

 SOURCE =              Quellprogramm
,MACRO-LIBRARY =       Benutzer-Makrobibliotheken
,COPY-LIBRARY =        Bibliotheken für COPY-Elemente
,SOURCE-PROPERTIES =   Formate des Quellprogramms, Befehlssatz,
                       Wert für &SYSPARM
```

2.4.1.1 SOURCE-Option

Funktion

Mit SOURCE kann angegeben werden, von wo das Quellprogramm eingelesen werden soll. Bei Weglassen der SOURCE-Option wird das Quellprogramm von SYSDTA gelesen.

Format



SOURCE = *SYSDTA
Das Quellprogramm wird von SYSDTA gelesen.

SOURCE = *SYSDTA-AFTER-BREAK
Nach dem Einlesen der Optionen erfolgt eine Unterbrechung. Über SYSCMD kann mit dem ASSIGN-SYSDTA-Kommando SYSDTA zugeordnet werden. Das Quellprogramm wird dann über SYSDTA eingelesen. Die neue Zuordnung von SYSDTA wird aber erst nach dem Abarbeiten aller Optionen wirksam.

SOURCE = <full-filename 1..54>
Name einer katalogisierten Datei, in der das Quellprogramm steht

SOURCE = *LIBRARY-ELEMENT(...)

LIBRARY = <full-filename 1..54>
Name einer PLAM-Bibliothek, in der das Quellprogramm steht

ELEMENT = <composed-name 1..64>(...)
Name eines Elements vom Typ S (Quellprogramm) der angegebenen PLAM-Bibliothek

VERSION = *HIGHEST-EXISTING
Das Element mit der höchsten existierenden Version wird verwendet.

VERSION = *UPPER-LIMIT
Das Element mit der höchstmöglichen Version wird verwendet.

VERSION = <composed-name 1..24>
Versionsbezeichnung des Elements

Hinweise

– Zur Operanden-Eingabe
Bei Angabe eines Bibliothekselements kann die Eingabe '*LIBRARY-ELEMENT (LIBRARY=...,ELEMENT=...)' weggelassen werden.

Beispiel

Anstelle von

```
//C SOURCE=*LIBRARY-ELEMENT(LIBRARY=bibl,ELEMENT=elem(VERSION=007))
```
kann

```
//C SOURCE=(bibl,elem(007))
```
geschrieben werden

– Zu Bibliotheken
Zusätzlich zu den PLAM-Bibliotheken sind auch noch OSM-Quellprogrammbibliotheken zugelassen.

2.4.1.2 MACRO-LIBRARY-Option

Funktion

Mit MACRO-LIBRARY können bis zu 100 benutzereigene PLAM-Bibliotheken angegeben werden, aus denen die Makro-Elemente gelesen werden sollen (PLAM-Bibliothekselemente vom Typ M).

Format

```
COMPILE

MACRO-LIBRARY = *NONE /
                list-poss(100): <full-filename 1..54 without-gen-vers> /
                                *LINK(...)

    *LINK(...)
       |
       |  LINK-NAME = <full-filename 1..8 without-gen-vers>
```

MACRO-LIBRARY = *NONE
Es wird keine benutzereigene Makro-Bibliothek zugewiesen.

MACRO-LIBRARY = list-poss(100): <full-filename 1..54>
Namen der PLAM-Bibliotheken, in der die Makro-Elemente stehen

MACRO-LIBRARY = list-poss(100): *LINK(...)

LINK-NAME = <full-filename 1..8>)
Nennt den zugeordneten Link-Namen einer Makro-Bibliothek.

Hinweise

– Zur Suchhierarchie
  Siehe Makro-Elemente 3.1.2, Suchreihenfolge.

– Zu list-possible
  Mischung von Bibliotheks-Namen und Link-Namen in einer Liste sind möglich.

  Beispiel

  ```
  /SET-FILE-LINK LINK-NAME=maclink,FILE-NAME=maclib
  ```

  Der Makrobibliothek maclib wird der Link-Name maclink zugewiesen

  ```
  //C MAC-LIB=(maclib1,maclib2,*LINK(maclink))
  ```

  Es werden die Makrobibliotheken maclib1 und maclib2 zugewiesen, sowie über den Link-Namen maclink die Makrobibliothek maclib.

– Zu Bibliotheken
  Zusätzlich zu den PLAM-Bibliotheken sind auch noch OSM-Makrobibliotheken im MLU-Format zugelassen.

2.4.1.3 COPY-LIBRARY-Option

Funktion

Mit COPY-LIBRARY können bis zu 100 benutzereigene PLAM-Bibliotheken angegeben werden, aus denen die COPY-Elemente gelesen werden sollen (PLAM-Bibliothekselemente vom Typ S oder M).

Format

```
COMPILE

COPY-LIBRARY = *NONE /
               list-poss(100):
               <full-filename 1..54 without gen-vers>(...) /
               *LINK(...)

    <full-filename 1..54 without gen-vers>(...)
       |
       |  ELEMENT-TYPE = SOURCE-ONLY / MACRO-ONLY / BOTH

    *LINK(...)
       |
       |  LINK-NAME = <full-filename 1..8 without-gen-vers>
       |
       | ,ELEMENT-TYPE = SOURCE-ONLY / MACRO-ONLY / BOTH
```

COPY-LIBRARY = *NONE
Es wird keine benutzereigene COPY-Bibliothek zugewiesen.

COPY-LIBRARY = list-poss(100): <full-filename 1..54>(...)
Name der PLAM-Bibliothek, in der die COPY-Elemente stehen.

ELEMENT-TYPE = SOURCE-ONLY / MACRO-ONLY / BOTH
Nennt den Element-Typ (S,M) der COPY-Elemente, die aus den angegebenen Bibliotheken gelesen werden sollen (bei BOTH zuerst S, dann M)

COPY-LIBRARY = *LINK(...)

LINK-NAME = list-poss(100): <full-filename 1..8>
Nennt den zugeordneten Link-Namen einer COPY-Bibliothek.

ELEMENT-TYPE = SOURCE-ONLY / MACRO-ONLY / BOTH
Nennt den Element-Typ (S,M) der COPY-Elemente, die aus den angegebenen Bibliotheken gelesen werden sollen (bei BOTH zuerst S, dann M).

Hinweise

– Zur Suchhierarchie
  Siehe COPY-Elemente 3.1.3, Suchreihenfolge.

– Zur ELEMENT-TYPE Spezifikation
  Die ELEMENT-TYPE Spezifikation gilt jeweils nur für die angegebene Bibliothek.

– Zu list-possible
  Mischung von Bibliotheks-Namen und Link-Namen in einer Liste sind möglich.

  Beispiel

  ```
  /SET-FILE-LINK LINK-NAME=coplink,FILE-NAME=coplib
  ```

  Der Bibliothek coplib wird der Link-Name coplink zugewiesen.

  ```
  //C COPY-LIB=(coplib1(ELEM-TYPE=MAC-O),coplib2,*LINK(coplink))
  ```

  Die Bibliotheken coplib1 und coplib2 werden zugewiesen, sowie über den Link-Namen coplink die Bibliothek coplib.

– Zu Bibliotheken
  Zusätzlich zu den PLAM-Bibliotheken sind auch noch OSM-Quellprogramm- und OSM-Makrobibliotheken im MLU-Format zugelassen.

2.4.1.4 SOURCE-PROPERTIES-Option

Funktion

Mit SOURCE-PROPERTIES können die Formate des Quellprogramms, der Befehlssatz, sowie ein Wert für den Systemparameter &SYSPARM bestimmt werden.

Format

```
COMPILE

SOURCE-PROPERTIES = STD / PARAMETERS(...)

    PARAMETERS(...)
        |  FROM-COLUMN = 1 / <integer 1..70>
        |
        | ,TO-COLUMN = 71 / <integer 2..255>
        |
        | ,CONTINUATION-COLUMN = 16 / <integer 1..255> / NO-CONTINUATION
        |
        | ,LOW-CASE-CONVERSION = NO / YES
        |
        | ,INSTRUCTION-SET = HOST-STD / BS2000-ESA / BS2000-XS / BS2000-NXS
        |                    / DUET
        |
        | ,PREDEFINED-VARIABLES = NONE / SYS(...)
        |
        |    SYS(...)
        |       |
        |       |  SYSPARM = <c-string 1..255>
```

SOURCE-PROPERTIES = STD
Es werden die Standardwerte der Struktur PARAMETERS(...) eingesetzt.

SOURCE-PROPERTIES = PARAMETERS(...)

FROM-COLUMN = 1 / <integer 1..70>
Nennt die Anfangsspalte für die Assemblierung einer Quellzeile.

TO-COLUMN = 71 / <integer 2..255>
Nennt die Endspalte für die Assemblierung einer Quellzeile.

CONTINUATION-COLUMN = 16 / <integer 1..255> / NO-CONTINUATION
Nennt die Anfangsspalte für die Fortsetzungszeile einer Instruktion in der Quelle. Bei NO-CONTINUATION wird keine Fortsetzung durchgeführt.

LOW-CASE-CONVERSION = NO / YES
Bei YES werden Klein-Buchstaben in Großbuchstaben umgewandelt (siehe ASSEMBH-Beschreibung [1])

INSTRUCTION-SET = HOST-STD / BS2000-ESA / BS2000-XS / BS2000-NXS / DUET
Nennt den Befehlssatz, der verwendet werden soll (siehe 11.3).
Bei HOST-STD gilt: Abhängig vom Hardware/Software-Interface der CPU wird BS2000-ESA, BS2000-XS oder BS2000-NXS genommen.

PREDEFINED-VARIABLES = NONE / SYS(...)
Übergibt externe Benutzerinformation an einen Systemparameter.

SYSPARM = <c-string 1..255>
Weist dem Systemparameter &SYSPARM einen Wert zu.

Hinweise

– Zur Operanden-Eingabe
Die Angaben ’SOURCE-PROPERTIES’ und ’PARAMETERS( )’ können weggelassen werden.

Beispiel

Anstelle von

```
//C SOURCE-PROPERTIES=PARAMETERS(FROM-COLUMN=2)
```
kann

```
//C SOURCE-PROPER=(FROM-COLUMN=2)
```
oder

```
//C S-PRO=(2)
```
oder

```
//C FROM-COLUMN=2
```
geschrieben werden.

Bei Eingabe von ’SYSPARM’ können auch ’PREDEFINED-VARIABLES’ und SYS( ) weggelassen werden.

Beispiel

Anstelle von

```
//C S-PRO=PREDEFINED-VARIABLES(SYS(SYSPARM='100'))
```
kann

```
//C S-PRO=(SYSPARM='100')
```
geschrieben werden.

– Zum Format des Quellprogramms

  Bei NO-CONTINUATION wird keine Fortsetzung einer Zeile durchgeführt. Ansonsten muß das Fortsetzungszeichen in Endspalte + 1 gesetzt werden und die Fortsetzungszeile ab Fortsetzungsspalte beginnen. Dabei gilt:

  Anfangsspalte ≤ Fortsetzungsspalte ≤ Endspalte

  Die Anfangsspalte muß kleiner als die Endspalte sein.

  Bei unzulässigen Angaben werden die Standardwerte verwendet.

– Zu Spaltenangaben bei

  Makro-Elementen

  FROM-COLUMN, TO-COLUMN, und CONTINUATION-COLUMN sind für Eingaben aus Makroelementen entsprechend dem Standardformat mit 1, 71 und 16 besetzt. Das gilt generell für alle Makroelemente (aus System- und aus benutzereigenen Makrobibliotheken).

  Sourcedeck-Makros

  Eine Makrodefinition im Quelltext wird wie eine Quellzeile behandelt, d.h. es gelten die für den Quelltext eingestellten Optionen.

  COPY-Elementen

  COPY-Elemente werden wie die Zeile, in der die COPY-Anweisung steht gelesen. D.h. eine COPY-Anweisung im Quelltext oder in einem Sourcedeck-Makro wird wie eine Quellzeile behandelt, d.h. es gelten die für den Quelltext eingestellten Optionen. Eine COPY-Anweisung in einem Bibliotheksmakro wird im Standardformat gelesen.

2.4.2 Optionen zur Modulerzeugung

Diese Optionen steuern die Ausgabe eines Objektmoduls oder Bindelademoduls.

COMPILE	
,COMPILER-ACTION =	Generierung eines Objektmoduls oder Bindelademoduls
,MODULE-LIBRARY =	Bibliothek für Module

2.4.2.1 COMPILER-ACTION-Option

Funktion

Mit COMPILER-ACTION wird gesteuert, ob ein Objektmodul oder ein Bindelademodul erzeugt oder nur ein Syntax-Check durchgeführt werden soll.

Format

```
COMPILE

COMPILER-ACTION = MODULE-GENERATION(...) / SYNTAX-CHECK(...)
    MODULE-GENERATION(...)
        MODE = STD / F-ASSEMB-COMPATIBLE
       ,MODULE-FORMAT = OM / LLM(...)
          LLM(...)
              EXTERNAL-NAMES = STD / TRUNCATED
    SYNTAX-CHECK(...)
        MODE = STD / F-ASSEMB-COMPATIBLE
```

COMPILER-ACTION = MODULE-GENERATION(...)

MODE = STD
Es wird eine syntaktische Überprüfung durchgeführt und ein Modul generiert.

MODE = F-ASSEMB-COMPATIBLE
Ausgewählte Inkompatibilitäten zum F-Assembler (ASSEMB) werden bei Angabe dieses Operanden vermieden. Die Bearbeitung erfolgt kompatibel zum ASSEMB V30.0A.

MODULE-FORMAT = OM
Es wird ein Modul im OM-Format (Objektmodul-Format) erzeugt, der entweder standardmäßig in die temporäre EAM-Objektmoduldatei oder als Bibliothekselement vom Typ R abgelegt wird (siehe Option MODULE-LIBRARY).

MODULE-FORMAT = LLM(...)

EXTERNAL-NAMES = STD / TRUNCATED

Es wird ein Modul im LLM-Format (Bindelademodul-Format) erzeugt.
Dabei werden externe Namen auf 32 Zeichen (STD) oder auf 8 Zeichen (TRUNCATED) verkürzt. Die Ablage kann nur als Bibliothekselement (vom Typ L) erfolgen (siehe Option MODULE-LIBRARY).

COMPILER-ACTION = SYNTAX-CHECK(...)

MODE = STD
Es wird kein Modul generiert, sondern nur ein Syntax-Check durchgeführt.

MODE = F-ASSEMB-COMPATIBLE
Ausgewählte Inkompatibilitäten zum F-Assembler (ASSEMB) werden bei Angabe dieses Operanden vermieden. Die Bearbeitung erfolgt kompatibel zum ASSEMB V30.0A.

Hinweise

Zu F-ASSEMB-COMPATIBLE

- Die betroffenen Programme sollten geändert werden, da dieser Operand wieder abgeschafft wird.

Ausgewählte Inkompatibilitäten

- SETA- und SETB-Operanden werden kompatibel zum F-Assembler behandelt.
- Die Schreibweise C'...' ist in SETA-, SETB- und Vergleichsausdrücken zugelassen. Das C wird ignoriert.
- Kann ein Zeichenwert nicht konvertiert werden, so erfolgt keine Meldung und es wird mit dem Ersatzwert Null weitergerechnet.
- Bei SPACE und EJECT werden fehlerhafte Operanden ignoriert.
- Kommentare bei MNOTE
  Fehlt das trennende Blank zwischen MNOTE-Operanden und Kommentarfeld, so wird nach dem schließenden Apostroph alles als Kommentar betrachtet. Fehler bei ungepaarten Apostrophen werden nicht erkannt.

2.4.2.2 MODULE-LIBRARY-Option

Funktion

Mit MODULE-LIBRARY kann angegeben werden, wohin der Modul (Objektmodul oder Bindelademodul) ausgegeben wird.

Format

```
COMPILE

MODULE-LIBRARY = *OMF / <full-filename 1..54 without gen-vers>(...)

     <full-filename 1..54 without gen-vers>(...)

          ELEMENT = *STD(...) /
                    <composed-name 1..64>(...)

           *STD(...)

               VERSION = *UPPER-LIMIT / *INCREMENT / *HIGHEST-EXISTING /
                         <composed-name 1..24>

           <composed-name 1..64>(...)

               VERSION = *UPPER-LIMIT / *INCREMENT / *HIGHEST-EXISTING /
                         <composed-name 1..24>
```

MODULE-LIBRARY = *OMF
Der Objektmodul wird in die temporäre EAM-Objektmoduldatei ausgegeben.

MODULE-LIBRARY = <full-filename 1..54 without gen-vers>(...)
Name der PLAM-Bibliothek, in die der Objektmodul (OM-Format) oder der Bindelademodul (LLM-Format) ausgegeben wird.
Für LLMs muß mit der MODULE-LIBRARY-Option eine Bibliothek angegeben werden. Wird keine Bibliothek angegeben, so erfolgt eine Meldung.

ELEMENT = *STD(...)
Name des Objektmoduls (Bibliothekselement vom Typ R) oder Bindelademoduls (Bibliothekselement vom Typ L). Das Element erhält den Namen des ersten Programmabschnitts. Ist der erste Programmabschnitt unbenannt, so wird kein Modul erzeugt; es wird eine Meldung ausgegeben.

VERSION = *UPPER-LIMIT
Das Element erhält die höchstmögliche Version.

VERSION = *INCREMENT
Das Element erhält die erhöhte Version.

VERSION = *HIGHEST-EXISTING
Das Element erhält die höchste existierende Version.

VERSION = <composed-name 1..24>
Versionsbezeichnung des Elements.

ELEMENT = <composed-name 1..64>(...)
Name des Elements

VERSION = *UPPER-LIMIT
Das Element erhält die höchstmögliche Version.

VERSION = *INCREMENT
Das Element erhält die erhöhte Version.

VERSION = *HIGHEST-EXISTING
Das Element erhält die höchste existierende Version.

VERSION = <composed-name 1..24>
Versionsbezeichnung des Elements.

Hinweise

- Zur Länge des Element-Namens
  Der Binder TSOSLNK verarbeitet zur Zeit nur Element-Namen mit maximal 8 Zeichen.
  Element-Namen von LLMs können zur Weiterverarbeitung mit dem Binder BINDER oder Bindelader DBL maximal 32 Zeichen lang sein.
- Zu '@'
  Ab PLAM V1.4 ist '@' als Version bei Objektmodulausgabe nicht mehr möglich.
- Zu VERSION = *INCREMENT (erhöhte Version)
  Siehe Versionsbezeichnung *INCREMENT und automatische Versionserhöhung im Benutzerhandbuch LMS [8].

2.4.3 Option zur CIF-Unterstützung COMPILATION-INFO-Option

Funktion

Diese Option steuert, ob die CIF-Information in einer PLAM-Bibliothek gespeichert werden soll.

Format

```
COMPILE

COMPILATION-INFO = NONE / PARAMETERS(...)

    PARAMETERS(...)
        |  INFORMATION = STD / MAXIMUM
        | ,OUTPUT = *LIBRARY-ELEMENT(...)
        |
        |    *LIBRARY-ELEMENT(...)
        |        |  LIBRARY = <full-filename 1..54 without gen-vers>
        |        |
        |        | ,ELEMENT = <composed-name 1..64>(...)
        |        |        |
        |        |        |  VERSION = *UPPER-LIMIT / *INCREMENT /
        |        |        |            *HIGHEST-EXISTING / <composed-name 1..24>
```

COMPILATION-INFO = NONE
CIF wird nur temporär zur Erstellung des Protokolls erzeugt.

COMPILATION-INFO = PARAMETERS(...)

INFORMATION = STD / MAXIMUM
Umfang der CIF-Information.

STD bedeutet: Nur diejenige Information wird zur Verfügung gestellt, die in der LISTING-Option angefordert wurde.

MAXIMUM bedeutet: die vollständige Information wird zur Verfügung gestellt. Das Standard-Listing wird davon nicht beeinflußt; dieses wird über die Option LISTING gesteuert.

OUTPUT = *LIBRARY-ELEMENT(...)

LIBRARY = <full-filename 1..54>
Name einer PLAM-Bibliothek, in die die CIF-Information abgelegt wird.

ELEMENT = <composed-name 1..64>(...)
Name des Bibliothekselements (Typ H).

VERSION = *UPPER-LIMIT
Das Element erhält die höchstmögliche Version.

VERSION = *INCREMENT
Das Element erhält die erhöhte Version.

VERSION = *HIGHEST-EXISTING
Das Element erhält die höchste existierende Version.

VERSION = <composed-name 1..24>
Versionsbezeichnung des Elements.

Hinweise

– Zur Bildung des Element-Namens bei Mehrfachübersetzung
  Bei Mehrfachübersetzung wird für jede Übersetzungseinheit ein CIF-Element in der angegebenen Bibliothek abgelegt. Der Element-Name für die n-te Übersetzungseinheit ($n \geq 2$) wird durch Anhängen von '.n' an den CIF-Element-Namen der 1. Übersetzungseinheit gebildet:

  cifelementname.n (Version bleibt erhalten)

– Zu VERSION = *INCREMENT (erhöhte Version)
  Siehe Versionsbezeichnung *INCREMENT und automatische Versionserhöhung im Benutzerhandbuch LMS [8].

2.4.4 Option zur Protokollunterstützung LISTING-Option

Funktion

Mit LISTING können das Format, der Umfang und der Ablageort des Assemblerprotokolls bestimmt werden.

Format

```
COMPILE

LISTING = STD / PARAMETERS(...)

  PARAMETERS(...)
    |
    |  SOURCE-PRINT = NO / WITH-OBJECT-CODE(...) / SOURCE-ONLY(...) /
    |                 ERRORS-ONLY(...)
    |
    |    WITH-OBJECT-CODE(...)
    |      |
    |      |  PRINT-STATEMENTS = ACCEPTED / IGNORED
    |      |  LINE-NUMBERING = NO / YES
    |
    |    SOURCE-ONLY(...)
    |      |
    |      |  PRINT-STATEMENTS = ACCEPTED / IGNORED
    |      |  LINE-NUMBERING = NO / YES
    |
    |    ERRORS-ONLY(...)
    |      |
    |      |  LINE-NUMBERING = NO / YES
    |
    |  ,SOURCE-FORMAT = STD / STRUCTURED(...)
    |
    |    STRUCTURED(...)
    |      |
    |      |  EVALUATED-NEST-LEVEL = 1 / ALL
    |      | ,INDENTATION-AMOUNT = 2 / <integer 1..8>
    |      | ,FIXED-AREA-START = NONE / <integer 60..255>
    |      | ,STRUCT-MACRO-PRINT = STD / OBJECT-CODE-ONLY /
    |      |                       WITH-OBJECT-CODE / NO-OBJECT-CODE
    |
    |  ,MACRO-PRINT = STD / PARAMETERS(...)
    |
    |    PARAMETERS(...)
    |      |
    |      |  NOPRINT-NEST-LEVEL = 255 / <integer 1..255>
    |      | ,NOPRINT-PREFIX = *NONE / list-poss(256): <name 1..64>
    |      | ,TITLE-STATEMENTS = ACCEPTED / IGNORED
    |      | ,MACRO-ORIGIN-INFO = SEPARATE / INSERTED
```

```
,MIN-MESSAGE-WEIGHT = NOTE / WARNING / SIGNIFICANT / SERIOUS / FATAL

,CROSS-REFERENCE = STD / ALL / NO / PARAMETERS(...)

   PARAMETERS(...)
      SYMBOL = NO / YES(...)

        YES(...)
           WITH-ATTRIBUTES = NO / YES
          ,REFERENCED-ONLY = NO / YES
          ,PREFIX = ALL / EXCEPT(...) / ONLY(...)

             EXCEPT(...)
               CHARACTERS = list-poss(256): <name 1..64>

             ONLY(...)
               CHARACTERS = list-poss(256): <name 1..64>

      ,LITERAL = NO / YES

      ,MACRO = NO / YES

      ,COPY = NO / YES

      ,DIAGNOSTICS = NO / YES

,EXTERNAL-DICTIONARY = NO / YES

,LAYOUT = STD / PARAMETERS(...)

   PARAMETERS(...)
      LINES-PER-PAGE = 60 / <integer 15..255>
     ,LASER-PRINTER = NO / ND2
     ,FORMAT = STD / F-ASSEMB-COMPATIBLE(...)

        F-ASSEMB-COMPATIBLE(...)
          MESSAGE-PLACEMENT = SEPARATE / INSERTED

,OUTPUT = *SYSLST / *NONE / <full-filename 1..54> /
          *LIBRARY-ELEMENT(...) / *SAVLST

   *LIBRARY-ELEMENT(...)
      LIBRARY = <full-filename 1..54 without-gen-vers>

      ,ELEMENT = <composed-name 1..64>(...)
          VERSION = *UPPER-LIMIT / *INCREMENT / *HIGHEST-EXISTING /
                    <composed-name 1..24>
```

LISTING = STD
Es werden die Standardwerte der Struktur PARAMETERS(...) eingesetzt.

LISTING = PARAMETERS(...)

SOURCE-PRINT =
Steuert die Protokollierung des Quellprogramms

SOURCE-PRINT = NO
Kein Protokoll des Quellprogramms

SOURCE-PRINT = WITH-OBJECT-CODE(...)
Protokoll der Source-Zeilen mit Objekt-Code

PRINT-STATEMENTS = ACCEPTED / IGNORED
Die Angaben NOGEN, OFF, NOCOPY der PRINT-Anweisung werden ausgeführt oder ignoriert.

LINE-NUMBERING = NO / YES
Angabe, ob im Übersetzungsprotokoll die Zeilen aus der Quelle im Identifikationsfeld (Spalten 73-80) durchnumeriert werden sollen.

SOURCE-PRINT = SOURCE-ONLY(...)
Nur Protokoll der Source-Zeilen ohne Objekt-Code

PRINT-STATEMENTS = ACCEPTED / IGNORED
Die Angaben NOGEN, OFF, NOCOPY der PRINT-Anweisung werden ausgeführt oder ignoriert.

LINE-NUMBERING = NO / YES
Angabe, ob im Übersetzungsprotokoll die Zeilen aus der Quelle im Identifikationsfeld (Spalten 73-80) durchnumeriert werden sollen.
Die Numerierung beginnt mit 100 in der Schrittweite 100 auf 8 Stellen.
Bei Source-Deck-Makros wird keine Numerierung durchgeführt.

SOURCE-PRINT = ERRORS-ONLY(...)
Nur Protokoll der fehlerhaften Source-Zeilen

LINE-NUMBERING = NO / YES
Angabe, ob im Übersetzungsprotokoll die Zeilen aus der Quelle im Identifikationsfeld (Spalten 73-80) durchnumeriert werden sollen.

SOURCE-FORMAT = STD
Es werden die Standardwerte der Struktur STRUCTURED(...) eingesetzt.

SOURCE-FORMAT = STRUCTURED(...)
Es wird ein strukturiertes Listing erzeugt, vorausgesetzt im Quellprogramm wurden die vordefinierten Makros der Strukturierten Programmierung (Strukturmakros, "@-Makros") eingesetzt.

EVALUATED-NEST-LEVEL = 1 / ALL
Entweder werden nur die in der Quelle auftretenden Aufrufe von Strukturmakros oder alle (auch die die durch Generierung aufgerufen wurden) protokolliert.

INDENTATION-AMOUNT = 2 / <integer 1...8>
Angabe des Einrückbetrages in Zeichenstellen (und damit auch der Abstand der senkrechten Strukturlinien).

FIXED-AREA-START = NONE / <integer 60...255>
Angabe, ab welcher Spalte das Quellprogramm durch die Strukturierung nicht verändert bzw. verschoben werden soll.

STRUCT-MACRO-PRINT =
Steuert die Protokollierung der Strukturmakros.

STRUCT-MACRO-PRINT = STD
Strukturmakros werden wie andere Makros protokolliert.

STRUCT-MACRO-PRINT = OBJECT-CODE-ONLY
Für alle Strukturmakros wird nur der generierte Objekt-Code ausgegeben. Die Wirkung entspricht der Angabe PRINT NOGEN,CODE. Die Option NOPRINT-PREFIX wird ignoriert.

STRUCT-MACRO-PRINT = WITH-OBJECT-CODE
Es wird der Objekt-Code mit der zugehörigen generierten Sourcedarstellung protokolliert. Bei Makros, die mit der Option NOPRINT-PREFIX oder durch eine PRINT NOGEN Quellanweisung von der Protokollierung ausgeschlossen werden, wird nur der Objekt-Code protokolliert.

STRUCT-MACRO-PRINT = NO-OBJECT-CODE
Für Strukturmakros wird der Objekt-Code nicht protokolliert.

MACRO-PRINT =
Steuert die Protokollierung der Makro-Elemente im Sourcelisting

MACRO-PRINT = STD
Es werden die Standardwerte der Struktur PARAMETERS(...) eingesetzt

MACRO-PRINT = PARAMETERS(...)

NOPRINT-NEST-LEVEL = 255 / <integer 1..255>
Bestimmt die maximale Makroverschachtelung, bis zu der die Generierung protokolliert wird.

NOPRINT-PREFIX = *NONE / <name 1..64>
Nennt eine Liste von Namensanfängen (256) der Makros, die nicht protokolliert werden sollen.
Der Operand PREFIX-EXCEPTION = <name 1..1> wird nur noch aus Kompatibilitätsgründen unterstützt. Wenn NOPRINT-PREFIX gesetzt ist, wird PREFIX-EXCEPTION nicht mehr ausgewertet.

TITLE-STATEMENTS = ACCEPTED / IGNORED
TITLE-Anweisungen, die durch Makros generiert werden, werden ausgeführt oder ignoriert.

MACRO-ORIGIN-INFO = SEPARATE / INSERTED
Bestimmt den Ort der Ablage der Makroidentifikationszeile (Version, Erstellungsdatum, Linkname der Makrobibliothek) im Protokoll; bei SEPARATE wird die Meldung in der Makro-XREF-Liste abgelegt, bei INSERTED erfolgt die Meldung zusätzlich nach dem Makroaufruf.

MIN-MESSAGE-WEIGHT = NOTE / WARNING / SIGNIFICANT / SERIOUS / FATAL
Nennt die Fehlerschwere, ab der Fehler protokolliert werden sollen; nur diese Fehler gehen in die Summenzeile ein.

CROSS-REFERENCE = STD / ALL / NO / PARAMETERS(...)
Steuert die Protokollierung der Cross-Referenzlisten

CROSS-REFERENCE = STD
Es werden die Standardwerte der Struktur PARAMETERS(...) eingesetzt.

CROSS-REFERENCE = ALL
Bedeutet, daß die Cross-Referenzlisten im größtmöglichen Umfang ausgegeben werden. Dabei sind folgende Werte gültig:
SYMBOL=YES (WITH-ATTRIBUTES=YES, REFERENCED-ONLY=NO, PREFIX=ALL), LITERAL=YES, MACRO=YES, COPY=YES, DIAGNOSTICS=YES

CROSS-REFERENCE = PARAMETERS(...)

SYMBOL = NO / YES(...)
Bestimmt die Ausgabe der Referenzliste für Symbole (Symbol-XREF).

WITH-ATTRIBUTES = NO / YES
Bestimmt die Ausgabe der jeweils zugehörigen Attribute, die auf die Zugriffsart hinweisen.
W Schreibender Zugriff
R Lesender Zugriff durch Befehle
A Adreßzugriff
E EQU/ORG-Anweisungen

REFERENCED-ONLY = NO / YES
Bestimmt, ob nur referenzierte Symbole ausgegeben werden sollen.

PREFIX = ALL / EXCEPT(...) / ONLY(...)
Bestimmt oder unterdrückt die Ausgabe von Symbolen mit bestimmtem Präfix.

PREFIX = EXCEPT(CHARACTERS=<name 1..64>)
Bestimmt das Präfix der Symbole, die nicht ausgegeben werden sollen (256).

PREFIX = ONLY(CHARACTERS=<name 1..64>)
Bestimmt das Präfix der Symbole, die ausgegeben werden sollen (256).

LITERAL = NO / YES
Bestimmt die Ausgabe der Referenzliste für Literale.

MACRO = NO / YES
Bestimmt die Ausgabe der Referenzliste für Makros.

COPY = NO / YES
Bestimmt die Ausgabe der Referenzliste für Copy-Elemente.

DIAGNOSTICS = NO / YES
Bestimmt die Ausgabe der Referenzliste für die aufgetretenen Assemblerflags.

EXTERNAL-DICTIONARY = NO / YES
Bestimmt die Protokollierung der Externverweise des übersetzten Moduls (ENTRY, EXTRN, WXTRN usw.).

LAYOUT =
Bestimmt das Format des Protokolls.

LAYOUT = STD
Es werden die Standardwerte der Struktur PARAMETERS(...) eingesetzt.

LAYOUT = PARAMETERS(...)

LINES-PER-PAGE = 60 / <integer 15..255>
Bestimmt die Anzahl der Zeilen je Seite des Protokolls.

LASER-PRINTER = NO / ND2
Bestimmt, ob ein für den Laser-Drucker spezifisches Protokoll ausgegeben werden soll.

FORMAT = STD
Es wird ein Listing im ASSEMBH-Standardformat erzeugt.

FORMAT = F-ASSEMB-COMPATIBLE(...)
Es wird ein F-Assembler (ASSEMB V30.0A) kompatibles Listing erzeugt.

MESSAGE-PLACEMENT = SEPARATE / INSERTED
Bestimmt den Ort der Ablage von Fehlermeldungen im Protokoll; bei SEPERATE erfolgt eine Flagkennung in der Quellzeile und ein Eintrag im Diagnostic-XREF-Listing, bei INSERTED erfolgt die Fehlermeldung zusätzlich nach der fehlerhaften Quellzeile.

OUTPUT =
Nennt das Ausgabemedium für das Assembler-Protokoll.

Falls Sie über das Diagnoseprogramm ASSDIAG (siehe Kapitel 8) eine Übersetzung starten und das Protokoll dazu wünschen, erhalten Sie dieses nur, wenn Sie den ASSDIAG mit END L beenden.
(Bei Angabe von END, ohne L, erhalten Sie kein Protokoll).

OUTPUT = *SYSLST
Das Assembler-Protokoll wird in die Systemdatei SYSLST ausgegeben.

OUTPUT = *NONE
Das Assembler-Protokoll wird nicht ausgegeben.

OUTPUT = <full-filename 1..54>
Das Assembler-Protokoll wird in eine katalogisierte Datei ausgegeben.

OUTPUT = *LIBRARY-ELEMENT(...)

LIBRARY = <full-filename 1..54>)
Nennt den Bibliotheksnamen für die Ausgabe des Assembler-Protokolls.

ELEMENT = <composed-name 1..64>(...)
Name des Elementes vom Typ P

VERSION = *UPPER-LIMIT
Das Element erhält die höchstmögliche Version.

VERSION = *INCREMENT
Das Element erhält die erhöhte Version.

VERSION = *HIGHEST-EXISTING
Das Element erhält die höchste existierende Version.

VERSION = <composed-name 1..24>
Versionsbezeichnung des Elements.

OUTPUT = *SAVLST
Das Assembler-Protokoll wird mit ISAM-Schlüssel ausgegeben (siehe COMOPT SAVLST).

Hinweise

Zur Operanden-Eingabe

– Die Angaben 'LISTING' und 'PARAMETERS( )' können weggelassen werden.

Beispiel

Anstelle von

```
//C LISTING=PARAMETERS(SOURCE-PRINT=ERRORS-ONLY)
```
kann

```
//C SOURCE-PRINT=ERRORS-ONLY
```
oder

```
//C S-PRI=ERR-O
```
geschrieben werden.

– Bei Eingaben zu 'MACRO-PRINT' ('NOPRINT-NEST-LEVEL' usw.) können die Angaben 'MACRO-PRINT' und 'PARAMETERS( )' weggelassen werden.

Beispiel

Anstelle von

```
//C MACRO-PRINT=PARAMETERS(NOPRINT-NEST-LEVEL=20)
```
kann

```
//C NOPRINT-NEST-LEVEL=20
```
geschrieben werden

– Bei Eingaben zu ’CROSS-REFERENCE’ können die Angaben ’PARAMETERS( )’, ’SYMBOL’ und ’YES( )’ weggelassen werden.

  Beispiel

  Anstelle von

  ```
  //C CROSS-REFERENCE=PARAMETERS(SYMBOL=YES(WITH-ATTRIBUTES=NO))
  ```
  kann

  ```
  //C CROSS-REF=(WITH-ATTR=NO)
  ```
  geschrieben werden

– Bei Eingaben zu ’LAYOUT’ kann die Angabe ’PARAMETERS( )’ weggelassen werden.

  Beispiel

  Anstelle von

  ```
  //C LAYOUT=PARAMETERS(LASER-PRINTER=ND2)
  ```
  kann

  ```
  //C LAYOUT=(LASER-PRINTER=ND2)
  ```
  geschrieben werden

– Bei Eingaben zu ’OUTPUT’ kann die Angabe ’*LIBRARY-ELEMENT( )’ weggelassen werden.

  Beispiel

  Anstelle von

  ```
  //C SOURCE=dateiname,OUTPUT=*LIB-ELEM(LIB=bibl)
  ```
  kann

  ```
  //C SOURCE=dateiname,OUTPUT=(bibl)
  ```
  geschrieben werden.

– Zu VERSION = *INCREMENT (erhöhte Version)
  Siehe Versionsbezeichnung *INCREMENT und automatische Versionserhöhung im Benutzerhandbuch LMS [8].

2.4.5 Option zur Testunterstützung TEST-SUPPORT-Option

Wird vom ASSEMBH-BC nicht unterstützt !

Funktion

Mit TEST-SUPPORT wird gesteuert, ob LSD-Informationen erzeugt und im Objektmodul abgelegt werden.
Die LSD-Information im Objektmodul ist Voraussetzung für das symbolische Testen mit AID (siehe Kapitel 9, Dialogtesthilfe AID; sowie das Handbuch "AID, Testen von ASSEMBH-Programmen" [2]).

Format

`COMPILE`
`TEST-SUPPORT = NO / NONE / AID`

TEST-SUPPORT = NO / NONE
Keine Unterstützung für Symbolisches Testen mit AID.

TEST-SUPPORT = AID
Unterstützung für Symbolisches Testen mit AID.
Der ASSEMBH legt im Objektmodul nach dem ersten Programmabschnitt eine 8 Byte lange Konsistenz-Konstante ab. Damit stellt AID die Konsistenz zwischen Objektmodul und LSD-Information sicher.

2.4.6 Option zum Übersetzungsabbruch COMPILER-TERMINATION-Option

Funktion

Mit COMPILER-TERMINATION können die vom Assembler zu interpretierenden Abbruchbedingungen und Schachtelungstiefen bestimmt werden.

Format

```
COMPILE

COMPILER-TERMINATION = STD / PARAMETERS(...)

    PARAMETERS(...)
        MAX-ERROR-WEIGHT = WARNING / SIGNIFICANT / SERIOUS / FATAL
        MAX-ERROR-NUMBER = 32767 / <integer 0..32767>
        MAX-MACRO-NEST-LEVEL = 255 / <integer 1..255>
        MAX-COPY-NEST-LEVEL = 5 / <integer 1..255>
```

COMPILER-TERMINATION = STD
Es werden die Standardwerte der Struktur PARAMETERS(...) eingesetzt.

COMPILER-TERMINATION = PARAMETERS(...)

MAX-ERROR-WEIGHT = WARNING / SIGNIFICANT / SERIOUS / FATAL
Abbruchkriterium Fehlergewicht; nennt die Fehlerklasse, bei der die Übersetzung abgebrochen werden soll.

MAX-ERROR-NUMBER = 32767 / <integer 0..32767>
Abbruchkriterium Fehleranzahl; nennt die Fehleranzahl, bei der die Übersetzung abgebrochen werden soll, sobald die Fehleranzahl überschritten wird.

MAX-MACRO-NEST-LEVEL = 255 / <integer 1..255>
Nennt die maximale Schachtelungstiefe von Makro-Elementen.

MAX-COPY-NEST-LEVEL = 5 / <integer 1..255>
Nennt die maximale Schachtelungstiefe von COPY-Elementen.

Hinweise

– Zur Operanden-Eingabe
  Die Angaben 'COMPILER-TERMINATION' und 'PARAMETERS( )' können weggelassen werden.

  Beispiel

  Anstelle von

  `//C COMPILER-TERMINATION=PARAMETERS(MAX-ERROR-NUMBER=10)` kann

  `//C MAX-ERROR-NUMBER=10` geschrieben werden

– Falls die maximale Schachtelungstiefe von Makro-Elementen (MAX-MACRO-NEST-LEVEL) und COPY-Elementen (MAX-COPY-NEST-LEVEL) überschritten wird, gilt:

  Bei Makro-Elementen: der Makroaufruf wird ignoriert
  Bei COPY-Elementen: der COPY-Aufruf wird ignoriert
  Bei COPY in Makrodefinitionen gilt der COPY-Level zum Zeitpunkt des Einlesens der Makrodefinition.

2.4.7 Option zum Aktivieren des Korrekturzyklus CORRECTION-CYCLE-Option

Wird vom ASSEMBH-BC nicht unterstützt !

Funktion

Mit CORRECTION-CYCLE kann angegeben werden, ob und unter welchen Bedingungen das Diagnoseprogramm ASSDIAG (siehe Kapitel 8) zur Diagnose der Übersetzung und zum Korrigieren der Quelle im Dialog aufgerufen wird.

Format

```
COMPILE

CORRECTION-CYCLE = NO / YES(...)

  YES(...)
    ACTIVATION-WEIGHT = ALWAYS / NOTE / WARNING / SIGNIFICANT / SERIOUS
```

CORRECTION-CYCLE = NO

CORRECTION-CYCLE = YES(...)

ACTIVATION-WEIGHT =
Fehlergewicht, bei dem der ASSDIAG aufgerufen wird.

ACTIVATION-WEIGHT = ALWAYS
Der ASSDIAG wird, unabhängig vom Übersetzungsergebnis, am Ende einer Übersetzungseinheit aufgerufen.

ACTIVATION-WEIGHT = NOTE / WARNING / SIGNIFICANT / SERIOUS
Der ASSDIAG wird beim Erreichen des entsprechenden Fehlergewichts am Ende einer Übersetzungseinheit aufgerufen.

Hinweis

Mit Hilfe des ASSDIAG lassen sich Quellzeilen im Quelltext korrigieren und eine erneute Übersetzung starten. Ein Übersetzungsprotokoll dazu wird nur ausgegeben, wenn Sie den ASSDIAG mit END L beenden. (Bei END, ohne L, wird kein Protokoll ausgegeben). Der Zyklus wird solange durchlaufen, bis das eingestellte Fehlergewicht nicht mehr erreicht wird (d.h. die Korrektur war erfolgreich und die Übersetzung läuft fehlerfrei ab) oder der Benutzer den Zyklus im ASSDIAG beendet (siehe Kapitel 8).

2.4.8 Option zur Maintenance-Unterstützung MAINTENANCE-OPTIONS-Option

Funktion

Mit MAINTENANCE-OPTIONS können Tests für CCW-Kanalbefehle ausgeführt werden.

Format

```
COMPILE

MAINTENANCE-OPTIONS = STD / PARAMETERS(...)

    PARAMETERS(...)
        |
        |  CHANNEL-INSTRUCTIONS = NO / YES
```

MAINTENANCE-OPTIONS = STD
Es werden die Standardwerte der Struktur PARAMETERS(...) eingesetzt.

MAINTENANCE-OPTIONS = PARAMETERS(...)

CHANNEL-INSTRUCTIONS = NO / YES
Unterstützung der Tests für die CCW-Kanalbefehle.

Hinweis

Diese Option wird nur im "Expert-Modus" ausgeführt, d.h. sie ist im geführten Dialogbetrieb ("Menü-Modus") nicht möglich.

2.4.9 Option zur Verringerung des virtuellen Adreßraumbedarfs COMPILATION-SPACE-Option

Funktion

Mit COMPILATION-SPACE kann die Übersetzung und die Listenerzeugung unter Performanceverlusten in einem kleineren virtuellen Adreßraum ausgeführt werden.

Format

`COMPILE`
`COMPILATION-SPACE = STD / SMALL`

COMPILATION-SPACE = STD
Die Übersetzung und die Listenerzeugung findet im virtuellen XS-Adreßraum statt.

COMPILATION-SPACE = SMALL
Die Übersetzung und die Listenerzeugung findet unter Performanceverlusten in einem verringerten virtuellen Adreßraum statt.

Hinweis

Falls der Benutzer auf einer 25-Bit-Anlage ein sehr großes Listing erstellen möchte, muß er einen CIF anlegen (durch Angabe der SDF-Option COMPILATION-INFO, siehe 2.4.3). Ansonsten besteht die Gefahr, daß es aufgrund der in den virtuellen Speicher abgelegten CIF-Information zu Speicherengpässen kommt, und die Übersetzung abgebrochen wird.

2.5 Der Stand-Alone-Listengenerator ASSLG

Wird vom ASSEMBH-BC nicht unterstützt !

Der Stand-Alone-Listengenerator wird mit folgendem Kommando gestartet:

```
/START-PROGRAM $ASSLG
```

2.5.1 GENERATE-Anweisung

Funktion

Der Stand-Alone-Listengenerator ASSLG erstellt aus der in einer Bibliothek abgespeicherten CIF-Information (siehe COMPILATION-INFO, 2.4.3) über die GENERATE-Anweisung die Listen.

Format

```
GENERATE

 COMPILER-INFO-FILE = *LIBRARY-ELEMENT(...)

    *LIBRARY-ELEMENT(...)
       |  LIBRARY = <full-filename 1..54 without gen-vers>
       | ,ELEMENT = <composed-name 1..64>(...)
       |    |  VERSION = *HIGHEST-EXISTING / *UPPER-LIMIT /
       |    |            <composed-name 1..24>

,SOURCE-PRINT = NO / WITH-OBJECT-CODE(...) / SOURCE-ONLY(...) /
                ERRORS-ONLY(...)

     WITH-OBJECT-CODE(...)
       |  LINE-NUMBERING = NO / YES

     SOURCE-ONLY(...)
       |  LINE-NUMBERING = NO / YES

     ERRORS-ONLY(...)
       |  LINE-NUMBERING = NO / YES

,SOURCE-FORMAT = STD / STRUCTURED(...)

     STRUCTURED(...)
       |  EVALUATED-NEST-LEVEL = 1 / ALL
       | ,INDENTATION-AMOUNT = 2 / <integer 1..8>
       | ,FIXED-AREA-START = NONE / <integer 60..255>
       | ,STRUCT-MACRO-PRINT = STD / OBJECT-CODE-ONLY /
       |                       WITH-OBJECT-CODE / NO-OBJECT-CODE

,MACRO-PRINT = STD / PARAMETERS(...)

     PARAMETERS(...)
       |  MACRO-ORIGIN-INFO = SEPARATE / INSERTED

,MIN-MESSAGE-WEIGHT = NOTE / WARNING / SIGNIFICANT / SERIOUS / FATAL

,CROSS-REFERENCE = STD / ALL / NO / PARAMETERS(...)

     PARAMETERS(...)
       |  SYMBOL = NO / YES(...)
```

```
        YES(...)
          WITH-ATTRIBUTES = NO / YES
         ,REFERENCED-ONLY = NO / YES
         ,PREFIX = ALL / EXCEPT(...) / ONLY(...)
            EXCEPT(...)
              CHARACTERS = list-poss(256): <name 1..64>
            ONLY(...)
              CHARACTERS = list-poss(256): <name 1..64>
       ,LITERAL = NO / YES
       ,MACRO = NO / YES
       ,COPY = NO / YES
       ,DIAGNOSTICS = NO / YES
,EXTERNAL-DICTIONARY = NO / YES
,LAYOUT = STD / PARAMETERS(...)
     PARAMETERS(...)
       LINES-PER-PAGE = 60 / <integer 15..255>
      ,LASER-PRINTER = NO / ND2
      ,FORMAT = STD / F-ASSEMB-COMPATIBLE(...)
        F-ASSEMB-COMPATIBLE(...)
          MESSAGE-PLACEMENT = SEPARATE / INSERTED
,OUTPUT = *SYSLST / *SAVLST / <full-filename 1..54 without gen-vers> /
          *LIBRARY-ELEMENT(...)
     *LIBRARY-ELEMENT(...)
       LIBRARY = <full-filename 1..54 without gen-vers>
      ,ELEMENT = <composed-name 1..64>(...)
         VERSION = *UPPER-LIMIT / *INCREMENT / *HIGHEST-EXISTING /
                   <composed-name 1..24>
,GENERATION-SPACE = STD / SMALL
```

COMPILER-INFO-FILE = *LIBRARY-ELEMENT(...)

LIBRARY = <full-filename 1..54>
Name der Bibliothek, in der die CIF-Informationen abgelegt sind (siehe COMPILATION-INFO-Option).

ELEMENT = <composed-name 1..64>(...)
Name des Bibliothekselements

VERSION = *HIGHEST-EXISTING
Das Element erhält die höchste existierende Version.

VERSION = *UPPER-LIMIT
Das Element erhält die höchstmögliche Version.

VERSION = <composed-name 1..24>
Versionsbezeichnung des Elements.

SOURCE-PRINT =
Steuert die Protokollierung des Quellprogramms

SOURCE-PRINT = NO
Kein Protokoll des Quellprogramms

SOURCE-PRINT = WITH-OBJECT-CODE(...)
Protokoll der Source-Zeilen mit Objekt-Code

LINE-NUMBERING = NO / YES
Angabe, ob im Übersetzungsprotokoll die Zeilen aus der Quelle im Identifikationsfeld (Spalten 73-80) durchnumeriert werden sollen.

SOURCE-PRINT = SOURCE-ONLY(...)
Nur Protokoll der Source-Zeilen ohne Objekt-Code

LINE-NUMBERING = NO / YES
Angabe, ob im Übersetzungsprotokoll die Zeilen aus der Quelle im Identifikationsfeld (Spalten 73-80) durchnumeriert werden sollen.

SOURCE-PRINT = ERRORS-ONLY(...)
Nur Protokoll der fehlerhaften Source-Zeilen

LINE-NUMBERING = NO / YES
Angabe, ob im Übersetzungsprotokoll die Zeilen aus der Quelle im Identifikationsfeld (Spalten 73-80) durchnumeriert werden sollen.

SOURCE-FORMAT = STD
Es werden die Standardwerte der Struktur STRUCTURED(...) eingesetzt.

SOURCE-FORMAT = STRUCTURED(...)
Es wird ein strukturiertes Listing erzeugt, vorausgesetzt im Quellprogramm wurden die vordefinierten Makros der Strukturierten Programmierung (Strukturmakros, "@-Makros") eingesetzt.

EVALUATED-NEST-LEVEL = 1 / ALL
Entweder werden nur die in der Quelle auftretenden Aufrufe von Strukturmakros oder alle (auch die die durch Generierung aufgerufen wurden) protokolliert.

INDENTATION-AMOUNT = 2 / <integer 1...8>
Angabe des Einrückbetrages in Zeichenstellen (und damit auch der Abstand der senkrechten Strukturlinien).

FIXED-AREA-START = NONE / <integer 60...255>
Angabe, ab welcher Spalte das Quellprogramm durch die Strukturierung nicht verändert bzw. verschoben werden soll.

STRUCT-MACRO-PRINT =
Steuert die Protokollierung der Strukturmakros.

STRUCT-MACRO-PRINT = STD
Strukturmakros werden wie andere Makros protokolliert.

STRUCT-MACRO-PRINT = OBJECT-CODE-ONLY
Für alle Strukturmakros wird nur der generierte Objekt-Code ausgegeben. Die Wirkung entspricht der Angabe PRINT NOGEN,CODE. Die Option NOPRINT-PREFIX wird ignoriert.

STRUCT-MACRO-PRINT = WITH-OBJECT-CODE
Es wird der Objekt-Code mit der zugehörigen generierten Sourcedarstellung protokolliert. Bei Makros, die mit der Option NOPRINT-PREFIX oder durch eine PRINT NOGEN Quellanweisung von der Protokollierung ausgeschlossen werden, wird nur der Objekt-Code protokolliert.

STRUCT-MACRO-PRINT = NO-OBJECT-CODE
Für Strukturmakros wird der Objekt-Code nicht protokolliert.

MACRO-PRINT =
Steuert die Protokollierung der Makro-Elemente im Sourcelisting

MACRO-PRINT = STD
Es werden die Standardwerte der Struktur PARAMETERS(...) eingesetzt

MACRO-PRINT = PARAMETERS(...)

MACRO-ORIGIN-INFO = SEPARATE / INSERTED
Bestimmt den Ort der Ablage der Makroidentifikationszeile (Version, Erstellungsdatum, Linkname der Makrobibliothek) im Protokoll; bei SEPARATE wird die Meldung in der Makro-XREF-Liste abgelegt, bei INSERTED erfolgt die Meldung zusätzlich nach dem Makroaufruf.

MIN-MESSAGE-WEIGHT = NOTE / WARNING / SIGNIFICANT / SERIOUS / FATAL
Nennt die Fehlerschwere, ab der Fehler protokolliert werden sollen; nur diese Fehler gehen in die Summenzeile ein.

CROSS-REFERENCE = STD / ALL / NO / PARAMETERS(...)
Steuert die Protokollierung der Cross-Referenzlisten

CROSS-REFERENCE = STD
Es werden die Standardwerte der Struktur PARAMETERS(...) eingesetzt.

CROSS-REFERENCE = ALL
Bedeutet, daß die Cross-Referenzlisten im größtmöglichen Umfang ausgegeben werden. Dabei sind folgende Werte gültig:
SYMBOL=YES (WITH-ATTRIBUTES=YES, REFERENCED-ONLY=NO, PREFIX=ALL), LITERAL=YES, MACRO=YES, COPY=YES, DIAGNOSTICS=YES

CROSS-REFERENCE = PARAMETERS(...)

SYMBOL = NO / YES(...)
Bestimmt die Ausgabe der Referenzliste für Symbole (Symbol-XREF).

WITH-ATTRIBUTES = NO / YES
Bestimmt die Ausgabe der jeweils zugehörigen Attribute, die auf die Zugriffsart hinweisen.

W Schreibender Zugriff
R Lesender Zugriff durch Befehle
A Adreßzugriff
E EQU/ORG-Anweisungen

REFERENCED-ONLY = NO / YES
Bestimmt, ob nur referenzierte Symbole ausgegeben werden sollen.

PREFIX = ALL / EXCEPT(...) / ONLY(...)
Bestimmt oder unterdrückt die Ausgabe von Symbolen mit bestimmtem Präfix.

PREFIX = EXCEPT(CHARACTERS=<name 1..64>)
Bestimmt das Präfix der Symbole, die nicht ausgegeben werden sollen (256).

PREFIX = ONLY(CHARACTERS=<name 1..64>)
Bestimmt das Präfix der Symbole, die ausgegeben werden sollen (256).

LITERAL = NO / YES
Bestimmt die Ausgabe der Referenzliste für Literale.

MACRO = NO / YES
Bestimmt die Ausgabe der Referenzliste für Makros.

COPY = NO / YES
Bestimmt die Ausgabe der Referenzliste für Copy-Elemente.

DIAGNOSTICS = NO / YES
Bestimmt die Ausgabe der Referenzliste für die aufgetretenen Assemblerflags.

EXTERNAL-DICTIONARY = NO / YES
Bestimmt die Protokollierung der Externverweise des übersetzten Moduls (ENTRY, EXTRN, WXTRN usw.).

LAYOUT =
Bestimmt das Format des Protokolls.

LAYOUT = STD
Es werden die Standardwerte der Struktur PARAMETERS(...) eingesetzt.

LAYOUT = PARAMETERS(...)

LINES-PER-PAGE = 60 / <integer 15..255>
Bestimmt die Anzahl der Zeilen je Seite des Protokolls.

LASER-PRINTER = NO / ND2
Bestimmt, ob ein für den Laser-Drucker spezifisches Protokoll ausgegeben werden soll.

FORMAT = STD
Es wird ein Listing im ASSEMBH-Standardformat erzeugt.

FORMAT = F-ASSEMB-COMPATIBLE(...)
Es wird ein F-Assembler (ASSEMB V30.0A) kompatibles Listing erzeugt.

MESSAGE-PLACEMENT = SEPARATE / INSERTED
Bestimmt den Ort der Ablage von Fehlermeldungen im Protokoll; bei SEPERATE erfolgt eine Flagkennung in der Quellzeile und ein Eintrag im Diagnostic-XREF-Listing, bei INSERTED erfolgt die Fehlermeldung zusätzlich nach der fehlerhaften Quellzeile.

OUTPUT =
Nennt das Ausgabemedium für das Assembler-Protokoll.

OUTPUT = *SYSLST
Das Assembler-Protokoll wird in die Systemdatei SYSLST ausgegeben.

OUTPUT = *SAVLST
Das Assembler-Protokoll wird mit ISAM-Schlüssel ausgegeben (siehe COMOPT SAVLST).

OUTPUT = <full-filename 1..54>
Das Assembler-Protokoll wird in eine katalogisierte Datei ausgegeben.

OUTPUT = *LIBRARY-ELEMENT(...)

LIBRARY = <full-filename 1..54>
Nennt den Bibliotheksnamen für die Ausgabe des Assembler-Protokolls.

ELEMENT = <composed-name 1..64>(...)
Name des Elementes vom Typ P

VERSION = *UPPER-LIMIT
Das Element erhält die höchstmögliche Version.

VERSION = *INCREMENT
Das Element erhält die erhöhte Version.

VERSION = *HIGHEST-EXISTING
Das Element erhält die höchste existierende Version.

VERSION = <composed-name 1..24>
Versionsbezeichnung des Elements.

GENERATION-SPACE = STD / SMALL
Bei SMALL werden die Listen unter Performanceverlusten in einem kleineren virtuellen Adreßraum generiert.

3 Ein-/Ausgabe des ASSEMBH

3.1 Eingabequellen des ASSEMBH

Die Eingaben des ASSEMBH bestehen aus dem Quelltext und den Anweisungen zur Benutzersteuerung, den Optionen (siehe Kapitel 2).
Der Quelltext steht in einem Quellprogramm. Teile des Quelltextes können während der Übersetzung über Makro-Elemente generiert und aus COPY-Elementen eingelesen werden. Die Optionen steuern den Ablauf der Übersetzung, die Eingaben und Ausgaben des Assemblers.

- Quellprogramm
  Ein Quellprogramm wird entweder
  - über die Systemdatei SYSDTA eingegeben, d.h.
    direkt von einer Datensichtstation oder
    indem SYSDTA einer Datei oder Bibliothek zugewiesen wird, oder
  - es wird aus einer Datei oder Bibliothek gelesen.

  Bibliothekselemente vom Typ S aus einer PLAM-Bibliothek oder OSM-Quellprogrammbibliothek sind zugelassen.
- Makro-Element
  Ein Makro-Element wird aus einer PLAM-Bibliothek (Element-Typ M) oder aus einer OSM-Makrobibliothek (MLU-Format) gelesen.
- COPY-Element
  Ein COPY-Element wird aus einer PLAM-Bibliothek (Element-Typ S oder M) oder aus einer OSM-Quellprogramm- oder OSM-Makrobibliothek (MLU-Format) gelesen.

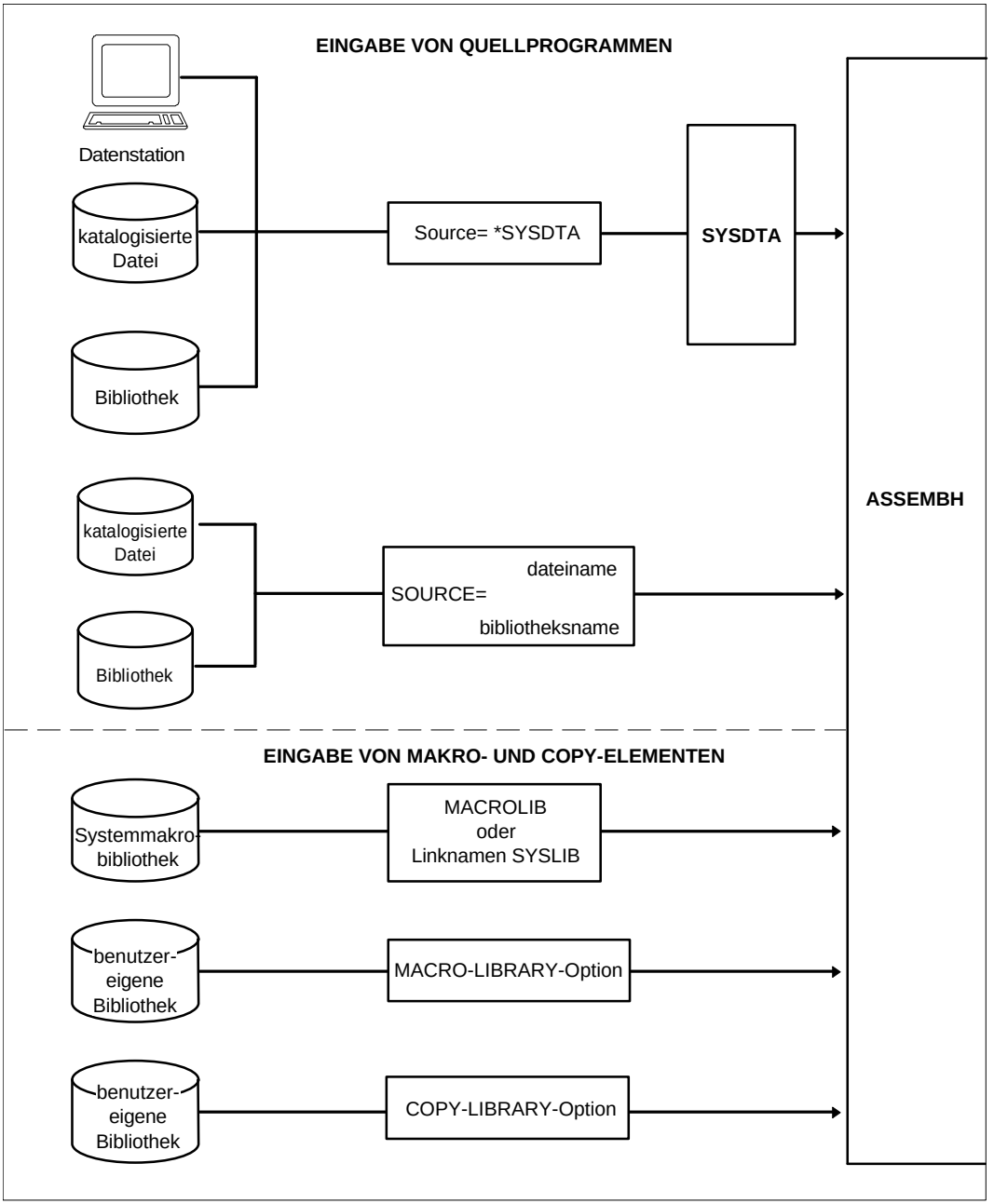


Bild 3-1: Die Eingabequellen des ASSEMBH

3.1.1 Eingabe des Quellprogramms

Der Assembler interpretiert den Inhalt einer Datei als Quelltext. Die Zeile eines Quelltextes darf maximal 255 Zeichen lang sein.
Mit der Option SOURCE-PROPERTIES, LOW-CASE-CONVERSION (siehe 2.4.1.4) können im Quelltext Groß- und Kleinbuchstaben verwendet werden (siehe "ASSEMBH", Beschreibung [1], Kapitel 2.1).
Die Voreinstellung für den zu interpretierenden Quelltext sind die Spalten 1 (Anfangsspalte), 71 (Endspalte), 72 (Fortsetzungszeichen) und 16 (Fortsetzungsspalte).
Mit Hilfe der Option SOURCE-PROPERTIES (siehe 2.4.1.4) kann diese Voreinstellung geändert werden.
Mit der ICTL-Anweisung (siehe "ASSEMBH", Beschreibung [1]) kann ebenfalls die Voreinstellung für die Anfangs-, End- und Fortsetzungsspalte geändert werden.

Eingabe über SYSDTA

Standardmäßig erfolgt die Eingabe des Quelltextes von der Datensichtstation über die Systemdatei SYSDTA. Nach dem Start des ASSEMBH und Eingabe von ’C’, d.h.

//COMPILE SOURCE=*SYSDTA, ...Standardwerte..., meldet sich der ASSEMBH mit ’*’ und verlangt die Eingabe des Quelltextes.

Soll der Quelltext über SYSDTA aus einer Datei oder einem Bibliothekselement gelesen werden, muß vor dem Aufruf des Assemblers SYSDTA einer katalogisierten Datei oder einem Bibliothekselement (Element-Typ S aus einer PLAM-Bibliothek oder aus einer OSM-Quellprogrammbibliothek) zugewiesen werden. Die Zuweisung erfolgt mit dem SDF-Kommando ASSIGN-SYSDTA. In der Datei oder dem Bibliothekselement müssen eine //COMPILE- und eine //END-Anweisung stehen.

Beispiel

```
/ASSIGN-SYSDTA TO-FILE=⎧ dateiname                                ⎫
                       ⎩ *LIB-ELEM(LIB=bibliothek,ELEM=element) ⎭

/START-PROGRAM $ASSEMBH
```

Eingabe aus Dateien

Die Eingabe aus katalogisierten SAM- oder ISAM-Dateien erfolgt durch die SOURCE-Option (siehe 2.4.1.1).

Beispiel

```
//COMPILE SOURCE=dateiname
```

Eingabe aus Bibliotheken

Die Eingabe des Quellprogramms aus Bibliotheken erfolgt durch die SOURCE-Option (siehe 2.4.1.1). Es sind Bibliothekselemente vom Typ S aus PLAM-Bibliotheken und aus OSM-Quellprogrammbibliotheken zugelassen.

Beispiel

```
//COMPILE SOURCE=(bibliothek,element)
```

3.1.2 Eingabe von Makro-Elementen

Makrodefinitionen stehen meistens nicht im Quellprogrammm, sondern werden als Makroelemente (siehe "ASSEMBH", Beschreibung [1]) in Makrobibliotheken abgelegt. Es sind Makroelemente vom Typ M aus PLAM-Bibliotheken und aus OSM-Makrobibliotheken (MLU-Format) zugelassen.
Im Quellprogramm selbst steht nur ein Makroaufruf. Beim Übersetzungslauf wird durch den Makroaufruf parametergesteuert aus der Makrodefinition eine Folge von Instruktionen generiert und in das Quellprogramm übernommen.
Die Spalten des Textes einer Makrodefinition werden standardmaßig interpretiert, d.h. Spalte 1 (Anfangsspalte), 71 (Endspalte), 72 (Fortsetzungszeichen) und Spalte 16 (Fortsetzungsspalte).
Ein Ändern über die SOURCE-PROPERTIES-Option oder die ICTL-Anweisung ist wirkungslos.

Es gibt zwei Arten von Makrobibliotheken:

- die benutzereigenen Makrobibliotheken und
- die System-Makrobibliothek MACROLIB ($TSOS.MACROLIB), die für alle Benutzer erreichbar ist

Benutzereigene Makrobibliotheken

Mit der MACRO-LIBRARY-Option (siehe 2.4.1.2) können bis zu 100 private Benutzer-Makrobibliotheken angegeben werden. Will man die benutzereigenen Makrobibliotheken über Linknamen ansprechen, muß man vor dem Start des Assemblers die entsprechenden SET-FILE-LINK-Kommandos absetzen.

Beispiel

```
/SET-FILE-LINK LINK-NAME=maclink,FILE-NAME=maclib
```

Der Makrobibliothek maclib wird der Link-Name maclink zugewiesen.

```
//C MAC-LIB=(maclib1,maclib2,*LINK(maclink))
```

Es werden die Makrobibliotheken maclib1 und maclib2 zugewiesen, sowie über den Link-Namen maclink die Makrobibliothek maclib.

System-Makrobibliothek

Die System-Makrobibliothek ist mit einem Dateisteuerblock (FCB) spezifiziert, der LINK=SYSLIB enthält. Will ein Benutzer seine eigene Datei als System-Makrobibliothek verwenden, so kann er dies auf zwei verschiedene Arten erreichen:

– der eigenen Datei wird mit dem SET-FILE-LINK Kommando der Dateikettungsnamen SYSLIB zugewiesen.
  Ein entsprechendes REMOVE-FILE-LINK-Kommando muß auch in diesem Falle der Benutzer abgeben.

  Beispiel
  ```
  /SET-FILE-LINK LINK-NAME=SYSLIB,FILE-NAME=dateiname
  ```

– die eigene Datei erhält den Standard-Dateinamen MACROLIB.

  Beispiel
  ```
  /MOD-FILE-ATTR FILE-NAME=dateiname,NEW-NAME=MACROLIB
  ```

3.1.2.1 Suchreihenfolge von Makro-Elementen

Bei einer Übersetzung können die System-Makrobibliothek und bis zu 100 benutzereigene Makrobibliotheken angesprochen werden. Bei der Bearbeitung eines Makroaufrufs wird eine unbekannte Makrodefinition nach folgender Reihenfolge gesucht und eingelesen:

1. benutzereigene Makrobibliothek
.
.
.
100. benutzereigene Makrobibliothek
System-Makrobibliothek MACROLIB

Die Suchreihenfolge für Makroelemente aus benutzereigenen Makrobibliotheken entspricht der Reihenfolge der in der MACRO-LIBRARY-Option angegebenen Bibliotheken.

Existieren in einer PLAM-Bibliothek mehrere Elemente vom Typ M mit gleichen Namen, aber verschiedenen Versionen, so wird stets das Element mit der höchsten Version verwendet.

Innere Makros von Makros aus der System-Makrobibliothek werden nur in dieser gesucht, falls die entsprechenden Makrodefinitionen nicht schon vorher eingelesen worden sind.

3.1.3 Eingabe von COPY-Elementen

Im Quellprogramm selbst steht nur die COPY-Anweisung. Beim Übersetzungslauf wird durch die Anweisung die abgespeicherte Folge von Instruktionen aus der Bibliothek gelesen und in das Quellprogramm übernommen.

Mit der COPY-Anweisung (siehe "ASSEMBH", Beschreibung [1]) können COPY-Elemente aus einer PLAM-Bibliothek (Element-Typ S oder M) oder aus einer OSM-Quellprogramm- oder OSM-Makrobibliothek (MLU-Format) in ein Quellprogramm kopiert werden.

Mit der COPY-LIBRARY-Option (siehe 2.4.1.3) können bis zu 100 benutzereigene Bibliotheken angegeben werden. Es sind PLAM-Bibliotheken (Element-Typ S und M), sowie OSM-Quellprogramm- und OSM-Makrobibliotheken (MLU-Format) zugelassen.
Mit dem Operanden ELEMENT-TYPE = SOURCE-ONLY oder MACRO-ONLY kann bestimmt werden, daß das Element nur in der Source-Abteilung oder nur in der Makro-Abteilung einer PLAM-Bibliothek gesucht wird.

Beispiele

```
/SET-FILE-LINK LINK-NAME=coplink,FILE-NAME=coplib
```

Der Bibliothek coplib wird der Link-Name coplink zugewiesen.

```
//C COPY-LIB=(coplib1(ELEMENT-TYPE=MACRO-ONLY),coplib2,*LINK(coplink))
```

Die Bibliotheken coplib1 und coplib2 werden zugewiesen, sowie über den Link-Namen coplink die Bibliothek coplib.

```
//C C-L=coplib3(S-O)
```

Die Bibliothek coplib3 (Element-Typ S) wird zugewiesen.

3.1.3.1 Suchreihenfolge von COPY-Elementen

Die Suchreihenfolge von COPY-Elementen entspricht der Reihenfolge der in der COPY-LIBRARY-Option angegebenen Bibliotheken (siehe 2.4.2).

Beispiel

```
//C COPY-LIB=(coplib2,coplib1,coplib3)
```

Die COPY-Elemente werden zuerst in der Bibliothek coplib2, dann in coplib1 und zuletzt in coplib3 gesucht.

Zu jeder Bibliothek wird die Typangabe ausgewertet. Bei der Typangabe BOTH wird zuerst in der S-Abteilung und dann in der M-Abteilung der Bibliothek gesucht.

Existieren in der Source- bzw. Makro-Abteilung einer PLAM-Bibliothek mehrere Elemente mit gleichen Namen, aber mit verschiedenen Versionen, so wird stets das Element mit der höchsten Version verwendet.

3.2 Ausgaben des ASSEMBH

Der ASSEMBH erzeugt, wählbar durch Optionen, folgende Ausgaben:

- Objektmodule
  Objektmodule (Object Modules, OMs) werden in eine PLAM-Bibliothek (Element-Typ R) oder in die *EAM-Datei (OMF) ausgegeben.
- Bindelademodule
  Bindelademodule (Load and Link Modules, LLMs) werden in eine PLAM-Bibliothek (Element-Typ L) ausgegeben.
- Meldungen
  Die Start- und Endemeldungen des ASSEMBH werden nach SYSOUT ausgegeben (zur Datensichtstation oder in die SYSOUT-Datei).
- Listen
  Die Listen werden durch die Option LISTING, OUTPUT (siehe 2.4.4) nach SYSLST oder in eine Datei oder in eine PLAM-Bibliothek (Element-Typ P) ausgegeben. Die Listen sind in Kapitel 6 beschrieben.
- Compiler Information File (CIF)
  Zur Erzeugung des Assemblerprotokolls wird CIF standardmäßig temporär erstellt. Mit der Option COMPILATION-INFO (siehe 2.4.3) kann die CIF-Information in eine PLAM-Bibliothek (Element-Typ H) ausgegeben werden.
- Monitorjobvariable (MONJV)
  Am Ende einer Übersetzung wird die überwachende Monitorjobvariable, falls sie vom Benutzer vereinbart wurde, vom Assembler mit einer Zustands- und einer Rückkehrcode-Anzeige versorgt (siehe 3.2.3).

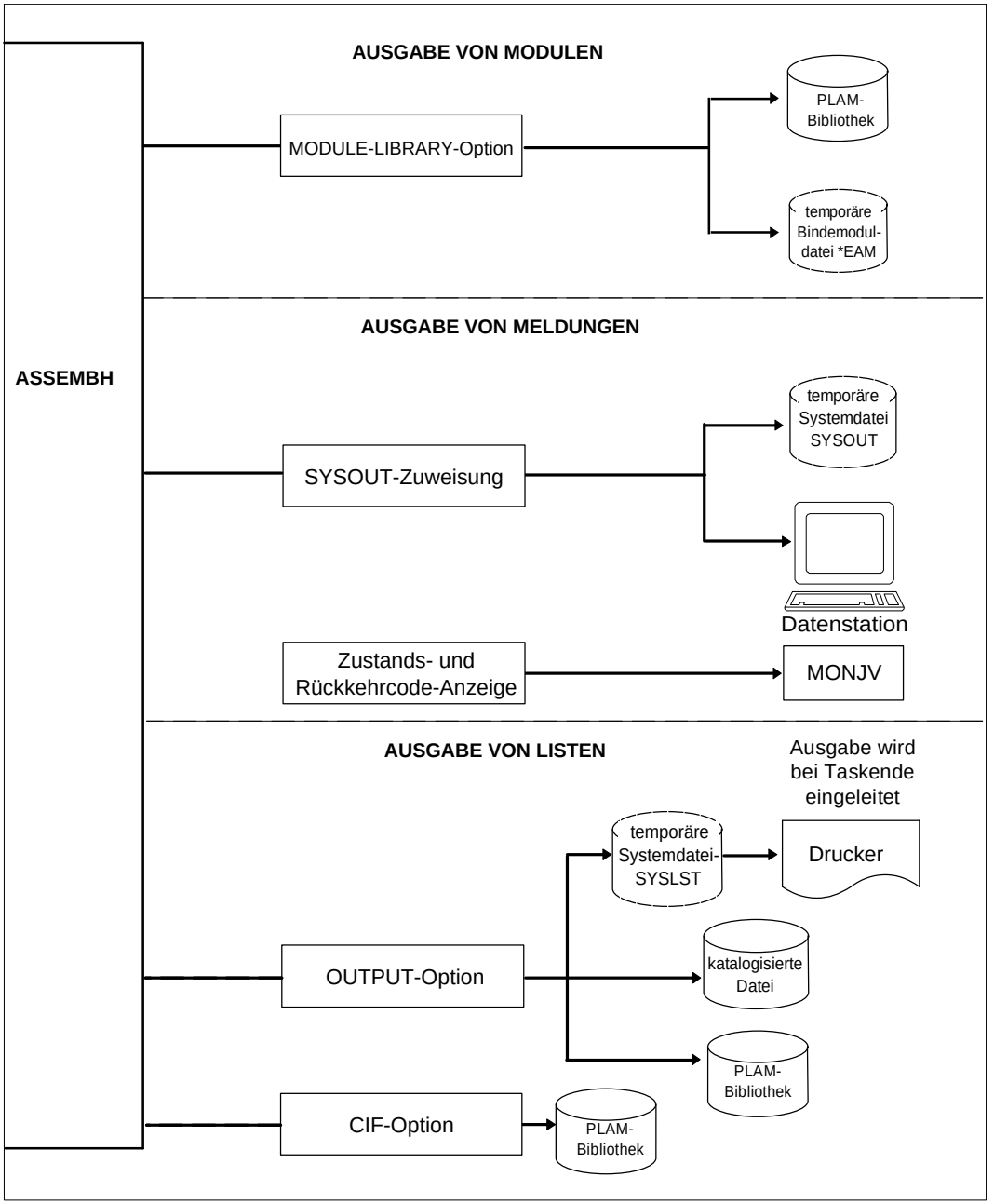


Bild 3-2: Die Ausgaben des ASSEMBH

3.2.1 Ausgabe eines Objektmoduls

Mit der COMPILER-ACTION-Option (siehe 2.4.2.1) wird standardmäsig ein Objektmodul erzeugt und mit der MODULE-LIBRARY-Option (siehe 2.4.2.2) kann der Ausgabeort bestimmt werden.

Beispiel

```
//C MOD-LIB=plambibl
```

Der Objektmodul mit dem Namen der ersten benannten CSECT wird in der PLAM-Bibliothek plambibl abgelegt.

Der Assembler übersetzt ein Quellprogramm in Maschinensprache. Dieses direkte Ergebnis einer Übersetzung ist ein Objektmodul (OM) oder ein Bindelademodul (LLM). Dieser Objektmodul besteht zwar bereits aus Maschinencode, kann aber erst ablaufen, nachdem er gebunden und geladen wurde (siehe Binden, Laden und Starten; Kapitel 5).

Die Assemblerbefehle und Anweisungen werden entsprechend dem gewählten Befehlssatz in Maschinenbefehle umgesetzt und als TXT-Einträge in den Objektmodul ausgegeben. Über eine Option besteht die Möglichkeit, unterschiedliche Befehlssätze auszuwählen (siehe 2.4.1.4, Option SOURCE-PROPERTIES, INSTRUCTION-SET).

Die zugehörige Binder- und Laderinformation wird in ESD- und RLD-Einträgen abgelegt.

Ein Objektmodul besteht standardmäßig aus den ESD-, TXT-, RLD- und END-Sätzen. Zusätzlich können auf Wunsch über die TEST-SUPPORT-Option (siehe 2.4.5) für das Symbolische Testen mit AID LSD-Einträge erzeugt werden (diese Funktion wird vom ASSEMBH-BC nicht unterstützt).

Einträge in den Objektmodul

ESD	Binder-Laderinformation (Definition und Referenz von externen Namen)
LSD	Testhilfeinformation für AID
TXT	Befehle und Anweisungen im Maschinencode
RLD	Binder-Laderinformation (Relativierung von Adressen)
END	Endinformation des Objektmoduls

ESD = External Symbol Dictionary
LSD = List for Symbolic Debugging
TXT = Textinformation
RLD = Relocation Directory

3.2.2 Ausgabe eines Bindelademoduls

Mit der COMPILER-ACTION-Option (siehe 2.4.2.1) kann ein Bindelademodul (LLM) erzeugt werden und mit der MODULE-LIBRARY-Option (siehe 2.4.2.2) muß die Bibliothek angegeben werden, in die der Modul ausgegeben wird.

Beispiel

```
//C COMP-ACT=(MODULE-FORMAT=LLM),MOD-LIB=plambibl
```

Der Bindelademodul mit dem Namen der ersten benannten CSECT wird in der PLAM-Bibliothek plambibl abgelegt.

Der Assembler übersetzt ein Quellprogramm in Maschinensprache. Dieses direkte Ergebnis einer Übersetzung ist ein Objektmodul oder ein Bindelademodul. Dieser Bindelademodul besteht zwar bereits aus Maschinencode, kann aber erst ablaufen, nachdem er gebunden und geladen wurde (siehe Binden, Laden und Starten; Kapitel 5).

Die Assemblerbefehle und Anweisungen werden entsprechend dem gewählten Befehlssatz in Maschinenbefehle umgesetzt und als TXT-Einträge in den Bindelademodul ausgegeben. Über eine Option besteht die Möglichkeit, unterschiedliche Befehlssätze auszuwählen (siehe 2.4.1.4, Option SOURCE-PROPERTIES, INSTRUCTION-SET).

Die zugehörige Binder- und Laderinformation wird in ESV- und LRLD-Einträgen abgelegt.

Ein Bindelademodul besteht standardmäßig aus den ESV-, TXT-, LRLD- und END-Sätzen. Zusätzlich können auf Wunsch über die TEST-SUPPORT-Option (siehe 2.4.5) für das Symbolische Testen mit AID LSD-Einträge erzeugt werden (diese Funktion wird vom ASSEMBH-BC nicht unterstützt).

Einträge in den Bindelademodul

ESV	Binder-Laderinformation (Definition und Referenz von externen Namen)
LSD	Testhilfeinformation für AID
TXT	Befehle und Anweisungen im Maschinencode
LRLD	Binder-Laderinformation (Relativierung von Adressen)
END	Endinformation des Bindelademoduls

ESV = External Symbols Vector
LSD = List for Symbolic Debugging
TXT = Textinformation
LRLD = Local Relocation Dictionary

3.2.3 Überwachung der Übersetzung mit der Monitorjobvariablen MONJV

Mit Hilfe des Softwareprodukts JV (Jobvariablen) können Aufträge und Programme im BS2000 überwacht und gesteuert werden (siehe "BS2000 Jobvariablen" Beschreibung [7]). Der Benutzer definiert eine sog. überwachende Jobvariable, die er bei einem LOGON-, ENTER-JOB- oder START-PROG-Kommando als Operand angibt. Das Betriebssystem trägt in diese Jobvariable Informationen über den aktuellen Zustand eines Programms ("Zustandsanzeige") und weitere auf Programmebene definierte Informationen ("Rückkehrcode-Anzeige") ein. Nach dem Programmende können diese Informationen vom Benutzer abgefragt werden; weitere Aufträge und Programme können in Abhängigkeit von diesen Informationen gesteuert werden.

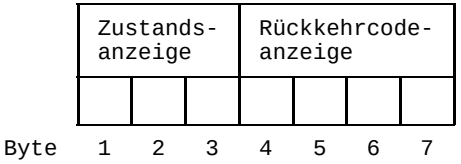
Am Ende einer Programmübersetzung mit dem Assembler wird die überwachende Monitorjobvariable mit einer Zustands- und einer Rückkehrcode-Anzeige versorgt.

Beim Aufruf des ASSEMBH über die Unterprogrammschnittstelle wird der MONJV-Wert über Parameter zurückgegeben.

Da Mehrfachübersetzungen und Restarts erlaubt sind, gilt folgendes:
Die programmüberwachende Jobvariable beinhaltet die Werte aus dem Übersetzungsabschnitt, in dem das höchste Fehlergewicht auftrat.

3.2.3.1 Aufbau der Monitorjobvariablen

Der MONJV-Wert gliedert sich in eine 3 Byte lange Zustandsanzeige und in eine 4 Byte lange Rückkehrcode-Anzeige.



Die Zustandsanzeige wird linksbündig in den ersten 3 Bytes, die Rückkehrcode-Anzeige im vierten bis siebten Byte gesetzt.

Zustandsanzeige

Die 3-stellige Zustandsanzeige in der überwachenden Jobvariablen wird durch den Assembler folgendermaßen besetzt:

Zustands-anzeige	Beendigungs-code BC	
$T_	0 1	Normale Beendigung
$A_	2 3	Abnormale Beendigung

Rückkehrcode-Anzeige

Der in der MONJV abgelegte 4-stellige Rückkehrcode hat folgenden Aufbau:

	BC	PI
Länge in Bytes	1	3

BC = Beendigungscode
PI = Programminformation

BC = Beendigungscode, kann die folgenden Werte annehmen:

BC	Erläuterung
0	Normale Beendigung. Es traten keine Warnungen oder Fehler auf, höchstens NOTES.
1	Normale Beendigung. Es traten Warnungen oder Fehler der Klassen WARNING/SIGNIFICANT/ SERIOUS auf (siehe nächste Tabelle PI).
2	Abnormale Beendigung. Es wurde ein durch eine Option gesetztes Abbruchkriterium erreicht (max. Fehlergewicht, max. Fehleranzahl; siehe 2.4.6).
3	Abnormale Beendigung. Es wurde ein Fehler der Klasse FATAL, ein Ein/Ausgabe-Fehler oder ein Assemblerfehler erkannt.

PI = Programminformation, kann die folgenden Werte annehmen:

PI	Erläuterung	Text:HIGHEST ERROR-WEIGHT:	
		am Terminal	im Listing
000	Keine Flags und keine MNOTES gemeldet. Keine Informationsmeldungen.	NO ERRORS	-
001	Informationsmeldungen wurden ausgegeben	NOTES	-
002	Höchste aufgetretene Fehlerklasse	WARNING	0
003	Höchste aufgetretene Fehlerklasse	SIGNIFICANT	1
004	Höchste aufgetretene Fehlerklasse	SERIOUS	2
005	Höchste aufgetretene Fehlerklasse	FATAL	3
006	Assemblerfehler, Ein/Ausgabe-Fehler	FAILURE	3

Mögliche Kombinationen

		Assembler-Fehlergewicht						
		-	Inf. Meld.	WAR	SIG	SER	FAT	Assem. fehler
Zustands-anzeige	BC	Programminformation PI						
		000	001	002	003	004	005	006
$T_	0	x	x					
$T_	1			x	x	x		
$A_	2*)			x	x	x		
$A_	3*)						x	x

*) In diesen Fällen wird zum Jobstep verzweigt.

Mit dem Kommando

```
/START-PROG $ASSEMBH,MONJV=jvname
```

wird die Jobvariable vom Betriebssystem eingerichtet.

jvname darf maximal 41 Zeichen lang sein und alle Buchstaben, die Ziffern 0 - 9 und die Sonderzeichen -, @, #, $ enthalten.

Beispiel

Der Übersetzungslauf wird mit der Jobvariablen JOBVAR überwacht. Der Binder soll nur dann aufgerufen werden, wenn der Assembler keine Fehler und keine Warnungen gemeldet hat.

```
/BEGIN-PROC LOGG=A,PAR=YES(PROC-PAR=(&PROG),ESC-CHAR=C'&')
/ASSIGN-SYSDTA TO-FILE=*SYSCMD
/START-PROG $ASSEMBH,MONJV=JOBVAR  ————————————————————————  (1)
//C SOURCE=(PLAMLIB,&PROG),MOD-LIB=PLAMLIB,LIST=(OUTPUT=(PLAMLIB))
//END
/SET-JOB-STEP
/SHOW-JV JV-ID=NAME(JV-NAME=JOBVAR)  ————————————————————————  (2)
/SKIP-COM TO-LABEL=ENDE,IF=JV(COND=(JOBVAR,4,4)>'0001')  ————————  (3)
/START-PROG $TSOSLNK
PROG &PROG,LIB=PLAMLIB
INCLUDE &PROG,PLAMLIB
END
/SKIP-COM TO-LABEL=ENDE
/SET-JOB-STEP
/.ENDE END-PROC
```

(1) Mit dem START-PROG Kommando wird JOBVAR dem Assembler als programmüberwachende Jobvariable zugeordnet.

(2) Mit dem SHOW-JV Kommando wird der Wert der Jobvariablen ausgegeben.

(3) Mit dem SKIP-COM Kommando wird geprüft, ob die Rückkehrcode-Anzeige (4. - 7. Byte) einen Wert größer '0001' enthält; in diesem Fall
hat der Assembler Fehler ab der Fehlerklasse "Warnings" gemeldet und die Prozedur verzweigt zur Marke ".ENDE".

4 Laufzeitsystem für die strukturierte Programmierung

4.1 Allgemeines

Die strukturierte Programmierung im Assembler (siehe "ASSEMBH", Beschreibung [1]) benötigt zum Ablauf eines Programms ein Laufzeitsystem, welches die Registersicherung und die Beschaffung und Freigabe von Speicher für Registersicherung, Automatic- und Controlled-Bereiche durchführt.

Dieses Laufzeitsystem wird als Modul IASSRTS in der Bibliothek SYSLIB.ASSEMBH.012 bereitgestellt.
Die Beschaffung und Freigabe von Speicher für Registersicherung, Automatic- und Controlled-Bereiche basiert auf Routinen von ILCS (Inter Language Communication Services). Der ILCS-Initialisierungsmodul IT0INITS aus der Bibliothek SYSLIB.ASSEMBH.012 lädt die benötigten ILCS-Routinen dynamisch nach.
In dieser Bibliothek wird auch der ILCS-Modul IT0ENTR bereitgestellt; in ihm sind alle Eingänge der ILCS-Routinen, außer IT0INIT(S), zusammengefaßt.

Die Speicherverwaltung ist voll dynamisch.

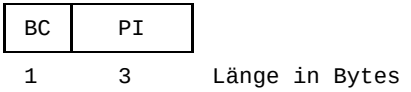
Assemblerobjekte mit strukturierter Programmierung, die mit dem COLUMBUS-Assembler V2.2F übersetzt wurden, sind auch mit dem neuen Laufzeitsystem kompatibel ablauffähig.

4.2 Unterstützung von Monitorjobvariablen

Das neue Laufzeitsystem unterstützt die Verwendung von Monitorjobvariablen. Dazu übergibt das Laufzeitsystem sowohl bei normaler als auch bei abnormaler Programmbeendigung entsprechende Rückkehrcodes an die ILCS-Routinen; dort werden sie an die Jobvariable weitergereicht.

In einigen Fehlersituationen, bei denen das ILCS noch nicht initialisiert ist, erfolgt das Setzen der Monitorjobvariablen direkt im Laufzeitsystem.

Das Laufzeitsystem erzeugt für MONJV den 4-stelligen Rückkehrcode in folgender Form:



BC = Beendigungscode mit folgenden Werten:

BC	Erläuterung
0	Normale Beendigung
3	Abnormale Beendigung

PI = Programminformation mit folgenden Werten

PI	Erläuterung
000	Während des Programmlaufs traten keine Fehler auf.
001 . . . nnn	Während des Programmlaufs traten Fehler auf. Die dreistelligen Nummern entsprechen den rechts stehenden 3 Zeichen der Meldungen des Laufzeitsystems (ASS7nnn, siehe 11.1)

Folgende Kombinationen sind möglich:

$T_0000 Normale Programmbeendigung
$A_3001 Abnormale Beendigung
.
.
$A_3nnn

5 Binden, Laden und Starten

5.1 Allgemeines

Das Ergebnis einer Übersetzung eines Quellprogramms sind ein oder mehrere Objektmodule oder Bindelademodule. Die vom Assembler erzeugten Objektmodule stehen in der temporären EAM-Datei (OMF) der aktuellen Task oder als Elemente (vom Typ R) in einer PLAM-Bibliothek. Die erzeugten Bindelademodule stehen als Elemente (vom Typ L) in einer PLAM-Bibliothek (siehe 3.2, Ausgaben des ASSEMBH). Diese Module bestehen zwar schon aus Maschinencode, müssen aber, damit ein ablauffähiges Programm entsteht, zu einem Lademodul gebunden werden. Das fertig gebundene Programm muß, bevor es ausgeführt werden kann, in den Arbeitsspeicher geladen werden. Deshalb wird das ablauffähige Programm auch als "Lademodul" (= zu ladender Modul) bezeichnet.

Es können auch weitere Module, wie z.B. getrennt übersetzte Quellprogramme oder Unterprogramme in anderen Sprachen eingebunden werden. Diese weiteren Module können dabei zu unterschiedlichen Zeiten und von verschiedenen Compilern übersetzt worden sein.

Die wichtigste Funktion des Binders besteht darin, die für die Ablaufeinheit erforderlichen Module aus den verschiedenen Quellen (Dateien, Bibliotheken) abzurufen und miteinander zu verknüpfen. Die Verknüpfung besteht darin, daß der Binder jeden Modul um diejenigen Adressen ergänzt, die sich auf Bereiche außerhalb des Moduls beziehen (Externverweise).

Für die Aufgaben des Bindens und Ladens stehen im BS2000 verschiedene Dienstprogramme zur Verfügung:

- **Der Binder BINDER (ab BS2000 V10.0)**

Der BINDER (siehe 5.2) bindet Objektmoduln (OMs) und Bindelademoduln (LLMs) zu einer logisch und physikalisch strukturierten ladbaren Einheit zusammen. Diese Einheit bezeichnet man als "Bindelademodul" (Link and Load Module, LLM). Abgespeichert wird ein LLM vom BINDER als Element vom Typ L in einer PLAM-Bibliothek.

- **Der dynamische Bindelader (DLL bis BS2000 V9.5; DBL ab BS2000 V10.0)**

Der dynamische Bindelader DBL (siehe 5.3) fügt in einem Arbeitsgang Objektmoduln (OMs) und Bindelademoduln (LLMs) zu einem temporären Programm zusammen, lädt es sofort in den Speicher und startet es. Nach Ablauf des Programms wird es automatisch gelöscht. Die Verwendung des DBL ist vor allem in der Testphase angebracht.

- **Der statische Binder TSOSLNK**

Der statische Binder TSOSLNK (siehe 5.4) bindet Objektmoduln und speichert das erzeugte ablauffähige Programm (auch "Lademodul" genannt) in einer katalogisierten Datei oder in einer PLAM-Bibliothek (Element-Typ C) ab.

- **Der statische Lader ELDE**

Der statische Lader ELDE (siehe 5.5) hat die Aufgabe, ein ablauffähiges Programm zu laden, das vom TSOSLNK gebunden wurde.

5.2 Binden mit dem BINDER

Mit dem BINDER können Objektmoduln (OMs) und Bindelademoduln (LLMs) zu einem LLM gebunden und als Element vom Typ L in einer PLAM-Bibliothek abgespeichert werden. Der BINDER ist ausführlich im Handbuch "BINDER" [10] beschrieben.

Die vom ASSEMBH erzeugten Objektmoduln stehen entweder in der EAM-Datei der aktuellen Task oder als Elemente vom Typ R in einer PLAM-Bibliothek.
Die LLMs stehen als Elemente vom Typ L in einer PLAM-Bibliothek.

Steueranweisungen für den BINDER (Auswahl)

```
/START-PROGRAM $BINDER                                                    (1)
 START-LLM-CREATION INT-NAME=name                                         (2)
 INCLUDE-MODULES LIB= {bibliothek} ,ELEM= {element}                       (3)
                     {*OMF       }        {*ALL   }
[INCLUDE-MODULES LIB=..., ELEM=...]                                       (4)
[RESOLVE-BY-AUTOLINK LIB=SYSLIB.ASSEMBH.012                               (5)
[RESOLVE-BY-AUTOLINK LIB=..., [SYMBOL-NAME=externverweis]]                (6)
[MODIFY-SYMBOL-VISIBILITY ..., VISIBLE=NO]                                (7)
 SAVE-LLM LIB=bibliothek, ELEM=element                                    (8)
 END                                                                      (9)
```

(1) Der BINDER wird aufgerufen.

(2) Diese Anweisung erzeugt einen neuen LLM im Arbeitsbereich mit dem internen Namen "name". Der erzeugte LLM wird mit der Anweisung SAVE-LLM (siehe 8) als Element vom Typ L in einer PLAM-Bibliothek gespeichert.

(3) Mit bibliothek wird der Name der PLAM-Bibliothek angegeben, in der sich die Moduln befinden bzw. mit *OMF die EAM-Datei.
Mit element wird der Name eines Moduls angegeben.
Bei Angabe von *ALL werden alle Moduln aus der angegebenen Eingabequelle eingebunden.

(4) Mit einer weiteren INCLUDE-MODULE-Anweisung können zusätzliche Moduln aus verschiedenen Bibliotheken eingebunden werden.

(5) Das ASSEMBH-Laufzeitsystem (falls Sie die strukturierte Programmierung anwenden wollen) wird mit RESOLVE-BY-AUTOLINK fest eingebunden.

(6) Mit weiteren RESOLVE-BY-AUTOLINK-Anweisungen werden dem BINDER die Externverweise (= Modulnamen) und die entsprechenden Bibliotheken bzw. nur die Bibliotheken mitgeteilt, die mit dem Autolink-Verfahren nach bisher unbefriedigten Externverweisen durchsucht werden sollen.

(7) Mit der MODIFY-SYMBOL-VISIBILITY-Anweisung können externe Symbole für weitere Binderläufe maskiert werden. Standardmäßig bleiben die Symbole sichtbar. Siehe Abschnitt weiter unten ’Maskierung von Symbolen’.

(8) Diese Anweisung speichert den aktuellen LLM, der mit START-LLM-CREATION erzeugt wurde, als Element vom Typ L in eine PLAM-Bibliothek.

(9) Mit der END-Anweisung wird der BINDER-Lauf beendet.

Bei den Anweisungen INCLUDE-MODULES und RESOLVE-BY-AUTOLINK kann anstelle des Bibliotheksnamens (LIB=bibliothek) auch LIB=*BLS-LINK angegeben werden. In diesem Fall müssen die zu durchsuchende Bibliotheken mit dem Linknamen BLSLIBnn ($00 \leq nn \leq 99$) zugewiesen werden. Dies geschieht vor Aufruf des BINDERS mit dem SET-FILE-LINK-Kommando, z.B.:

```
/SET-FILE-LINK LINK-NAME=BLSLIB01,FILE-NAME=SYSLIB.ASSEMBH.012
```

Ein mit dem BINDER erzeugter LLM kann - sofern alle Externverweise befriedigt sind - mit dem DBL ohne Zuweisung alternativer Bibliotheken geladen und gestartet werden:

```
START-PROGRAM *MODULE(LIB=bibliothek,ELEM=modul,RUN-MODE=ADVANCED)
```

Maskierung von Symbolen

Im Gegensatz zum TSOSLNK werden beim Binden mit dem BINDER Symbole (CSECTs, ENTRYs) standardmäßig nicht maskiert und bleiben für spätere Bindeläufe mit dem BINDER oder DBL sichtbar.

Beim dynamischen Binden mit dem DBL hat dies u.a. folgende Auswirkungen: Wenn in einer PLAM-Bibliothek sowohl vom ASSEMBH generierte Einzelmoduln als auch LLMs mit eingebundenem Laufzeitsystem stehen, werden beim dynamischen Binden der Einzelmoduln die Externverweise auf das Laufzeitsystem aus irgendeinem vorgebundenen Modul befriedigt und nicht aus der Laufzeitbibliothek. Bei dieser Gelegenheit erhält man diverse "DUPLICATES"-Warnungen vom DBL. Aufgrund des Autolink-Mechanismus wird zuerst die Bibliothek durchsucht, in der der Einzelmodul steht und erst anschließend die mit den Linknamen BLSLIBnn zugewiesenen Laufzeitbibliotheken.

Um sicherzustellen, daß beim Binden die Externverweise immer aus der aktuellen Laufzeitbibliothek und nicht aus einem beliebigen Modul befriedigt werden, empfehlen wir

- entweder Einzelmoduln und vorgebundene Moduln in unterschiedlichen Bibliotheken zu halten
- oder beim Binden mit dem BINDER die Symbole mit der Anweisung MODIFY-SYMBOL-VISIBILITY zu maskieren

5.3 Dynamisches Binden und Laden mit dem DBL

Mit dem dynamischen Bindelader DBL werden in einem Arbeitsgang Objektmoduln (OMs) und Bindelademoduln (LLMs) temporär zu einem Programm gebunden, dann in den Speicher geladen und ausgeführt. Das erzeugte Programm wird automatisch nach Programmablauf gelöscht.
Die Arbeitsweise des DBL ist im Handbuch "Bindelader-Starter" [9] ausführlich beschrieben.

Der DBL arbeitet in zwei Betriebsmodi. Der Betriebsmodus wird mit dem Operanden RUN-MODE der Kommandos START-PROGRAM und LOAD-PROGRAM ausgewählt.

RUN-MODE=STD (Voreinstellung)
In diesem Modus verhält sich der DBL kompatibel zum DLL bis einschließlich BS2000 V9.5. Es können nur Objektmoduln und keine Bindelademoduln (LLMs) verarbeitet werden.

RUN-MODE=ADVANCED
In diesem Modus können Objektmoduln und Bindelademoduln (LLMs) verarbeitet werden. Dieser Betriebsmodus wird hier nicht beschrieben. Eine ausführliche Beschreibung befindet sich im Handbuch "Bindelader-Starter" [9].

Die vom Assembler erzeugten Moduln stehen entweder in der temporären EAM-Datei der aktuellen Task oder als Elemente (vom Typ R/L) in einer PLAM-Bibliothek.

Sollen Objektmoduln aus der EAM-Datei gebunden werden, so ist vor der Übersetzung die EAM-Datei mit DEL-SYS-FILE OMF zu löschen.

Der Bindelauf mit dem DBL wird mit dem START-PROG oder LOAD-PROG Kommando gestartet. Nach dem START-PROG Kommando wird das Programm sofort ausgeführt. Nach dem LOAD-PROG Kommando besteht die Möglicheit, weitere Kommandos (z.B. Testhilfe Kommandos) einzugeben. Das Programm kann daraufhin mit dem RESUME-PROG Kommando gestartet werden.

Kommandos für den DBL

```
/[START-PROG]                                    [*OMF [,ELEM=*ALL]          ]
 [          ]  [FROM-FILE=] *MODULE (LIB= {*OMF ,ELEMENT=modul             }
 [LOAD-PROG ]                             [bibliothek,ELEM=modul             ]
                                          [,RUN-MODE=STD/ADVANCED]
```

*OMF bezeichnet die temporäre EAM-Datei (OMF), in die der Assembler den Objektmodul ausgegeben hat.

modul gibt den Namen des Moduls an, der geladen werden soll.

bibliothek gibt den Namen einer PLAM-Bibliothek an, in der sich der Modul (OM/LLM) als Element mit dem Namen modul befindet. Dieses Element muß dabei vom Typ R/L sein. Sind mehrere Elemente gleichen Namens in der Bibliothek gespeichert, wird das Elemnt mit der höchsten Version genommen.

RUN-MODE=ADVANCED
Diese Angabe ist immer erforderlich, wenn Bindelademoduln (LLMs) verarbeitet werden sollen.

5.4 Statisches Binden mit dem TSOSLNK

Mit dem Statischen Binder TSOSLNK können Objektmoduln zu einem Programm gebunden und dieses Programm kann in einer katalogisierten Datei oder als Element (vom Typ C) in einer PLAM-Bibliothek abgelegt werden.

Steueranweisungen für den TSOSLNK

```
/START-PROG $TSOSLNK ──────────────────────────────────────────── (1)

*PROGRAM programm [,{FILENAM=datei                    }] ──────── (2)
                    {LIB=bibliothek [,ELEM=element]   }

*INCLUDE {modul,bibliothek / *      } ──────────────────────────── (3)
         {(modul,...),bibliothek / *}

[*RESOLVE [externverweis],bibliothek] ──────────────────────────── (4)

*END ──────────────────────────────────────────────────────────── (5)
```

(1) Der Statische Binder TSOSLNK wird aufgerufen.

(2) Die PROGRAM-Anweisung legt fest, wo das Programm abgelegt werden soll.

programm	Hier ist der Name anzugeben, den das Programm erhalten soll. Ist kein weiterer Operand ("FILENAM" oder "LIB") angegeben, erhält die katalogisierte Datei diesen Namen.
FILENAM=datei	Mit "datei" wird ein Name gewählt, den die katalogisierte Datei erhalten soll. Die max. Länge einschließlich cat-id und user-id beträgt 54 Zeichen.
LIB=bibliothek, ELEM=element	Das Programm wird in der PLAM-Bibliothek namens "bibliothek" als C-Element unter dem Namen "element" abgelegt. Ist nur der Operand "LIB" angegeben, wird "programm" als Elementname angenommen.

(3) Mit der INCLUDE-Anweisung können ein oder mehrere Moduln aus einer Bibliothek eingebunden werden. Eine Liste von Moduln muß durch runde Klammern eingeschlossen werden. Als Bibliotheksname kann auch * für die EAM-Datei (OMF) angegeben werden.
Durch eine Folge von INCLUDE-Anweisungen können Moduln aus verschiedenen Bibliotheken eingebunden werden.

(4) Mit der RESOLVE-Anweisung werden dem Binder die Externverweise (= Objektmodul-Namen) und die entsprechenden Bibliotheken bzw. nur die Bibliotheken mitgeteilt, die mit dem Autolink-Verfahren (siehe nächste Seite) nach bisher unbefriedigten Externverweisen durchsucht werden sollen.

(5) Mit der END-Anweisung müssen die Eingaben für den TSOSLNK abgeschlossen werden.

Autolink-Verfahren des TSOSLNK

Findet der TSOSLNK in einem Objektmodul Externverweise, die nicht durch die Moduln befriedigt werden können, die in INCLUDE-Anweisungen angegeben wurden, so geht er nach folgendem Autolink-Verfahren vor:

- Zuerst sucht der TSOSLNK in der Bibliothek, die in der RESOLVE-Anweisung explizit in Verbindung mit dem Externverweis angegeben wurde.
- Kann der TSOSLNK im ersten Schritt den Externverweis nicht befriedigen, so durchsucht er sämtliche Bibliotheken, die in RESOLVE-Anweisungen angegeben wurden. Dabei wird die letzte RESOLVE-Anweisung zuerst berücksichtigt, die vorletzte als zweite usf.
  Nicht zu durchsuchende Bibliotheken können durch EXCLUDE-Anweisungen von der Suche ausgeschlossen werden.
- Ist es dem TSOSLNK auch im zweiten Schritt nicht gelungen, den Externverweis zu befriedigen, durchsucht er die Bibliothek TASKLIB, sofern dies nicht durch die Anweisung NCAL oder eine entsprechende EXCLUDE-Anweisung verhindert wurde. Falls es unter der Benutzerkennung der laufenden Task keine Bibliothek namens TASKLIB gibt, verwendet der TSOSLNK dabei die Bibliothek des Systems, $TSOS.TASKLIB.

Sind auch nach dem Autolink-Verfahren noch unbefriedigte Externverweise vorhanden, gibt der TSOSLNK ihre Namen in einer Liste nach SYSOUT und SYSLST aus.

Beispiel für einen Bindelauf mit dem TSOSLNK und Starten mit ELDE

Es sollen Objektmoduln, die aus der getrennten Übersetzung von zwei Programmteilen entstanden sind, zu einem Programm gebunden und gestartet werden.
Die Moduln PROG1 und UP1 stehen in der Bibliothek PLAMLIB.

```
/START-PROG $TSOSLNK
%  BLS0500 PROGRAM 'TSOSLNK', VERSION 'V21.0E00' OF '1992-01-07' LOADED.
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1991. ALL
   RIGHTS RESERVED
PROGRAM PROG1,FILENAM=TESTASS
INCLUDE PROG1,PLAMLIB
INCLUDE UP1,PLAMLIB
END
%  LNK0500 PROG BOUND
%  LNK0503 PROG FILE WRITTEN: TESTASS
%  LNK0504 NUMBER PAM PAGES USED:      3
/START-PROG TESTASS
%  BLS0500 PROGRAM 'PROG1', VERSION ' ' OF '90-05-24' LOADED.
HIER IST PROG1
HIER IST UP1
HIER IST WIEDER PROG1
```

5.5 Laden und Starten von Programmen mit dem Lader ELDE

Damit ein fertig gebundenes Programm ablaufen kann, muß es in den Hauptspeicher geladen werden. Für diese Aufgabe steht im BS2000 der Lader ELDE zur Verfügung. Er wird - wie der DBL - implizit mit den Kommandos START-PROG und LOAD-PROG aufgerufen:

– Das START-PROG Kommando weist den ELDE an, das Programm in den Speicher zu laden und zu starten. Da der Programmablauf unmittelbar nach dem Laden beginnt, müssen bereits vorher die Dateien, die das Programm benötigt, zugewiesen werden.

– Das LOAD-PROG Kommando weist den ELDE an, das Programm in den Speicher zu laden, ohne es zu starten. Dadurch lassen sich vor dem Programmablauf weitere Kommandos eingeben - etwa zur Testhilfe. Das Programm kann daraufhin mit dem RESUME-PROG Kommando gestartet werden.

Nachfolgend sind die wichtigsten Angaben der Kommandos START-PROG und LOAD-PROG aufgeführt. Die ausführliche Beschreibung der beiden Kommandos finden Sie im Handbuch "BS2000/OSD-BC Kommandos" [6], ein Beispiel ist auf vorheriger Seite.

```
/{LOAD-PROG } FROM-FILE = {dateiname                                          }
 {START-PROG}             {*PHASE(LIB=bibliothek,ELEM=modul,VERS=version)}
```

dateiname — ist der Name der katalogisierten Datei, die das vom TSOSLNK erzeugte Programm enthält.

bibliothek — ist der Name der PLAM-Bibliothek, die das vom TSOSLNK erzeugte Programm als Element vom Typ C enthält.

modul — ist der Name des Bibliothekselements, in dem das Programm gespeichert ist.

version — ist die Version des Bibliothekselements mit maximal 24 Zeichen Länge.

5.6 Übersetzen und Binden eines strukturierten Assembler-Programms

Beim Übersetzen muß die Bibliothek SYSLIB.ASSEMBH.012 für die Makros der strukturierten Programmierung als Makrobibliothek angegeben werden.

```
/START-PROGRAM  $ASSEMBH
//COMPILE SOURCE=quelldatei,-
//       MACRO-LIBRARY=SYSLIB.ASSEMBH.012,-
//       MODULE-LIBRARY=modulbibliothek
//END
```

Beim Binden muß die Bibliothek SYSLIB.ASSEMBH.012 zum Anbinden des Assembler-Laufzeitsystems angegeben werden.

```
/START-PROGRAM  $TSOSLNK
 PROG strukturiertes-programm,...
 INCLUDE strukturiertes-programm, modulbibliothek
  .
  .
  .
 RESOLVE,SYSLIB.ASSEMBH.012
 END
```

5.7 XS-Unterstützung

Ab der Version 9.0 unterstützt das BS2000 neben der herkömmlichen Hardware auch die XS-Anlagen (XS steht für e**X**tended **S**ystem), die dem Benutzer einen erheblich erweiterten virtuellen Adreßraum zur Verfügung stellen: Gegenüber 16 Megabyte auf den bisherigen NXS-Anlagen (NXS steht für Nicht-XS) können auf ihnen bis zu 2 Gigabyte adressiert werden.

Diese Erweiterung des Adreßraums auf XS-Anlagen wird dadurch ermöglicht, daß zur Adreßbildung nicht 24 Bit (wie bei NXS-Anlagen), sondern 31 Bit eines Adreßwortes herangezogen werden.

Die XS-Programmierung ist in dem Handbuch "Einführung in die XS-Programmierung für Assembler-Programmierer" [4] beschrieben.

5.8 ESA-Unterstützung

Ab BS2000 V11.0 wird ein neuer Adressierungsmodus zur Erweiterung des virtuellen Adreßraums unterstützt. Der erweiterte Adressierungsmodus steht nur auf Anlagen zur Verfügung, die die entsprechende neue Hardware (z.B. H130) besitzen. Auf diesen sogenannten ESA-Anlagen (Enterprise Systems Architecture) werden neben dem bisherigen Adreßraum weitere Adreßräume für Daten zur Verfügung gestellt.
Auf ESA-Anlagen können Sie zum einen mit 24-Bit-Adressen oder 31-Bit-Adressen arbeiten und zum anderen mit Datenräumen oder nur im Programmraum (siehe Benutzerhandbuch "Makroaufrufe an den Ablaufteil" [12]).
Der Assembler ASSEMBH (≥ V1.2A) unterstützt die ESA-Befehle mit dem Operand INSTRUCTION-SET = BS2000-ESA der Option SOURCE-PROPERTIES (2.4.1.4).
Die ESA-Befehle sind im Anhang 11.3 aufgelistet und in der Sprachbeschreibung "Assemblerbefehle (BS2000)" [11] beschrieben.

6 Beschreibung der Listen

Mit der LISTING-Option (siehe 2.4.4)
können das Format, der Umfang und der Ablageort der Listen bestimmt werden.

Die Listen werden nicht direkt vom ASSEMBH erzeugt, sondern von einem Listengenerator erstellt.

Zusätzlich besteht die Möglichkeit die Listen über den Stand-Alone-Listengenerator ASSLG zu erstellen (siehe 2.5). Bedingung dafür ist, daß beim Übersetzen mit der Option COMPILATION-INFO (siehe 2.4.3) die CIF-Information (Protokoll-Information) in eine Bibliothek gespeichert wird.

Die Listen können in fünf verschiedenen Formaten erstellt werden:

- Listen im Standardformat (ASSEMBH)
- ASSEMB V30 kompatible Liste
- Laserdruckerspezifische Liste
- SAVLST (Liste mit ISAM-Schlüssel)
- Strukturierte Liste

6.1 Listen im Standardformat

Der ASSEMBH erzeugt während des Übersetzungsvorgangs ein Assemblerprotokoll (dieses Protokoll besteht aus nachfolgend beschriebenen Listen).
Abhängig von den in der LISTING-Option angegebenen Werten können folgende Listen erstellt werden:

- eine Optionenliste (OPTIONS LISTING); diese Liste wird immer erstellt
- eine ESD-Liste (EXTERNAL SYMBOL DICTIONARY)
- eine Quellprogrammliste (SOURCE LISTING)
- eine Liste der benutzten Dateien und Bibliotheken
- Querverweislisten (XREF LISTINGS)

6.1.1 Optionenliste (OPTIONS LISTING)

Die Optionenliste enthält alle für den aktuellen Übersetzungslauf gültigen Optionen der COMPILE-Anweisung mit den dazugehörenden Operanden und Operandenwerten.
Die Optionenliste wird immer erstellt, d.h. die Erstellung kann nicht unterdrückt werden.

Bei Steuerung mit *COMOPT werden die benutzten Optionen wie bei SDF-Steuerung ausgegeben. Die eingegebenen COMOPT's werden in einer zusätzlichen Liste aufgeführt (siehe 6.2, ASSEMB V30 kompatible Liste).

Wird die Liste über den Stand-Alone-Listengenerator erstellt, so werden die gültigen Operanden der GENERATE-Anweisung als zusätzliche Liste vorangestellt.

```
ASSEMBH LISTING                                                                    10:36:24  1994-03-07   PAGE 0001

SOURCE=:01KH:$HASSEMB.MES.XREF.ENGL,

MACRO-LIBRARY=MES.PLAM,

COPY-LIBRARY=MES.PLAM
  (ELEMENT-TYPE=BOTH),

SOURCE-PROPERTIES=PARAMETERS
  (FROM-COLUMN=1,TO-COLUMN=71,CONTINUATION-COLUMN=16,LOW-CASE-CONVERSION=NO,INSTRUCTION-SET=BS2000-XS,
  PREDEFINED-VARIABLES=NONE),

COMPILER-ACTION=MODULE-GENERATION
  (MODE=STD,MODULE-FORMAT=OM),

MODULE-LIBRARY=MES.PLAM
  (ELEMENT=*STD
    (VERSION=*UPPER-LIMIT)),

COMPILATION-INFO=NONE,

LISTING=PARAMETERS
  (SOURCE-PRINT=WITH-OBJECT-CODE
    (PRINT-STATEMENTS=ACCEPTED,LINE-NUMBERING=NO),
  SOURCE-FORMAT=STD,
  MACRO-PRINT=PARAMETERS
    (NOPRINT-NEST-LEVEL=255,NOPRINT-PREFIX=*NONE,TITLE-STATEMENTS=IGNORED,MACRO-ORIGIN-INFO=SEPARATE),
  MIN-MESSAGE-WEIGHT=SIGNIFICANT,CROSS-REFERENCE=PARAMETERS
    (SYMBOL=YES
      (WITH-ATTRIBUTES=YES,REFERENCED-ONLY=NO,PREFIX=ALL),
    LITERAL=YES,MACRO=YES,COPY=YES,DIAGNOSTICS=YES),
  EXTERNAL-DICTIONARY=YES,LAYOUT=PARAMETERS
    (LINES-PER-PAGE=60,LASER-PRINTER=NO,FORMAT=STD),
  OUTPUT=MES.LIST.XREF),

TEST-SUPPORT=NO,

COMPILER-TERMINATION=PARAMETERS
  (MAX-ERROR-WEIGHT=FATAL,MAX-ERROR-NUMBER=32767,MAX-MACRO-NEST-LEVEL=255,MAX-COPY-NEST-LEVEL=5),

CORRECTION-CYCLE=NO,

MAINTENANCE-OPTIONS=PARAMETERS
  (CHANNEL-INSTRUCTIONS=NO),

COMPILATION-SPACE=STD
```

6.1.2 ESD-Liste (EXTERNAL SYMBOL DICTIONARY)

Die ESD-Liste ist eine Liste über die Definition und Referenz externer Namen von:

- Programmabschnitten (CSECT, inklusive AMODE und RMODE)
- gemeinsamen Hilfsabschnitten (COM)
- Pseudoabschnitten (DSECT, sowie externen Pseudoabschnitten XDSEC)
- Pseudoregistern (DXD)
- Einsprungadressen in der eigenen Übersetzungseinheit (ENTRY)
- Einsprungadressen bzw. Adreßverweise in andere Übersetzungseinheiten (V-Konstanten, EXTRN, WXTRN)

Die protokollierten ESD-Informationen entsprechen den ESD-Sätzen, die bei der Übersetzung erzeugt und im Modul abgelegt werden. Diese Informationen benötigt der Binder und Lader, um Moduln zu einem ablauffähigen Programm zu verknüpfen.

Die ESD-Liste wird standardmäßig erstellt. Die Erstellung kann mit der Option LISTING(EXTERNAL-DICTIONARY=NO) unterdrückt werden.

Die Spalten in der ESD-Liste haben folgende Bedeutung:

Spalte **Bedeutung**

SYMBOL Externer Name
Dieser wird entweder vom Benutzer in den entsprechenden Anweisungen (siehe Spalte TYPE) angegeben oder er wird vom ASSEMBH erzeugt.
Unbenannte CSECT- und COM-Abschnitte werden mit %CSECT bzw. mit %COM protokolliert.
Externe Namen, die vom Binder bearbeitet werden, sind auf acht Zeichen beschränkt. Längere externe Namen werden auf acht Zeichen verkürzt weiterverarbeitet und mit einer Meldung versehen.
Bei einer Modulausgabe im LLM-Format sind maximal 32 Zeichen möglich (siehe 6.6).

TYPE Art des externen Namens

- CM Name eines gemeinsamen Hilfsabschnitts
  (CM ≙ Common Memory; COM-Anweisung).
  Ein unbenannter Hilfsabschnitt wird mit %COM in der Spalte SYMBOL protokolliert.
- DS Name eines Pseudoabschnitts
  (DS ≙ Dummy Section; DSECT-Anweisung).
  Zusätzlich wird diese Zeile, vor der Spalte SYMBOL, mit der Bezeichnung (DUMMY) versehen.
- DX Name eines Pseudoregisters

(DXD-Anweisung).

ER Name einer externen Verknüpfungsadresse
(ER ≙ External Reference; EXTRN-Anweisung).

LD Name einer Verknüpfungsadresse
(LD ≙ Label Definition; ENTRY-Anweisung).

SD Name eines Programmabschnitts
(SD ≙ Section Definition; CSECT- oder START-Anweisung).
Ein unbenannter Programmabschnitt wird mit %CSECT in der Spalte SYMBOL protokolliert.

VC Name einer externen Verknüpfungsadresse
(VC ≙ V-Constant; V-Konstante).

XD Name eines externen Pseudoabschnitts
(XDSEC-Anweisung mit Operand D).
Zusätzlich wird diese Zeile, vor der Spalte SYMBOL, mit der Bezeichnung (DUMMY) versehen.

XR Name über die Referenz eines externen Pseudoabschnitts
(XDSEC-Anweisung mit Operand R).
Zusätzlich wird diese Zeile, vor der Spalte SYMBOL, mit der Bezeichnung (DUMMY) versehen.

WX Name einer bedingten externen Verknüpfungsadresse
(WXTRN-Anweisung).

ID Nummer des externen Namens
(ID ≙ Identification).
Die externen Namen werden für jeden Modul mit 0001 beginnend durchnumeriert.

ADDR Distanz zum Modulbeginn, wenn es ein Modul im OM-Format ist; Distanz zum Beginn der entsprechenden CSECT eines Moduls, wenn es ein Modul im LLM-Format ist (siehe 6.6) bei Definitionen von externen Namen.
Diese Distanz wird sedezimal in Bytes angegeben.

LENGTH Länge eines Abschnitts oder Hilfsabschnitts, sedezimal in Bytes.
Bei V-Konstanten und Verknüpfungsadressen wird keine Längenangabe gemacht.

A/R-MODE In der linken Spalte (A-MODE) wird der Adressierungsmode (24/31/ANY) eines Programmabschnitts protokolliert (AMODE-Anweisung).

In der rechten Spalte (R-MODE) wird das Ladeattribut (24/ANY) eines Programmabschnitts protokolliert (RMODE-Anweisung).

```
ASSEMBH LISTING                                                                                  10:36:24  1994-03-07   PAGE 0002
          SYMBOL   TYPE ID  ADDR     LENGTH  A/R-MODE                EXTERNAL SYMBOL DICTIONARY
          TESTXREF  SD 0001 00000000 000068  24  24
          ADDRCOM   VC 0002
  (DUMMY) BEGIN     DS 0003 00000000 000020
          ADDRCOM   SD 0004 00000068 000020  24  24
          HCOM      CM 0005 00000000 000009  24  24
          %CSECT    SD 0006 00000088 000002  24  24
          %COM      CM 0007 00000000 000008  24  24
```

6.1.3 Quellprogrammliste (SOURCE LISTING)

Die Ausgabe der Quellprogrammliste ist über die Option LISTING(SOURCE-PRINT=) steuerbar.

Die Quellprogrammliste enthält das Quellprogramm und standardmäßig den Objekt-Code. Am Ende der Quellprogrammliste steht eine Fehlersummen-Meldung. Danach folgt die Endemeldung des Assemblers mit Versions-, Datum- und Uhrzeitangabe.

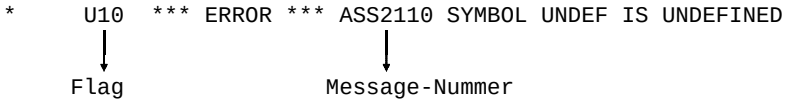
Die Spalten in der Quellprogrammliste haben folgende Bedeutung:

Spalte	**Bedeutung**
LOCTN	Adreßpegel, sedezimal (3 Byte lang).
OBJECT CODE	Objektcode, sedezimal (6 Byte lang).
ADDR1	Adresse des ersten Operanden, sedezimal (4 Byte lang).
ADDR2	Adresse des zweiten Operanden, sedezimal (4 Byte lang).
STMNT	Fortlaufende Zeilennummer, beginnend mit 1.
M	Eine Ziffer bezeichnet die Schachtelungstiefe der Makros und COPYs: 1 Stufe 1 2 Stufe 2 usw. + bedeutet, daß diese Instruktionen durch Makroanweisungen im Quellprogramm generiert wurden.
SOURCE STATEMENT	Text des Quellprogramms Eine Zeile des Quellprogramms kann aus fünf Einträgen bestehen. Von links nach rechts können dort stehen: Namen, Operationen, Operanden, Kommentare und Fortsetzungszeichen.

Zum Inhalt der Adreßfelder LOCTN, ADDR1 und ADDR2 bei einem Modul im LLM-Format siehe dazu Abschnitt 6.6

Anzeige im Fehlerfall

Der Assembler erzeugt bei der Übersetzung im Fehlerfall Diagnosemeldungen (siehe Anhang 11.1). Diese stehen hinter den Zeilen, auf die sie sich beziehen.
Die Zeile beginnt mit einem * und kann wie folgt aussehen:



```
ASSEMBH LISTING                                                                                    10:36:24  1994-03-07   PAGE 0003
LOCTN  OBJECT CODE    ADDR1    ADDR2     STMNT  M  SOURCE STATEMENT
000000                                       1     TESTXREF START
                                             2               PRINT NOGEN
                                             3     *
                                             4               COPY  MES.EQU
                      00000005               5   1 R5        EQU   5
                      00000006               6   1 R6        EQU   6
                      00000007               7   1 R7        EQU   7
                      00000008               8   1 R8        EQU   8
                      00000009               9   1 R9        EQU   9
                      0000000A              10   1 R10       EQU   10
                      0000000E              11   1 R14       EQU   14
                      0000000F              12   1 R15       EQU   15
                                            13     *
000000 05 50                                14     BEG       BALR  R5,0
000002                00000002              15               USING *,R5
000002 D2 02 50505053 00000052 00000055     16               MVC   FIELD,NUMBER
000008 47 F0 0000                           17               B     UNDEF
*    U10  *** ERROR *** ASS2110 SYMBOL UNDEF IS UNDEFINED
00000C D2 02 5050505E 00000052 00000060     18               MVC   FIELD,=C'456'
000012 47 F0 5053     00000055              19               B     NUMBER
*    D7   *** ERROR *** ASS0407 ALIGNMENT ERROR IN OPERAND 1
                                            20     *
                                            21               MNOTE 152,'BRANCH ADDRESS IS WRONG'
                                            22     *
000016 41 60 5030     00000032              23               LA    R6,INPUT
00001A                00000000              24               USING BEGIN,R6
00001A 58 F0 5056     00000058              25               L     R15,=V(ADDRCOM)
00001E 05 EF                                26               BALR  R14,R15
                                            27     *
000020                                      28               TERM
                                            31   2                 *,VERSION 010                                    00001300
                                            43     *
000032                                      44     INPUT     DS    CL32
000052                                      45     FIELD     DS    CL3
000055 F1F2F3                               46     NUMBER    DC    C'123'
                                            47     *
000000                                      48     BEGIN     DSECT
000000                                      49     NR        DS    CL2
000002                                      50     NAME      DS    CL10
00000C                                      51     STREET    DS    CL20
                                            52     *
000068                                      53     ADDRCOM   CSECT
000068 05 70                                54               BALR  R7,0
00006A                0000006A              55               USING *,R7
00006A 58 80 505A     0000005C              56               L     R8,=A(HCOM)
00006E                00000000              57               USING HCOM,R8
00006E D2 04 80045061 00000004 00000063     58               MVC   COM2,=C'12345'
000074                                      59               TERM
                                            62   2                 *,VERSION 010                                    00001300
                                            74     *
000000                                      75     HCOM      COM
000000                                      76     COM1      DS    F
000004                                      77     COM2      DS    CL5
                                            78     *
000088                                      79               CSECT
```

```
ASSEMBH LISTING                                                                                    10:36:24  1994-03-07   PAGE 0004
LOCTN  OBJECT CODE    ADDR1    ADDR2     STMNT  M  SOURCE STATEMENT
000088 05 90                                80             BALR  R9,0
00008A                0000008A              81             USING *,R9
                                            82     *
000000                                      83             COM
000000                                      84     COM3    DS    F
000004                                      85     COM4    DS    F
                                            86     *
000000                                      87             END   BEG
000058 00000000                             88                   =V(ADDRCOM)
00005C 00000000                             89                   =A(HCOM)
000060 F4F5F6                               90                   =C'456'
000063 F1F2F3F4F5                           91                   =C'12345'
FLAGS IN 00002 STATEMENTS, 000 PRIVILEGED FLAGS, 001 MNOTES
HIGHEST ERROR-WEIGHT : SERIOUS ERROR
THIS PROGRAM WAS ASSEMBLED BY ASSEMBH     V 1.2A00     ON 1994-03-07 AT 10:33:03
```

6.1.4 Liste der benutzten Dateien und Bibliotheken

In dieser Liste sind die benutzte Source-Quelle, der erzeugte Modul sowie die verwendeten Makro- und COPY-Bibliotheken aufgeführt.

```
ASSEMBH LISTING                                                                10:36:24  1994-03-07   PAGE 0005
USED FILES AND LIBRARIES
SOURCE FILE   :   :01KH:$HASSEMB.MES.XREF.ENGL
MODULE LIBRARY  :      :01KH:$HASSEMB.MES.PLAM
MODULE ELEMENT  :      TESTXREF
VERS/DATE       :      @/1994-03-07
MACRO-LIBRARIES     LINKNAME  LIBRARY-NAME
                              MES.PLAM
                              :D:$TSOS.MACROLIB
COPY-LIBRARIES      LINKNAME  LIBRARY-NAME
                              :01KH:$HASSEMB.MES.PLAM
```

6.1.5 Querverweislisten

Querverweislisten liefern in aufsteigender Reihenfolge Informationen über den Ort im Quellprogramm von

- Namen (SYMBOL-XREF)
- Literalen (LITERAL-XREF)
- Namen von Makros (MACRO-XREF)
- Namen von COPY-Elementen (COPY-XREF)
- nichtdefinierte Namen (UNDEFND SYMBOL-XREF)
- vom Assembler festgestellte Fehler, sowie benutzereigene Meldungen (DIAGNOSTIC-XREF: FLAG-XREF und MNOTE-XREF)

Standardmäßig werden die FLAG-XREF und die MNOTE-XREF erstellt. Alle übrigen Querverweislisten können mit der Option LISTING(CROSS-REFERENCE) angefordert werden.

Bei Anforderung einer SYMBOL-XREF wird zusätzlich die UNDEFND SYMBOL-XREF erzeugt.
Standardmäßig wird in beiden Listen in der Spalte 'REFERENCES' das zum Symbol gehörende Attribut mit ausgegeben. Die Attribute weisen auf die Zugriffsart hin.
Folgende Attribute sind möglich:

A : Adreßzugriff
E : EQU / ORG-Anweisungen
R : Lesender Zugriff durch Befehle
W : Schreibender Zugriff

In der SYMBOL-XREF werden unbenannte CSECT- und COM-Abschnitte mit %CSECT bzw. mit %COM protokolliert.

Zum Inhalt des Adreßfeldes VALUE der SYMBOL-XREF und LITERAL-XREF bei einem Modul im LLM-Format siehe dazu Abschnitt 6.6

```
ASSEMBH LISTING                                                                 10:36:24  1994-03-07   PAGE 0006
SYMBOL    LEN  VALUE    DEFN    REFERENCES
%COM     00008 00000000 000083
%CSECT   00002 00000088 000079
ADDRCOM  00032 00000068 000053
ADDRCOM  00000 00000000 000000  000025A
BEG      00002 00000000 000014  000087
BEGIN    00032 00000000 000048  000024  000048
COM1     00004 00000000 000076
COM2     00005 00000004 000077  000058W
COM3     00004 00000000 000084
COM4     00004 00000004 000085
FIELD    00003 00000052 000045  000016W 000018W
HCOM     00009 00000000 000075  000056A 000057  000075
INPUT    00032 00000032 000044  000023A
NAME     00010 00000002 000050
NR       00002 00000000 000049
NUMBER   00003 00000055 000046  000016R 000019A
R10      00001 0000000A 000010
R14      00001 0000000E 000011  000026W
R15      00001 0000000F 000012  000025W 000026R
R5       00001 00000005 000005  000014W 000015
R6       00001 00000006 000006  000023W 000024
R7       00001 00000007 000007  000054W 000055
R8       00001 00000008 000008  000056W 000057
R9       00001 00000009 000009  000080W 000081
STREET   00020 0000000C 000051
TESTXREF 00104 00000000 000001
```

```
ASSEMBH LISTING                                                                 10:36:24  1994-03-07   PAGE 0007
LITERAL   LEN  VALUE    DEFN    REFERENCES
=A(HCOM)
        00004 0000005C 000089  000056
=C'12345'
        00005 00000063 000091  000058
=C'456'
        00003 00000060 000090  000018
=V(ADDRCOM)
        00004 00000058 000088  000025
```

```
ASSEMBH LISTING                                                                 10:36:24  1994-03-07   PAGE 0008
MACRO-NAME                                          LINKNAME TYPE     VERSION                 DATE       DEF-STMNT
         LIBRARY-NAME/SOURCE-NAME                   ELEMENT-NAME
         REFERENCES
##BAL                                                        M        010                     1988-06-14
         :D:$TSOS.MACROLIB                          ##BAL
         000034  000065
#INTF                                                       M        919                     1987-12-11
         :D:$TSOS.MACROLIB                          #INTF
         000029  000060
IDLKG                                                       M        002                     1987-12-11
         :D:$TSOS.MACROLIB                               IDLKG
         000030  000061
TERM                                                        M        010                     1988-06-15
         :D:$TSOS.MACROLIB                               TERM
         000028  000059
```

```
ASSEMBH LISTING                                                                 10:36:24  1994-03-07   PAGE 0009
COPY-NAME LIBRARY-NAME                              LINKNAME TYPE     VERSION                 DATE
         REFERENCES
MES.EQU   :01KH:$HASSEMB.MES.PLAM                                S        @                       1992-02-28
         000004
```

```
ASSEMBH LISTING                                                    10:36:24  1994-03-07   PAGE 0010
UNDEFND-SYMBOL REFERENCES
UNDEF          000017A



ASSEMBH LISTING                                                    10:36:24  1994-03-07   PAGE 0011
                                        DIAGNOSTICS
FLAG  MESSAGE AND STATEMENT NUMBERS
D7    ASS0407 ALIGNMENT ERROR IN OPERAND
                000019
U10   ASS2110 SYMBOL   IS UNDEFINED
                000017



ASSEMBH LISTING                                                    10:36:24  1994-03-07   PAGE 0012
                                        DIAGNOSTICS
       SEVERITY CODES OF MNOTES AND STATEMENT NUMBERS
       MNOTE WITH SEVERITY CODE 0152          000021
```

6.1.6 Endemeldung

Assembly time : Zeit für eine Übersetzung ohne die Zeit für die Listenerzeugung

Endemeldung des Listengenerators mit Versionsangabe

```
ASSEMBLY TIME :          0.543     SEC.
THIS LISTING WAS GENERATED BY THE LISTING GENERATOR V 1.2A00.
```

6.2 ASSEMB V30 kompatible Liste

Mit der Option LISTING=PAR(LAYOUT=PAR(FORMAT=F-ASSEMB-COMPATIBLE)) kann eine ASSEMB V30 kompatible Liste erzeugt werden.

Bei Steuerung mit *COMOPT wird stets eine ASSEMB V30 kompatible Liste erzeugt. Die angegebenen COMOPT's werden in einer zusätzlichen Optionenliste (USER'S OPTIONS) aufgeführt.

```
ASSEMBH LISTING - FORMAT: F_ASSEMB_COMPATIBLE                                                    11:46:25  94-03-07   PAGE 0001

SOURCE=:01KH:$HASSEMB.MES.TEST1F.ENGL,

MACRO-LIBRARY=*NONE,

COPY-LIBRARY=*NONE,

SOURCE-PROPERTIES=PARAMETERS
  (FROM-COLUMN=1,TO-COLUMN=71,CONTINUATION-COLUMN=16,LOW-CASE-CONVERSION=NO,INSTRUCTION-SET=BS2000-NXS,
  PREDEFINED-VARIABLES=NONE),

COMPILER-ACTION=MODULE-GENERATION
  (MODE=F-ASSEMB-COMPATIBLE,MODULE-FORMAT=OM),

MODULE-LIBRARY=MES.PLAM
  (ELEMENT=*STD
    (VERSION=*UPPER-LIMIT)),

COMPILATION-INFO=NONE,

LISTING=PARAMETERS
  (SOURCE-PRINT=WITH-OBJECT-CODE
    (PRINT-STATEMENTS=ACCEPTED,LINE-NUMBERING=NO),
  SOURCE-FORMAT=STD,
  MACRO-PRINT=PARAMETERS
    (NOPRINT-NEST-LEVEL=255,NOPRINT-PREFIX=*NONE,TITLE-STATEMENTS=ACCEPTED,MACRO-ORIGIN-INFO=SEPARATE),
  MIN-MESSAGE-WEIGHT=SIGNIFICANT,CROSS-REFERENCE=PARAMETERS
    (SYMBOL=YES
      (WITH-ATTRIBUTES=NO,REFERENCED-ONLY=NO,PREFIX=ALL),
    LITERAL=YES,MACRO=YES,COPY=NO,DIAGNOSTICS=YES),
  EXTERNAL-DICTIONARY=YES,LAYOUT=PARAMETERS
    (LINES-PER-PAGE=60,LASER-PRINTER=NO,FORMAT=F-ASSEMB-COMPATIBLE
      (MESSAGE-PLACEMENT=SEPARATE)),
  OUTPUT=*SAVLST-AND-SYSLST),

TEST-SUPPORT=YES,

COMPILER-TERMINATION=PARAMETERS
  (MAX-ERROR-WEIGHT=FATAL,MAX-ERROR-NUMBER=32767,MAX-MACRO-NEST-LEVEL=255,MAX-COPY-NEST-LEVEL=5),

CORRECTION-CYCLE=NO,

MAINTENANCE-OPTIONS=PARAMETERS
  (CHANNEL-INSTRUCTIONS=NO),

COMPILATION-SPACE=STD




ASSEMBH LISTING - FORMAT: F_ASSEMB_COMPATIBLE                                                    11:46:25  94-03-07   PAGE 0002
          *** USER'S OPTIONS ***
*COMOPT SOURCE=MES.TEST1F.ENGL
*COMOPT XREF,ISD,SAVLST
*COMOPT MODULE=MES.PLAM
*END HALT




ASSEMBH LISTING - FORMAT: F_ASSEMB_COMPATIBLE                                                    11:46:25  94-03-07   PAGE 0003
          SYMBOL   TYPE ID  ADDR     LENGTH  A/R-MODE                EXTERNAL SYMBOL DICTIONARY
          TEST1F    SD 0001 00000000 00002E  24  24
```

```
ASSEMBH LISTING - FORMAT: F_ASSEMB_COMPATIBLE                                                    11:46:25  94-03-07   PAGE 0004
FLAG LOCTN OBJECT CODE    ADDR1  ADDR2  STMNT M  SOURCE STATEMENT
   000000                                   1    TEST1F   START
                         000005             2    R5       EQU   5
                                            3    *
   000000 05 50                             4    BEGIN    BALR  R5,0
   000002                000002             5             USING *,R5
   000002 D2 02 501C501F 00001E 000021      6             MVC   FIELD,NUMBER
   000008 47 F0 501F     000021             7             B     NUMBER
                                            8             TERM
                                            9  1          #INTF INTNAME=TERM,REFTYPE=REQUEST,INTCOMP=001
                                           10  1          IDLKG VER=010,ALIGN=F
                                           11  2                *,VERSION 010                                          00001300
   00000C                                  12  2          CNOP  0,4                                                  00002800
   00000C                                  13  2          DS    0F                                                   00003500
                                           14  1          ##BAL 1,*+16
   00000C 45 10 501A     00001C            15  2          BAL   1,*+16
   000010 01                               16  1          DC    XL1'01'
   000011 00                               17  1          DC    XL1'00'
   000012 00                               18  1          DC    XL1'00'
   000013 04                               19  1          DC    XL1'04'
   000014 40404040                         20  1          DC    CL4'    '
   000018 00000075                         21  1          DC    XL4'00000075'
   00001C 0A 09                            22  1          SVC   9
                                           23    *
   00001E                                  24    FIELD    DS    CL3
   000021 F1F2F3                           25    NUMBER   DC    C'123'
   000024 07 00                            26    END      NOPR  0
   000000                                  27             END   BEGIN
   000026 9203101514384858                 28                   =X'9203101514384858' CONSISTENCY CONSTANT FOR AID
FLAGS IN 00001 STATEMENTS, 000 PRIVILEGED FLAGS, 000 MNOTES
HIGHEST ERROR-WEIGHT : 1
THIS PROGRAM WAS ASSEMBLED BY ASSEMBHC    V 1.2A00
```

```
ASSEMBH LISTING - FORMAT: F_ASSEMB_COMPATIBLE                                                    11:46:25  94-03-07   PAGE 0005
USED FILES AND LIBRARIES
SOURCE FILE   :   :01KH:$HASSEMB.MES.TEST1F.ENGL
MODULE LIBRARY  :      :01KH:$HASSEMB.MES.PLAM
MODULE ELEMENT  :      TEST1F
VERS/DATE       :      @/1994-03-07
SYSTEM MACROLIBRARY :  :D:$TSOS.MACROLIB
MACRO-LIBRARIES     LINKNAME  LIBRARY-NAME
                    SYSLIB    :D:$TSOS.MACROLIB
```

```
ASSEMBH LISTING - FORMAT: F_ASSEMB_COMPATIBLE                                                    11:46:25  94-03-07   PAGE 0006
SYMBOL    LEN  VALUE    DEFN    REFERENCES
BEGIN    00002 00000000 000004  000027
END      00002 00000024 000026
FIELD    00003 0000001E 000024  000006
NUMBER   00003 00000021 000025  000006  000007
R5       00001 00000005 000002  000004  000005
TEST1F   00046 00000000 000001
```

```
ASSEMBH LISTING - FORMAT: F_ASSEMB_COMPATIBLE                                                    11:46:25  94-03-07   PAGE 0007
MACRO NAME VERS/DATE  DEFN      REFERENCES
##BAL      010/880614 SYSLIB    000014
#INTF      919/871211 SYSLIB    000009
IDLKG      002/871211 SYSLIB    000010
TERM       010/880615 SYSLIB    000008
```

```
ASSEMBH LISTING - FORMAT: F_ASSEMB_COMPATIBLE                                    11:46:25  94-03-07   PAGE 0008
                                              DIAGNOSTICS
FLAG  MESSAGE AND STATEMENT NUMBERS
D7    ASS0407 ALIGNMENT ERROR IN OPERAND
                000007
ASSEMBLY TIME :          0.212     SEC.
THIS LISTING WAS GENERATED BY THE LISTING GENERATOR V 1.2A00.
```

6.3 Laserdruckerspezifische Liste

Mit der Option LISTING=PAR(LAYOUT=PAR(LASER-PRINTER=ND2)) kann eine laserdruckerspezifische Liste (ND-Liste) erstellt werden. Sie enthält folgende Unterschiede zu der Liste im Standardformat:

- Die ND-Liste (Quellprogrammliste) ist in drei Abschnitte unterteilt:
  - Objektcode
  - Quellprogramm
  - Zusatzinformation

  Objektcode und Quellprogramm sind identisch mit der Standardliste.
  Die Zusatzinformation besteht aus:
  - ISAM-Schlüssel, falls das übersetzte Programm in einer ISAM-Datei steht.
  - Abschnittsnamen der Symbole, die Adressen in Befehlen darstellen.
  - OPSYN-Protokoll weist den mnemotechnischen Operationscode auf, der mittels einer OPSYN-Anweisung verändert wurde.
  - STACK-Level gibt bei jeder STACK- oder UNSTK-Anweisung die Schachtelungstiefe an:
    U_x für USING (wobei: $1 \leq x \leq 4$)
    P_x für PRINT (wobei: $1 \leq x \leq 4$)
  - MTRAC-Information wird vollständig ausgegeben.
    Einschränkung: Der Wert von SETC-Variablen wird bis zu 50 Zeichen ausgedruckt.

- In allen Querverweis- und Diagnoselisten wird die Anzahl der Anweisungsnummern auf 24 pro Zeile erhöht.

- Ausdruck einer laserdruckerspezifischen Liste
  Mit folgender Optionenangabe erstellt der ASSEMBH eine ND-Liste und legt diese in einer Datei ab:

  ```
  // COMPILE...,LISTING=PAR(LAYOUT=PAR(LASER-PRINTER=ND2),OUTPUT=dateiname)
  ```

  Die Zeilenlänge in der ND-Liste beträgt maximal 205 Zeichen. Daher muß für die Ausgabe des Protokolls auf den Laserdrucker das entsprechende Papierformat (FORM-NAME=) und der entsprechende Zeichensatz (CHARACTER-SETS=) im PRINT-FILE-Kommando angegeben werden. Die entsprechenden Werte sind beim Systemverwalter zu erfragen.

  Mit folgendem Kommando kann ein Protokoll ausgegeben werden:

  ```
  /PRINT-FILE FILE-NAME=dateiname,LAYOUT-CONTROL=PAR(FORM-NAME=format,
   CHARACTER-SETS=chars)
  ```

6.4 SAVLST (Liste mit ISAM-Schlüssel)

Mit der Option LISTING=PAR(OUTPUT=*SAVLST) kann eine Liste im SAVLST-Format, die zum ASSEMB V30 kompatibel ist, erstellt werden.

Bei einer Modulausgabe im LLM-Format ändern sich die Inhalte der Adreßfelder ADDR in der ESD-Liste; LOCTN, ADDR1, ADDR2 in der Quellprogrammliste und VALUE in den Querverweislisten SYMBOL-XREF und LITERAL-XREF (siehe 6.6).
Das Namensfeld (SYMBOL) in der ESD-Liste ist auf 32 Zeichen vergrößert.

```
0001000    ASSEMBH LISTING - FORMAT: F_ASSEMB_COMPATIBLE                                    11:46:25  94-03-07   PAGE 0001
0001001              SYMBOL   TYPE ID  ADDR     LENGTH  A/R-MODE                EXTERNAL SYMBOL DICTIONARY
0001002
0001003              TEST1F    SD 0001 00000000 00002E  24  24
```

```
0000001000 ASSEMBH LISTING - FORMAT: F_ASSEMB_COMPATIBLE                                    11:46:25  94-03-07   PAGE 0002
0000002000 FLAG LOCTN OBJECT CODE    ADDR1  ADDR2  STMNT M  SOURCE STATEMENT
0000101001     000000                                  1    TEST1F   START
0000201001                           000005            2    R5       EQU   5
0000301001                                             3    *
0000401001     000000 05 50                            4    BEGIN    BALR  R5,0
0000501001     000002                000002            5             USING *,R5
0000601001     000002 D2 02 501C501F 00001E 000021     6             MVC   FIELD,NUMBER
0000701001 D   000008 47 F0 501F     000021            7             B     NUMBER
0000801001                                             8             TERM
0000901001                                             9  1          #INTF INTNAME=TERM,REFTYPE=REQUEST,INTCOMP=001
0001001001                                            10  1          IDLKG VER=010,ALIGN=F
0001101001                                            11  2                *,VERSION 010                                00001300
0001201001     00000C                                 12  2          CNOP  0,4                                          00002800
0001301001     00000C                                 13  2          DS    0F                                           00003500
0001401001                                            14  1          ##BAL 1,*+16
0001501001     00000C 45 10 501A     00001C           15  2          BAL   1,*+16
0001601001     000010 01                              16  1          DC    XL1'01'
0001701001     000011 00                              17  1          DC    XL1'00'
0001801001     000012 00                              18  1          DC    XL1'00'
0001901001     000013 04                              19  1          DC    XL1'04'
0002001001     000014 40404040                        20  1          DC    CL4'    '
0002101001     000018 00000075                        21  1          DC    XL4'00000075'
0002201001     00001C 0A 09                           22  1          SVC   9
0002301001                                            23    *
0002401001     00001E                                 24    FIELD    DS    CL3
0002501001     000021 F1F2F3                          25    NUMBER   DC    C'123'
0002601001     000024 07 00                           26    END      NOPR  0
0002701001     000000                                 27             END   BEGIN
0002801001     000026 9203101514384858                28                   =X'9203101514384858' CONSISTENCY CONSTANT FOR AID
0002802000 FLAGS IN 00001 STATEMENTS, 000 PRIVILEGED FLAGS, 000 MNOTES
0002803000 HIGHEST ERROR-WEIGHT : 1
0002804000 THIS PROGRAM WAS ASSEMBLED BY ASSEMBHC    V 1.2A00
           SYSTEM MACROLIBRARY :  :D:$TSOS.MACROLIB
           DIAGNOSTIC FILE :      :01KH:$HASSEMB.SAVLST.ASSEMBH.TEST1F
```

```
         0 ASSEMBH LISTING - FORMAT: F_ASSEMB_COMPATIBLE                                  11:46:25  94-03-07   PAGE 0003
         0 SYMBOL    LEN  VALUE    DEFN    REFERENCES
         0
BEGIN    1 BEGIN
         2          00002 00000000 000004  000027
END      1 END
         2          00002 00000024 000026
FIELD    1 FIELD
         2          00003 0000001E 000024  000006
NUMBER   1 NUMBER
         2          00003 00000021 000025  000006  000007
R5       1 R5
         2          00001 00000005 000002  000004  000005
TEST1F   1 TEST1F
         2          00046 00000000 000001
```

```
        0 ASSEMBH LISTING - FORMAT: F_ASSEMB_COMPATIBLE                                  11:46:25  94-03-07   PAGE 0004
        0 MACRO NAME VERS/DATE  DEFN      REFERENCES
        0
##BAL   1 ##BAL
        2           010/880614 SYSLIB    000014
#INTF   1 #INTF
        2           919/871211 SYSLIB    000009
IDLKG   1 IDLKG
        2           002/871211 SYSLIB    000010
TERM    1 TERM
        2           010/880615 SYSLIB    000008



D070000010 ASSEMBH LISTING - FORMAT: F_ASSEMB_COMPATIBLE                                 11:46:25  94-03-07   PAGE 0005
D070000020                                          DIAGNOSTICS
D070000030 FLAG  MESSAGE AND STATEMENT NUMBERS
D070300011 D7    ASS0407 ALIGNMENT ERROR IN OPERAND
D070300022                 000007
0000000001 94-03-07 11:13:38 V 1.2A00     TEST1F
           :01KH:$HASSEMB.MES.TEST1F.ENGL                          51

           THIS LISTING WAS GENERATED BY THE LISTING GENERATOR V 1.2A00.
```

6.5 Die strukturierte Liste

Durch die Integration der Strukturierung von Assembler-Listen in den ASSEMBH vereinfacht sich die bisherige Benutzerschnittstelle (siehe 'Dienstprogramme für die strukturierte Programmierung', Kapitel 10) wesentlich.
Die Strukturierungsfunktion wird mit der Option LISTING=(,SOURCE-FORMAT= STRUCTURED,) ausgewählt (siehe 2.4.4) und direkt im Anschluß an die Übersetzung durchgeführt.
Voraussetzung für die Erstellung einer strukturierten Liste ist die Anwendung der vordefinierten Makros (folgend auch Strukturmakros genannt) der Strukturierten Programmierung (siehe "ASSEMBH, Beschreibung" [1]).

Die strukturierte Liste kann auch mit dem Stand-Alone-Listengenerator (siehe 2.5) aus einem in einer vorangegangenen Übersetzung erstellten permanenten CIF (H-Element in einer PLAM-Bibliothek) erzeugt werden. Die Strukturierungsfunktion wird mit der Option SOURCE-FORMAT=STRUCTURED der GENERATE-Anweisung ausgewählt. Hierbei ist zu beachten, daß für alle Möglichkeiten der strukturierten Liste, die Option COMPILATION-INFO=PAR(INFO=MAX) bei der CIF-Erzeugung angegeben war (siehe 2.4.3).

Die strukturierte Liste wird im ASSEMBH-Standardformat erzeugt. Soll eine Liste mit der Option LISTING=(LAYOUT=(FORMAT=F-ASSEMB-COMPATIBLE)) strukturiert werden, so müssen weiterhin die Dienstprogramme (siehe Kapitel 10) verwendet werden.

6.5.1 Leistungen der Strukturierungsfunktion

1. Kennzeichnung von Strukturblöcken durch waagrechte und senkrechte Striche. Einrückung der Instruktionen und Kommentare entsprechend der Strukturtiefe um einen festgelegten Einrückbetrag.
2. Festlegung des Einrückbetrages für jede Strukturstufe und Festlegung von einem nicht verschiebbaren Spaltenbereich.
   Option: `SOURCE-FORMAT=STRUCT(IDENTATION-AMOUNT=...,FIXED-AREA-START=...)`
3. Steuerung der Protokollierung für Strukturmakros
   Option: `SOURCE-FORMAT=STRUCT(,STRUCT-MACRO-PRINT=...)`

Beispiele

1. Übersetzung mit dem ASSEMBH und Strukturierung des Protokolles in einem Lauf.

```
START-PROG $ASSEMBH

  Optionen:
  SOURCE    = Eingabedatei mit nicht-strukturierter Source
  MACRO-LIB = Bibliothek mit den vordefinierten Makros
  LISTING   = (OUTPUT=assemb.list,
               NOPRINT-PREFIX=@,
               SOURCE-FORMAT=STRUCTURED(,INDENT-AMOUNT=n,
               STRUCT-MACRO-PRINT=OBJECT-CODE-ONLY),
               TITLE-STATEM=ACCEPTED,
               LINES-PER-PAGE=n)
```

2. Erstellung und Strukturierung des Protokolles mit dem Stand-Alone-Listengenerator aus der in einer Bibliothek abgelegten CIF-Information.

   Bei der Übersetzung muß die Option COMPILATION-INFO=PAR(INFO=MAX) angegeben werden.

```
START-PROG $ASSLG

  Optionen:
  COMPILER-INFO-FILE = Eingabeelement mit permanentem CIF
  SOURCE-FORMAT      = STRUCTURED(INDENT-AMOUNT=n,
                       STRUCT-MACRO-PRINT=OBJECT-CODE-ONLY),
                       LINES-PER-PAGE=n
  OUTPUT             = assemb.list
```

Behandlung von Strukturfehlern

Strukturfehler werden durch die Strukturmakros über MNOTES gemeldet.
Nach Auftreten eines Fehlers wird die Fortsetzung der Strukturierung versucht.

6.5.2 Aufbereitung des Übersetzungsprotokolls

1. Strukturblöcke

Ein Strukturblock beginnt mit einer Strukturblock-Anfangsanweisung; dazu gehören die Strukturmakros @BEGIN, @IF, @CASE, @CAS2, @WHILE, @CYCLE und @THRU. Der Strukturblock wird mit der Endanweisung @BEND abgeschlossen. Zur Kennzeichnung eines Strukturblockes in einer aufbereiteten Liste wird vom Aufruf des Strukturmakros eine waagrechte Linie zum rechten Rand der Sourcezeile gezogen; dort wird der aktuelle Strukturlevel eingetragen. Beginn und Ende eines Strukturblockes werden durch eine senkrechte Linie verbunden.

Alle zu einem Strukturblock gehörenden Instruktionen und Kommentare werden gemäß der Strukturschachtelung und dem Einrückbetrag protokolliert.

2. Prozeduren

Prozedurkörper werden durch einen waagrechten Strich nach @ENTR und vor @END gekennzeichnet.
Instruktionen außerhalb eines Prozedurkörpers, d.h. vor @ENTR bzw. nach @END, werden nicht eingerückt. Das ist auch nicht notwendig, weil Prozedurvereinbarungen nicht geschachtelt sein können. Außerdem werden auch Instruktionen nach @ENTR bis zum ersten Strukturblock und nach dem letzten Strukturblock bis @END nicht eingerückt. Hier werden vom Benutzer im allgemeinen Datendeklarationen bzw. DSECT's abgelegt, deren Struktur unverändert übernommen wird.

@ENTR, @END und der erste Strukturblock nach @ENTR beginnen in Spalte 10.

3. Verlassen von Strukturblöcken

Durch @BREAK, @EXIT und @PASS können Strukturblöcke verlassen werden. Diese Strukturmakros werden durch einen nach links weisenden Pfeil vor der Anweisung gekennzeichnet. Ein möglicher Namensfeldeintrag wird in einer neuen Zeile vor dem Aufruf des Strukturmakros eingetragen.

```
146    |||@THEN *-----------------------------------------------------
146    ||||     IN ORDNUNG
152  3 ||||
156   <------@PASS NAME=PROC1
162  1 ||||
164  2 ||||
165    |||@ELSE *-----------------------------------------------------
```

4. Behandlung von Instruktionen und Kommentaren

Die Behandlung von Instruktionen ist in Abschnitt 6.5.2.1, die Behandlung von Kommentaren ist in Abschnitt 6.5.2.2 beschrieben.

5. Protokollierungs-Anweisungen

Die Anweisungen EJECT, SPACE und TITLE werden nicht protokolliert.

6. Statementnummern

Muß durch die Einrückung eine Statementzeile in mehrere aufgeteilt werden, so erhält die so erzeugte Folgezeile im Protokoll die gleiche Statementnummer wie die erste Zeile.

6.5.2.1 Behandlung von Instruktionen

1. Namensfeldeintrag bei den Aufrufen der Strukturmakros

Namesfeldeinträge werden nicht eingerückt. Bei den durch einen waagrechten Strich gekennzeichneten Aufrufen der Strukturmakros wird der Name in eine neue Zeile davor eingetragen, wenn der Name größer als 8 Zeichen ist. Die Zeile behält die Nummer des Originalstatements.
Bei Namen ≤ 8 Zeichen bleibt dieser in der Zeile des Aufrufs.

2. Namensfeldeintrag bei Assemblerinstruktionen

Namensfeldeinträge werden nicht eingerückt. Die Struktur bleibt auch bei langen Namensfeldeinträgen erhalten. Erreicht der Name die rechteste senkrechte Strukturlinie, wird der Rest der Zeile ab Opcode in einer Folgezeile abgelegt.

```
        |  |  |  | @BEGI *-----------------------------------------------6-
        |  |  |  |  LR  1,1
SYMBOL_TRANSPORT MVC 0(1,2),0(3)
SEHR_LANGER_NAMENSFELDEINTRAG
        |  |  |  |  MVC ZIEL,QUELLE  LANGES SYMBOL
        |  |  |  |  AR  2,2
        |  |  |  | @BEND *-----------------------------------------------6-
```

3. Sourcezeile

Die Instruktion wird eingerückt und gegebenenfalls auf mehrere Zeilen aufgeteilt. Bei Instruktionen werden der Opcode, die Operanden und das Kommentarfeld analog der ICTL-Standardwerte (10,16) eingerückt. Paßt die Instruktion trotz Blankkomprimierung nicht in die Zeile, werden Folgezeilen generiert.

4. Folgezeile

Die Instruktion wird einschließlich ihrer Folgezeilen nach einer möglichen Blankkomprimierung neu aufgebrochen und eingerückt.
Bevor durch den Einrückbetrag bedingt Folgezeilen erzeugt werden, wird eine Blankkomprimierung zwischen Opcode, Operanden und Kommentarfeld durchgeführt.
Folgezeilen werden entsprechend der Schachtelungstiefe eingerückt, die Operanden werden zusätzlich entsprechend der ICTL-Standardwerte (10,16) eingerückt.

```
@BEGI *-----------------------------------------------3-
 @BEGI *-----------------------------------------------4-
  MVC SEHR_LANGER_ZIELNAME(L'SEHR_LANGER_ZIELNAME),LANGE*
      R_QUELLNAME
 @BEND *-----------------------------------------------4-
 LR  1,1
@BEND *-----------------------------------------------3-
```

5. Makroaufruf mit Operanden

Opcode und Operanden werden gemäß der Verschachtelung eingerückt; muß die Zeile aufgebrochen werden, so werden in den generierten Folgezeilen die Operanden ebenfalls ausgerichtet.

```
    @BEGI *-----------------------------------------6-
     @BEGI *-----------------------------------------7-
      @DATA CLASS=C,BASE=BASISREG,LENGTH=2000,INIT=ADR*
            INIT
     @BEND *-----------------------------------------7-
    @BEND *-----------------------------------------6-
```

6. Makroaufruf im alternativen Format

Opcode und Operanden werden gemäß der Verschachtelung eingerückt; in den Folgezeilen werden die Operanden analog ausgerichtet. Reicht der Platz in der Zeile nicht, so werden auch Folgezeilen aufgebrochen.

```
    @BEGI *-----------------------------------------6-
     @BEGI *-----------------------------------------7-
      @DATA CLASS=C,BASE=BASISREG,                    F
            LENGTH=2000,                              F
            INIT=ADRINIT
     @BEND *-----------------------------------------7-
    @BEND *-----------------------------------------6-
```

6.5.2.2 Behandlung von Kommentaren

1. Kommentar bei Aufrufen von Strukturmakros

Der Kommentar wird generell vom Aufruf getrennt, um die waagrechte Verbindungslinie zur Levelangabe am rechten Rand nicht zu durchbrechen; dazu wird er in eine separate Kommentarzeile nach dem Aufruf übernommen und eingerückt.

Die Kommentarzeile wird mit der Originalstatementnummer vor der Makroexpansion abgesetzt. Dabei wird unter dem einleitenden "*" begonnen, wenn ein einzeiliger Kommentar noch in die Zeile paßt. Reicht der Platz nicht oder liegt ein mehrzeiliger Kommentar vor (Fortsetzungszeilen), wird der Kommentar auf mehrere Zeilen aufgeteilt und im Opcodefeld begonnen.

```
M SOURCE STATEMENT
        @THEN *-------------------------------------------------3-
 *      |     der bei @THEN stehende Strukturwort-Kommentar
3       |
3       | generierte Statements
3       |
```

2. Kommentar bei Assemblerinstruktionen

Der Kommentar bleibt in der Zeile; Mehrfachblanks werden, wenn notwendig, vom Zeilenende ausgehend auch im Text komprimiert. Reicht der Platz nicht aus, erfolgt Blankkomprimierung zwischen Operanden- und Kommentarfeld. Wenn der Platz trotz Blankkomprimierung nicht ausreicht, wird der Kommentar aufgebrochen und auf mehrere Folgezeilen verteilt, wobei die Einträge in den Folgezeilen, je nach Restlänge, zum Kommentarbeginn bzw. zum Operandenfeld ausgerichtet werden. Die Fortsetzung im Operandenfeld wird mit einem zusätzlichen "*" eingeleitet.

```
        | | | | | | | @BEGI *-------------------------10-
        | | | | | | |  MVC ZIEL1,QUELLE1 KOMMENTAR
**      | | | | | | |                                    *
        | | | | | | |  MVC ZIEL2,QUELLE2 KOMMENTARFELD-E*
*       | | | | | | |                    INTRAG 2
**      | | | | | | |                                    *
        | | | | | | |  MVC ZIEL3,QUELLE3 LANGER-KOMMENTA*
*       | | | | | | |      *RFELD-EINTRAG-DREI
        | | | | | | | @BEND *-------------------------10-
```

3. Kommentarzeile

Bei Kommentarzeilen bleibt der einleitende * in Spalte 1 stehen, der Kommentartext wird entsprechend der Schachtelung eingerückt. Reicht der Platz trotz Blankkomprimierung nicht aus, wird die Zeile aufgebrochen und dann eingerückt. Die erzeugten Folgezeilen werden zusätzlich mit einem Kommentarstern gekennzeichnet.

```
        | | | | | | | | @BEGI *-----------------------------------------10-
*       | | | | | | | |      DIESE ZEILE IST AUSGERICHTET
        | | | | | | | |  AR  1,1
*       | | | | | | | |  KOMMENTARZEILE IST AUSGERICHTET UND IN EINER FOLGEZ*
*       | | | | | | | |  EILE FORTGESETZT
        | | | | | | | |  AR  2,2
        | | | | | | | | @BEND *-----------------------------------------10-
```

4. Kommentarzeilen für Kommentarkästen

Durch Belegen der Spalten 2 und 71 kann ein Kommentar als nicht verschiebbar gekennzeichnet werden. Die senkrechten Striche der Strukturblockklammerung werden unterbrochen.

```
        | | | | | | | | @BEGI *-----------------------------------------10-
        | | | | | | | |
*Kommentar wird nicht eingerückt                                           *
        | | | | | | | |  AR  1,1
*****************************************************************************
**                                                                         *
**    KOMMENTARKASTEN:                                                     *
**                                                                         *
**    DIE STRUKTURLINIEN IN EINEM KOMMENTARKASTEN                          *
**    WERDEN BIS ZUM ENDE DES KOMMENTARKASTENS                             *
**    UNTERBROCHEN                                                         *
**                                                                         *
**    DER TEXT IST GUT LESBAR                                              *
**                                                                         *
*****************************************************************************
        | | | | | | | |  LR  2,2
        | | | | | | | | @BEND *-----------------------------------------10-
```

5. Definition eines nicht verschiebbaren rechtsbündigen Kommentarfeldes

Über eine Option steuerbar kann ein nicht verschiebbarer Spaltenbereich definiert werden. Dieser Bereich kann zur Kennzeichnung von Korrekturständen und Ähnlichem verwendet werden.
Dies gilt nicht für durch Makros (auch Strukturmakros) generierte Zeilen.

Option:

```
LISTING = (OUT=assemb.list,SOURCE-FORMAT=STRUCT(,FIXED-AREA-START=m))
```

Die neue Option FIXED-AREA-START darf Werte ab 60 bis 255 (Defaultwert=NONE) annehmen. Sie gibt an, ab welcher Spalte die Quelle bei der Strukturierung nicht verändert werden soll.

Beispiel

Das folgende Beispiel zeigt ein nicht strukturiertes Quellprogramm mit der LISTING-Option, SOURCE-FORMAT=STD (voreingestellter Standardwert).

```
*** STRUKTURIERTES LISTING ***                                                    17:03:59  1994-01-13   PAGE 0003
 LOCTN  OBJECT CODE    ADDR1    ADDR2     STMNT  M  SOURCE STATEMENT
                                              1             PRINT NOGEN,CODE
                                              2             TITLE '*** STRUKTURIERTES LISTING ***'
 000000                                       3     STRUKTLG    @ENTR TYP=E
 000000 90 EC D00C     0000000C               112   1
 000004 18 AF                                 113   1
 000006 58 F0 A110     00000110               118   2
 00000A 05 EF                                 126   3
 00000C 00000060                              127   2
 000010 E2E3D9E4D2E3D3C7                      128   2
 000018                                       131     LANGES_SYMBOL @BEGIN
 000018                                       139     RTCCHECK @IF   ZE            RTC ABPRUEFEN
 000018 12 11                                 146             LTR   1,1
 00001A                                       147             @THEN               IN ORDNUNG
 00001A 47 70 A028     00000028               153   3
 00001E                                       156             @PASS NAME=PROC1
 00001E 58 F0 A114     00000114               162   1
 000022 05 EF                                 169   2
 000024                                       170             @ELSE               NICHT OKAY
 000024 47 F0 A0B8     000000B8               174   1
 000028                                       178     FEHLER   @CASE (1)          FEHLERBEHANDLUNG
 000028 89 10 0001                            185   1
 00002C 48 11 A0B0     000000B0               186   1
 000030 47 F1 A030     00000030               187   1
 000034                                       188     FALL1    @BEGI               FALL 1
 000034 1A 11                                 196             AR          1,1
 000036                                       197             @BEND               ENDE FALL 1
 000036 47 F0 A0B8     000000B8               203   1
 00003A                                       204     FALL2    @BEGI               FALL 2
                                              212     *        MAKROAUFRUF MIT OPERANDEN
                                              213             @BEGI
 00003A                                       219     DATAC1   @DATA            CLASS=C,BASE=5,LENGTH=1000
 00003A 58 F0 A118     00000118               225   1
 00003E 58 50 A11C     0000011C               226   1
 000042 05 EF                                 234   2
 000044 001C                                  235   1
 000046 C35C                                  236   1
 000048 00000000                              237   1
                                              238     *        MAKROAUFRUF IM ALTERNATIVEN FORMAT
 00004C                                       239     DATAC2   @DATA      CLASS=C,BASE=BASISREG,                              F
                                              239                   LENGTH=2000,                                            F
                                              239                   INIT=ADRINIT
 00004C 58 F0 A118     00000118               245   1
 000050 58 60 A120     00000120               246   1
 000054 0700                                  247   1
 000056 05 EF                                 254   2
 000058 0020                                  255   1
 00005A C3C1                                  256   1
 00005C 00000108                              257   1
 000060 1A 11                                 258             AR             1,1
                                              259     * KOMMENTAR AUSGERICHTET
                                              260     * DIESER KOMMENTAR PASST IN DER AKTUELLEN STRUKTUR NICHT IN EINE ZEILE
                                              261             PRINT GEN
                                              262             COLMAC@             ── MAKRO MIT STRUKTURELEMENTEN
                                              263   1 * MAKRO MIT STRUKTURELEMENTEN
                                              264   1         @IF   EQ
```

```
*** STRUKTURIERTES LISTING ***                                                    17:03:59  1994-01-13   PAGE 0004
 LOCTN  OBJECT CODE    ADDR1    ADDR2     STMNT  M  SOURCE STATEMENT
 000062 15 11                               271   1          CLR   1,1
                                            272   1          @THEN
 000064 47 70 A07C     0000007C             278   4
                                            281   1          @IF EQ
 000068 15 11                               288   1          CLR 1,1
                                            289   1          @THEN
 00006A 47 70 A07C     0000007C             295   4
                                            298   1          @IF   EQ
 00006E 15 11                               305   1          CLR   1,1
                                            306   1          @THEN
 000070 47 70 A07A     0000007A             312   4
 000074 1A 11                               315   1          AR    1,1
                                            316   1          @ELSE
 000076 47 F0 A07C     0000007C             320   2
 00007A 1A 12                               324   1          AR    1,2
                                            325   1          @BEND
                                            332   1          @BEND
                                            339   1          @BEND
                                            346              PRINT NOGEN,CODE
                                            347              @BEND
 00007C                                     353     DATAF1   @FREE BASE=5
 00007C 58 F0 A124     00000124             362   2
 000080 05 EF                               369   3
 000082 001C                                370   2
 000084 5C                                  371   2
 00008A                                     372     DATAF2   @FREE BASE=BASISREG
 00008A 58 F0 A124     00000124             381   2
 00008E 05 EF                               388   3
 000090 0020                                389   2
 000092 5C                                  390   2
 000098                                     391     FALL2_E  @BEND                ENDE FALL 2
 000098 47 F0 A0B8     000000B8             398   1
 00009C                                     399              @BEGI                FALL 3
                                            406              PRINT GEN
                                            407              COLMAC               ── MAKRO OHNE STRUKTURELEMENTE
                                            408   1 * MAKRO OHNE STRUKTURELEMENTE
 00009C 1A 11                               409   1          AR  1,1
 00009E 18 11                               410   1          LR  1,1
 0000A0 1B 11                               411   1          SR  1,1
 0000A2 15 11                               412   1          CLR 1,1
 0000A4 41 10 0002                          413   1          LA    1,2
                                            414              PRINT NOGEN,CODE
 0000A8                                     415              @BEND                ENDE FALL 3
 0000A8 47 F0 A0B8     000000B8             421   1
 0000AC                                     422              @BEND                FEHLERBEH.-ENDE
 0000AC 47 F0 A0B8     000000B8             428   1
 0000B0 0003                                429   1
 0000B2 0004                                430   1
 0000B4 000A                                431   1
 0000B6 006C                                432   1
 0000B8                                     434     RTC_END  @BEND                RTC-ENDE
                                            442              @BEND
                                            448     *
                                            449              @BEGI
 0000B8                                     455              @PASS NAME=PROC2
```

```
*** STRUKTURIERTES LISTING ***                                                    17:03:59  1994-01-13   PAGE 0005
 LOCTN  OBJECT CODE    ADDR1    ADDR2     STMNT  M  SOURCE STATEMENT
 0000B8 58 F0 A128     00000128            461   1
 0000BC 05 EF                              468   2
                                           469              @BEGI
                                           475              @BEGI
 0000BE 41 20 0001                         481     SEHR_LANGER_NAMENSFELD_EINTRAG     LA    2,1
                                           482              @BEGI
                                           488              @BEGI
                                           494              @BEGI
                                           500              @BEGI
                                           506              @BEGI
                                           512              @BEGI
                                           518     *                                                              *
 0000C2 18 11                              519     NAME1    LR 1,1
 0000C4 1A 22                              520     NAME2    AR        2,2        KOMMENTAR
 0000C6 D2 03 A0FCA0F8 000000FC 000000F8   521              MVC       LANGES_ZIELFELD(L'LANGES_ZIELFELD),LANGE_QUELLE
                                           522     *                                                              *
 0000CC D2 03 A0F4A0F0 000000F4 000000F0   523     SEHRLANGER_NAMENS_FELD_EINTRAG  MVC SEHR_LANGES_ZIEL,SEHR_LANGE_QUELLE
                                           524     *                                                              *
 0000D2 41 20 0002                         525              LA    2,2        ASSEMBLERANWEISUNG   MIT   LANGEM   KOMMENTAR
                                           526     *                                                              *
                                           527     *  BLANKKOMPRIMIERUNG IM KOMMENTARFELD                         *
 0000D6 D2 03 A104A100 00000104 00000100   528              MVC  ZIEL,QUELLE    KOMM-FELD         MIT            LUECKEN
                                           529     *
                                           530     ** KOMMENTAR  WIRD  NICHT  EINGERUECKT                         *
 0000DC 1A 11                              531              AR    1,1
                                           532     ************************************************************************
                                           533     **                                                             *
                                           534     **    KOMMENTARKASTEN:                                         *
                                           535     **                                                             *
                                           536     **    DIE STRUKTURLINIEN IN EINEM KOMMENTARKASTEN              *
                                           537     **    WERDEN BIS ZUM ENDE DES KOMMENTARKASTENS                 *
                                           538     **    UNTERBROCHEN                                             *
                                           539     **                                                             *
                                           540     **    DER TEXT IST GUT LESBAR                                  *
                                           541     **                                                             *
                                           542     ************************************************************************
 0000DE 18 22                              543              LR    2,2
                                           544     *
                                           545              @BEND
 0000E0                                    551              @IF   ZE
 0000E0 12 11                              558              LTR   1,1
 0000E2                                    559              @THEN
 0000E2 47 70 A0E8     000000E8            565   3
 0000E6 1A 11                              568              AR    1,1
 0000E8                                    569              @BEND
                                           576              @BEND
                                           582              @BEND
                                           588              @BEND
                                           594              @BEND
                                           600              @BEND
                                           606              @BEND
                                           612              @BEND
                                           618              @BEND
 0000E8                                    624              @EXIT
 0000E8 58 F0 A12C     0000012C            633   2
```

```
*** STRUKTURIERTES LISTING ***                                                    17:03:59  1994-01-13   PAGE 0006
 LOCTN  OBJECT CODE    ADDR1    ADDR2     STMNT  M  SOURCE STATEMENT
 0000EC 05 EF                                640   3
 0000EE F1F0                                 641   2
 0000F0                                      642     SEHR_LANGE_QUELLE    DS      F      LANGER SYMBOLNAME FUER ZIELADRESSE
 0000F4                                      643     SEHR_LANGES_ZIEL     DS      F      LANGER NAME FUER QUELLADR.
 0000F8                                      644     LANGE_QUELLE         DS      F
 0000FC                                      645     LANGES_ZIELFELD      DS      F
 000100                                      646     QUELLE               DS      F
 000104                                      647     ZIEL                 DS      F
 000108                                      648     ADRINIT  DS    F
                       00000006              649     BASISREG EQU   6
 000110                                      650     ENTR_END @END
 000110 00000000                             654   1
 000114 00000130                             655   1
 000118 00000000                             656   1
 00011C 000003E8                             657   1
 000120 000007D0                             658   1
 000124 00000000                             659   1
 000128 00000168                             660   1
 00012C 00000000                             661   1
 000130                                      665     PROC1    @ENTR TYP=I
 000130 90 EC D00C     0000000C              672   1
 000134 18 AF                                673   1
 000136 58 F0 A028     00000158              678   2
 00013A 05 EF                                686   3
 00013C 00000060                             687   2
 000140 D7D9D6C3F1404040                     688   2
 000148                                      691               @PASS NAME=PROC2
 000148 58 F0 A02C     0000015C              697   1
 00014C 05 EF                                704   2
 00014E                                      705               @EXIT
 00014E 58 F0 A030     00000160              714   2
 000152 05 EF                                721   3
 000154 F1F0                                 722   2
 000158                                      723               @END
 000158 00000000                             727   1
 00015C 00000168                             728   1
 000160 00000000                             729   1
 000168                                      733     PROC2    @ENTR TYP=L
 000168                                      741               @EXIT
 000168 07 FE                                747   1
 000170                                      748               @END
                                             755               END
FLAGS IN 00000 STATEMENTS, 000 PRIVILEGED FLAGS, 000 MNOTES
HIGHEST ERROR-WEIGHT : NO ERRORS
THIS PROGRAM WAS ASSEMBLED BY ASSEMBH     V 1.2A00     ON 1994-01-13 AT 17:02:51
```

```
*** STRUKTURIERTES LISTING ***                                                    17:03:59  1994-01-13   PAGE 0007
USED FILES AND LIBRARIES
SOURCE LIBRARY  :      :U:$ASS1.ESC.TSTLIB
SOURCE ELEMENT  :      STR.QUELLE
VERS/DATE       :      @/1993-08-24
MACRO-LIBRARIES     LINKNAME  LIBRARY-NAME
                              :U:$ASS1.ESC.TSTLIB
                              :U:$ASS1.VO.LIB
                              :U:$ASS1.ASS1.LIB
                              :H:$TSOS.SYSLIB.ASSEMBH.011
ASSEMBLY TIME :         16.081     SEC.
THIS LISTING WAS GENERATED BY THE LISTING GENERATOR V 1.2A00.
```

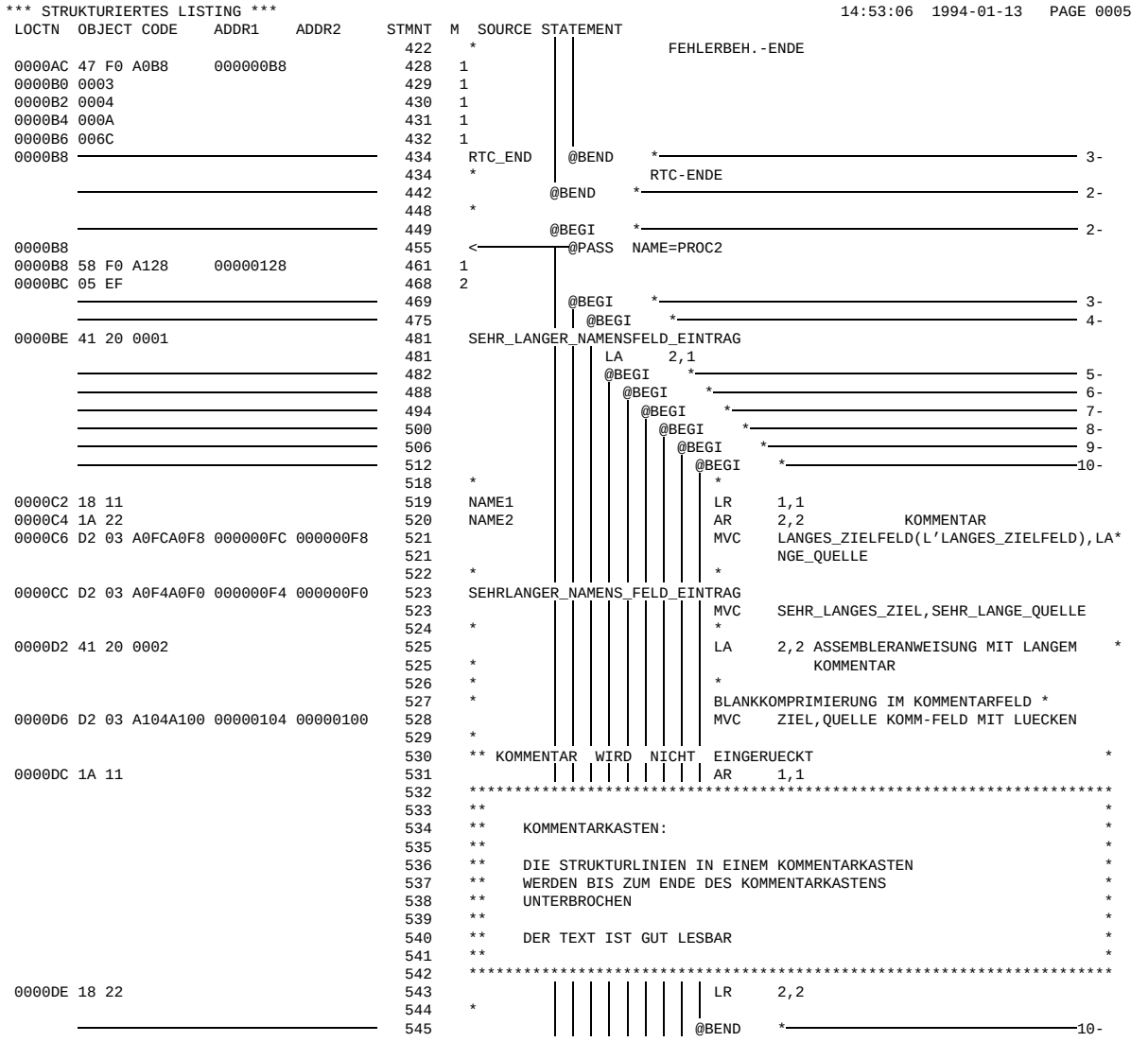
Beispiel

Das folgende Beispiel zeigt ein mit der LISTING-Option, SOURCE-FORMAT= STRUCTURED strukturiertes Quellprogramm (siehe 2.4.4).

```
*** STRUKTURIERTES LISTING ***                                                    14:53:06  1994-01-13   PAGE 0001

SOURCE=*LIBRARY-ELEMENT
  (LIBRARY=:U:$ASS1.ESC.TSTLIB,ELEMENT=STR.QUELLE
    (VERSION=*UPPER-LIMIT)),

MACRO-LIBRARY=
  (ESC.TSTLIB,VO.LIB,ASS1.LIB,$TSOS.SYSLIB.ASSEMBH.011,$TSOS.SYSLIB.BS2CP.100),

COPY-LIBRARY=*NONE,

SOURCE-PROPERTIES=PARAMETERS
  (FROM-COLUMN=1,TO-COLUMN=71,CONTINUATION-COLUMN=16,LOW-CASE-CONVERSION=NO,INSTRUCTION-SET=BS2000-XS,
  PREDEFINED-VARIABLES=NONE),

COMPILER-ACTION=MODULE-GENERATION
  (MODE=STD,MODULE-FORMAT=OM),

MODULE-LIBRARY=*OMF,

COMPILATION-INFO=PARAMETERS
  (INFORMATION=STD,OUTPUT=*LIBRARY-ELEMENT
    (LIBRARY=ESC.TSTLIB,ELEMENT=STR.PROT
      (VERSION=6789-9876543210))),

LISTING=PARAMETERS
  (SOURCE-PRINT=WITH-OBJECT-CODE
    (PRINT-STATEMENTS=ACCEPTED,LINE-NUMBERING=NO),
  SOURCE-FORMAT=STRUCTURED
    (EVALUATED-NEST-LEVEL=ALL,INDENTATION-AMOUNT=2,FIXED-AREA-START=NONE,STRUCT-MACRO-PRINT=OBJECT-CODE-ONLY),
  MACRO-PRINT=PARAMETERS
    (NOPRINT-NEST-LEVEL=255,NOPRINT-PREFIX=@,TITLE-STATEMENTS=IGNORED,MACRO-ORIGIN-INFO=SEPARATE),
  MIN-MESSAGE-WEIGHT=SIGNIFICANT,CROSS-REFERENCE=PARAMETERS
    (SYMBOL=NO,LITERAL=NO,MACRO=NO,COPY=NO,DIAGNOSTICS=YES),
  EXTERNAL-DICTIONARY=YES,LAYOUT=PARAMETERS
    (LINES-PER-PAGE=60,LASER-PRINTER=NO,FORMAT=STD),
  OUTPUT=ESLL.STR.QUELLE),

TEST-SUPPORT=NO,

COMPILER-TERMINATION=PARAMETERS
  (MAX-ERROR-WEIGHT=FATAL,MAX-ERROR-NUMBER=32767,MAX-MACRO-NEST-LEVEL=255,MAX-COPY-NEST-LEVEL=5),

CORRECTION-CYCLE=NO,

MAINTENANCE-OPTIONS=PARAMETERS
  (CHANNEL-INSTRUCTIONS=NO),

COMPILATION-SPACE=STD



*** STRUKTURIERTES LISTING ***                                                    14:53:06  1994-01-13   PAGE 0002
          SYMBOL   TYPE ID  ADDR     LENGTH  A/R-MODE                EXTERNAL SYMBOL DICTIONARY
  (DUMMY) @SAV      DS 0001 00000000 000058
          STRUKTLG  SD 0002 00000000 000170  24  24
          IASSENTR  VC 0003
          IASSCNTR  VC 0004
          IASSFREE  VC 0005
          IASSEXIT  VC 0006
```

```
*** STRUKTURIERTES LISTING ***                                                14:53:06  1994-01-13   PAGE 0003
 LOCTN  OBJECT CODE    ADDR1    ADDR2     STMNT  M  SOURCE STATEMENT
                                              1            PRINT NOGEN,CODE
                                              3   *-------------------------------------------------------------------------
000000                                        3    STRUKTLG @ENTR  TYP=E
000000 90 EC D00C     0000000C              112   1
000004 18 AF                                113   1
000006 58 F0 A110     00000110              118   2
00000A 05 EF                                126   3
00000C 00000060                             127   2
000010 E2E3D9E4D2E3D3C7                     128   2
000018                                      131    LANGES_SYMBOL DS 0H
                                            131            @BEGIN    *------------------------------------------------ 2-
000018 ------------------------------------ 139    RTCCHECK  @IF    ZE    *------------------------------------------- 3-
                                            139    *                       RTC ABPRUEFEN
000018 12 11                                146                    LTR    1,1
00001A ------------------------------------ 147                   @THEN    *------------------------------------------ 3-
                                            147    *                       IN ORDNUNG
00001A 47 70 A028     00000028              153   3
00001E                                      156    <-------------@PASS  NAME=PROC1
00001E 58 F0 A114     00000114              162   1
000022 05 EF                                169   2
000024 ------------------------------------ 170                   @ELSE    *------------------------------------------ 3-
                                            170    *                       NICHT OKAY
000024 47 F0 A0B8     000000B8              174   1
000028 ------------------------------------ 178    FEHLER       @CASE  (1)    *------------------------------------- 4-
                                            178    *                           FEHLERBEHANDLUNG
000028 89 10 0001                           185   1
00002C 48 11 A0B0     000000B0              186   1
000030 47 F1 A030     00000030              187   1
000034 ------------------------------------ 188    FALL1          @BEGI    *------------------------------------- 5-
                                            188    *                       FALL 1
000034 1A 11                                196                    AR     1,1
000036 ------------------------------------ 197                   @BEND    *------------------------------------- 5-
                                            197    *                       ENDE FALL 1
000036 47 F0 A0B8     000000B8              203   1
00003A ------------------------------------ 204    FALL2          @BEGI    *------------------------------------- 5-
                                            204    *                       FALL 2
                                            212    *                     MAKROAUFRUF MIT OPERANDEN
                                            213                          @BEGI    *-------------------------------- 6-
00003A                                      219    DATAC1                  @DATA  CLASS=C,BASE=5,LENGTH=1000
00003A 58 F0 A118     00000118              225   1
00003E 58 50 A11C     0000011C              226   1
000042 05 EF                                234   2
000044 001C                                 235   1
000046 C35C                                 236   1
000048 00000000                             237   1
                                            238    *
                                            239    *                     MAKROAUFRUF IM ALTERNATIVEN FORMAT
00004C                                      239    DATAC2                  @DATA  CLASS=C,BASE=BASISREG,                F
                                            239                                   LENGTH=2000,                          F
                                            239                                   INIT=ADRINIT
00004C 58 F0 A118     00000118              245   1
000050 58 60 A120     00000120              246   1
000054 0700                                 247   1
000056 05 EF                                254   2
000058 0020                                 255   1
00005A C3C1                                 256   1
00005C 00000108                             257   1
000060 1A 11                                258                          AR     1,1
```

```
*** STRUKTURIERTES LISTING ***                                                14:53:06  1994-01-13   PAGE 0004
 LOCTN  OBJECT CODE    ADDR1    ADDR2     STMNT  M  SOURCE STATEMENT
                                            259    *                  KOMMENTAR AUSGERICHTET
                                            260    *                  DIESER KOMMENTAR PASST IN DER AKTUELLEN STRUKTUR    *
                                            260    *                  NICHT IN EINE ZEILE
                                            261                       PRINT  GEN
                                            262                       COLMAC@ ---- MAKRO MIT STRUKTURELEMENTEN
                                            263   1*                  MAKRO MIT STRUKTURELEMENTEN
                                            264   1                   @IF    EQ    *------------------------------------- 7-
000062 15 11                                271   1                    CLR    1,1
                                            272   1                   @THEN    *------------------------------------- 7-
000064 47 70 A07C     0000007C              278   4                    |
                                            281   1                    @IF    EQ    *------------------------------------- 8-
000068 15 11                                288   1                    | CLR    1,1
                                            289   1                    @THEN    *------------------------------------- 8-
00006A 47 70 A07C     0000007C              295   4                    |
                                            298   1                    | @IF    EQ    *------------------------------------- 9-
00006E 15 11                                305   1                    | | CLR    1,1
                                            306   1                    | @THEN    *------------------------------------- 9-
000070 47 70 A07A     0000007A              312   4                    | |
000074 1A 11                                315   1                    | | AR     1,1
                                            316   1                    | @ELSE    *------------------------------------- 9-
000076 47 F0 A07C     0000007C              320   2                    | |
00007A 1A 12                                324   1                    | | AR     1,2
                                            325   1                    | @BEND    *------------------------------------- 9-
                                            332   1                    @BEND    *------------------------------------- 8-
                                            339   1                   @BEND    *------------------------------------- 7-
                                            346                       PRINT  NOGEN,CODE
                                            347                     @BEND    *------------------------------------- 6-
00007C                                      353    DATAF1           @FREE  BASE=5
00007C 58 F0 A124     00000124              362   2
000080 05 EF                                369   3
000082 001C                                 370   2
000084 5C                                   371   2
00008A                                      372    DATAF2          @FREE  BASE=BASISREG
00008A 58 F0 A124     00000124              381   2
00008E 05 EF                                388   3
000090 0020                                 389   2
000092 5C                                   390   2
000098 ------------------------------------ 391    FALL2_E         @BEND    *------------------------------------- 5-
                                            391    *                        ENDE FALL 2
000098 47 F0 A0B8     000000B8              398   1
00009C ------------------------------------ 399                     @BEGI    *------------------------------------- 5-
                                            399    *                |        FALL 3
                                            406                     | PRINT  GEN
                                            407                     | COLMAC ---- MAKRO OHNE STRUKTURELEMENTE
                                            408   1*                | MAKRO OHNE STRUKTURELEMENTE
00009C 1A 11                                409   1                 | AR     1,1
00009E 18 11                                410   1                 | LR     1,1
0000A0 1B 11                                411   1                 | SR     1,1
0000A2 15 11                                412   1                 | CLR    1,1
0000A4 41 10 0002                           413   1                 | LA     1,2
                                            414                     | PRINT  NOGEN,CODE
0000A8 ------------------------------------ 415                     @BEND    *------------------------------------- 5-
                                            415    *                        ENDE FALL 3
0000A8 47 F0 A0B8     000000B8              421   1
0000AC ------------------------------------ 422                   @BEND    *------------------------------------- 4-
```


*** STRUKTURIERTES LISTING ***
14:53:06 1994-01-13 PAGE 0005
LOCTN OBJECT CODE ADDR1 ADDR2 STMNT M SOURCE STATEMENT
FEHLERBEH.-ENDE
RTC_END @BEND
RTC-ENDE
@BEND
@BEGI
@PASS NAME=PROC2
@BEGI
@BEGI
SEHR_LANGER_NAMENSFELD_EINTRAG
LA 2,1
@BEGI
@BEGI
@BEGI
@BEGI
@BEGI
@BEGI
NAME1 LR 1,1
NAME2 AR 2,2 KOMMENTAR
MVC LANGES_ZIELFELD(L'LANGES_ZIELFELD),LA*
NGE_QUELLE
SEHRLANGER_NAMENS_FELD_EINTRAG
MVC SEHR_LANGES_ZIEL,SEHR_LANGE_QUELLE
0000D2 41 20 0002
LA 2,2 ASSEMBLERANWEISUNG MIT LANGEM *
KOMMENTAR
BLANKKOMPRIMIERUNG IM KOMMENTARFELD *
0000D6 D2 03 A104A100 00000104 00000100
MVC ZIEL,QUELLE KOMM-FELD MIT LUECKEN
** KOMMENTAR WIRD NICHT EINGERUECKT *
0000DC 1A 11
AR 1,1
**
** KOMMENTARKASTEN:
**
** DIE STRUKTURLINIEN IN EINEM KOMMENTARKASTEN
** WERDEN BIS ZUM ENDE DES KOMMENTARKASTENS
** UNTERBROCHEN
**
** DER TEXT IST GUT LESBAR
**
0000DE 18 22
LR 2,2
@BEND

```
*** STRUKTURIERTES LISTING ***                                                    14:53:06  1994-01-13   PAGE 0006
 LOCTN  OBJECT CODE    ADDR1    ADDR2     STMNT  M  SOURCE STATEMENT
0000E0 ──────────────────────────────────  551                 @IF    ZE    *──────────────────────────────10-
0000E0 12 11                               558                  LTR    1,1
0000E2 ──────────────────────────────────  559                 @THEN    *──────────────────────────────10-
0000E2 47 70 A0E8     000000E8             565   3
0000E6 1A 11                               568                  AR     1,1
0000E8 ──────────────────────────────────  569                 @BEND    *──────────────────────────────10-
       ──────────────────────────────────  576                @BEND    *──────────────────────────────9-
       ──────────────────────────────────  582               @BEND    *──────────────────────────────8-
       ──────────────────────────────────  588              @BEND    *──────────────────────────────7-
       ──────────────────────────────────  594             @BEND    *──────────────────────────────6-
       ──────────────────────────────────  600            @BEND    *──────────────────────────────5-
       ──────────────────────────────────  606           @BEND    *──────────────────────────────4-
       ──────────────────────────────────  612          @BEND    *──────────────────────────────3-
       ──────────────────────────────────  618         @BEND    *──────────────────────────────2-
0000E8                                     624    <─────────@EXIT
0000E8 58 F0 A12C     0000012C             633   2
0000EC 05 EF                               640   3
0000EE F1F0                                641   2
0000F0                                     642    SEHR_LANGE_QUELLE    DS      F      LANGER SYMBOLNAME FUER ZIELADRESSE
0000F4                                     643    SEHR_LANGES_ZIEL     DS      F      LANGER NAME FUER QUELLADR.
0000F8                                     644    LANGE_QUELLE         DS      F
0000FC                                     645    LANGES_ZIELFELD      DS      F
000100                                     646    QUELLE               DS      F
000104                                     647    ZIEL                 DS      F
000108                                     648    ADRINIT  DS    F
                      00000006             649    BASISREG EQU   6
000110                                     650    ENTR_END @END
──────────────────────────────────────────  651    *──────────────────────────────────────────────────────────
000110 00000000                            654   1
000114 00000130                            655   1
000118 00000000                            656   1
00011C 000003E8                            657   1
000120 000007D0                            658   1
000124 00000000                            659   1
000128 00000168                            660   1
00012C 00000000                            661   1
──────────────────────────────────────────  665    *──────────────────────────────────────────────────────────
000130                                     665    PROC1    @ENTR  TYP=I
000130 90 EC D00C     0000000C             672   1
000134 18 AF                               673   1
000136 58 F0 A028     00000158             678   2
00013A 05 EF                               686   3
00013C 00000060                            687   2
000140 D7D9D6C3F1404040                    688   2
000148                                     691    <─────────@PASS  NAME=PROC2
000148 58 F0 A02C     0000015C             697   1
00014C 05 EF                               704   2
00014E                                     705    <─────────@EXIT
00014E 58 F0 A030     00000160             714   2
000152 05 EF                               721   3
000154 F1F0                                722   2
000158                                     723            @END
──────────────────────────────────────────  724    *──────────────────────────────────────────────────────────
000158 00000000                            727   1
00015C 00000168                            728   1
000160 00000000                            729   1
──────────────────────────────────────────  733    *──────────────────────────────────────────────────────────
000168                                     733    PROC2    @ENTR  TYP=L
000168                                     741    <─────────@EXIT
```

```
*** STRUKTURIERTES LISTING ***                                                    14:53:06  1994-01-13   PAGE 0007
 LOCTN  OBJECT CODE    ADDR1    ADDR2     STMNT  M  SOURCE STATEMENT
 000168 07 FE                               747   1
 000170                                     748             @END
 ------------------------------------------ 749    *-------------------------------------------------------------
                                            755             END
FLAGS IN 00000 STATEMENTS, 000 PRIVILEGED FLAGS, 000 MNOTES
HIGHEST ERROR-WEIGHT : NO ERRORS
THIS PROGRAM WAS ASSEMBLED BY ASSEMBH     V 1.2A00     ON 1994-01-13 AT 14:51:58



*** STRUKTURIERTES LISTING ***                                                    14:53:06  1994-01-13   PAGE 0008
USED FILES AND LIBRARIES
SOURCE LIBRARY  :      :U:$ASS1.ESC.TSTLIB
SOURCE ELEMENT  :      STR.QUELLE
VERS/DATE       :      @/1993-08-24
MACRO-LIBRARIES     LINKNAME  LIBRARY-NAME
                              :U:$ASS1.ESC.TSTLIB
                              :U:$ASS1.VO.LIB
                              :U:$ASS1.ASS1.LIB
                              :H:$TSOS.SYSLIB.ASSEMBH.011
ASSEMBLY TIME :         16.175     SEC.
THIS LISTING WAS GENERATED BY THE LISTING GENERATOR V 1.2A00.
```

6.6 Änderungen in Listen bei Ausgabe des Moduls im LLM-Format

Im bisherigen Objektmodul werden die einzelnen CSECTs zusammenhängend und aufsteigend adressiert (Modul-relative Adressierung).
In der Modulausgabe im LLM-Format wird die CSECT-relative Adressierung verwendet, d.h. jede CSECT des Moduls beginnt bei Adreßpegel 0 (Location 0).
Diesbezüglich verhält sich eine CSECT wie eine DSECT. Dasselbe gilt für die entsprechende Information in der SAVLST (siehe 6.4).

Durch die CSECT-relative Adressierung ändern sich die Inhalte der Adreßfelder in der ESD- und Quellprogrammliste sowie in den Querverweislisten SYMBOL-XREF und LITERAL-XREF. Alle Adreßwerte sind Abstände (Offsets) zum Beginn der entsprechenden CSECT, die jeweils bei Adreßpegel 0 beginnt.

Liste	Feld
ESD	ADDR
SOURCE	LOCTN ADDR1 ADDR2
SYMBOL-XREF, LITERAL-XREF	VALUE

Das Namensfeld (SYMBOL) in der ESD-Liste ist auf 32 Zeichen vergrößert.

Die nachfolgenden Beispiele zeigen die Listen im OM- und LLM-Format.

6.6.1 Listen im OM-Format

ESD-Liste

```
         SYMBOL   TYPE ID  ADDR     LENGTH  A/R-MODE                  EXTERNAL SYMBOL DICTIONARY
         C1        SD 0001 00000000 000020  24  24
         LONGER_N  ER 0002
         C2        SD 0003 00000020 000008  24  24
  (DUMMY) D1        DS 0004 00000000 000004
```

Quellprogrammliste (SOURCE LISTING)

```
LOCTN  OBJECT CODE    ADDR1    ADDR2     STMNT  M  SOURCE STATEMENT
000000                                       1     C1       CSECT
                                             2              EXTRN LONGER_NAME
000000 05 A0                                 3              BALR  10,0
000002                 00000002              4              USING *,10
000002                 00000020              5              USING C2,11
000002                 00000000              6              USING D1,12
000002 58 B0 A016      00000018              7              L     11,=A(C2)
000006 41 20 A012      00000014              8              LA    2,C1_AD1
00000A 41 20 B004      00000024              9              LA    2,C2_AD2
00000E 41 20 C000      00000000             10              LA    2,D1_AD1
000014                                      11   C1_AD1     DS    F
                                            12              LTORG
000018 00000020                             13                    =A(C2)
                                            14              SPACE ,
000020                                      15   C2         CSECT READ
000020                                      16   C2_AD1     DS    CL4
000024                                      17   C2_AD2     DS    F
                                            18              SPACE ,
000000                                      19   D1         DSECT
000000                                      20   D1_AD1     DS    F
000000                                      21              END   C1
```

Querverweisliste (SYMBOL-XREF)

```
SYMBOL    LEN  VALUE    DEFN    REFERENCES
C1       00032 00000000 000001  000021
C1_AD1   00004 00000014 000011  000008A
C2       00008 00000020 000015  000005  000007A 000015
C2_AD1   00004 00000020 000016
C2_AD2   00004 00000024 000017  000009A
D1       00004 00000000 000019  000006  000019
D1_AD1   00004 00000000 000020  0000010A
LONGER_NAME
         00000 00000000 000002
```

6.6.2 Listen im LLM-Format

Nicht kompatible Feldinhalte zum OM-Format sind fett gedruckt.

ESD-Liste

```
         SYMBOL                        TYPE ID  ADDR     LENGTH  A/R-MODE     EXTERNAL SYMBOL DICTIONARY
         C1                             SD 0001 00000000 000020  24  24
         LONGER_NAME                    ER 0002
         C2                             SD 0003 00000000 000008  24  24
  (DUMMY) D1                            DS 0004 00000000 000004
```

Quellprogrammliste (SOURCE LISTING)

```
LOCTN  OBJECT CODE    ADDR1    ADDR2     STMNT  M  SOURCE STATEMENT
000000                                       1     C1       CSECT
                                             2              EXTRN LONGER_NAME
000000 05 A0                                 3              BALR  10,0
000002                 00000002              4              USING *,10
000002                 00000000              5              USING C2,11
000002                 00000000              6              USING D1,12
000002 58 B0 A016      00000018              7              L     11,=A(C2)
000006 41 20 A012      00000014              8              LA    2,C1_AD1
00000A 41 20 B004      00000004              9              LA    2,C2_AD2
00000E 41 20 C000      00000000             10              LA    2,D1_AD1
000014                                      11   C1_AD1     DS    F
                                            12              LTORG
000018 00000000                             13                    =A(C2)
                                            14              SPACE ,
000000                                      15   C2         CSECT READ
000000                                      16   C2_AD1     DS    CL4
000004                                      17   C2_AD2     DS    F
                                            18              SPACE ,
000000                                      19   D1         DSECT
000000                                      20   D1_AD1     DS    F
000000                                      21              END   C1
```

Querverweisliste (SYMBOL-XREF)

```
SYMBOL    LEN  VALUE    DEFN    REFERENCES
C1       00032 00000000 000001  000021
C1_AD1   00004 00000014 000011  000008A
C2       00008 00000000 000015  000005  000007A 000015
C2_AD1   00004 00000000 000016
C2_AD2   00004 00000004 000017  000009A
D1       00004 00000000 000019  000006  000019
D1_AD1   00004 00000000 000020  0000010A
LONGER_NAME
         00000 00000000 000002
```

7 Sprachverknüpfungen

7.1 Symbolische Verknüpfung von Assembler-Programmen

Der Text eines Assembler-Quellprogramms besteht aus einer oder mehreren Übersetzungseinheiten. Eine Übersetzungseinheit fängt in der Regel mit einer START- oder CSECT-Anweisung an und wird durch eine END-Anweisung abgeschlossen. Eine Übersetzungseinheit wird meist als ein "Programm" bezeichnet.
Jede Übersetzungseinheit wird in einen Modul übersetzt.
Eine Übersetzungseinheit kann aus einem oder mehreren Programmabschnitten bestehen, die als Teile eines Moduls übersetzt werden.
Ein oder mehrere Module können zu einem ablauffähigen Programm gebunden werden (siehe 5.4).

Durch entsprechende Anweisungen (siehe weiter unten) ist es möglich

- von einem Programmteil in einen anderen Programmteil zu verzweigen und
- sich auf Daten zu beziehen, die in einem anderen Programmteil definiert sind.

Zu diesem Zweck muß man die Kommunikation zwischen den Programmteilen ermöglichen.

- Jeder einzelne Programmabschnitt, der angesprochen wird, muß symbolisch adressierbar sein.
- Die zwei oder mehr Übersetzungseinheiten, die verbunden werden sollen, müssen symbolisch verknüpft werden.

Die symbolische Programmverknüpfung ermöglicht es, symbolische Adressen, die in einer Übersetzungseinheit definiert sind, von einer anderen aus anzusprechen. Der Assembler benötigt dafür entsprechende Informationen, die er über ESD-Einträge an den Binder weitergibt. Der Binder ersetzt diese symbolischen Bezugnahmen vor bzw. beim Laden durch aktuelle Adressen.
Eine symbolische Adresse, die von einer anderen Übersetzungseinheit aus angesprochen werden soll, muß dem Assembler und dem Binder durch die ENTRY-Anweisung (siehe "ASSEMBH", Beschreibung [1]) kenntlich gemacht werden. Sie ist damit als symbolische Adresse einer Einsprungstelle definiert.

Wenn in einer Übersetzungseinheit symbolische Adressen verwendet werden, die in einer anderen Übersetzungseinheit definiert sind, müssen sie durch die EXTRN- oder WXTRN-Anweisung (siehe "ASSEMBH", Beschreibung [1]) kenntlich gemacht werden. In der Übersetzungseinheit, die die EXTRN-Adresse verwendet, muß ein Basisregister für Zugriffe auf diese Adresse bereitgestellt werden. Der Wert der Adresse muß über eine A-Konstante in das Basisregister geladen werden (siehe "ASSEMBH", Beschreibung [1], DC-Anweisung).

Eine weitere Möglichkeit zur symbolischen Verknüpfung ist die Verwendung von V-Konstanten (siehe "ASSEMBH", Beschreibung [1], DC-Anweisung). Diese Konstanten werden als indirekte Verknüpfungspunkte betrachtet, die aus einer extern definierten symbolischen Adresse erzeugt werden. In diesem Fall darf die symbolische Adresse nicht mit der EXTRN-Anweisung gekennzeichnet sein.
V-Konstanten können für Sprünge in andere Übersetzungseinheiten verwendet werden, jedoch nicht für Bezugnahmen auf Daten in anderen Übersetzungseinheiten.

Die Bezugnahme auf Daten kann z.B. mit den Anweisungen COM, DXD oder XDSEC (siehe "ASSEMBH", Beschreibung [1]) erfolgen.

Beim Programmablauf sind für alle zu einem Programm gebundenen Module die Mehrzweckregister 0-15 nur einmal vorhanden. Diese Register sind die gemeinsame Kommunikationsebene. Für die Verknüpfung von Programmen gilt die Forderung:
Alle Teilprogramme sollen über sämtliche Mehrzweckregister verfügen können.
Das bedeutet, daß beim Verzweigen von einem Modul in einen nachfolgenden Modul die Registerinhalte des rufenden Moduls sichergestellt und beim Rücksprung aus dem gerufenen Modul wieder geladen werden müssen.

Zur Verknüpfung von Assembler-Programmen siehe auch das Kapitel 3.2 ’Programmunterteilung und Programmverknüpfung’ im Handbuch "ASSEMBH", Beschreibung [1].

7.1.1 Verknüpfung mit anderen Sprachen

- Assembler wird von einer anderen Sprache gerufen
  Das bedeutet, das Assembler-Programm muß die Parameterübergabe der rufenden Sprache berücksichtigen und beim Rücksprung entsprechend den Konventionen der rufenden Sprache die Register restaurieren.
- Assembler ruft eine andere Sprache
  Dies geschieht über Transferroutinen oder das Assembler-Programm muß die Konventionen der gerufenen Sprache berücksichtigen (Parameterübergabe und Registerkonvention). Die Sprachumgebung der gerufenen Sprache muß initialisiert sein, d.h. das Laufzeitsystem der gerufenen Sprache muß initialisiert sein.

Die Verknüpfung mit anderen Sprachen wie z.B. COBOL, C, FORTRAN ist in den Benutzerhandbüchern dieser Sprachen beschrieben.

7.2 Verknüpfung von strukturierten Assembler-Programmen

Mit dem Prozedur- und Datenkonzept der Strukturierten Programmierung (siehe "ASSEMBH", Beschreibung [1]) können mehrere Unterprogramme (Prozeduren) mit einem Hauptprogramm (Hauptprozedur) verknüpft werden. Eine Prozedur beginnt mit @ENTR und endet statisch mit @END. @PASS ruft eine andere Prozedur auf und in die gerufene Prozedur können Parameter übergeben werden. @EXIT beendet die gerufene Prozedur (dynamisches Prozedurende) und gibt die Steuerung an die aufrufende Prozedur zurück.

Den Zusammenhang zwischen statischer Programmstruktur und dynamischer Prozedurverknüpfung zeigt folgendes Bild.

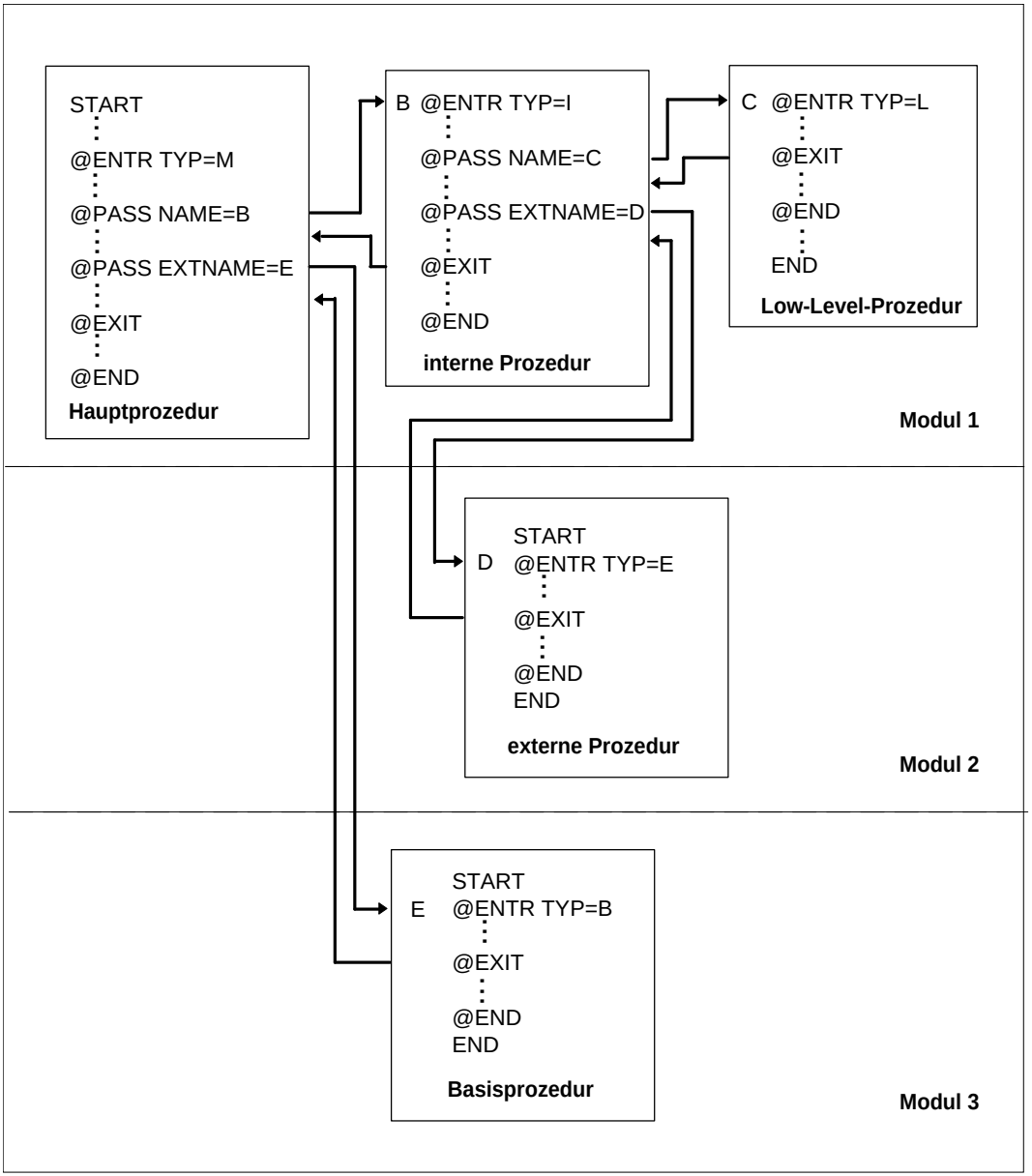


Bild 7-1: Statische Programmstruktur und dynamische Prozedurverknüpfung

7.2.1 Verknüpfung von strukturierten Assembler-Programmen mit C-Programmen

Für C-Programme wird die Möglichkeit geboten, strukturierte Assembler-Programme zu verwenden, die sich wie C-Programme verhalten. D.h. die Stackverwaltung und die Parameterversorgung entsprechen den C-Konventionen.
Über den Parameter &ENV=C des @ENTR-Makros (siehe "ASSEMBH", Beschreibung [1]) wird Code generiert, der für die Save-Area-Verwaltung und die Stackverwaltung den Program-Manager für C-Programme aufruft. Von den @EXIT- und @END-Makros wird ebenfalls der Program-Manager für C-Programme aufgerufen.
Mit den @PASS- und @PAR-Makros wird die Parameterversorgung entsprechend den C-Konventionen generiert.
Der @DATA-Makro kann nur eingeschränkt verwendet werden.
Bei der Benutzung von Speicher der Klasse C (controlled) über die @DATA- und @FREE-Makros muß die C-Umgebung initialisiert sein, d.h. das Hauptprogramm muß ein C-Programm sein.

Für die Registerbenutzung, Speicheranforderung und Parameterübergabe gilt zusätzlich folgendes:

- Register 12 darf nicht benutzt werden.
  Mit dem Parameter LOADR12=YES des @ENTR-Makros wird die Adresse des C-Program-Managers in das Register 12 geladen.
- Register 13 darf nicht benutzt werden.
  Der C-Program-Manager benutzt das Register 13 als Laufzeit-Stack-Register.
- Die Speicherklasse A (Automatic) darf nicht vereinbart werden.
- Es sind nur Prozeduren vom Typ M und E zugelassen.
- Die Parameterübergabe darf nur in STANDARD-Form erfolgen, d.h. Register 1 enthält die Adresse der Parameterliste.
  Diese Übergabe ist für C-Programme zwingend, d.h. der PASS-Parameter des @ENTR- bzw. @PASS-Makros darf nicht auf OPTIMAL-Form geändert werden.

7.2.2 Anschluß von strukturierten Assembler-Programmen an COBOL- und FORTRAN-Programme

Wird von einem COBOL- oder FORTRAN-Hauptprogramm ein strukturiertes Assembler-Programm, das nicht ausschließlich aus Prozeduren vom Typ B, L und D besteht, aufgerufen, so muß vor dem ersten Aufruf einmal der Entry IASSIN zur Initialisierung des Assembler-Laufzeitsystems aufgerufen werden.

Der Initialisierungsentry IASSIN führt die gleichen Funktionen wie bei einem strukturierten Assembler-Hauptprogramm (@ENTR TYP=M) durch.
Der IASSIN-Aufruf erfolgt normalerweise ohne Parameter, wobei ein Standard-Initialstack angelegt wird.

Hinweise

– Aus Kompatibilität zu bestehenden COBOL- und FORTRAN-Objekten mit COLBIN-Aufruf, wird dieser Entry analog zu IASSIN unterstützt, wobei KL5SP-Angaben ignoriert werden.

– Will man die Größe des Initialstacks bestimmen, so muß beim IASSIN-Entry ein Parameter angegeben werden,

  im COBOL vom Typ COMPUTATIONAL PIC 9(n) mit $5 \leq n \leq 9$
  und für FORTRAN vom Typ INTEGER

  Dieser Parameter beeinflußt analog zur STACK-Angabe beim @ENTR TYP=M die Größe des Initialstacks in Bytes, wobei gilt:

$$\text{Größe des Initialstack} = \begin{cases} 1 \text{ Seite} & \text{Parameter } n \leq 0 \\ n \text{ Bytes} & \text{Parameter } n > 0 \end{cases}$$

Parameterübergabe an strukturierte-Assembler-Prozeduren

Von COBOL- bzw. FORTRAN-Programmen werden Parameter immer in der STANDARD-Form übergeben und sind deshalb in der STANDARD-Form zu übernehmen (siehe "ASSEMBH", Beschreibung [1]).

Aufruf von strukturierten Assembler-Prozeduren

Aufruf aus COLUMBUS-COBOL bzw. COBOL

mit Parameter: @PASS name:TYP=E:USING Parameter. bzw.
CALL "name" USING Parameter.

ohne Parameter: @PASS name.:TYP=E: bzw.
CALL "name".

Aufruf aus COLUMBUS-FORTRAN bzw. FORTRAN

mit Parameter: @PASS name(Parameter) bzw.
CALL name(Parameter)

ohne Parameter: @PASS name bzw.
CALL name

Bei Aufruf von Prozeduren des Typs B, L oder D muß der Anwender selbst für die Sicherstellung und die Wiederherstellung der Registerinhalte durch die gerufene Prozedur sorgen.

Rückkehr in das aufrufende COBOL- bzw. FORTRAN-Programm

Die strukturierte Assembler-Prozedur gibt mit der Anweisung

```
@EXIT
```

die Kontrolle an das aufrufende Programm zurück. Die Operanden RC, RESTORE und TO sind nicht zugelassen. Bei Rückkehr in den FORTRAN-Teil muß der Operand PROG=FORTRAN angegeben werden.

7.2.3 Anschluß von strukturierten Assembler-Programmen an Assembler-Programme

Parameterübergabe an strukturierte Assembler-Prozeduren

– Übernahmeform OPTIMAL
  Register 1 bis 4 enthalten die Parameter
  bzw. Register 1 enthält die Adresse einer Parameteradressenliste für die Parameter 4 und folgende.
  Register 2 bis 4 enthalten die Parameter 1 bis 3.

– Übernahmeform STANDARD
  Standardparameterschnittstelle: Register 1 enthält die Adresse der Parameteradressenliste.

Aufruf von Prozeduren ohne Laufzeitsystem (Typ B/L/D)

Strukturierte Assembler-Prozeduren werden über die Standardschnittstelle aufgerufen: Register 15 enthält die Prozeduradresse und Register 14 enthält die Rückkehradresse.

Strukturierte Assembler-Prozeduren können also aufgerufen werden durch:

```
          ⎧A⎫
L    15,= ⎨V⎬ (Prozedurname)
BALR 14,15
```

Der Anwender muß selbst für die Sicherstellung und Wiederherstellung der Registerinhalte durch die gerufene Prozedur sorgen.

Aufruf von Prozeduren mit Laufzeitsystem (Typ E/I)

Aufruf wie bei Prozeduren ohne Laufzeitsystem (Sicherstellung und Wiederherstellung der Registerinhalte wird vom Assembler-Laufzeitsystem übernommen).
Wie bei COBOL- bzw. FORTRAN-Programmen in Verbindung mit strukturierten Assembler-Programmen muß auch hier das Laufzeitsystem dadurch initialisiert werden, daß vor dem ersten Aufruf einer strukturierten Assembler-Prozedur einmal der Entry IASSIN über eine Standardschnittstelle aufgerufen wird.

Schnittstelle für IASSIN = Standardschnittstelle

Register 1	verweist auf ein Wort, das entweder 0 (kein Parameter) oder die Adresse des Parameters (Wortformat) enthält. Bedeutung der Parameterangabe wie bei Anschluß an COBOL- bzw. FORTRAN-Programme (siehe Abschnitt 7.2.2)
Register 15	enthält die Adresse des Laufzeitentry IASSIN
Register 14	enthält die Rückkehradresse
Register 13	verweist auf den standardmäßigen Registersicherstellungsbereich

Registersicherstellungsbereich

Wort 1	intern belegt
Wort 2	verweist auf Vorgänger, bzw. enthält 0, falls kein Vorgänger vorhanden
Wort 3	verweist auf Nachfolger
Wort 4-18	Sicherungsbereich für Register 14,...,12
Wort 19	reserviert
Wort 20	enthält den Zeiger auf die PCD (siehe 7.3.2). Der Zeiger wird vom Laufzeitsystem zurückgemeldet.

Rückkehr in das aufrufende Assembler-Programm

Die strukturierte Assembler-Prozedur gibt mit der Anweisung

```
@EXIT
```

die Kontrolle an das aufrufende Programm zurück. Der Operand TO ist nicht zugelassen.

Beispiel für den Aufruf von IASSIN und einer strukturierten Assembler-Prozedur

1. Ohne Parameterübergabe mit Anlage eines Standard-Initialstacks

```
                    .
                    .
                    LA        13,SAVE
                    .
                    .
                    LA        1,PARAMLIS
                    L         15,=V(IASSIN)
                    BALR      14,15
                    .
                    .
                    L         15,=V(Prozedurname)
                    BALR      14,15
                    .
                    .
PARAMLIS            DC        A(0)           keine Parameterübergabe
                    .
                    .
SAVE                DS        OD
                    DC        3F'0'
                    DS        17F
                    .
                    .
```

2. Mit Parameterübergabe und einem Initialstack von 10000 Bytes

```
                .
                .
STACK           EQU     10000
                .
                .
                LA      13,SAVE
                .
                .
                LA      1,PARAMLIS
                L       15,=V(IASSIN)
                BALR    14,15
                .
                .
                L       15,=V(Prozedurname)
                BALR    14,15
                .
                .
PARAMLIS        DS      OF
                DC      X'80'
                DC      A(PAR1+X'80000000')
                .
                .
PAR1            DC      A(STACK)
                .
                .
SAVE            DS      OD
                DC      3F'O'
                DS      17F
                .
                .
```

Hinweis

Der gerufenen Prozedur und dem Laufzeitentry IASSIN muß in Register 13 die Adresse desselben Registersicherungsbereiches übergeben werden !

7.2.4 Anschluß von COBOL- und FORTRAN-Programmteilen an strukturierte Assembler-Programme

Parameterübergabe von strukturierten Assembler-Prozeduren

COBOL- und FORTRAN-Programmteile akzeptieren Parameterlisten nur in der STANDARD-Form. Bei Aufruf eines COBOL-Teils ohne Parameterübergabe muß Register 1 vor dem Aufruf mit der Adresse eines Wortes, welches 0 enthält, geladen werden. Bei Aufruf eines FORTRAN-Teils ohne Parameterübergabe muß Register 1 vor dem Aufruf mit dem Wert 1 geladen werden.

Aufruf aus strukturierten Assembler-Prozeduren

Die strukturierte Assembler-Prozedur übergibt mit der Anweisung @PASS EXTNAME= die Kontrolle an den COBOL-bzw. FORTRAN-Teil.

Bei Aufruf aus Prozeduren des Typs B, L und D muß der Anwender selbst dafür sorgen, daß Register 13 die Adresse eines Sicherstellungsbereiches enthält.

Rückkehr in die aufrufende strukturierte Assembler-Prozedur

Rückkehr aus	COLUMBUS-COBOL bzw. COBOL @EXIT/@END bzw. EXIT PROGRAM
Rückkehr aus	COLUMBUS-FORTRAN bzw. FORTRAN @EXIT/@END bzw. RETURN/END

7.2.5 Anschluß von Assemblerprogrammteilen an strukturierte Assembler-Programme

Parameterübergabe von strukturierten Assembler-Prozeduren

- Übergabeform OPTIMAL
  Register 1 bis 4 enthalten die Parameter
  bzw. Register 1 enthält die Adresse einer Parameteradressenliste für die Parameter 4 und folgende.
  Register 2 bis 4 enthalten die Parameter 1 bis 3.
- Übergabeform STANDARD
  Standardparameterschnittstelle: Register 1 enthält die Adresse der Parameteradressenliste.

Aufruf aus strukturierten Assembler-Prozeduren

Die strukturierte Assembler-Prozedur übergibt mit der Anweisung @PASS EXTNAME= die Kontrolle an den Assemblerteil, danach enthält Register 14 die Rückkehradresse und Register 15 die Adresse des aufgerufenen Programmteils.

Bei Aufruf aus Prozeduren des Typs B, L und D muß der Anwender selbst dafür sorgen, daß ein Register mit der Adresse eines Sicherstellungsbereiches geladen ist. Bei Aufruf aus Prozeduren des Typs M, E und I ist Register 13 mit der Adresse eines Sicherstellungsbereiches geladen.

Rückkehr in die aufrufende, strukturierte Assembler-Prozedur

Für den Rücksprung muß Register 14, das durch @PASS mit der Rückkehradresse geladen wurde, benützt werden.

7.3 Die Programm-Kommunikationsschnittstelle ILCS

Durch die Programm-Kommunikationsschnittstelle ILCS (Inter-Language Communication Services) wird die Kommunikation zwischen dem Hauptprogramm und den externen Unterprogrammen sowie zwischen den Unterprogrammen untereinander sprachübergreifend vereinheitlicht. Der Anwender kann jeden Programmteil in einer beliebigen ILCS-fähigen Programmiersprache ohne weitere Vorkehrungen (wie etwa Ansprung von Sprachinitialisierungsroutinen, Connection-Module etc.) schreiben.

ILCS ist eine Kombination aus Software und Schnittstellen-Konvention:
Es beinhaltet zum einen Laufzeitroutinen, die in einer PLAM-Bibliothek zusammengefaßt sind, zum anderen garantiert ILCS auch die den "Standard-Linkage-Konventionen im BS2000" entsprechende Kommunikationsschnittstelle; d.h. jeder von einem ILCS-fähigen Compiler erzeugte Modul ist entsprechend den Standard-Linkage-Konventionen für die Verknüpfung mit gleich- und verschiedensprachigen Programmen vorbereitet.

Die Bibliothek der ILCS-Laufzeitroutinen wird mit jedem ILCS-fähigen Compiler - als gleichsam zusätzliches Laufzeitsystem - ausgeliefert.

Im einzelnen bietet ILCS folgende Funktionen:

- multilaterale Konvention zur Verknüpfung verschiedensprachiger Programme,
- einheitliche Richtlinien zur Ereignisbehandlung
- Speicherverwaltung (Stack- und Heap-Speicher)
- Behandlung der Programmaske

In diesem Abschnitt wird nur die von der stukturierten Programmierung des ASSEMBH genutzte ILCS-Funktion Programmverknüpfung mit den grundlegenden ILCS-Datenstrukturen beschrieben.

Hinweis

Programme, die mit ILCS-fähigen Compilern übersetzt wurden, werden obligatorisch mittels ILCS zu einem Programmsystem verknüpft. Enthält ein Programmsystem Programme, die sich nicht gemäß den ILCS-Konventionen verhalten, müssen sie ggf. so umgestaltet werden, daß sie den ILCS-Konventionen entsprechen. Andernfalls besteht - zumindest bei der Verknüpfung verschiedensprachiger Programme - die Gefahr der Inkompatibilität.

7.3.1 ILCS-Registerkonventionen

Registerversorgung bei Aufruf

Die nachfolgende Tabelle gibt eine Übersicht über die Registerversorgung, die das aufrufende Programm vor dem Ansprung des aufgerufenen Programms durchführt.

Register-nummer	Inhalt
0	Anzahl der Parameter
1	Anfangsadresse der Parameteradreßliste
2 - 12	Programmdaten
13	Anfangsadresse des Sicherstellungsbereiches des aufrufenden Programms
14	Adresse des Rückkehrpunktes ins aufrufende Programm
15	Adresse des Einsprungpunktes im aufgerufenen Programm.
PM	Programmaske: Wert aus PCD-Feld "Programmaske"

Registerversorgung bei Rückkehr ins aufrufende Programm

Die nachfolgende Tabelle gibt eine Übersicht über die Registerversorgung, die das aufgerufene Programm beim Rücksprung ins aufrufende Programm durchführt.

Register-nummer	Inhalt
0 - 1	Returnwerte von Ganzzahl-Funktionen oder undefiniert
2 - 14	wie bei Aufruf-Versorgung
15	undefiniert
PM	Programmaske: Wert aus PCD-Feld "Programmaske"

7.3.2 ILCS-Datenstrukturen

Save Area (Sicherstellungsbereich)

Das aufrufende Programm liefert die Adresse eines Sicherstellungsbereiches, in der das aufgerufene Programm die aktuellen Registerstände ablegen kann. Das aufgerufene Programm legt einen neuen Sicherstellungsbereich an und verkettet die beiden Sicherstellungsbereiche.
Der Sicherstellungsbereich ist folgendermaßen aufgebaut:

Byte	Inhalt
1-4	1. Byte: 1. Bit: Aktivitätsbit (1: Programm aktiv, 0: Programm inaktiv) 2.-7. Bit: reserviert 8. Bit = im allgemeinen 0 2. Byte: Version = X'01' 3. und 4. Byte: X'FEFF'
5-8	enthält die Anfangsadresse des Sicherstellungsbereiches des aufrufenden Programms. Im **ersten** aufrufenden Programm ist der Inhalt dieses Feldes -1.
9-12	enthält ggf. die Anfangsadresse des nächsten (geketteten) Sicherstellungsbereiches.
13-16	Inhalt von Register 14
17-20	Inhalt von Register 15
21-24	Inhalt von Register 0
25-28	Inhalt von Register 1
29-32	Inhalt von Register 2
.	.
.	.
69-72	Inhalt von Register 12
73-76	reserviert für FOR1
77-80	Adresse der PCD
81-84	Adresse der EHL (Event Handler List): Ist keine EHL definiert, enthält das Feld den Wert -1.
85-128	reserviert

Prosys Common Data Area (PCD)

Die PCD ist ein allgemeiner Datenbereich, der allen Programmiersprachen zur Verfügung steht. Er hat eine Größe von 4096 Byte.
Der erste Teil enthält die von ILCS verwendeten Datenbereiche, u.a. auch (in Byte 148) das Feld "Programmaske", das mit dem Wert X'0C' vorbelegt ist. Der zweite Teil der PCD enthält die jeweils 128 Byte großen Programmiersprachen-Bereiche, die den Laufzeitsystemen der verschiedenen Sprachen zur Verfügung stehen.

7.3.3 Initialisierung des Programmsystems

Die Initialisierung eines Programmsystems verläuft in zwei Stufen: Zuerst ruft das Hauptprogramm die passende ILCS-Initialisierungsroutine auf. Diese ILCS-Initialisierungsroutine ruft dann ihrerseits alle nötigen sprachspezifischen Initialisierungen auf, so daß vor Ausführung der ersten Programmanweisung die für das gesamte Programmsystem benötigten Sprachumgebungen eingerichtet sind.

7.3.4 Programmasken-Behandlung durch ILCS

Die Programmaske für den Programmablauf wird im Wege der Initialisierung auf den Wert des PCD-Feldes "Programmaske" (vorbelegt mit X'0C') gesetzt. Wird sie während des Programmablaufs verändert, muß sie vor dem nächsten Programmaufruf bzw. -rücksprung auf den Wert des PCD-Feldes "Programmaske" zurückgesetzt werden.

7.3.5 Parameterübergabe in ILCS-Programmsystemen

Die Semantik der Datentypen weist bei den durch ILCS verknüpfbaren Programmiersprachen starke Unterschiede auf. Im folgenden werden jene Datentypen aufgeführt, die in den einzelnen Programmiersprachen eine gleiche Datendarstellung besitzen und daher problemlos als Parameter übergeben werden können. Bei der Verwendung anderer Datentypen als Parameter ist die genaue Kenntnis der jeweiligen Datenablage erforderlich, um den korrekten Programmablauf sicherzustellen.

Compiler	Datentypen			
	Binär Wort	Gleitpunkt Wort	Gleitpunkt Doppelwort	String
COBOL85	PIC S9(i) COMP SYNCHRONIZED 5<=i<=9	COMP-1	COMP-2	USAGE DISPLAY
FOR1	INT*4	REAL*4	REAL*8	CHAR*i
Pascal-XT	long_integer	short_real	long_real	packed array [<range>]of char
PLI1	BIN FIXED(31) ALIGNED	BIN FLOAT(21) DEC FLOAT(6)	BIN FLOAT(53) DEC FLOAT(16)	CHAR(i)
C	long	float	double	char <var> [<size>]
ASSEMBH (@-Makros)	F	E	D	C
RPG3	binäres Feld mit 0 Dezimal-stellen	——	——	alphanum. Feld (feste Länge)

Die Daten müssen immer ausgerichtet abgelegt werden; d.h. 32-Bit-Ganzzahlen in binärer Darstellung liegen auf Wortgrenze, Gleitpunktzahlen auf Wort- bzw. Doppelwortgrenze, Strings auf Bytegrenze. Die Länge von Strings ist konstant und dem gerufenen Programm bekannt.

Die Übergabe erfolgt "by reference", d.h. es wird die Adresse des Datums übergeben. Das aufrufende Programm legt eine Liste der übergebenen Adressen an. Die Anzahl der Parameter wird in Register 0, die Adresse der Liste in Register 1 übergeben (siehe 7.3.1).

Übergabe von Funktions-Returnwerten

Returnwerte von Ganzzahl-Funktionen werden in den Registern 0 und 1, Returnwerte von Gleitpunkt-Funktionen werden im Gleitpunktregister 0 übergeben.
Die Übergabe von Returnwerten mit anderen Datentypen in Register 0 und 1 ist möglich, aber nicht durch ILCS festgelegt. Ihre Darstellung ist den einzelnen Programmiersprachen freigestellt.

7.3.6 Hinweise zum Binden von ILCS-Programmsystemen

Statisches Binden

Enthält ein Programmsystem ausschließlich strukturierte ASSEMBH-Programme, genügt - wie bisher - die Einbindung der ASSEMBH-Laufzeitbibliothek (SYSLIB.ASSEMBH.012) mit der RESOLVE-Anweisung des TSOSLNK.

Enthält ein Programmsystem verschiedensprachige Programme, muß die Initialisierungsroutine IT0INITS explizit aus der ILCS-Bibliothek eingebunden werden.

Dynamisches Binden

Programmsysteme mit ausschließlich gleichsprachigen Programmen können - wie bisher - mit Hilfe der TASKLIB-Zuweisung der Laufzeitbibliothek dynamisch gebunden werden.

Programmsysteme mit verschiedensprachigen Programmen können dynamisch gebunden werden, wenn der Benutzer dafür sorgt, daß die ILCS-Initialisierungsroutine IT0INITS in der mit TASKLIB zugewiesenen Laufzeitbibliothek steht.

Beim Dynamischen Binden mit dem DBL (ab BS2000-Version 10.0) kann im RUN-MODE=ADVANCED mit dem Linknamen BLSLIBnn als weitere zu durchsuchende Bibliothek die ILCS-Bibliothek zugewiesen werden.

Binden von Großmoduln

Werden Großmodule gebunden, so darf die ILCS-Routine IT0INITS nur in dem Großmodul eingebunden werden, der das Hauptprogramm enthält. Die Einsprungpunkte und Externverweise der Großmoduln müssen sichtbar bleiben.

7.4 Programmverknüpfungen von strukturierten Assembler-Programmen über die ILCS-Schnittstelle

Mit der Version V1.1A des ASSEMBH werden die Makros zur Strukturierten Programmierung (@-Makros) dahingehend erweitert, daß der Anwender auch in Assembler ILCS-fähige Programme schreiben kann.
Die Strukturierte Programmierung wird nicht vom ASSEMBH-BC unterstützt !

Die vereinheitlichte Prozedurschnittstelle bringt dem Anwender folgende Vorteile:

- die Assemblerprogrammteile können in einer beliebigen ILCS-fähigen Sprachumgebung verwendet werden
- die Parameterübergabe und die Registerkonventionen sind über alle Sprachen vereinheitlicht
- der Initialisierungsaufruf COLBIN des ASSEMBH-RTS kann entfallen
- die sprachspezifischen Makros (wie z. B. CENT, CEX und CCALL von C), die bisher die Programmierung in Assembler in der jeweiligen Sprachumgebung ermöglichten, können ersatzlos durch die @-Makros abgelöst werden, da sich das ILCS-fähige Assemblerprogramm in einer ILCS-Umgebung zu jeder Fremdsprache gleich verhält.

Neben der Umstellung der Prozedurschnittstelle auf die ILCS-Konventionen werden im ASSEMBH V1.1A durch neue @-Makros folgende neuen ILCS-Leistungen für den Assemblerprogrammierer angeboten (siehe "ASSEMBH", Beschreibung [1]).

- einheitliche Behandlung von Ereignissen (Event Handling)
- einheitliche Behandlung von Folgeprozessen (Contingency Handling)
- einheitliche Behandlung von STXIT-Ereignissen (STXIT Handling)
- Behandlung von Programmasken
- Setzen der Monitorjobvariablen
- Sprachinitialisierung bei nachgeladenen Moduln
- Anmeldung benutzereigener Routinen zur Speicherbeschaffung und Speicherfreigabe für Stack- und Heap-Speicher
- Anmeldung benutzereigener Beendigungsroutinen
- Angabe der minimalen Stack-Extent-Größe

Die neuen @-Makros generieren den Aufruf entsprechender Eingänge im Standard Event Handler (SEH), im Standard Contingency Handler (SCH), sowie im Standard STXIT Handler (SSH) der ILCS-Schnittstelle.

Damit können auf Prozedurebene Benutzerbehandlungsroutinen an- und abgemeldet werden. Der Aufruf dieser Prozeduren erfolgt nach ILCS-Konventionen.

7.4.1 Erstellen eines ASSEMBH-ILCS-Objektes

Die folgenden Voraussetzungen müssen erfüllt sein, wenn mit dem ASSEMBH ein ablauffähiges ILCS-Objekt mit Anschluss an das ASSEMBH-RTS (ASSEMBH-Laufzeitsystem) erzeugt werden soll:

- Falls das Objekt eine Assembler-Hauptprozedur (TYP=M) enthält, muss diese mit dem @ENTR-Parameter ILCS=YES übersetzt werden, damit die ILCS-Umgebung eingerichtet und das ASSEMBH-RTS sowie die Laufzeitsysteme aller weiteren beteiligten Sprachen initialisiert werden.
- Wird das ASSEMBH-ILCS-Objekt von einer Fremdsprachen-Prozedur gerufen, so muss diese die ILCS-Umgebung und damit das ASSEMBH-RTS initialisiert haben.
- Prozeduren vom Typ E/I müssen mit dem Parameter ILCS=YES übersetzt werden.
- Die ASSEMBH-Laufzeitbibliothek muß eingebunden werden (siehe 7.3.6).

7.5 ILCS-Linkage-Kombinationen

7.5.1 ILCS-Objekt ruft ASSEMBH-ILCS-Objekt

Rufende Prozedur
Die ILCS-Umgebung ist von der ILCS-Fremdprozedur initialisiert worden. Damit ist auch die sprachspezifische Initialisierung des ASSEMBH-Laufzeitsystems (ASSEMBH-RTS) durchgeführt.

Der bisherige COLBIN-Aufruf in der Fremdprozedur kann entfallen.

Gerufene Prozedur
Prozedurprolog:
Durch den Makro @ENTR mit ILCS=YES. Die Parameter werden im STANDARD-Format durch "call by reference" erwartet.

Prozedurepilog:
War der @ENTR-Parameter RETURNS=YES gesetzt, wird der Funktionswert von R1 nach R0 kopiert. Die Register R2 bis R14 werden restauriert.

7.5.2 ASSEMBH-ILCS-Objekt ruft ASSEMBH-ILCS-Objekt

Rufende Prozedur
Die Initialisierung der ILCS-Umgebung und des ASSEMBH-RTS ist in der Hauptprozedur (Typ M) durch @ENTR mit ILCS=YES erfolgt. Die Parameterübergabe erfolgt im STANDARD-Format durch "call by reference".

Gerufene Prozedur
Prozedurprolog:
Durch den Makro @ENTR mit ILCS=YES. Die Parameter werden im STANDARD-Format durch "call by reference" erwartet.

Prozedurepilog:
War der @ENTR-Parameter RETURNS=YES gesetzt, wird der Funktionswert von R1 nach R0 kopiert. Die Register R2 bis R14 werden restauriert.

7.5.3 ASSEMBH-ILCS-Objekt ruft Nicht-ILCS-ASSEMBH-Objekt

Rufende Prozedur
Die Initialisierung der ILCS-Umgebung und des ASSEMBH-RTS ist in der Hauptprozedur (Typ M) durch @ENTR mit ILCS=YES erfolgt. Die Parameterübergabe erfolgt im STANDARD-Format durch "call by reference".

Gerufene Prozedur
Prozedurprolog:
Durch den Makro @ENTR mit ILCS=NO (Voreinstellung). Die Parameter werden im STANDARD-Format durch "call by reference" erwartet.

Prozedurepilog:
Der Funktionswert wird nicht von R1 nach R0 kopiert.

Beachte
Gerufene Prozedur:
- darf kein @EXIT mit TO-Operanden enthalten
- in R0 wird die Anzahl der Parameter übergeben

Rufende Prozedur:
- darf sich nicht darauf verlassen, daß von der gerufenen Prozedur die ILCS-Konventionen eingehalten werden, z.B. wird der Funktionswert nicht von R1 nach R0 kopiert.

7.5.4 Nicht-ILCS-ASSEMBH-Objekt ruft ASSEMBH-ILCS-Objekt

Rufende Prozedur
Die Initialisierung der ILCS-Umgebung und des ASSEMBH-RTS ist in der Nicht-ILCS-Hauptprozedur (Typ M) durch @ENTR mit ILCS=NO erfolgt. Die Parameterübergabe erfolgt im STANDARD-Format durch "call by reference".

Gerufene Prozedur
Prozedurprolog:
Durch den Makro @ENTR mit ILCS=YES. R0 enthält i.a. nicht die Anzahl der Parameter. Die Parameter werden im STANDARD-Format durch "call by reference" erwartet.

Prozedurepilog:
War der @ENTR-Parameter RETURNS=YES gesetzt, wird der Funktionswert von R1 nach R0 kopiert. Die Register R2-R14 werden restauriert.

Beachte
Gerufene Prozedur:
– darf in R0 nicht die Anzahl der Parameter erwarten

7.5.5 Nicht-ILCS-Objekt ruft ASSEMBH-ILCS-Objekt

Rufende Prozedur
Der COLBIN-Aufruf zur Initialisierung der ILCS-Umgebung ist notwendig.
Die Parameterübergabe erfolgt im STANDARD-Format durch "call by reference".

Gerufene Prozedur wie 7.5.4

Beachte
Gerufene Prozedur:
Im Makro @EXIT darf PROG=FORTRAN nicht angegeben werden.

7.5.6 Long-Jump (@EXIT mit Parameter TO)

Enthält ein Programm ILCS-Objekte und Nicht-ILCS-ASSEMBH-Objekte, so ist vom Anwender sicherzustellen, dass innerhalb der Nicht-ILCS-ASSEMBH-Objekte an keiner Stelle ein Long-Jump vorgesehen ist. Andernfalls sind Programmfehler möglich!

8 ASSEMBH-Diagnoseprogramm ASSDIAG

Wird vom ASSEMBH-BC nicht unterstützt !

8.1 Anwendung

Das Diagnoseprogramm ASSDIAG wird vom ASSEMBH im Dialogbetrieb im Rahmen des Assembler-Korrektur-Zyklus aufgerufen und erfüllt folgende Grundfunktionen:

- SDF- (bzw. COMOPT-) gesteuerter impliziter Start durch den ASSEMBH, wenn ein bestimmtes Fehlergewicht erreicht wurde.
- Ausgeben von Diagnoseinformation über eine vorausgegangene Assemblierung.
- Formatierte Ein-Ausgabe im Dialogbetrieb.
- Möglichkeit der Quellprogrammkorrektur im Dialogbetrieb mit Hilfe des Dateibearbeiters EDT.
- Restart des ASSEMBH mit den eingestellten Optionen.

Der ASSDIAG wird vom ASSEMBH aus der Bibliothek
<userid>.SYSLNK.ASSEMBH.011 geladen und gestartet.

Dieser Start wird vorbereitet durch die Angabe folgender Option (siehe 2.4.7):

CORRECTION-CYCLE=YES(ACTIVATION-WEIGHT=<fehlergewicht>)
(im COMOPT-Fall durch *COMOPT ADIAG=n)

und wirkt, wenn bei der Übersetzung ein entsprechendes Fehlergewicht festgestellt wurde.
Bei der Angabe CORRECTION-CYCLE=YES(ACTIVATION-WEIGHT=ALWAYS)
(bzw. *COMOPT ADIAG=0) wird der ASSDIAG unabhängig von aufgetretenen Fehlern gestartet.

Softwarevoraussetzung

Das Diagnoseprogramm arbeitet beim Korrigieren von Quellzeilen mit dem Dateibearbeiter EDT, die dafür erforderliche Version ist der Freigabemitteilung zu entnehmen.

8.2 Begriffe

Diagnosedatei

Eine vom ASSEMBH erzeugte temporäre ISAM-Datei, die bei Beendigung des ASSDIAG wieder gelöscht wird.

ASSDIAG-Kommando

Anweisungen an das Diagnoseprogramm bestimmte Leistungen zu erbringen.

Fehlerklasse

Jeder vom ASSEMBH erkannte Fehler bei der Übersetzung eines Programmes wird einer der nachfolgend beschriebenenen Fehlerklassen zugeordnet:

Code	Beschreibung	Fehlerklassen	
		SDF	COMOPT
NOT	NOTE - Hinweis Erfolgreicher Programmlauf möglich	NOTE	-
WAR	WARNING - Warnung Erfolgreicher Programmlauf möglich	WARNING	1
SIG	SIGNIFICANT ERROR - Fehler Fehlerhafter Programmlauf möglich	SIGNIFICANT	2
SER	SERIOUS ERROR - Schwerer Fehler Programmlauf nicht möglich	SERIOUS	3
FAT	FATAL ERROR - Abbruchfehler Assemblierung wird abgebrochen; Diagnosedatei ist unvollständig	-	-
FAL	FAILURE - Assemblerfehler Assemblierung wird abgebrochen; Diagnosedatei ist unvollständig	-	-
MNO	MNOTE - Meldung Durch variablen Parameter erzeugte Meldung mit deren MNOTE-Nummer eine entsprechende Fehlerklasse erzeugt werden kann. (Reaktionen dabei wie oben beschrieben)	-	-

Flagtyp

Einer der Buchstaben A-Z mit dem eine Flagkennung beginnt.

Flagkennung

Flagtyp und nachfolgend ein bis zwei Ziffern, mit denen ein Fehler durch ein Flag gekennzeichnet wird.

Meldungstext

Verbale Beschreibung einer Flagkennung.

8.3 Start des Diagnoseprogrammes

Die Option CORRECTION-CYCLE=YES(ACTIVATION-WEIGHT=<gewicht>) (bzw. die *COMOPT-Angabe ADIAG=n) ermöglicht es dem Benutzer das Diagnoseprogramm ASSDIAG abhängig von der höchsten bei der Übersetzung aufgetretenen Fehlerklasse am Ende einer Übersetzungseinheit starten zu lassen.

Die Werte <gewicht> bzw. <n> haben folgende Bedeutung:

<gewicht>	<n>	Start des ASSDIAG:
ALWAYS	0	nach einer Übersetzungseinheit, unabhängig vom Übersetzungsergebnis
NOTE	-	beim Auftreten der Fehlerklasse NOTE oder größer
WARNING	1	beim Auftreten der Fehlerklasse WAR oder größer
SIGNIFICANT	2	beim Auftreten der Fehlerklasse SIG oder größer
SERIOUS	3	beim Auftreten der Fehlerklasse SER

Nach dem Eröffnen der Diagnosedatei wertet der ASSDIAG die über das Assemblerprotokoll hinausgehende Information aus, gibt diese aus und erwartet die Eingabe von ASSDIAG-Kommandos.

Der Eröffnungsschirm enthält folgende Angaben:
- Name der Diagnosedatei
- Erstellungsdatum
- Name des Quellprogramms (Sourcemodul)
- Versionsnummer des Assemblers
- Anzahl der aufgetretenen Flags gegliedert nach Fehlergewicht

Angezeigt und weiter bearbeitbar sind nur die Flags, die durch die SDF-Angabe MIN-MESSAGE-WEIGHT= (bzw. *COMOPT-Angabe ERRPR=) erkennbar bleiben.

8.4 Unterbrechen des Programmablaufes

Wenn der Benutzer zur Eingabe eines ASSDIAG-Kommandos aufgefordert wird, kann durch Betätigen der BREAK-Taste der Datensichtstation in den Systemmodus verzweigt werden und BS2000-Kommandos können eingegeben werden (dies kann auch mit dem ASSDIAG-Kommando SYSTEM geschehen, siehe 8.5.9).
Durch Eingabe des RESUME-PROGRAM Kommandos wird der unterbrochene ASSDIAG-Lauf fortgesetzt.

8.5 ASSDIAG Kommandos

Die Steuerung des ASSDIAG erfolgt über Kommandos. Sie werden im Dialogbetrieb mit dem WRTRD-Makro gelesen und sind immer in der Kommandozeile der Bildschirmmaske einzugeben.
Die Kommandos können, soweit Eindeutigkeit gewährleistet ist, bis auf ein Zeichen abgekürzt werden.

Allgemeines Format

```
<kommandoname> [<parameter>[,<parameter>...]]
```

Blätterfunktion

Erstreckt sich die Ausgabe über mehrere Bildschirme, so kann durch eine leere Eingabe zum nächsten Schirm weitergeschaltet werden.

Übersicht über die ASSDIAG Kommandos

Kommando	Funktion
CDT	Aufruf des Dateibearbeiters EDT zur Korrektur von Quellzeilen
CONTINUE-CDT	Fortsetzen der Korrekturbearbeitung an der Unterbrechungsstelle
DISPLAY	Ausgabe von Fehlerursachen und Nummern der betroffenen Statements auf der Datensichtstation
END	Beenden des Diagnoseprogrammes
HELP	Auflisten und Erläutern von ASSDIAG-Kommandos
LIST	Wie Kommando DISPLAY, jedoch Ausgabe nach SYSLST
PRINT	Auflistung von Statements
RERUN	Start des Assemblers mit den gültigen COMOPT’s bzw.SDF-Options
SYSTEM	Ausführung eines System-Kommandos
TAGS	Auflistung aller Symbole, die undefiniert bzw. mehrfach definiert sind
XREF	Ausgabe von Querverweisen

8.5.1 Kommando CDT

Format

```
C[DT] [<parameter>]
```

Parameter

```
                ⌈ALL            ⌉
                |SO[URCE]       |
                |<fehlerklasse> |
<parameter>::={                 }
                |<statementnr.> |
                |<flagkennung>  |
                ⌊<flagtyp>      ⌋
```

Funktion

Der Dateibearbeiter EDT wird als Unterprogramm gestartet. Die Quellprogrammdatei oder das Source-Element einer PLAM-Bibliothek wird eröffnet und zur Korrektur angeboten.

1. Parameter ALL (Voreinstellung)

   Die Fehlerinformation wird für alle aufgetretenen Fehler hinter dem fehlerverursachenden Quellstatement in der EDT-Arbeitsdatei mit den Attributen ’halbhell’ und ’schreibgeschützt’ eingemischt. Die Information besteht aus Flagkennung, Fehlergewicht, Meldungsnummer und Meldungstext.
   Die fehlerhafte Quellzeile selbst wird ’hell’ und ’überschreibbar’ eingestellt. Das EDT-Fenster wird auf die erste Fehlerzeile - 2 Zeilen positioniert, um etwas von der Fehlerumgebung aufzuzeigen.
   Bei Fehlern in Makro- und COPY-Elementen wird der äußerste Aufruf in der Quelle als fehlerhaft gekennzeichnet. Dabei wird die Fehlerinformation durch die fehlerhafte generierte Zeile ergänzt. Sie ist für nachfolgende Aktionen mit der Statementnummer versehen.

2. Parameter SOURCE

   Die Source wird in der EDT-Arbeitsdatei ohne Fehlerinformation zur Korrektur angeboten und kann mit den EDT-Möglichkeiten bearbeitet werden. Es obliegt dem Benutzer fehlerhafte Quellzeilen zu suchen und zu korrigieren.
   Eine Bearbeitung von Bibliothekselementen (Makro, COPY) ist nicht möglich.

3. Parameter < >

   Wie unter 1. beschrieben, jedoch nur für den spezifizierten Fehlerumfang.

CDT-Kommandobearbeitung

Blätterfunktion:

Bei den Funktionen 1. und 3. kann durch Betätigen der <K1>-Taste das EDT-Fenster auf die nächste fehlerhafte Quellzeile positioniert werden.

Beenden der CDT-Bearbeitung:

Für das Beenden des EDT und das Sichern der bearbeiteten Assemblerquellen (Source/Makro/Copy) gibt es mehrere Möglichkeiten:

1. EDT-Kommando RETURN oder
   Abschluß der Blätterfunktion durch letzte <K1>-Taste

   Die geöffneten Assemblerquellen werden zurückgeschrieben und die EDT-Bearbeitung wird beendet. Bei PLAM-Elementen wird die Versionsbezeichnung nicht verändert.

2. EDT-Kommando HALT

   Der EDT wird beendet. Die bearbeiteten Elemente werden nicht zurückgeschrieben.

3. Durch die ASSDIAG-Kommandos RERUN oder END

Unterbrechen:

1. ASSDIAG-Kommando außer CDT
2. CDT-Anweisung

ASSDIAG-Kommandoeingabe direkt im CDT-Korrekturschirm

Durch die Eingabe von ASSDIAG-Kommandos während der CDT-Korrekturbearbeitung mit dem EDT wird diese unterbrochen und die entsprechende ASSDIAG-Leistung aufgerufen. Eine Fortsetzung der unterbrochenen CDT-Korrekturbearbeitung an der bisher erreichten Stelle ist nur durch das parameterlose ASSDIAG-Kommando C[ONTINUE]-C[DT] möglich. Das ASSDIAG-Kommando CDT aus dem CDT-Korrekturschirm heraus ist nicht erlaubt und wird zurückgewiesen. Nach einer unterbrochenen Korrekturbearbeitung ist nur das Kommando C-C erlaubt.
Die Kommandos LIST und SYSTEM sind aus dem Korrekturschirm im Sinne des EDT erlaubt.

Besonderheiten bei ASSDIAG-Kommandos aus CDT-Korrekturschirm:

Abkürzungen für ASSDIAG-Kommandos aus dem CDT-Korrekturschirm unterscheiden sich von den sonst gültigen minimalen Abkürzungen. Siehe zweite Zeile in der Formatbeschreibung der Kommandos (8.5.3 ff).

– END-Kommando

Alle zur Korrektur geöffneten Dateien/Bibliothekselemente werden, nach Anfrage und positiver Antwort, auf den Datenträger zurückgeschrieben und geschlossen. Sonst werden die Programmteile unverändert geschlossen. Der Compile-Korrektur-Zyklus wird beendet. Das Übersetzungsprotokoll wird bei angegebenen Parameter L ausgegeben. Das Kommando darf aus Gründen der Eindeutigkeit innerhalb des Korrekturmodus nicht abgekürzt eingegeben werden.

– RERUN-Kommando

Alle zur Korrektur geöffneten Dateien/Bibliothekselemente werden ohne Anfrage auf den Datenträger zurückgeschrieben und geschlossen. Der Compile-Korrektur-Zyklus mit den geänderten Programmteilen wird erneut angestoßen.

8.5.1.1 CDT-Anweisungen

Durch CDT-Anweisungen ist es möglich, zusätzliche Bearbeitungsschritte an beteiligten Assemblerprogramm-Elementen in die laufende Source-Korrektur einzuschieben. Diese Anweisungen sind nur während der CDT-Korrekturbearbeitung erlaubt und führen außerhalb des CDT-Kommandos zu einer Fehlermeldung. Die Anweisungen (1.-4.) sind vom CDT-Korrekturschirm heraus in der Kommandozeile angebbar.

Beim Öffnen des Elements wird im zweiten Fenster folgende Information bereitgestellt (gilt für die Anweisungen 2.-4.):

– Bibliotheksname des geöffneten Elements
– Elementname des geöffneten Elements
– Version und Typ des geöffneten Elements
– Elementname des zurückgeschriebenen Elements

CDT-Anweisungsbearbeitung

Blättern: bei SHOW-DEF in der untersten Kommandozeile mit EDT-Blätterfunktion (+/-)
Unterbrechung beenden: <K1>-Taste
Rückkehrpunkt: letzte CDT-Kommandobearbeitung
Beenden: RETURN, RERUN, END, HALT

CDT-Kommandobearbeitung bzw. Anweisungsbearbeitung unterbrechen: Eingabe eines ASSDIAG-Kommandos
Unterbrechung beenden: CONTINUE-CDT
Rückkehrpunkt: letzte CDT-Kommandobearbeitung

1. Einblenden der Definitionszeile[n] in die Quelle

Format

```
S[HOW]-D[EFINITION] <name>
```

Funktion

Durch diese Anweisung werden alle Definitionszeilen aus der Liste mit dem Symbol <name> in der EDT-Arbeitsdatei abgelegt. Die EDT-Arbeitsdatei wird als zweites Fenster unten auf dem Bildschirm abgebildet. Die eingemischten Listen-Zeilen werden komprimiert dargestellt, bestehend aus:
Flagspalte, Location-Counter, Statementnummer, Makro-/COPY-Level und Source-Statement.
EDT-Anweisungen zum Blättern innerhalb der EDT-Arbeitsdatei (+, -, ++, --) sind in der Eingabezeile der EDT-Arbeitsdatei einzugeben, andere CDT-Anweisungen sind in der Eingabezeile des oberen Fensters einzugeben (z.B. EDIT-DEFINITION).

2. Korrigieren einer Definitionszeile

Format

```
E[DIT]-D[EFINITION] <name>
```

Funktion

Diese Anweisung positioniert das EDT-Fenster auf die erste Definitionszeile des angegebenen Symbols im entsprechenden Quellelement (Source/Makro/COPY). Befindet sich die Definitionszeile in einem Makro-/COPY-Element wird das Element, sofern nicht schon geladen, in eine freie EDT-Arbeitsdatei eingelesen.

<name> Bezeichnet das erste Auftreten der Symboldefinition <name> in der Quelle. Soll eine Definitionszeile bearbeitet werden, die nicht erste Definitionszeile des Symbols <name> ist, so kann dies über die CDT-Anweisungen SHOW-DEFINITION <name> und EDIT <statementnr.> erreicht werden.

Nach EDIT-DEFINITION wird durch die <K1>-Taste die Bearbeitung fortgesetzt.

3. Bearbeiten eines Source-Makro-/COPY-Statements mit eingeblendeter Fehlerinformation

Format

```
E[DIT] <statementnummer>
```

Funktion

Statementnummer im Source:
Befindet sich die angegebene Statementnummer im Sourceteil der Übersetzungseinheit, so wird die Arbeitsdatei auf die entsprechende Sourcezeile positioniert. Durch die <K1>-Taste wird zum folgenden fehlerhaften Statement weiterpositioniert bzw. die Korrekturbearbeitung wird beendet.

Statementnummer im Makro-/COPY-Element:
In der eingeblendeten Fehlerinformation zum generierten Statement beim Makroaufruf wird die Statementnummer mit ausgegeben. Mit dieser Information kann die Beziehung zum Quellelement und der Zeilennummer hergestellt werden. Durch die Anweisung wird das entsprechende Makro-/COPY-Element in eine freie Arbeitsdatei eingelesen und auf die entsprechende Quellzeile positioniert. Mit Hilfe der eingeblendeten Fehlerinformation kann der Anwender die Zeile korrigieren. Mit der Funktionstaste <K1> wird die Bearbeitung an der Position fortgesetzt, die vor der EDIT-Anweisung erreicht war.

4. Bearbeiten eines Makro-/COPY-Elementes ohne eingeblendete Fehlerinformation

Format

```
        ⎧ M[ACRO] ⎫
E[DIT]-⎨          ⎬ <name>
        ⎩ C[OPY]  ⎭
```

Funktion

Das Element wird ohne eingeblendete Fehlerinformation bearbeitet. Das EDT-Fenster wird auf den Elementanfang positioniert. Die Zuweisung der EDT-Arbeitsdatei erfolgt analog zur Verarbeitung mit eingeblendeter Fehlerinformation. Die Rückkehr zur weiteren Bearbeitung nach der Elementkorrektur erfolgt mit der <K1>-Taste. Gleichnamige Makros werden nicht unterstützt.

8.5.2 Kommando CONTINUE-CDT

Format

```
C[ONTINUE]-C[DT]
```

Funktion

Eine durch eine ASSDIAG-Kommandoeingabe unterbrochene CDT-Korrekturbearbeitung wird an der Unterbrechungsstelle wieder fortgesetzt.
Wurde keine CDT-Korrekturbearbeitung unterbrochen, so wird das Kommando als nicht erlaubt zurückgewiesen.

8.5.3 Kommando DISPLAY

Format

```
⎧D[ISPLAY]⎫
⎨         ⎬ [<parameter>]
⎩DIS[PLAY]⎭
```

In der zweiten Zeile steht die Minimalabkürzung für die Kommandoeingabe aus dem CDT-Korrekturschirm (8.5.1).

Parameter

```
                ⎧NOT                                  ⎫
                ⎪WAR                                  ⎪
                ⎪SIG                                  ⎪
                ⎪SER                                  ⎪
<parameter>::=  ⎨FAT                                  ⎬
                ⎪FAL                                  ⎪
                ⎪MNO                                  ⎪
                ⎪<statementnr.> [-<statementnr.>]     ⎪
                ⎪<flagtyp>                            ⎪
                ⎩<flagkennung>                        ⎭
```

Funktion

Ausgabe der ausgewählten Fehlerursachen auf der Datensichtstation unter der Voraussetzung, daß die Assemblierung nicht abgebrochen wurde (siehe Sonderfall).

- Nichtparametrisierte Anweisung:
  Auflistung aller erkannten Flags und MNOTES geordnet nach Fehlerklasse, Flagkennung mit Meldungstext und Verweisen auf die betroffenen Statementnummern.
- NOT:
  Wie oben, jedoch nur die Fehlerklasse NOTE.
- WAR:
  Wie oben, jedoch nur die Fehlerklasse WARNING.
- SIG:
  Wie oben, jedoch nur die Fehlerklasse SIGNIFICANT ERROR.
- SER:
  Wie oben, jedoch nur die Fehlerklasse SERIOUS ERROR.
- FAT:
  Wie oben, jedoch nur die Fehlerklasse FATAL ERROR.
- FAL:
  Wie oben, jedoch nur die Fehlerklasse FAILURE.

– MNO:
  Auflistung aller MNOTES mit zugeordnetem Severity-Code und Text, gefolgt von der Statementnummer.

– <statementnr.> [-<statementnr.>]
  Das fehlerhafte Statement bzw. die fehlerhaften Statements (keine MNOTES), die im angegebenen Nummernbereich auftraten, werden mit jeweils nachfolgender Flagkennung und dem Meldungstext auf Datensichtstation ausgegeben.

– <flagtyp>
  Auflistung der Fehler eines Flagtyps, gefolgt von genauer Flagkennung und Meldungstext und den betroffenen Statementnummern.

– <flagkennung>
  Wie oben, jedoch nur die spezifizierte Flagkennung.

Blätterfunktion
Erstreckt sich die Ausgabe über mehrere Bildschirme, so kann durch eine leere Eingabe zum nächsten Schirm weitergeschaltet werden.

Sonderfall: Abbruch der Assemblierung

Wird die Assemblierung abgebrochen, so kann keine vollständige Diagnosedatei erzeugt werden. Abhängig von der Abbruchursache sind dann 2 Ausgaben möglich:

1. Abbruch bei Fehlergewicht SERIOUS (d.h. keine Fortsetzung des Assemblerlaufes möglich):

   Es wird der Meldungstext ausgegeben mit Hinweisen auf die möglichen Ursachen. In diesem Fall werden sämtliche Parameter in der DISPLAY-Anweisung ignoriert.

2. Bei einem gesteuerten Abbruch durch Angabe eines höchsten Fehlergewichtes:

   Es werden sämtliche bisher fehlerhaften Statements ausgegeben. Mindestens aber das Statement, welches den Abbruch hervorgerufen hat (z.B. MNOTE mit SEVCODE=255).
   In diesem Fall werden DISPLAY-Parameter eingeschränkt akzeptiert (Statement-Referenzen sind hier immer ausgeschlossen).

8.5.4 Kommando END

Format

```
⎧E[ND]⎫
⎨     ⎬ [L]
⎩END  ⎭
```

In der zweiten Zeile steht die Schreibweise für die Kommandoeingabe aus dem CDT-Korrekturschirm (8.5.1).

Funktion

Das Diagnoseprogramm wird beendet und es wird zum ASSEMBH zurückgekehrt. Mit dem Zusatz 'L' wird das Übersetzungsprotokoll entsprechend den über SDF-Steuerung (bzw. *COMOPT) angegebenen Options ausgegeben. Die eröffneten Dateien werden geschlossen.
Eine weitere Übersetzungseinheit in derselben Quelle wird nicht mehr bearbeitet.

8.5.5 Kommando HELP

Format

```
⎧H[ELP]⎫
⎨      ⎬ [<kommando>]
⎩HEL[P]⎭
```

In der zweiten Zeile steht die Minimalabkürzung für die Kommandoeingabe aus dem CDT-Korrekturschirm (8.5.1).

Funktion

Auflistung aller ASSDIAG-Kommandos bzw. Beschreibung ausgewählter ASSDIAG-Kommandos auf Datensichtstation.

8.5.6 Kommando LIST

Format

```
L[IST] [<parameter>]
```

Parameter

siehe 8.5.3, Kommando DISPLAY

Funktion

Entsprechend Kommando DISPLAY jedoch Ausgabe auf SYSLST.

8.5.7 Kommando PRINT

Format

```
         ⎡<statementnr.>[-<statementnr.>]⎤
P[RINT] [⎨                               ⎬][,L][,S]
         ⎣<symbol>[-<symbol>]            ⎦
```

Die Minimalabkürzung ist auch für die Kommandoeingabe aus dem CDT-Korrekturschirm gültig (8.5.1).

Funktion

Auflisten einer bestimmten Anweisung oder eines bestimmten Bereiches von Anweisungen, so wie sie im Assemblerprotokoll erscheinen würden. Bei fehlender Bereichsangabe werden alle Statements ausgegeben. Dabei werden jedoch auf Datensichtstation die über 80 Spalten hinausgehenden rechten Zeichen einer Zeile abgeschnitten.

Eine Angabe ’S’ bewirkt, daß nur folgende Information einer Zeile ausgegeben wird:
- Location counter
- Statementnummer
- und Source Statement (ggf. mit zugehöriger Meldungstextzeile)

Eine Angabe ’L’ bewirkt eine zusätzliche, vollständige Ausgabe der Druckzeilen nach SYSLST. Die Flagkennungen und Meldungstexte sind hinter den betroffenen Statements eingefügt.

Blätterfunktion
Erstreckt sich die Ausgabe über mehrere Bildschirme, so kann durch eine leere Eingabe zum nächsten Schirm weitergeschaltet werden.

Anmerkung

Bei Abbruch der Assemblierung u.U. nicht möglich.

8.5.8 Kommando RERUN

Format

```
R[ERUN]
RER[UN]
```

In der zweiten Zeile steht die Minimalabkürzung für die Kommandoeingabe aus dem CDT-Korrekturschirm (8.5.1).

Funktion

Der ASSDIAG wird beendet und der ASSEMBH wird veranlaßt, die betroffene Übersetzungseinheit erneut zu übersetzen, mit den Optionen, die bei der vorhergehenden Übersetzung eingestellt waren.

Hinweise

- Das Kommando RERUN wird zurückgewiesen, wenn die Quelle über SYSDTA gelesen wurde.
- Nach einem Abbruch der Assemblierung ist ein RERUN nicht möglich.
- Ein Correction-Cycle mit Modulausgabe nach *OMF ist nicht sinnvoll, da bei einem RERUN der erzeugte Modul einen im *OMF vorhandenen Modul gleichen Namens nicht überschreibt.

8.5.9 Kommando SYSTEM

Format

```
S[YSTEM]<parameter>
```

Parameter

```
<parameter>::='Systemkommando'
```

Funktion

Das in Hochkomma eingeschlossene Systemkommando kann mit oder ohne Schrägstrich angegeben werden. Es wird sofort ausgeführt und nach der Ausführung wird mit dem ASSDIAG fortgefahren, sofern dies das zuvor gegebene Systemkommando erlaubt.

Es sind alle Kommandos zugelassen, die über den CMD-Makro (siehe Handbuch: 'Makroaufrufe an den Ablaufteil') aufgerufen werden können.

Anmerkung

Während der Kommandoausführung bleibt der ASSDIAG geladen und die geöffneten Dateien werden nicht geschlossen.

8.5.10 Kommando TAGS

Format

```
⎧T[AGS]⎫
⎨      ⎬ [<art>[,<art>]][,X[REF]
⎩TAG[S]⎭
```

In der zweiten Zeile steht die Minimalabkürzung für die Kommandoeingabe aus dem CDT-Korrekturschirm (8.5.1).

Parameter

```
<art>::=⎧M⎫
        ⎨ ⎬
        ⎩U⎭
```

Voreinstellung: M,U

Funktion

Anzeige aller undefinierten (U) und/oder mehrfach definierten (M) Symbole. Bei Angabe von XREF mit den Querverweisen.

Blätterfunktion
Erstreckt sich die Ausgabe über mehrere Bildschirme, so kann durch eine leere Eingabe zum nächsten Schirm weitergeschaltet werden.

Anmerkung

Die Funktion ist bei Abbruch der Assemblierung u.U. nicht möglich.

8.5.11 Kommando XREF

Format

```
X[REF] <parameter>[,<parameter>[,<parameter>]]
```

Die Minimalabkürzung ist auch für die Kommandoeingabe aus dem CDT-Korrekturschirm gültig (8.5.1).

Parameter

```
                ⎡<symbol>[-<spezifikation>]⎤
<parameter>::=⎨ *<makroname>               ⎬
                ⎣<literal>                 ⎦

                      ⎡ ⎤
                      ⎢A⎥
                      ⎢R⎥
<spezifikation>::=⎨   ⎬
                      ⎢W⎥
                      ⎢E⎥
                      ⎣O⎦
```

Funktion

Die Querverweise zu den angegebenen Symbolen, Makronamen oder Literalen werden auf Datensichtstation angezeigt.
Sofern bei der Übersetzung der Wunsch nach Referenzen mit Attributen angegeben wurde, kann bei Symbolen eine bestimmte Auswahl von Verweisen verlangt werden:

A: Adreßzugriffe
R: Nicht schreibende Zugriffe durch Befehle
W: Schreibende Zugriffe
E: Symbol von EQU-/ORG-Anweisung
O: Sonstige Zugriffe durch Assembler-Anweisungen

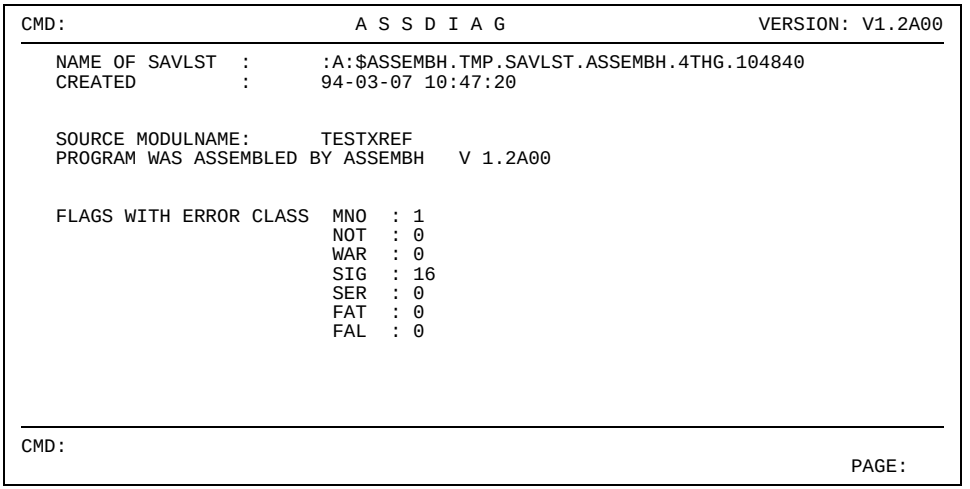
Blätterfunktion
Erstreckt sich die Ausgabe über mehrere Bildschirme, so kann durch eine leere Eingabe zum nächsten Schirm weitergeschaltet werden.

Anmerkung

Bei Abbruch der Assemblierung u.U. nicht möglich bzw. wenn bei der Übersetzung kein XREF gewünscht war.

8.6 Formatierte Bildschirm-E/A

8.6.1 Prinzipieller Aufbau der ASSDIAG-Formate



8.6.2 Beispiel: Kommando DISPLAY

```
CMD: DISPLAY                A S S D I A G                    VERSION: V1.2A00

CLASS FLAG  MESSAGE AND STATEMENT NUMBERS
FAL          NONE
FAT          NONE
SER          NONE
SIG   B42   ASS0242 'COPY' MEMBER NOT FOUND
            000004
      D7    ASS0407 ALIGNMENT ERROR IN OPERAND
            000011
      U10   ASS2110 SYMBOL   IS UNDEFINED
            000006  000007  000009  000015  000016  000017  000018  000018
            000046  000047  000048  000049  000072  000073
WAR          NONE
NOT          NONE
MNO          MNOTE WITH SEVERITY CODE 0152
            000013


CMD:
 END OF OUTPUT                                                   PAGE: 1
```

8.6.3 Beispiel: Kommando TAGS

```
CMD: TAGS                    A S S D I A G                    VERSION: V1.2A00

UNDEFND SYMBOL
 R14
 R15
 R5
 R6
 R7
 R8
 R9
 UNDEF

CMD:
 END OF OUTPUT                                                      PAGE: 1
```

9 Die Dialogtesthilfe AID

9.1 Einführung

Auch ein syntaktisch richtiges Assembler-Programm enthält möglicherweise noch logische Fehler und läuft daher nicht in der gewünschten Weise ab. Für das Auffinden und Beseitigen solcher Fehler steht dem Assembler-Programmierer die Dialogtesthilfe AID (**A**dvanced **I**nteractive **D**ebugger) zur Verfügung. Sie erfordert keine Vorkehrungen bei der Programmierung und erlaubt es, im geladenen Programm während dessen Ausführung Fehler zu suchen und korrigierend in den Ablauf einzugreifen.

In diesem Benutzerhandbuch soll AID nur kurz vorgestellt werden. Die ausführliche Beschreibung dieser Testhilfe finden Sie im Handbuch "AID, Testen von ASSEMBH-Programmen" [2].

AID zeichnet sich durch folgende Leistungsmerkmale aus:

1. AID bietet die Möglichkeit, "symbolisch" zu testen. D.h., in den Kommandos können anstelle sedezimaler Adressen symbolische Namen aus dem Quellprogramm angegeben werden, wenn die dafür nötigen LSD-Informationen beim Übersetzen erzeugt und später an das geladene Programm weitergegeben wurden (siehe 9.2).

   Dabei ist es nicht erforderlich, diese Informationen immer für das Gesamtprogramm zusammen mit diesem Programm zu laden. AID erlaubt ein Nachladen der LSD-Informationen für jede Übersetzungseinheit, falls die zugehörigen Moduln (mit den LSD-Informationen) in einer PLAM-Bibliothek stehen. Dadurch lassen sich Betriebsmittel wirtschaftlicher einsetzen:

   – Der Programmspeicher wird entlastet, da LSD-Informationen nur dann geladen werden müssen, wenn sie zum Testen benötigt werden (der Speicherbedarf für ein Programm steigt durch das Mitladen der LSD-Information ca. auf das Fünffache).

   – Ein Programm, das im Test fehlerfrei bleibt, muß für den Produktiveinsatz nicht unbedingt neu (ohne LSD-Information) übersetzt und gebunden werden.

   – Falls sich für ein Programm während des Produktiveinsatzes ein Test als nötig erweist, stehen dafür LSD-Informationen zur Verfügung, ohne daß das Programm erneut übersetzt und gebunden werden muß.

2. AID stellt Funktionen zur Verfügung, die es gestatten,

   – den Programmablauf an festgelegten Stellen oder beim Eintreten definierter Ereignisse zu unterbrechen, um AID- oder BS2000-Kommandos (Subkommandos) ausführen zu lassen,

   – sich die Inhalte von Feldern entsprechend der Datendefinition im Quellprogramm ausgeben zu lassen,

   – die Inhalte von Feldern zu verändern.

3. AID unterstützt neben der Diagnose geladener Programme auch die Analyse von Speicherabzügen in Plattendateien.

9.2 Voraussetzungen für das symbolische Testen

Beim Testen auf symbolischer Ebene erlaubt es AID, Namen anzusprechen, die im Quellprogramm definiert sind und sich auf Quellprogrammzeilen beziehen. Dafür müssen AID Informationen über die symbolischen Namen zur Verfügung gestellt werden. Diese Informationen bestehen aus zwei Teilen:

- der LSD (List for Symbolic Debugging), in der die im Modul definierten Namen und Instruktionen verzeichnet sind und
- dem ESD (External Symbol Dictionary), in dem die Externbezüge eines Moduls registriert sind.

Die Erzeugung, bzw. Weitergabe dieser Informationen kann bei jedem der folgenden Schritte veranlaßt oder unterdrückt werden:
- Übersetzen mit ASSEMBH,
- Binden und Laden mit DLL (bis BS2000 V9.5), DBL ab BS2000 V10.0 oder
- Binden mit TSOSLNK und
- Laden mit ELDE oder
- Binden mit BINDER (ab BS2000 V10.0) und Laden mit DBL

Dabei werden ESD-Informationen standardmäßig generiert und weitergegeben, während die LSD-Informationen AID auf zwei Wegen zugänglich gemacht werden können.

Wenn LSD-Informationen beim Übersetzen erzeugt worden sind, ist es möglich,
- sie zusammen mit dem Gesamtprogramm zu laden oder
- sie erst bei Bedarf für jede Übersetzungseinheit nachzuladen, falls die zugehörigen Moduln in einer PLAM-Bibliothek stehen.

Die folgende Tabelle gibt für beide Fälle einen Überblick über die entsprechenden Kommandos und die Operanden, die ihnen bei jedem Schritt zugeordnet werden müssen (zur genaueren Information siehe "AID, Testen von ASSEMBH-Programmen" [2]).

Schritte in der Programmentwicklung	Kommandos mit Operanden	
	wenn die LSD-Information zusammen mit dem Programm geladen werden soll	wenn die LSD-Information bei Bedarf durch AID nachgeladen werden soll *
Übersetzen mit ASSEMBH	//COMPILE SOURCE=..., TEST-SUPPORT=AID	//COMPILE SOURCE=..., TEST-SUPPORT=AID, MODULE-LIBRARY=...
Binden und Laden mit DLL/DBL	/LOAD-PROGRAM..., oder /START-PROGRAM..., TEST-OPTION=AID	/LOAD-PROGRAM..., oder /START-PROGRAM..., TEST-OPTION=NONE
Binden mit TSOSLNK	*PROGRAM...,SYMTEST=ALL	*PROGRAM...[,SYMTEST=MAP]
Binden mit BINDER (und Laden mit DBL, siehe weiter oben)	START-LLM-CREATION..., INCLUSION-DEFAULTS= (TEST-SUPPORT=YES) SAVE-LLM LIB=..., TEST-SUPPORT=YES	das Nachladen von LSD-Sätzen ist für Module im LLM-Format nicht möglich
Laden mit ELDE	/LOAD-PROGRAM..., oder /START-PROGRAM..., TEST-OPTION=AID	/LOAD-PROGRAM..., oder /START-PROGRAM..., TEST-OPTION=NONE

* Dies ist nur möglich, wenn die zugehörigen Moduln in einer PLAM-Bibliothek stehen und mit %SYMLIB zugewiesen wurden.

9.3 Beispiel für einen Testablauf

In diesem Beispiel wird eine AID-Testsitzung für ein kleines Assembler-Programm gezeigt. Anhand dieser Testsitzung können Sie die Anwendung und Wirkung einiger AID-Kommandos nachvollziehen, die Vorgehensweise ist bewußt einfach gehalten. Das Assembler-Programm ist in Abschnitt 9.3.1 dargestellt, der Testablauf in Abschnitt 9.3.2 Zur besseren Lesbarkeit sind Eingaben fettgedruckt.

9.3.1 Assembler-Programm

Aufgabenbeschreibung

Das Programm SUMME soll maximal 10 zweistellige Zahlen einlesen und ihre Summe ausgeben. Als Endekennzeichen wird die Zahl 00 eingegeben.
Bei mehr als 10 Zahlen wird eine Meldung und die errechnete Summe ausgegeben.

Quellprogrammliste

```
BERECHNUNG DER SUMME VON N ZAHLEN (N <= 10)                                                                    11:17:32  94-03-08
 LOCTN  OBJECT CODE    ADDR1    ADDR2     STMNT  M  SOURCE STATEMENT
 000000                                       1     SUMME    START
                                              2              TITLE 'BERECHNUNG DER SUMME VON N ZAHLEN (N <= 10)'
                                              3              PRINT NOGEN
 000000                 00000000              4     R0       EQU   0
 000000                 00000001              5     R1       EQU   1
 000000                 00000002              6     R2       EQU   2
 000000                 00000003              7     R3       EQU   3
 000000                 00000004              8     R4       EQU   4
 000000                 00000005              9     R5       EQU   5
                                             10     SUMME    AMODE ANY
                                             11     SUMME    RMODE ANY
                                             12              GPARMOD 31
                                             14   2                *,VERSION 010
 000000 0D 20                                15              BASR  R2,R0
 000002                 00000002             16              USING *,R2
                                             17     ANFANG   WROUT MELD1,ENDE
                                             24   2                *,FHDR VERSION 105 / 1988-06-13
                                             48   2                *,@DCEO      952    900503   53531004
                                             51   1                *,WROUT      005    910215   53121058
 000026 58 50 2176      00000178             52              L     R5,=F'1'
 00002A 5A 50 2176      00000178             53     SCHLEIFE A     R5,=F'1'
 00002E 49 50 2138      0000013A             54              CH    R5,ZEHN
 000032 47 20 20BE      000000C0             55              BH    FEHLER
                                             56     LESEN    RDATA EINGABE,ENDE
                                             63   2                *,FHDR VERSION 105 / 1988-06-13
                                             92   2                *,@DCEI      920    881104   53531002
                                             95   1                *,RDATA      006    910215   53121057
 000062 D5 05 2121213A 00000123 0000013C     96     VERGL    CLC   EINGABE+4,NULL
 000068 47 80 207A      0000007C             97              BE    AUS
 00006C F2 11 21232121 00000125 00000123     98     ADD      PACK  PACK,EINGABE+4(2)
 000072 FA 31 213C2123 0000013E 00000125     99              AP    GESAMT,PACK
 000078 47 F0 2028      0000002A            100              B     SCHLEIFE
 00007C F3 63 2131213C 00000133 0000013E    101     AUS      UNPK  ERGEB,GESAMT
 000082 D3 00 21372140 00000139 00000142    102              MVZ   ERGEB+6(1),ZONE
                                            103              WROUT MELD2,ENDE
                                            109   2                *,FHDR VERSION 105 / 1988-06-13
                                            133   2                *,@DCEO      952    900503   53531004
                                            136   1                *,WROUT      005    910215   53121058
                                            137     ENDE     TERM  DUMP=Y
                                            140   2                *,VERSION 010
                                            152     FEHLER   WROUT MELD3,ENDE
                                            159   2                *,FHDR VERSION 105 / 1988-06-13
                                            183   2                *,@DCEO      952    900503   53531004
                                            186   1                *,WROUT      005    910215   53121058
 0000E2 47 F0 207A      0000007C            187              B     AUS
                                            188              EJECT
                                            189     *
                                            190     * DEFINITIONEN
                                            191     *
 0000E6 0039                                192     MELD1    DC    Y(L'M1+5)
 0000E8 404001                              193              DC    X'404001'
 0000EB C2C9E3E3C540C2C9                    194     M1       DC    C'BITTE BIS ZU 10 2-STELLIGE ZAHLEN EINGEBEN! ENDE:
 00011F 000000000000                        195     EINGABE  DC    XL6'00'
 000125 000C                                196     PACK     DC    PL2'0'
                                            197     *
 000128 0012                                198     MELD2    DC    Y(L'M2+L'ERGEB+5)
 00012A 404001                              199              DC    X'404001'
 00012D E2E4D4D4C57A                        200     M2       DC    C'SUMME:'
 000133 40404040404040                      201     ERGEB    DC    CL7' '
                                            202     *
 00013A 000A                                203     ZEHN     DC    H'10'
 00013C F0F0                                204     NULL     DC    C'00'
 00013E 0000000C                            205     GESAMT   DC    PL4'0'
 000142 F0                                  206     ZONE     DC    X'F0'
                                            207     *
 000144 0034                                208     MELD3    DC    Y(L'M3+5)
 000146 404001                              209              DC    X'404001'
 000149 C5E240D2D6C5D5D5                    210     M3       DC    C'ES KOENNEN MAXIMAL 10 ZAHLEN VERARBEITET WERDEN'
```

```
 000000                                  211              END   SUMME
 000178 00000001                         212                    =F'1'
 00017C 9101221427002852                 213                    =X'9101221427002852' CONSISTENCY CONSTANT FOR AID
FLAGS IN 00000 STATEMENTS, 000 PRIVILEGED FLAGS, 000 MNOTES
HIGHEST ERROR-WEIGHT : NO ERRORS
THIS PROGRAM WAS ASSEMBLED BY ASSEMBH                   V1.2A00     ON 1994-03-08 AT 11:15:54
```

9.3.2 Testablauf

1. Schritt

Das Assembler-Quellprogramm SUMME in der Datei SOURCE.TEST wird mit dem ASSEMBH übersetzt. Auf Grund der Angabe TEST-SUPPORT=YES wird vom ASSEMBH LSD-Information erzeugt und an den Bindemodul übergeben. Das Quellprogramm wird fehlerfrei übersetzt.

```
/DEL-SYS-FILE OMF
/START-PROG $ASSEMBH

%  BLS0500 PROGRAM 'ASSEMBH', VERSION '1.XXXX' OF 'yy-mm-dd' LOADED.
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG. 1991. ALL
   RIGHTS RESERVED
%  ASS6010 V 1.XXXX OF BS2000 ASSEMBH READY

//COMPILE SOURCE=SOURCE.TEST,
          TEST-SUPPORT=AID

%  ASS6011 ASSEMBLY TIME: 80 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 102 MSEC

//END

%  ASS6012 END OF ASSEMBH
```

2. Schritt

Das Programm SUMME soll zum Ablauf gebracht werden.

```
/START-PROG (*OMF)

%  BLS0517 MODULE 'SUMME' LOADED

BITTE BIS ZU 10 2-STELLIGE ZAHLEN EINGEBEN. ENDE: 00
*05
*16
*48
*00
*0
*00
*EN
/
```

Das Programm verzweigt immer wieder zur Eingabe, es liegt ein Programmfehler vor. Das Programm wird durch Drücken der K2-Taste unterbrochen.

3. Schritt

Um das Programm symbolisch testen zu können, wird es mit der Angabe TEST-OPTION=AID erneut geladen.

```
/LOAD-PROG (*OMF),TEST-OPTION=AID

%  BLS0517 MODULE 'SUMME' LOADED

/%IN S'96' <%D EINGABE;%STOP>
/%R
```

Mit dem %INSERT-Kommando wird ein Testpunkt auf die Zeile mit der Statementnummer 96 gesetzt, also auf den CLC-Befehl. Jedesmal, wenn das Programm an dieser Adresse angekommen ist, soll der Inhalt des Feldes EINGABE ausgegeben werden. Nach der Ausgabe soll das Programm in den Zustand STOP übergehen, damit neue Kommandos eingegeben werden können.

Mit %RESUME wird das geladene Programm gestartet.

```
BITTE BIS ZU 10 2-STELLIGE ZAHLEN EINGEBEN. ENDE: 00
*05

** ITN: #'004B012E' *** TSN: 2069 ****************************************
SRC_REF:    96 SOURCE: SUMME     PROC: SUMME ********************************
EINGABE         = 00060000 F0F5   ....05
STOPPED AT LABEL: VERGL , SRC_REF:  96, SOURCE: SUMME ,PROC: SUMME

/%R
*48
EINGABE         = 00060000 F4F8   ....48
STOPPED AT LABEL: VERGL , SRC_REF:  96, SOURCE: SUMME ,PROC: SUMME

/%R
*16
EINGABE         = 00060000 F1F6   ....16
STOPPED AT LABEL: VERGL , SRC_REF:  96, SOURCE: SUMME ,PROC: SUMME

/%R
*00
EINGABE         = 00060000 F0F0   ....00
STOPPED AT LABEL: VERGL , SRC_REF:  96, SOURCE: SUMME ,PROC: SUMME
```

Das Feld EINGABE enthält jeweils den richtigen Wert. Das Programm erkennt offensichtlich das Endekriterium nicht.

4. Schritt

Mit dem %DISASSEMBLE-Kommando sollen jetzt von der Zeile 96, also vom CLC-Befehl an, 5 Zeilen "rückübersetzt" ausgegeben werden.

```
/%DA 5 FROM S'96'

SUMME+62          CLC   121(6,R2),13A(R2)      D5 05 2121 213A
SUMME+68          BC    B'1000',7A(R0,R2)      47 80 207A
SUMME+6C          PACK  123(2,R2),121(2,R2)    F2 11 2123 2121
SUMME+72          AP    13C(4,R2),123(2,R2)    FA 31 213C 2123
SUMME+78          BC    B'1111',28(R0,R2)      47 F0 2028
```

Dabei stellt sich heraus, daß das Längenfeld des CLC-Befehls '6' statt '2' enthält. Deshalb wird das Endekriterium nicht erkannt.

Der Assemblerbefehl müßte korrekt lauten:

```
VERGL     CLC  EINGABE+4(2),NULL
```

5. Schritt

Dieser Fehler kann vorläufig mit dem %SET-Kommando behoben werden. Dazu wird das Programm erneut geladen.

```
/LOAD-PROG (*OMF),TEST-OPTION=AID

%  BLS0517 MODULE 'SUMME' LOADED

/%SET X'D5012121213A' INTO VERGL
/%DA 1 FROM VERGL

SUMME+62          CLC   121(2,R2),13A(R2)      D5 01 2121 213A

/%R
```

Mit dem %SET wird der Speicherinhalt an der Adresse VERGL geändert.
Übertragen wird ein AID-Literal in der Länge des CLC-Befehls, das die Längenangabe '01' statt '05' enthält. Anschließend wird mit %DISASSEMBLE der CLC-Befehl kontrolliert und mit %RESUME wieder gestartet.

```
BITTE BIS ZU 10 2-STELLIGE ZAHLEN EINGEBEN. ENDE: 00
*05
*16
*48
*12
*10
*15
*17
*19
*29
ES KOENNEN MAXIMAL 10 ZAHLEN VERARBEITET WERDEN
SUMME:0000171

%  IDA0N51 PROGRAM INTERRUPT AT LOCATION '000000BE (SUMME), (CDUMP), EC=90'
%  IDA0N45 DUMP DESIRED? REPLY (Y=USER-/AREADUMP;Y,SYSTEM=SYSTEMDUMP;N=NO)?Y
%  IDA0N53 DUMP BEING PROCESSED. PLEASE HOLD ON
%  IDA0N54 USERDUMP WRITTEN TO FILE 'benutzerkennung.DUMP.name.2069.00001'
%  IDA0N55 TITLE: 'TSN-2069  UID-benutzerkennung  AC#-xxxxxxxx  USERDUMP
   PC-0000BE EC=90  VERS-110  DUMP-TIME  11:37:42  94-03-08'
```

Es liegt ein weiterer Programmfehler vor, da der Benutzer bisher nur 9 Zahlen eingegeben hat. Deshalb wird bei Beendigung des Programms ein Dump zur weiteren Diagnose erzeugt.

6. Schritt

Mit dem %DUMPFILE-Kommando wird die Dump-Datei eröffnet und ihr der LINK-Name D1 zugewiesen. Durch %BASE wird der AID-Arbeitsbereich auf die geöffnete Dump-Datei umgeschaltet. Von jetzt an wird mit einer Adresse ohne eigene Basisqualifikation immer auf die Daten der Dump-Datei zugegriffen.

```
/%DUMPFILE D1=DUMP.NAME.2069.00001
/%BASE E=D1

/%D EINGABE
** D1: DUMP.NAME.2069.00001 ************************************************
EINGABE         = 00060000 F2F9   ....29

/%D _R5
_R5             = 0000000B
```

Die zuletzt eingegebene Zahl des Feldes EINGABE soll ausgegeben werden. Die Eingabe stimmt mit dem Protokoll überein.

Da im Register 5 die Anzahl der Eingaben gezählt wird, wird jetzt der Stand des Registers 5 untersucht.

Register 5 enthält den Wert '11', obwohl erst 9 Zahlen eingegeben wurden. Ein Vergleich mit dem Übersetzungsprotokoll ergibt, daß Register 5 mit '1' vorbesetzt ist, statt mit '0'.

Der Assemblerbefehl müßte korrekt lauten: `L  R5,=F'0'`

7. Schritt

Dieser Fehler kann vorläufig mit Hilfe des %SET-Kommandos behoben werden. Dazu wird das Programm erneut geladen.

```
/LOAD-PROG (*OMF),TEST-OPTION=AID

%  BLS0517 MODULE 'SUMME' LOADED

/%BASE
/%SET X'D5012121213A' INTO VERGL
/%IN SCHLEIFE <%SET #'0' INTO _R5; %REM %.>

/MOD-TEST-OPT DUMP=NO
/%R
```

Als erstes muß mit %BASE wieder das geladene Programm als AID-Arbeitsbereich zugewiesen werden.

Durch das erneute Laden des Programms sind vorherige Korrektureingaben gelöscht. Um einen fehlerfreien Programmablauf zu gewährleisten, setzen wir deshalb das %SET-Kommando aus dem 5. Schritt hier nochmals ab.

Mit dem %INSERT-Kommando wird ein *testpunkt* auf den Assemblerbefehl mit dem Namenseintrag SCHLEIFE gesetzt. AID soll das folgende *subkdo* also vor dem Additionsbefehl ausführen.

Das %SET-Kommando, das Register 5 mit '0' vorbesetzt, steht im *subkdo* von %INSERT. Mit %REM %. wird dieses *subkdo* nach dem ersten Durchlauf gelöscht (da kein weiteres Subkommando zu diesem *testpunkt* eingetragen ist, wird auch der *testpunkt* gelöscht), und das Programm läuft dann weiter.

Da im Quellprogramm der Makroaufruf TERM mit dem Operanden DUMP=Y definiert ist, wird bei jedem Programmende ein Speicherabzug (Dump) angeboten. Dieses kann vor dem Start des Programms (%RESUME) mit folgendem Kommando verhindert werden: /MODIFY-TEST-OPTIONS DUMP=NO

```
BITTE BIS ZU 10 2-STELLIGE ZAHLEN EINGEBEN. ENDE: 00
*05
*16
*48
*12
*10
*15
*17
*19
*29
*11
ES KOENNEN MAXIMAL 10 ZAHLEN VERARBEITET WERDEN
SUMME:0000182

%  IDA0N51 PROGRAM INTERRUPT AT LOCATION '000000BE (SUMME), (CDUMP), EC=90'
%  IDA0N47 DUMP PROHIBITED BY /OPTION COMMAND
/
```

Nach dieser Korrektur läuft das Programm fehlerfrei ab. Die Fehler können endgültig im Quellprogramm behoben werden.

10 Dienstprogramme für die strukturierte Programmierung

Die strukturierte Programmierung des ASSEMBH wird durch folgende Dienstprogramme unterstützt:

COLLIST	zur Erstellung von Strukturlisten
COLNAS	zur Erzeugung von Nassi-Shneiderman-Diagrammen
COLINDA	zum Einrücken von strukturierten Quellprogrammen
COLNUMA	zum Verbinden von Struktur- und Assemblerinformationen in einer Liste

Bild 10-1 zeigt in einer Übersicht die Funktion von COLLIST, COLNAS und COLNUMA an einem kleinen Beispiel.

Das Format und die Bedeutung der Sprachelemente der strukturierten Programmierung sind in der Sprachbeschreibung des ASSEMBH [1] ausführlich beschrieben.
Vom ASSEMBH-BC wird die strukturierte Programmierung nicht unterstützt !

Die Dienstprogramme COLLIST und COLNAS sind sprachunabhängig, d.h. die Eingabe dieser Programme darf auch aus Pseudo-Code bestehen.

COLINDA erzeugt aus dem Primärprogramm ein eingerücktes Quellprogramm, so daß beim Assemblieren ein übersichtliches Protokoll entsteht.

Das Dienstprogramm COLNUMA führt wahlweise eine der beiden folgenden Funktionen aus:

- es erweitert die von COLLIST erstellte Strukturliste eines Quellprogramms durch Informationen aus dem entsprechenden Assemblerprotokoll;
- es ergänzt die Assemblerliste eines von COLINDA aufbereiteten Quellprogramms zur stärkeren Hervorhebung der Programm-Struktur.

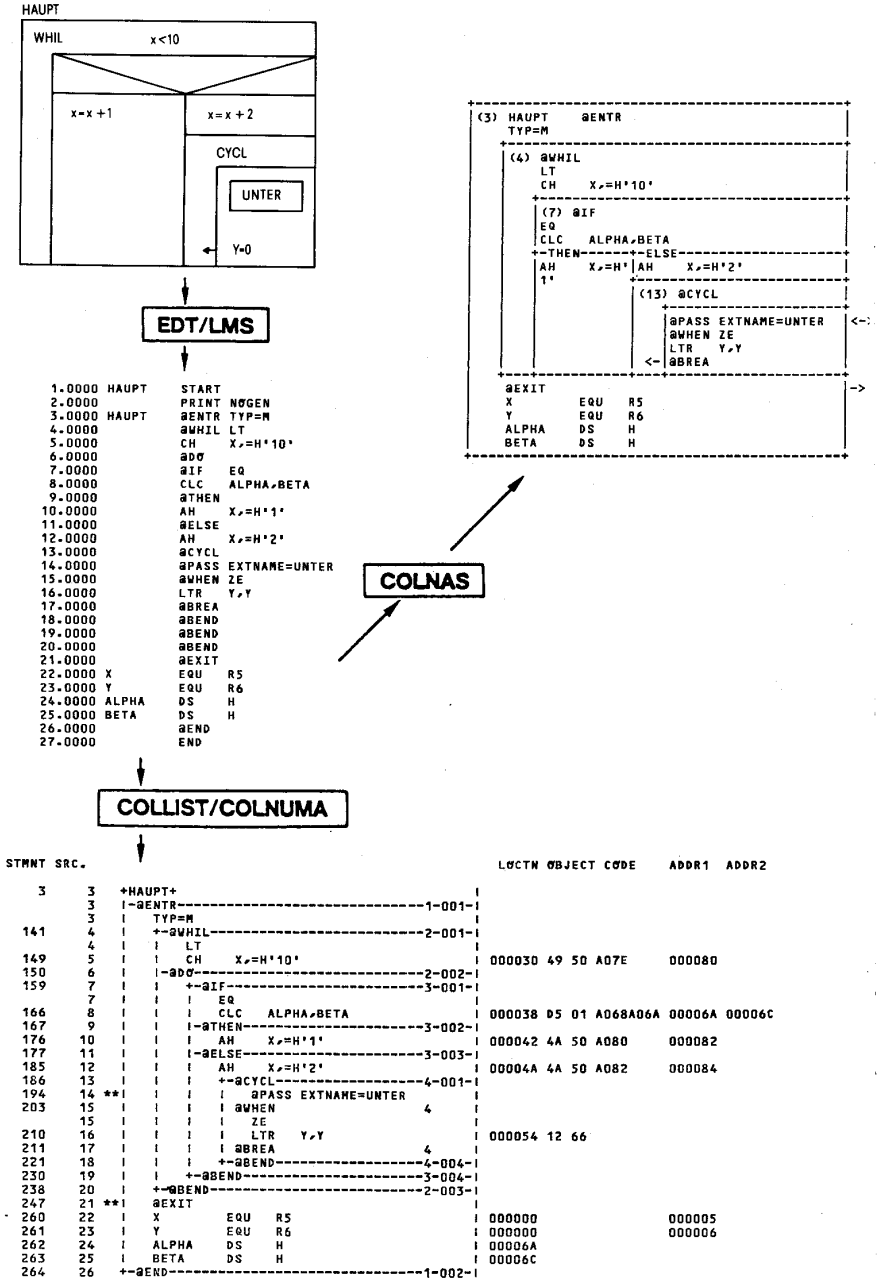


Bild 10-1: Funktion von COLNAS und COLNUMA

10.1 Dienstprogramme, die das strukturierte Quellprogramm aufbereiten

COLLIST, COLNAS, COLINDA

Die Dienstprogramme COLLIST, COLNAS und COLINDA bereiten Quellprogramme der strukturierten Programmierung auf. Alle drei Programme nehmen eine Prüfung der Syntax vor und melden erkannte Verstöße an den auftretenden Stellen.

10.1.1 COLLIST

Das Dienstprogramm COLLIST führt wahlweise 2 Funktionen aus:

1. Erstellen einer Strukturliste zur Darstellung der Strukturblockschachtelung.
2. Erstellen einer Prozedurliste zur Darstellung der Aufrufhierarchie und einer Tabelle über mehrfach verwendete Prozeduren.

10.1.1.1 Strukturliste

Behandlung der Blöcke

- Die Strukturwörter sowie Prozedurkopf und Prozedurende werden in jeweils besondere Zeilen gesetzt, deren rechte Enden mit einer waagerechten Linie und einer vierstelligen Nummer aufgefüllt werden. Die erste Stelle der Nummer gibt die Schachtelungstiefe an, die restlichen drei Stellen sind eine laufende Nummer innerhalb einer Schachtelungstiefe und einer Prozedur.

  Falls die maximale Schachtelungstiefe innerhalb einer Prozedur größer als 9 ist, werden nur fortlaufende Nummern beginnend mit 1-000 erzeugt.
- Die Strukturwörter eines Strukturblocks sind gleich ausgerichtet und durch eine senkrechte Linie verbunden.
- Die zu den Strukturwörtern gehörenden Unterblöcke sind diesen gegenüber eingerückt.
- Der rechte Rand ist durch eine senkrechte Linie abgeschlossen.
- Ein Blockname wird seinem Block in einer gesonderten Zeile vorangestellt.

Behandlung der Zeilen

- Zwischen den Linien der Strukturwörter werden die Grundanweisungen und Kommentare aus dem Quellprogramm übernommen und gegenüber den Strukturwörtern eingerückt. Führende Zwischenräume werden entfernt.
- Die Grundanweisungen @PASS und @EXIT werden durch Sterne am linken Rand gekennzeichnet.
- Reicht der Platz für eine Quellzeile in der Listenzeile nicht aus, so wird die Quellzeile auf einer oder mehreren Listenzeilen fortgesetzt. Ist der Platz in der Listenzeile sehr gering, d.h. kleiner als 12 Zeichen, so wird die Übernahme der Quellzeilen unterdrückt und die Meldung "LINE SUPPRESSED" ausgedruckt.
- Für jede Listenzeile wird die Nummer der korrespondierenden Quellprogrammzeile in den ersten Spalten ausgegeben.
- Hinter dem rechten Rand kann wahlweise eine Zeilenkennzeichnung aus der Quellprogrammzeile (Spalten 73-80) oder der Schlüssel des Eingabesatzes ausgegeben werden. Wird keine Zeilenkennzeichnung gewünscht, so schiebt COLLIST den senkrechten Strich ganz nach rechts.

Fehlermeldungen

- Fehlermeldungen und Warnungen aus der Syntaxprüfung werden in einer Zeile vor dem fehlerhaften Schlüsselwort ausgedruckt.

Seitenvorschub

– Für jede Prozedur wird ein Seitenvorschub und eine Überschrift mit Namen der Eingabedatei, Datum, Uhrzeit und Seitennummer erzeugt. In einer zweiten Überschrift wird die Prozedurnummer angegeben.

  Ebenso erfolgt ein Seitenvorschub mit der Ausgabe einer Überschrift vor allen Programmteilen außerhalb von Prozeduren.

  Sollen mehrere kleine Prozeduren auf eine Seite, so müssen diese Prozeduren im Quellprogramm durch Zeilen getrennt werden, die nur einen Stern in Spalte 1 enthalten. In der Strukturliste erscheinen dann entsprechend viele Leerzeilen zwischen den Prozeduren. Es wird nur eine Überschrift mit Prozedurnummer erzeugt.

– Um Strukturlisten, die größer als eine Druckseite sind, sinnvoll zu unterteilen, kann man durch eine Kommentarzeile mit *: in den Spalten 1-2 einen Seitenvorschub bewirken.
  Diese Kommentarzeile wird in der Strukturliste nicht ausgegeben.

  Steht das Seitenvorschubzeichen *: vor einem Prozeduranfang (@ENTRY) so wird nur ein Seitenvorschub vorgenommen. Dieses Seitenvorschubzeichen dient nur der optischen Auflockerung der Eingabedatei.

  Die Ausgabe der Überschrift kann dabei gesteuert werden.

– Die maximale Anzahl der Druckzeilen je Seite kann der Benutzer festlegen.

Beispiel

Das folgende Beispiel zeigt die wesentlichen Merkmale einer COLLIST-Strukturliste.

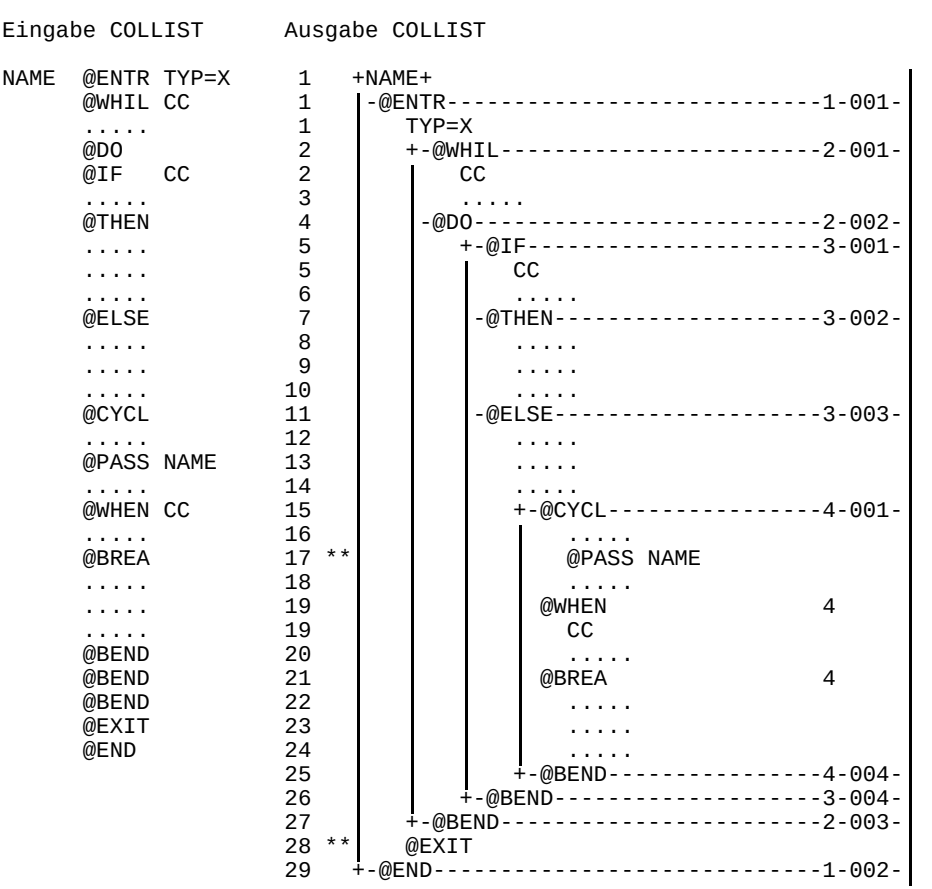


Bild 10-2: COLLIST-Strukturliste

10.1.1.2 Prozedurliste

Eine Prozedurliste ist eine den Strukturlisten ähnliche Darstellung der Aufrufhierarchie eines Primärprogramms, wobei alle @ENTR- und @PASS-Anweisungen ausgewertet werden. Den Listen der Aufrufhierarchie folgt eine abschließende Tabelle mit Informationen über mehrfach verwendete Prozeduren.
Im Beispiel sind die Details der Prozedurliste näher erläutert.

Die Erstellung der Prozedurlisten kann derart gesteuert werden, daß beim wiederholten Aufruf einer im Programm enthaltenen Prozedur deren Unterstruktur nicht mehr ausgegeben wird (Kennzeichnung durch **).

```
BEISPIEL      DATE 09/03/94   TIME 14:28:03

┌──CALLING HIERARCHY OF PROCEDURE──────────┐
│                                          │
│  CETLST  LN=1                            │
├──────────────────────────────────────────2┤
│  AFORM (3)   ──►                         │
│                                          │
│  IOP (9)   ──►                           │
│                                          │
│  GETIME ──►                              │
│                                          │
│  LISTAB  LN=1782                         │
│  ┌──────────────────────────────────────3┤
│  │  GETPRC (2)  LN=1647                 │
│  │  ┌──────────────────────────────────4┤
│  │  │  ILPDPF ──►                      │
│  │  └──────────────────────────────────4┤
│  │  SYSERR (7) ──►                      │
│  └──────────────────────────────────────3┤
│  SETPSS  LN=478                          │
│  ┌──────────────────────────────────────3┤
│  │  DRUCK (11)  LN=1146                 │
│  │  HEADER (2)  LN=1289                 │
│  │  ┌──────────────────────────────────4┤
│  │  │  ERS (7) ──►                     │
│  │  │  IOP (2) ──►                     │
│  │  └──────────────────────────────────4┤
│  └──────────────────────────────────────3┤
│  HFGTL1  LN=1981                         │
│  ┌──────────────────────────────────────3┤
│  │  RFOLGE (3)  LN=2067                 │
│  │  ┌──────────────────────────────────4┤
│  │  │  AUSGAB  LN=2199                 │
│  │  │  ┌──────────────────────────────5┤
│  │  │  │  HEADER  **  LN=1289          │
│  │  │  │  STRL ──►                     │
│  │  │  └──────────────────────────────5┤
│  │  └──────────────────────────────────4┤
│  └──────────────────────────────────────3┤
│  TERM UNIT=STEP,MODE=ABNORMAL ──►        │
└──────────────────────────────────────────2┘
```

```
BEISPIEL      DATE 09/03/94   TIME 14:28:03

MULTIPLE CALLED PROCEDURES
```

PROCEDURE	CALLED FROM
IOP →	CETLST (9) HEADER (2)
HEADER (LN=1289)	SETPSS (2) AUSGAB

Bild 10-3: Beispiel für eine Prozedurliste

Erläuterung des Beispiels

- BEISPIEL ist der Dateiname des Programms.
- Von der Prozedur CETLST wird dreimal die Prozedur AFORM aufgerufen, neunmal IOP, einmal GETIME usw. Die mit → gekennzeichneten Prozeduren sind nicht im ausgewerteten Programm enthalten.
- Die Prozedur LISTAB dagegen ist Bestandteil des Programms BEISPIEL. Die entsprechende @ENTR-Anweisung ist in Zeile 1782. LISTAB ruft die Prozeduren GETPRC und SYSERR auf.
- Mit ** gekennzeichnete Prozeduren sind im ausgewerteten Programm enthalten. Die ihnen untergeordneten Prozeduren sind weiter oben in der Prozedurliste bereits dargestellt und werden hier nicht mehr ausgedruckt (Parameter FULPCLST=NO).
- Die abschließende Tabelle enthält in der linken Spalte die Namen der mehrfach verwendeten Prozeduren IOP und HEADER. Liegt die gerufene Prozedur im analysierten Programm, so wird auch hier der Beginn der Prozedur angezeigt.
- Die rechte Spalte der Tabelle enthält die Namen der rufenden Prozeduren und in Klammern die Anzahl der Aufrufe, falls diese größer als 1 ist. Die Prozedur IOP wird neunmal von der Prozedur CETLST und zweimal von HEADER aufgerufen.

Hinweis

- Das Primärprogramm darf höchstens 300 verschiedene Prozedurnamen (in @ENTR- und @PASS-Anweisungen) enthalten.
- Das Primärprogramm darf höchstens 1000 @ENTR- und @PASS-Anweisungen enthalten, wobei mehrere Aufrufe derselben Prozedur innerhalb einer Prozedur nur einmal zählen.

10.1.2 COLNAS

In der von COLNAS ausgegebenen Liste werden die Prozeduren der strukturierten Programmierung als Nassi-Shneiderman-Diagramme dargestellt.

Insbesondere werden zusammengehörende Then- und Else-Unterblöcke bzw. Case-Unterblöcke nicht untereinanderstehend - wie von COLLIST - sondern nebeneinanderstehend dargestellt. So wird der Steuerfluß stärker hervorgehoben als durch bloßes Einrükken:
Ein Strukturblock wird prinzipiell nur über die abschließende horizontale Linie verlassen, die über einen der eventuell nebeneinandergestellten Unterblöcke erreicht wird.

Das Nebeneinanderstellen hat neben Vorteilen (Klarheit, Ausnützen der zweiten Dimension) jedoch auch Nachteile, die vor allem aus der begrenzten Zeichenstellenzahl je Zeile entstehen: In tiefer geschachtelten Blöcken stehen für einen Unterblock nur noch so wenige Zeichen der Zeile zur Verfügung, daß möglicherweise die Gesamtinformation aus dem Quellprogramm nicht sinnvoll dargestellt werden kann.

Soweit aber die Information des Quellprogramms untergebracht werden kann, bietet die Liste eine gute Überprüfbarkeit der Programmstruktur, insbesondere der Verschachtelung von Auswahlstrukturblöcken.

Fehler, die bei der Struktursyntaxprüfung erkannt werden, führen zu einem Unterdrükken der Diagramme für die jeweilige Prozedur.

10.1.2.1 Aufbau der Liste

Einrückbetrag

Der Anwender bestimmt über Steuerparameter einen "Einrückbetrag" (siehe INDAMT, 10.3.5). Sein Standardwert beträgt 4 Zeichen. Der Einrückbetrag bestimmt die Einrükkung des Schleifenunterblocks in Wiederholungsstrukturblöcken (@WHIL, @CYCL, @THRU) und des Unterblocks der Sequenz (@BEGI).

Auch bei den Entscheidungen (@IF) spielt der Einrückbetrag eine Rolle:

Darstellung der Entscheidungen

- Die Bedingung wird in einen rechteckigen Kasten unter das Strukturwort IF gestellt, da keine schrägen Linien gezogen werden können.
- Die Then- bzw. Else-Unterblöcke werden durch die Worte THEN bzw. ELSE in der oberen Kante der Unterblöcke gekennzeichnet, falls in der Zeile noch Platz dafür vorhanden ist.
- Falls der Else-Unterblock fehlt, wird dies durch einen leeren Streifen am rechten Rand in der Breite des Einrückbetrages dargestellt.

  Die Aufteilung des zur Verfügung stehenden Zeilenteils in die Unterblöcke ist vor allem von dem Verhältnis der Zeilenzahlen zwischen @THEN und @ELSE bzw. @ELSE und @BEND abhängig. Wenn das Verhältnis ungefähr 1 ist (0,8 - 1,2) oder der Zeilenteil relativ klein (6-facher Einrückbetrag), wird der Zeilenteil halbiert. Sonst wird der Zeilenteil entsprechend dem Zeilenverhälnis in ganzen Vielfachen des Einrückbetrags aufgeteilt. Beim Aufteilen nach Zeilenverhältnis werden für einen Unterblock minimal drei Einrückbeträge zur Verfügung gestellt.

Informationen, die nicht in die Liste übernommen werden

- Es fehlen Verweise auf Quellprogrammzeilen (wie sie COLLIST ausgibt). Sie werden nur für die einen Strukturblock einleitenden Schlüsselwörter erzeugt.
- Der Inhalt der Spalten 73-80 oder der Satzschlüssel werden nicht übernommen.
- Die Ausgabe des Inhalts von Strukturblöcken bzw. von Unterblöcken wird unterdrückt, wenn für den Block weniger als zwei Einrückbeträge in der Zeile zur Verfügung stehen.
- Falls die Ausgabe des Inhalts von Blöcken unterdrückt wird, wird der für den Block noch freie Platz im Diagramm mit Sternen (*) aufgefüllt.
- Auf dem Protokollgerät (SYSOUT) werden die Zeilennummern der unterdrückten Zeilen ausgegeben.

Seitenvorschub

– Für jede Prozedur wird ein Seitenvorschub in der Liste erzeugt sowie eine Überschrift mit Namen der Eingabedatei, Datum, Uhrzeit und Seite gedruckt. In einer zweiten Überschrift wird die Nummer der Prozedur ausgegeben.
  Ebenso erfolgt ein Seitenvorschub mit der Ausgabe einer Überschrift vor allen Programmteilen außerhalb von Prozeduren.

– Die Vorschubsteuerzeichen *: und * (siehe 10.1.1, COLLIST-Seitenvorschub) werden bei COLNAS nur außerhalb von Prozeduren berücksichtigt.

– Der Anwender kann steuern, wieviele Zeilen pro Seite ausgegeben werden und ob auf jeder Seite (auch zwischen Prozeduren) eine Überschrift steht.

Fehlermeldungen

– Werden in einer Prozedur von der Struktursyntaxprüfung Fehler festgestellt, so wird für die Prozedur kein Strukturdiagramm ausgegeben, sondern nur eine Liste der Strukturfehlermeldungen mit dem Text der dazugehörigen Zeilen.

Beispiel

Das folgende Beispiel zeigt die wesentlichen Merkmale einer Liste von COLNAS.

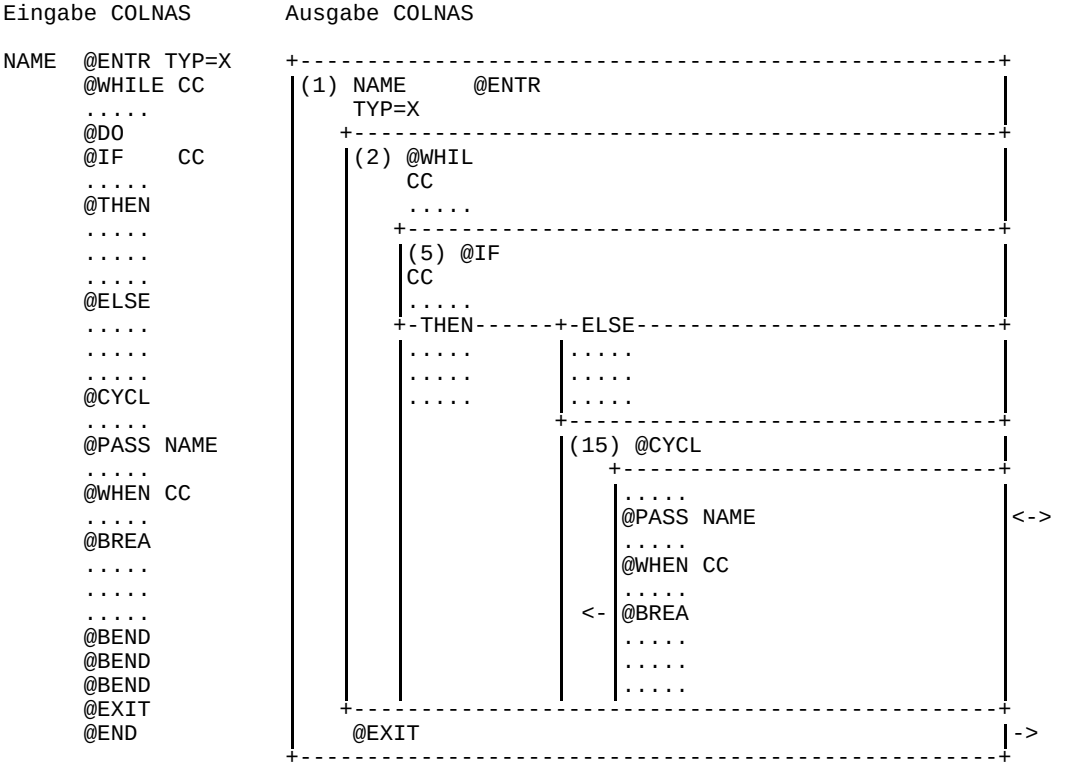


Bild 10-4: COLNAS-Liste

10.1.3 COLINDA

Das Dienstprogramm COLINDA erstellt aus einem strukturierten Quellprogramm ein gemäß Strukturblockschachtelung eingerücktes strukturiertes Quellprogramm.

10.1.3.1 Ausgabe von COLINDA

Das Programm COLINDA formt ein strukturiertes Quellprogramm so um, daß die Operations-, Operanden- und Kommentarteile einer Anweisung gemäß der Strukturblockschachtelung eingerückt sind und die mit einem Strukturwort beginnenden Zeilen durch eine waagerechte Linie abgeschlossen werden, an deren Ende die Schachtelungstiefe angegeben ist. Die Namensfelder des Assemblerformats werden erkannt und am linken Rand stehengelassen.

Das erzeugte eingerückte Quellprogramm dient als Eingabe für den Assembler und wird in dessen Übersetzungsprotokoll sichtbar.

Aufbau des ausgegebenen Quellprogramms und des damit erzeugten Übersetzungsprotokolls

- Alle Strukturwörter eines Strukturblocks werden - auf dieselbe Schreibstelle eingerückt - ausgegeben.
- Alle Strukturwörter sind durch eine Linie im Kommentarteil hervorgehoben.
- Alle Strukturwörter enthalten am Ende der Linie eine Nummer, die die Schachtelungstiefe angibt.
- Kommentare zu Strukturwörtern werden in der nächsten Zeile als Kommentarzeile ausgegeben.
- Namen von Strukturblöcken und von Unterblöcken werden getrennt und mit

  ```
  DS OH
  ```

  vor den Blockanfang gestellt.
- Unterblöcke werden gegenüber den Strukturwörtern, zu denen sie gehören, eingerückt.
- Vor dem ersten und nach dem letzten Strukturblock innerhalb jeder Prozedur unterbleibt das Einrücken (wichtig für Datendefinitionen).
- Kommentare in Assembler-Anweisungen werden mit eingerückt oder, falls der Platz in der Zeile nicht ausreicht, in eine eigene Kommentarzeile davor gesetzt.
- Der signifikante Teil von Kommentarzeilen (alle Spalten von der ersten bis zur letzten nicht leeren Spalte im Bereich 2 bis 71) in Unterblöcken wird entsprechend der Schachtelungstiefe nur dann ausgerichtet, falls er durch das Einrücken nicht über Spalte 71 hinausgeschoben werden muß. Soll eine Kommentarzeile in einem Unterblock nicht verändert werden (z.B. Umrahmungen von Kommentaren), so kann dies durch Belegen von Spalte 2 und 71 erreicht werden.
- Die Spalten 73-80 der Ausgabezeile werden durchnumeriert, wobei die Spalten 73-79 die laufende Nummer der Eingabezeile innerhalb des Eingabe-Quellprogramms darstellen und die Spalte 80 immer auf Null gesetzt wird. Damit kann im Assemblerprotokoll ein eindeutiger Hinweis auf das ursprüngliche strukturierte Quellprogramm gefunden werden. Diese Numerierung kann unterdrückt werden; dann wird die Zeilenkennzeichnung aus Spalte 73-80 der Eingabe übernommen.

Beispiel

```
Eingabe COLINDA     Ausgabe COLINDA

NAME  @ENTR TYP=X   *                       GENERATED BY COLINDA
      @WHIL CC      NAME     @ENTR TYP=X                                  00000010
      .....         *-------------------------------------------------------
      @DO                    @WHIL CC      *----------------------------------2- 00000020
      @IF   CC                      .....                                  00000030
      .....                  @DO           *----------------------------------2- 00000040
      @THEN                         @IF   CC      *--------------------------3- 00000050
      .....                           .....                                  00000060
      .....                         @THEN         *--------------------------3- 00000070
      .....                           .....                                  00000080
      @ELSE                           .....                                  00000090
      .....                           .....                                  00000100
      .....                         @ELSE         *--------------------------3- 00000110
      .....                           .....                                  00000120
      @CYCL                           .....                                  00000130
      .....                           .....                                  00000140
      @PASS NAME                      @CYCL ,       *------------------4- 00000150
      .....                                 .....                            00000160
      @WHEN CC                                @PASS NAME                     00000170
      .....                                   .....                          00000180
      @BREA                           @WHEN CC      *------------------4- 00000190
      .....                                   .....                          00000200
      .....                           @BREA         *------------------4- 00000210
      .....                                   .....                          00000220
      @BEND                                   .....                          00000230
      @BEND                                   .....                          00000240
      @BEND                           @BEND         *--------------------- 00000250
      @EXIT                         @BEND         *--------------------------3- 00000260
      @END                  @BEND         *----------------------------------2- 00000270
                            @EXIT                                            00000280
                      *-------------------------------------------------------
                            @END                                             00000290
```

Bild 10-5: COLINDA-Ausgabe

10.1.3.2 Strukturfunktionen im TOM-Editor

Die Funktion des Dienstprogramms COLINDA kann im TOM-Editor TOM-TI unmittelbar angewandt werden. Hierzu stehen die beiden TOM-TI-Kommandos COLINDA und COLA zur Verfügung.

COLINDA
: bereitet ein strukturiertes Assembler-Programm im Arbeitsbereich des TOM-TI genauso auf, wie es der Ausgabe des Dienstprogramms COLINDA entspricht.

COLA
: rückt das Programm ebenso ein wie COLINDA; es fehlen aber die Zeilennummern in Spalte 73-80, die waagrechten Striche zur Hervorhebung der Strukturanweisungen, die Angabe der Schachtelungstiefe sowie "Name DS 0H" vor Strukturblockanfängen.

10.2 COLNUMA

COLNUMA ist das Dienstprogramm, das alle Informationen zusammenfaßt und damit dem Anwender das Testen auf der "Ebene der strukturierten Programmierung" ermöglicht. Was es leistet, bestimmen die zugewiesenen Eingabedateien.

10.2.1 Erweiterung der Strukturliste

Ist eine Strukturliste des Quellprogramms (COLLIST-Ausgabe) zugewiesen, so wird sie durch Informationen aus dem Assemblerprotokoll vervollständigt.

Voraussetzungen dazu sind:

- Das Quellprogramm wird in den Stellen 73-80 mit einer aufsteigenden Numerierung versehen. Das läßt sich durch folgendes EDT-Kommando erreichen:

```
@SEQ[UENCE]
```

- Die Übersetzung wird mit folgender Assembler-Anweisung durchgeführt:

```
PRINT NOGEN
```

- Die Strukturliste von COLLIST wird mit folgendem Steuerparameter erzeugt:

```
LSTCOL=100
```

Eingabedateien von COLNUMA sind dann folgende von einem Quellprogramm mit Numerierung in den Stellen 73-80 ausgehende Listen:

- Strukturliste (COLLIST-Ausgabe)
- Assemblerprotokoll (Assembler-Ausgabe)

Ausgabe von COLNUMA ist eine COLLIST-Strukturliste mit folgenden Einfügungen:

Am rechten Rand der Strukturliste werden die entsprechenden sedezimalen Adressen und der erzeugte Objektcode (linker Teil einer Zeile des Assemblerprotokolls) eingefügt. Am linken Rand wird die Statementnummer des Assemblerprotokolls eingefügt.

Hinweise

- Die Ein- und Ausgabe kann mit den Parametern oder mit den Linknamen CLIST, ASMLST oder EWCLIST gesteuert werden.
- Durch Makros generierte Zeilen sind in der Ausgabeliste nicht enthalten.
- Wird nicht mit PRINT NOGEN assembliert, so fehlt die Location-Angabe in der Makroaufrufzeile.
- Die Assemblermeldungen werden nicht übernommen.

10.2.2 Ergänzung der Assemblerliste eines von COLINDA aufbereiteten Programms

Fehlt die Strukturliste, ist also nur eine Assembler-Liste vorhanden, so prüft COLNUMA, ob diese Liste auf einem von COLINDA aufbereiteten Programm basiert. Ist dies der Fall, so bearbeitet COLNUMA diese Assembler-Liste. Handelt es sich nicht um eine Assembler-Liste eines von COLINDA aufbereiteten Programms, so wird der COLNUMA-Lauf mit einer Meldung beendet.

Die folgenden Punkte beschreiben die Liste, die COLNUMA aus der COLINDA-Assembler-Liste erzeugt:

- Die Zeichen "@" der Strukturblock-Anfangsanweisungen (@BEGIN, @IF, @CASE, @CAS2, @WHILE, @CYCLE, @THRU) und der dazugehörigen Strukturblock-Endanweisungen (@BEND) werden durch senkrechte Striche miteinander verbunden. Das Zeichen "@" und andere Zeichen ungleich Blank (z.B. aus Namensfeldern) in dazwischenliegenden Zeilen werden dabei nicht überschrieben. Die außerhalb von Strukturblöcken liegenden Teile einer Prozedur werden dadurch nicht berührt. Auch COLINDA verändert diese nicht. Das Strichezieh en beschränkt sich auf den Bereich der Prozedur, in dem in der Spalte 70 die Schachtelungstiefe eingetragen ist. Bei Strukturfehlern (z.B. fehlendes @BEND) enden die Striche spätestens bei dem waagrechten Strich vor dem @END. In dieser von COLNUMA ergänzten COLINDA-Assembler-Liste ist somit die Struktur ebenso gut ersichtlich, wie in einer von COLLIST erstellten Strukturliste.
- Vor jeder Prozedur erfolgt ein Seitenvorschub und die Ausgabe einer Überschriftszeile.
- Der Prozedurkörper ist durch die waagrechten Striche nach dem @ENTR bzw. vor dem @END bereits von COLINDA hervorgehoben.
- Der Operationscode der Strukturblöcke wird ab Spalte 10 ausgegeben. Damit entsteht ein linker Rand für das Namensfeld, der die senkrechte Begrenzung der Strukturblöcke nicht tangiert.

  Im Beispiel auf der nachfolgenden Seite sind diese senkrechten Striche eingetragen.

Hinweis

Die Ein- und Ausgabe kann mit den Parametern oder mit den Linknamen ASMLST und EWCLIST gesteuert werden.

Beispiel

```
 SOURCE STATEMENT                                          10:24:17  94-03-09

 BLD      START
 ** ERROR: STRUCTURE  201 IN @ENTRY   BLOCK
 *                       GENERATED BY COLINDA
 EINS     @ENTR TYP=B
*-------------------------------------------------------------------------------

DATEN    @DATA CLASS=S, INIT=GLOBALS
         MNOTE 225, BASE MISSING
         MNOTE 225, FAILURES FOUND : ALL SKIPPED

RAHMEN   DS    OH
         @BEGIN     *-------------------------------------------------------2-
FALLS      DS      OH
           @IF     LE *-------------------------------------------------------3-
           |  CR       R4, R5
           @THEN      *-------------------------------------------------------3-
LOOP       |  DS      OH
           |  @CYCLE   (R7) *-------------------------------------------------4-
           |  |  TRALALA
           |  |  LH      R4, X
AUSIS      |  |  DS      OH
           |  @WHEN ZE   *-------------------------------------------------4-
           |  |  LTR     R4, R4
           |  @BREAK     *-------------------------------------------------4-
LOOPEND    |  |  DS      OH
           |  @BEND      *-------------------------------------------------4-
FALLSEND   |  DS      OH
           @BEND      *-------------------------------------------------------3-
MVC X,Y
@BEND      |
EINSENDE @EXIT
REINS    EQU   R2
** ERROR |STRUCTURE  301 IN @BEGIN   BLOCK
*--------|----------------------------------------------------------------------
           @END
           MNOTE 250, SYNTAX ERROR : @END ON WRONG PLACE : SKIPPED
 DEINS     EQU   1
           END
```

Bild 10-6: Ausschnitt einer von COLNUMA verbesserten COLINDA-Assembler-Liste

10.3 Bedienung der Dienstprogramme COLLIST, COLNAS und COLINDA

Datenfluß der strukturierten Assembler-Programme

Der gegenüberliegende Ablaufplan zeigt den Datenfluß der strukturierten Assembler-Programme.

Das strukturierte Quellprogramm dient als Eingabedatei für die Programme COLLIST, COLNAS und COLINDA. Im EDT können die Zeilen durchnumeriert werden. Die Ausgabe von COLLIST und COLNAS ist mit "liste" gekennzeichnet. Die Ausgabe von COLINDA ist eine eingerückte Datei mit dem Quellprogramm.

Sowohl die numerierte Quellprogrammdatei als auch die eingerückte Quellprogrammdatei dienen als Eingabe für den ASSEMBH, zu dem noch die Makrobibliothek für die strukturierte Programmierung zugewiesen wird. Außerdem ist die numerierte Quellprogrammdatei auch die Eingabedatei für das Dienstprogramm COLLIST.

Die von COLLIST ausgegebene Strukturliste und die Assemblerliste können von COLNUMA weiterverarbeitet werden.

Der ASSEMBH legt den Modul über die Option MODULE-LIBRARY (siehe 2.4.2.2) in einer "modulbibliothek" ab. Der Binder TSOSLNK verknüpft den Modul mit dem Assembler-Laufzeitsystem und erzeugt ein ablauffähiges Programm ("lademodul"). Siehe Kapitel 5.6, ’Übersetzen und Binden eines strukturierten Assembler-Programmes’.

Die in dieser Skizze verwendeten Bezeichnungen werden teilweise auch in den folgenden Bedienungsbeschreibungen verwendet, sofern sie zur Hervorhebung der unterschiedlichen Ein- und Ausgabedateien notwendig sind.

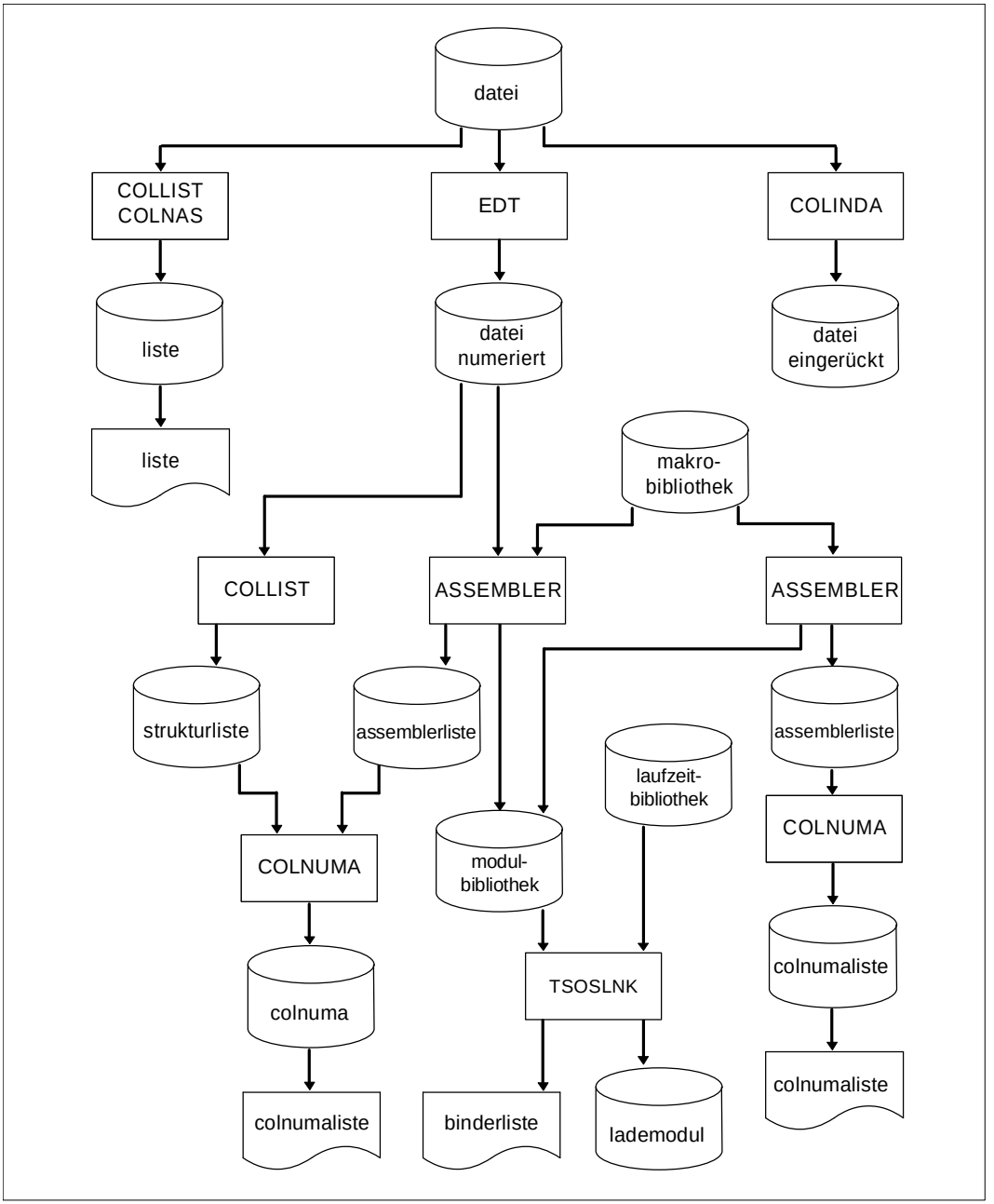


Bild 10-7: Datenfluß der strukturierten Assembler-Programme

10.3.1 Eingabe für COLLIST, COLNAS und COLINDA

Die Eingabe für die Dienstprogramme COLLIST, COLNAS und COLINDA können folgende strukturierte Quellprogramme sein:

- SAM-Dateien
- ISAM-Dateien
- Elemente einer PLAM-Bibliothek

Zu beachten ist, daß die ISAM-Dateien numerische, 8-Byte lange Schlüssel haben. Die Satzlänge ist variabel.

Die Zuweisung der Dateien und Bibliotheken erfolgt entweder über das FILE-Kommando oder über Parameter. Es gibt also folgende drei Möglichkeiten:

1. Mit dem LINK-Namen für SAM- und ISAM-Dateien nach dem Muster

```
/SET-FILE-LINK LINK-NAME=CINPUT,FILE-NAME=datei
```

2. Falls kein FILE-Kommando angegeben wurde, kann die Zuweisung über den Parameter CINPUT erfolgen:

```
           ⌈datei               ⌉
PAR CINPUT=⎨                    ⎬
           ⌊bibliothek(element) ⌋
```

3. Falls die Eingabe aus dem Element einer PLAM-Bibliothek erfolgt, kann die Zuweisung über den LINK-Namen SRCLIB und den Parameter SRCELEM erfolgen:

```
/SET-FILE-LINK LINK-NAME=SRCLIB,FILE-NAME=bibliothek
PAR SRCELEM=element
```

10.3.2 Ausgabe von COLLIST und COLNAS

- Die Ausgabe der Listen von COLLIST und COLNAS erfolgt in eine SAM-Datei, die standardmäßig durch das Suffix "CLIST" gekennzeichnet ist. Ist also "datei" die strukturierte Quellprogramm-Datei, dann erfolgt die Ausgabe in:

```
datei.CLIST
```

- Die Ausgabedatei kann über das FILE-Kommando und den LINK-Namen CLIST nach folgendem Schema festgelegt werden:

```
/SET-FILE-LINK LINK-NAME=clist,FILE-NAME=ausgabedatei
```

- Die Zuweisung der Ausgabedatei kann auch über Parameter erfolgen:

```
PAR CLIST=ausgabedatei
```

- In eine PLAM-Bibliothek erfolgt die Ausgabe über die Parameterzuweisung

```
PAR CLIST=bibliothek(element)
```

- Die ausgegebenen Listen können ausgedruckt werden mit dem Kommando:

```
/PRINT-FILE FILE-NAME=ausgabedatei,DELETE-FILE=YES,-
LAYOUT-CONTROL=PAR(FORM-NAME=format, CHARACTER-SETS=chars,-
CONTROL-CHARACTERS=EBCDIC)
```

10.3.3 Ausgabe COLINDA

- Die Ausgabe des von COLINDA erzeugten eingerückten Quellprogramms erfolgt standardmäßig in eine SAM-Datei, die durch das Suffix COUT gekennzeichnet ist. Ist also "datei" der Name des strukturierten Quellprogramms, dann ist die Ausgabedatei:

  ```
  datei.COUT
  ```

- Die Ausgabedatei kann ebenfalls über das FILE-Kommando und LINK-Namen zugewiesen werden:

  ```
  /SET-FILE-LINK LINK-NAME=COUTPUT,FILE-NAME=ausgabedatei
  ```

- Die Ausgabedatei kann auch über Parameter zugewiesen werden:

  ```
  PAR COUTPUT=ausgabedatei
  ```

- Soll die Ausgabe in das Element einer PLAM-Bibliothek erfolgen, dann erfolgt die Zuweisung über Parameter:

  ```
  PAR COUTPUT=bibliothek(element)
  ```

- Die ausgegebene Datei dient als Eingabe für den nachfolgenden Assemblerlauf. Damit wird ein Assemblerprotokoll in eingerückter Form erzeugt. Wir empfehlen nicht, die erzeugte Datei auszudrucken, denn die damit gewonnene Information wird besser durch COLLIST oder nach der Bearbeitung mit COLNUMA dargestellt. Die erzeugte Datei kann aber (nach eventuellen Korrekturen) erneut als Eingabe für COLINDA dienen.

Zusammenfassung

Aufgrund der Erläuterungen der vorangegangenen Abschnitte lassen sich folgende Kommandos zur Bedienung der Dienstprogramme COLLIST, COLNAS und COLINDA zusammenstellen:

Eingabe- und Ausgabedateien sind BS2000-Dateien

```
/LOGON ...
/SET-FILE-LINK LINK-NAME=CINPUT,FILE-NAME=datei

[/SET-FILE-LINK LINK-NAME=COUTPUT,FILE-NAME=datei-eingerückt]          (1)
[/SET-FILE-LINK LINK-NAME=CLIST,FILE-NAME=liste]                       (2)
[/ASSIGN-SYSDTA TO-FILE=parameter]                                     (3)

             ⌈ASSEMBH.COLLIST⌉
/START-PROG  {ASSEMBH.COLNAS  }
             ⌊ASSEMBH.COLINDA⌋
.
.
.
Parametereingabe
.
.
.
[/ASSIGN-SYSDTA=*PRIMARY]                                              (3)

/PRINT-FILE FILE-NAME=ausgabedatei,DELETE-FILE=YES,-
LAYOUT-CONTROL=PAR(FORM-NAME=format, CHARACTER-SETS=chars,-
CONTROL-CHARACTERS=EBCDIC)
/LOGOFF
```

(1) bei COLINDA

(2) bei COLLIST und COLNAS

(3) wahlweise wenn Parameter über SYSDTA eingegeben werden (Einzelheiten im Abschnitt 10.3.4)

Eingabe- und Ausgabedateien sind Elemente einer PLAM-Bibliothek

```
/LOGON ...
[/SET-FILE-LINK LINK-NAME=SRCLIB,FILE-NAME=bibliothek]
[/ASSIGN-SYSDTA TO-FILE=parameter]                                    (3)
.
.
.
             ⌈ASSEMBH.COLLIST⌉
/START-PROG  {ASSEMBH.COLNAS  }
             ⌊ASSEMBH.COLINDA⌋
.
.
.
[PAR SRCELEM=eingabe-element]
[PAR CINPUT=eingabe-bibliothek(element)]
[PAR CLIST=ausgabe-bibliothek(element)]                               (1)
[PAR COUTPUT=ausgabe-bibliothek(element)]                             (2)
.
.
weitere Parametereingabe
.
.
[/ASSIGN-SYSDTA=*PRIMARY]                                             (3)

/PRINT-FILE FILE-NAME=ausgabedatei,DELETE-FILE=YES,-                  (4)
LAYOUT-CONTROL=PAR(FORM-NAME=format, CHARACTER-SETS=chars,-
CONTROL-CHARACTERS=EBCDIC)
/LOGOFF
```

(1) bei COLLIST und COLNAS

(2) bei COLINDA

(3) wahlweise wenn Parameter über SYSDTA eingegeben werden (Einzelheiten im Abschnitt 10.3.4)

(4) Wenn die Ausgabe in eine Bibliothek gelegt wird, muß das Element vor dem Ausdruck in eine Datei gebracht werden.

10.3.4 Steuerung von COLLIST, COLNAS und COLINDA

Die Dienstprogramme COLLIST, COLNAS und COLINDA lassen folgende Möglichkeiten offen, die der Anwender durch Parameter steuern kann.

- Zuweisung von Dateien, PLAM- oder LMS-Elementen für die Eingabe (CINPUT).
- Zuweisung von PLAM- oder LMS-Elementen für die Eingabe (SRCELEM).
- Zuweisung von Dateien oder PLAM-Elementen für die Ausgabe (COUTPUT).
- Zuweisung von Dateien oder PLAM-Elementen für die Ausgabe von Listen (CLIST).
- Anpassung der internen Speicherverwaltung an die maximale Prozedurgröße (PROCSIZE).
- Ersetzen der standardmäßigen Syntaxzeichen

  ```
  @    :  *  ,
  ```

  durch andere (DELIM).
- Festlegung des Einrückbetrages (INDAMT).

Nur bei COLLIST und COLNAS

- Die Ausgabe von Anweisungen der Zielsprache wird ganz oder teilweise unterdrückt (STATMENT).
- Die Ausgabe von Kommentarzeilen wird unterdrückt. Es werden nur die Anweisungen der Zielsprache in den Strukturblöcken ausgegeben (COMMENT).
- Werden sowohl Anweisungen der Zielsprache als auch Kommentarzeilen unterdrückt, so werden nur die Strukturanweisungen in ihrer Aneinanderreihung und Schachtelung ausgegeben.
- Unterdrücken der Ausgabe von Strukturblöcken ab einer angegebenen Schachtelungstiefe (LEVLIM).
- Länge der Zeile; es können schmalere und breitere Seiten erzeugt werden (z.B. DIN-Formate) (LSTCOL).
- Festlegen der maximal zulässigen Länge der Eingabesätze (RECLEN).
- Variable Zeilenkennzeichnung (LINEID).
- Steuern des Seitenvorschubs (LINELIM).
- Ausgabesteuerung der Seitenüberschrift (HEADLINE).

Nur bei COLLIST

– Anfordern von Struktur- und/oder Prozedurlisten (LIST).

– Wiederholte Ausgabe von Unterstrukturen in der Prozedurliste (FULPCLST).

Nur bei COLNAS

– Unterdrücken der Ausgabe von Strukturblöcken bis zu einer angegebenen Schachtelungstiefe (LEVBEG).

Nur bei COLINDA

– Durchnumerierung der Ausgabezeilen (RENUM).

Die Ein- und Ausgabe von bzw. in Dateien kann auch mit Linknamen gesteuert werden (siehe 10.3.1 und 10.3.3).

10.3.5 Parameter

Die Programme werden über Parameter gesteuert, die man aber nur in Ausnahmefällen angeben muß. Im Normalfall gelten für die Parameter Standardwerte.

Die Parameter können entweder im Dialog- oder Stapel-Betrieb eingegeben werden. Der Benutzer steuert die Eingabe im Dialog-Betrieb mit der Beantwortung der Terminal-Anforderung:

```
PARAMS? (STANDARD/SYSDTA/DIALOG)
```

Bedeutung der Antworten:

STA[NDARD] Keine weitere Parametereingabe. Standardwerte werden für alle Parameter angenommen.

[SYS[DTA]] Im Dialogmodus werden Parameteranweisungen des Formats

$\texttt{PAR param}_1\texttt{=wert}_1\texttt{,param}_2\texttt{=wert}_2\texttt{...}$

bis zur END-Anweisung von SYSDTA gelesen. Statt der Antwort SYS[DTA] kann man sofort Parameteranweisungen eingeben.

Falls für alle Parameter Standardwerte gelten sollen, genügt die Eingabe von END.

Die Antwort SYS[DTA] ist für den Fall gedacht, daß die Parameter in einer Datei enthalten sind, die vor dem Aufruf des Programms mit /ASSIGN-SYSDTA=parameter zugewiesen wurden.

DIA[LOG] Im Dialog werden über das Datensichtgerät mit dem (WRTRD-Makro) die einzelnen Parameter angefordert mit der Frage:

$param_i$? (Kurzbeschreibung)

Antwort:

$wert_i$

Neben den gewünschten Werten sind auf die Frage "param?" auch die Angaben STANDARD (Standardwerte für den gefragten Parameter) und END (Standardwert für den gefragten und alle folgenden Parameter) zugelassen. Wenn man nicht mit END antwortet, werden alle Parameter abgefragt.

Im Stapel-Betrieb (ENTER-Prozesse) sind Parameteranweisungen in der Form

PAR $param_1$=$wert_1$,$param_2$=$wert_2$,.....

beendet durch

END

in SYSDTA, d.h. normalerweise unmittelbar hinter dem /START-PROG-Kommando, bereitzustellen. Soll mit Standardparametern gearbeitet werden, so ist nur eine END-Anweisung anzugeben.

Beschreibung der Parameter

Die folgende Tabelle zeigt, welche Parameter für die einzelnen Programme (bzw. Funktionen) gelten. Anschließend ist die Bedeutung der Parameter läutert.

Parameter	Wertebereich	COLINDA	COLLIST STR	COLLIST PRC	COLNAS
CINPUT={datei / bibliothek(element)}		X	X	X	X
CLIST={datei / bibliothek(element)}			X	X	X
COMMENT={YES / NO}			X		X
COUTPUT={datei / bibliothek(element)}		X			
DELIM={'abcd' / '@:*,'}		X	X	X	X
FULPCLST={YES / NO}				X	
HEADLINE={PROC / PAGE}			X		X
INDAMT=n	1≤n≤8	X(n=3)	X(n=4)X		X(n=4)
LEVBEG={n / 1}	1≤n≤20				X
LEVLIM={n / 20}	1≤n≤20		X		X

Parameter	Wertebereich	COLINDA	COLLIST STR	COLLIST PRC	COLNAS
LINELIM={n / 64}	0≤n≤144		X	X	X
LINEID={YES / NO / KEY}			X	X	X
LIST={STR / PRC / ALL}			X	X	
LSTCOL={n / 100}	52≤n≤240		X	X	X
PROCSIZE={n / 250}	10≤n≤4000	X	X	X	X
RECLEN={n / 80}	80≤n≤255		X	X	X
RENUM={YES / NO}		X			
SRCELEM=element		X	X	X	X
STATMENT={YES / NO / CON}			X		X

Innerhalb der geschweiften Klammern sind die möglichen Parameterwerte angegeben. Die Standardwerte sind unterstrichen.

Bedeutung der Parameter

CINPUT — Mit dem Parameter CINPUT können nach Aufruf eines Dienstprogrammes Dateien oder Elemente einer PLAM- oder LMS-Bibliothek zugewiesen werden.

CLIST — Datei oder Element einer PLAM-Bibliothek, wohin die Ausgabe von COLLIST und COLNAS erfolgen soll.

COMMENT — Bei COMMENT=YES werden Kommentare in die Ausgabe übernommen.

COUTPUT — Datei oder Element einer PLAM-Bibliothek, wohin die Ausgabe von COLINDA erfolgen soll.

DELIM — 4 abdruckbare Zeichen als Ersatz für die Delimiter in der Reihenfolge @ : * ,

Es bedeuten:

@ Führendes Zeichen von Strukturwörtern.

: Vorschubsteuerung bei COLLIST in Verbindung mit dem Kommentarzeichen.

* Kommentarkennzeichen.

, Trennzeichen für Parameter.

Die Zeichen "@", "*" und "," dürfen nur im Pseudocode, nicht im strukturierten Assembler-Programm verändert werden.

Auch wenn nicht alle 4 Delimiter ersetzt werden, sind 4 Zeichen anzugeben.

Bei der Anforderung des Wertes für DELIM im Dialog durch das Programm entfallen die Hochkommata, die sonst die 4 Zeichen einschließen müssen.

FULPCLST — Steuerung der Prozedurlistenausgabe von COLLIST:

=YES — Volle Prozedurliste (Standard).

=NO — Beim wiederholten Aufruf einer im Programm enthaltenen Prozedur wird deren Unterstruktur nicht mehr ausgegeben (Kennzeichnung durch **).

HEADLINE — Ausgabesteuerung der Seitenüberschrift

=PROC — Ausgabe der Seitenüberschrift auf der ersten Seite und auf jeder Seite, die mit einer neuen Prozedur beginnt.

=PAGE — Ausgabe der Überschrift auf jeder Seite unabhängig davon, wodurch der Seitenvorschub ausgelöst wurde.

Ein Seitenvorschub kann erfolgen, wenn

- eine neue Prozedur oder ein Programmteil außerhalb von Prozeduren beginnt;
- eine Prozedur oder ein Programmteil mehr Zeilen hat, als im Parameter LINELIM vorgegeben;
- wenn der Anwender mit *: einen Vorschub erzwingt.

INDAMT — Numerischer Wert zwischen 1 und 8 für den Einrückbetrag.

LEVBEG — Numerischer Wert zwischen 1 und 20.
1. Stufe der Verschachtelung, ab der gedruckt werden soll.
Der Parameter LEVBEG gilt nur für COLNAS.

LEVLIM — Numerischer Wert zwischen 1 und 20 (Standardwert 20).
Letzte Stufe der Verschachtelung, die ausgedruckt werden soll.

LINEID

=YES — Als Zeilenkennzeichnung gilt Spalte 73-80 der Eingabe. Diese wird von COLLIST an den rechten Rand der Liste gesetzt. COLNAS gibt die Zeilenkennzeichnung nicht aus.

Der Parameter RECLEN muß den Wert 80 haben; andernfalls erscheint eine Warnung und LINEID wird von COLLIST und COLNAS auf NO gesetzt.

=NO — Spalte 73-80 der Eingabe werden nicht als Zeilenkennzeichnung, sondern als Bestandteil des Programmtextes betrachtet. Die Dienstprogramme COLLIST und COLNAS verarbeiten die gesamte Zeile.

COLLIST verschiebt in der Strukturliste den senkrechten Strich am rechten Rand ganz nach rechts, so daß mehr Platz für den Text zur Verfügung steht.

=KEY — Als Zeilenkennzeichnung gilt der Schlüssel des betreffenden Eingabesatzes. Bei ISAM-Dateien ist dies der ISAM-Schlüssel (nur Keypos=5 und Keylen=8 ist erlaubt), bei SAM-Dateien besteht der Schlüssel aus den ersten 8 Zeichen des Satzes (kann im EDT mit "@WRITE'eingabe'KEY" erzeugt werden). Der Schlüssel wird auf numerischen Inhalt geprüft. Ist der Schlüssel nicht numerisch, so wird eine Fehlermeldung ausgegeben.

COLLIST und COLNAS verarbeiten den gesamten Zeileninhalt (nach dem Schlüssel). COLLIST gibt den Schlüssel in der Strukturliste neben dem senkrechten Strich am rechten Rand aus.

LINELIM

LINELIM=0 als Untergrenze.
LINELIM=64 als Standardwert.
LINELIM=144 als Obergrenze für den Wert von nn.

Festlegung, nach jeweils wievielen Zeilen in der Struktur- und Prozedurliste (Ausgabe von COLLIST) sowie im Struktogramm (Ausgabe von COLNAS) das Programm eine neue Seite beginnen soll, wenn nicht vorher durch den Anfang oder das Ende einer Prozedur sowie bei COLLIST durch "*:" in den Spalten 1-2 ein Seitenvorschub erzwungen wurde.

Dieser automatische Seitenvorschub nach nn Zeilen unterbleibt, wenn LINELIM den Wert 0 hat.

LIST

Steuerung der Listenfunktion von COLLIST:

=STR

Die Strukturliste wird erstellt (Standard).

=PRC

Die Prozedurliste wird erstellt (falls keine Strukturfehler).

=ALL

Beide Listentypen werden erzeugt (falls Strukturfehler: nur Strukturliste).

LSTCOL

Numerischer Wert zwischen 52 und 240 (Standardwert 100).
Letzte Druckstelle in der Zeile.

Ein Wert über 132 sollte nur angegeben werden, wenn ein entsprechender Drucker zur Verfügung steht.

PROCSIZE

Numerischer Wert zwischen 10 und 4000 (Standardwert 250). Der Parameter bestimmt die Größe des Speicherbereichs für interne Listen.
PROCSIZE wird in Schritten von 200 aufgerundet und für jedes Vielfache von 200 eine Seite (4 KB) virtueller Speicher angefordert.
Maßgebend für die Größe des Speicherbereichs, der für die internen Listen benötigt wird, ist die Anzahl der der Strukturanweisungen in einer Prozedur. Der Wert X des Parameters PROCSIZE kann überschlägig nach folgender Formel berechnet werden:

```
X = 3 * S
```

wobei S die Anzahl der Strukturanweisungen ist.

Bei Struktogrammen entspricht der Wert von PROCSIZE, also ungefähr der Anzahl der Quellprogrammzeilen der größten Prozedur (@ENTR bei @END).

Die Prozedur mit dem höchsten Wert für X bestimmt den Wert des Parameters PROCSIZE. Der Standardwert 250 dürfte also in den meisten Fällen ausreichen, zumal der Speicher stets in Abschnitten von 4 KB angefordert wird. Sollte der Platz nicht ausreichen, so bringt COLLIST eine Meldung mit einem empfohlenen Wert für PROCSIZE.

RECLEN — RECLEN=80 als Untergrenze und Standardwerte.
RECLEN=255 als Obergrenze.

Mit dem Parameter RECLEN kann die maximal zulässige Länge der Eingabesätze gesteuert werden, damit bei der Arbeit mit Pseudocode auch Sätze mit mehr als 80 Zeichen verwendet werden können.

Liegt der Wert von RECLEN über 80, so wird der Inhalt von Spalte 73-80 nicht mehr als Zeilenkennzeichnung aufgefaßt, sondern als Bestandteil des Programmtextes behandelt.
In diesem Fall darf der Parameter LINEID nur die Werte NO und KEY haben.

RENUM — Numerierung der Ausgabe von COLINDA in den Spalten 73-80.

=YES — Fortlaufende Numerierung der Zeilen in einer Schrittweite von 10.

=NO — Der Inhalt von Spalte 73-80 der Eingabezeile wird in die Ausgabezeile übernommen.

SRCELEM — Mit diesem Parameter wird das Element einer Bibliothek für die Eingabe bestimmt, nachdem die Bibliothek mit dem Linknamen SCRLIB zugewiesen worden ist.

STATMENT — Übernahme von Zielspracheanweisungen in die Ausgabe. Bei STATMENT=CON werden nur die Bedingungen, d.h. die Zielsprachetexte zwischen @IF und @THEN, @WHILE und @DO, @THRU und @DO, @WHEN und @BREAK sowie @CASE2 und dem 1. @OF ausgegeben.

Beispiele

Dialogbetrieb

Arbeiten mit den Standardwerten (für alle Parameter):

```
/SET-FILE-LINK LINK-NAME=CINPUT,FILE-NAME=TEST-PROGRAMM
/START-PROG ASSEMBH.COLLIST
%  BLS0500 PROGRAM 'COLLIST', VERSION '41B11' OF '1991-05-16' LOADED
%  BLS0551 COPYRIGHT (C) SNI 1991. ALL RIGHTS RESERVED
COLLIST VERSION 41B11 - 01.12.91 STARTED
PARAMS?(STANDARD/SYSDTA/DIALOG)
*STA
COLLIST COMPLETED
```

Eingabe von aktuellen Parametern aus einer Datei, die über SYSDTA eingelesen wird:

```
/START-PROG $EDT
%  BLS0500 PROGRAM 'EDT', VERSION '16.4A' OF '1992-06-24' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1992. ALL
   RIGHTS RESERVED
@EDT
   1.
PAR INDAMT=6
   2.
PAR RENUM=NO
   3.
END
   4.
@W'TEST.PARAMETER'
   4.
@H
EDT NORMAL END
/SET-FILE-LINK LINK-NAME=CINPUT,FILE-NAME=TEST-PROGRAMM
/ASSIGN-SYDTA TO-FILE=TEST.PARAMETER
/START-PROG ASSEMBH.COLINDA
%  BLS0500 PROGRAM 'COLINDA', VERSION '22F11' OF '1991-02-11' LOADED
%  BLS0551 COPYRIGHT (C) SNI 1991. ALL RIGHTS RESERVED
COLINDA VERSION 2.2F11 - 01.12.91 STARTED
PARAMS?(STANDARD/SYSDTA/DIALOG)
COLINDA COMPLETED
/ASSIGN-SYSDTA TO-FILE=*PRIMARY
```

Eingabe von aktuellen Parametern im Dialogmodus:

```
/START-PROG ASSEMBH.COLLIST
%  BLS0500 PROGRAM 'COLLIST', VERSION '41B11' OF '1991-05-16' LOADED
%  BLS0551 COPYRIGHT (C) SNI 1991. ALL RIGHTS RESERVED
COLLIST VERSION 41B11 - 01.12.91 STARTED
PARAMS?(STANDARD/SYSDTA/DIALOG)
*DIA
CINPUT?(NAME OF COLUMBUS-INPUT)
TEST.PROGRAMM
COMMENT?(YES/NO)
YES
STATMENT?(YES/NO/CON)
CON
PROCSIZE?(MAX SIZE OF PROCEDURES IN NO OF STMTS)
STA
LSTCOL?(LAST COLUMN IN LISTING)
60
LINELIM?(LIMIT OF LINES PER PAGE)
END
COLLIST COMPLETED
```

Eingabe von aktuellen Parametern über PAR-Anweisungen:

```
/SET-FILE-LINK LINK-NAME=CINPUT,FILE-NAME=TEST-PROGRAMM
/START-PROG ASSEMBH.COLNAS
%  BLS0500 PROGRAM 'COLNAS', VERSION '41B11' OF '1991-05-17' LOADED
%  BLS0551 COPYRIGHT (C) SNI 1991. ALL RIGHTS RESERVED
COLNAS VERSION 41B11 - 01.12.91 STARTED
PARAMS?(STANDARD/SYSDTA/DIALOG)
*PAR LSTCOL=80,INDAMT=6
*PAR STATMENT=CON
*END
COLNAS COMPLETED
```

Stapelbetrieb

Im Stapelbetrieb (ENTER-Prozeß) hat der Benutzer folgende Möglichkeiten:

- Arbeiten mit den Standardwerten; hierzu muß dem START-PROG-Kommando zum Laden und Starten des Dienstprogramms unmittelbar die Anweisung "END" folgen.
- Zuweisen von aktuellen Parameterwerten; hierzu sind die aktuellen Werte in Form von PAR-Anweisungen, abgeschlossen durch die Anweisung "END" (Format siehe oben in diesem Abschnitt), über SYSDTA bereitzustellen, d.h. normalerweise unmittelbar im Anschluß an das START-PROG-Kommando.

Das folgende Beispiel zeigt den Aufbau einer ENTER-Datei, wobei bestimmten Parametern aktuelle Werte zugewiesen werden.

```
/LOGON ...
/SET-FILE-LINK LINK-NAME=CINPUT,FILE-NAME=ERB.CON
/START-PROG ASSEMBH.COLLIST
PAR LSTCOL=80,INDAMT=6
PAR STATMENT=CON
END
/REMOVE-FILE-LINK LINK-NAME=CINPUT
/LOGOFF
```

10.4 Bedienung des Dienstprogramms COLNUMA

Die Funktion, welche COLNUMA ausüben soll, wird durch die Eingabedateien bestimmt. Gestartet wird COLNUMA mit dem Kommando

```
/START-PROG ASSEMBH.COLNUMA
```

Die Dateien und die Elemente der PLAM-Bibliotheken für die Ein- und Ausgabe werden über Linknamen oder Parameter zugewiesen.

10.4.1 Erweiterung der Strukturliste

Eingabe

Für die Eingabe müssen eine Strukturliste, die von COLLIST erzeugt wurde, und eine Assemblerliste, die vom ASSEMBH erzeugt wurde, zur Verfügung stehen.

Für die Strukturliste stehen folgende Zuweisungsmöglichkeiten zur Verfügung:

Linkname:

```
SET-FILE-LINK LINK-NAME=CLIST,FILE-NAME=datei
```

Parameter:

```
          ⎧datei               ⎫
PAR CLIST=⎨                    ⎬
          ⎩bibliothek(element) ⎭
```

Für die Assemblerliste stehen folgende Zuweisungsmöglichkeiten zur Verfügung:

Linkname:

```
SET-FILE-LINK LINK-NAME=ASMLST,FILE-NAME=datei
```

Parameter:

```
           ⎧datei               ⎫
PAR ASMLST=⎨                    ⎬
           ⎩bibliothek(element) ⎭
```

Hinweise

– Die Eingabe-Assemblerliste muß mit folgender Option erstellt worden sein:
  LISTING=PAR(LAYOUT=PAR(FORMAT=F-ASSEMB-COMPATIBLE))

  Wird die Eingabe-Assemblerliste aus einer Bibliothek eingegeben, so muß dieses Bibliothekselement vom Typ P sein.

– Das strukturierte Assembler-Quellprogramm, von dem die beiden Eingaben von COLNUMA - die Strukturliste und die Assemblerliste - stammen, muß vor Verarbeitung durch COLLIST bzw. Assembler mit dem Programm $EDT in den Spalten 73-80 numeriert worden sein. Diese Nummern werden sowohl in die Strukturliste als auch in die Assemblerliste übernommen und stellen den Bezug zwischen beiden Listen her.

– Die COLLIST-Strukturliste muß mit dem Parameter LSTCOL=100 (Standardwert) erzeugt worden sein.

Ausgabe

Die Ausgabe der erweiterten Strukturliste in eine SAM-Datei oder in das Element einer PLAM-Bibliothek kann folgendermaßen zugewiesen werden:

Linkname:

```
SET-FILE-LINK LINK-NAME=EWCLIST,FILE-NAME=ausgabedatei
```

Parameter:

```
                ⌈datei               ⌉
PAR EWCLIST=    |                    |
                ⌊bibliothek(element) ⌋
```

Die Parametereingabe wird mit END abgeschlossen.

Zusammenfassung

Mit den folgenden Kommandos können die Eingabedateien für COLNUMA erstellt und COLNUMA ausgeführt werden:

```
/LOGON
/START-PROG $EDT
@READ'datei'
@SEQ
@W'datei-numeriert'
@H
/SET-FILE-LINK LINK-NAME=CINPUT,FILE-NAME=datei-numeriert
/SET-FILE-LINK LINK-NAME=CLIST,FILE-NAME=strukturliste
/START-PROG ASSEMBH.COLLIST
*STA
/DELETE-SYSTEM-FILE OMF
/START-PROG $ASSEMBH
//COMPILE SOURCE=datei-numeriert,-
//        LISTING=PAR(LAYOUT=PAR(FORMAT=F-ASSEMB-COMPATIBLE),-
//        PAR(OUTPUT=assemblerliste)),-
//        MACRO-LIBRARY=makrobibliothek
//END
/SET-FILE-LINK LINK-NAME=CLIST,FILE-NAME=strukturliste
/SET-FILE-LINK LINK-NAME=ASMLST,FILE-NAME=assemblerliste
/SET-FILE-LINK LINK-NAME=EWCLIST,FILE-NAME=colnuma-liste
/START-PROG ASSEMBH.COLNUMA
[PAR CLIST=strukturliste]
[PAR ASMLST=assemblerliste]
[PAR EWCLIST=colnuma-liste]
[END]
/DELETE-FILE datei-numeriert
/DELETE-FILE assemblerliste
/DELETE-FILE strukturliste
/PRINT-FILE FILE-NAME=ausgabedatei,DELETE-FILE=YES,-
LAYOUT-CONTROL=PAR(FORM-NAME=format, CHARACTER-SETS=chars,-
CONTROL-CHARACTERS=EBCDIC)
```

10.4.2 Ergänzung der Assemblerliste eines von COLINDA aufbereiteten Programms

Eingabe

Das strukturierte Assembler-Quellprogramm muß zuerst durch COLINDA bearbeitet werden (siehe 10.3.1 und 10.3.3). Das aufbereitete Programm wird dann assembliert. Die Assemblerliste, erzeugt durch ASSEMBH auf SYSLST, ist die Eingabe für COLNUMA. Die Eingabe kann über Linknamen oder Parameter zugewiesen werden:

Linkname:

```
SET-FILE-LINK LINK-NAME=ASMLST,FILE-NAME=datei
```

Parameter:

```
              ⎧datei                ⎫
PAR ASMLST=   ⎨                     ⎬
              ⎩bibliothek(element)  ⎭
```

Hinweis

– Die Eingabe-Assemblerliste muß mit folgender Option erstellt worden sein:
  LISTING=PAR(LAYOUT=PAR(FORMAT=F-ASSEMB-COMPATIBLE))

  Wird die Eingabe-Assemblerliste aus einer Bibliothek eingegeben, so muß dieses Bibliothekselement vom Typ P sein.

Ausgabe

Die Assemblerliste wird durch senkrechte Striche und Seitenüberschriften ergänzt. Die Zuweisung der Datei bzw. des Elements einer PLAM-Bibliothek für die Ausgabe erfolgt entweder über Linknamen oder Parameter.

Linkname:

```
SET-FILE-LINK LINK-NAME=EWCLIST,FILE-NAME=ausgabedatei
```

Parameter:

```
              ⎧datei                ⎫
PAR EWCLIST=  ⎨                     ⎬
              ⎩bibliothek(element)  ⎭
```

Die Parametereingabe wird mit END abgeschlossen.

Zusammenfassung

Mit folgenden Kommandos kann die Eingabe vorbereitet und COLNUMA ausgeführt werden:

```
/LOGON
/SET-FILE-LINK LINK-NAME=CINPUT,FILE-NAME=datei
/SET-FILE-LINK LINK-NAME=COUTPUT,FILE-NAME=datei-eingerückt
/START-PROG ASSEMBH.COLINDA
*PAR param=wert,...
*END
/DELETE-SYS-FILE OMF
/START-PROG $ASSEMBH
//COMPILE SOURCE=datei-eingerückt,-
//        LISTING=PAR(LAYOUT=PAR(FORMAT=F-ASSEMB-COMPATIBLE),-
//        PAR(OUTPUT=assemblerliste)),-
//        MACRO-LIBRARY=makrobibliothek
//END
[/SET-FILE-LINK LINK-NAME=ASMLST,FILE-NAME=assemblerliste]
[/SET-FILE-LINK LINK-NAME=EWCLIST,FILE-NAME=colnuma-liste]
/START-PROG ASSEMBH.COLNUMA
[PAR ASMLST=assemblerliste]
[PAR EWCLIST=datei-eingerückt]
[END]
/DELETE-FILE datei-eingerückt
/DELETE-FILE assemblerliste
/PRINT-FILE FILE-NAME=colnuma-liste,DELETE-FILE=YES,-
LAYOUT-CONTROL=PAR(FORM-NAME=format, CHARACTER-SETS=chars)
```

10.4.3 Parameter

Die folgende Tabelle zeigt, welche Parameter für COLNUMA gelten. Anschließend ist die Bedeutung der Parameter erläutert.

Parameter
ASMLST=⎧datei⎫ ⎩bibliothek(element)⎭
EWCLST=⎧datei⎫ ⎩bibliothek(element)⎭
CLIST=⎧datei⎫ ⎩bibliothek(element)⎭

ASMLST — Datei oder Element einer PLAM-Bibliothek, in der die Assemblerliste steht.

EWCLST — Datei oder Element einer PLAM-Bibliothek, wohin die Ausgabe von COLNUMA erfolgen soll.

CLIST — Datei oder Element einer PLAM-Bibliothek, in der die von COLLIST erzeugte Strukturliste steht.

10.5 Meldungen der Dienstprogramme

Innerhalb der strukturierten Programmierung gibt es drei Klassen von Fehlermeldungen. Es handelt sich dabei um

- Bedienungsfehler- und Systemmeldungen,
- Fehlermeldungen in bezug auf Verstöße gegen die Syntax der strukturierten Programmierung und
- Fehlermeldungen, die beim Ablauf eines strukturierten Programms auftreten können.

10.5.1 Bedienungsfehler- und Systemmeldungen

Bedienungsfehler- und Systemmeldungen werden auf SYSOUT ausgegeben.

COLLIST, COLNAS und COLINDA

In der folgenden Tabelle sind die Fehlernummern für die einzelnen Komponenten zusammengefaßt, soweit sie für die strukturierte Programmierung relevant sind.

Es bedeuten:

PR Parameter Behandlung
IO Primäreingabe und -ausgabe
RQ Speicheranforderung
IL Zwischensprache
PL Prozedurlisten
SH String Handling

Meldung nnn iiii cn xxx	Bedeutung	Auswirkung
001 - RQ	Der mit dem Parameter PROCSIZE ermittelte Speicherbereich steht nicht zur Verfügung. Das Programm wird beendet.	Abbruch

Meldung nnn iiii cn xxx	Bedeutung	Auswirkung
003 - IL	Der mit dem Parameter PROCSIZE ermittelte Speicherbedarf reicht nicht aus, um eine Prozedur abzuarbeiten. Das Programm wird beendet.	Abbruch
005 - PR	Systemfehler bei RDATA-Makro.	Abbruch
006 - PR	Systemfehler bei WRTRD-Makro.	
007 - PR	Systemfehler bei WROUT-Makro.	
011 - PR Parameteranweisung	Fehlerhafte Parameteranweisung.	Die Standardwerte der Parameter werden angenommen.
014 - PR Parameteranweisung	Fehlerhafte Parameteranweisung RECLEN > 80 und LINEID = YES	Parameter LINEID wird von den Dienstprogrammen COLLIST und COLNAS auf NO gesetzt.
001 u) IO y) Name der Bibliothek	Fehler beim Offnen der Bibliothek	Abbruch.
003 u) IO y) Name der Bibliothek	Fehler beim Schließen der Bibliothek	
004 u) IO y) Name von Element und Bibliothek	Fehler beim Lesen eines Satzes aus einem Bibliothekselement	

Meldung nnn iiii cn xxx	Bedeutung	Auswirkung
005 u) IO y) Name von Element	Fehler beim Schreiben eines Satzes in ein Bibliothekselement	Abbruch
007 - IO	Systemfehler bei WROUT-Makro.	Abbruch.
008 - IO	Unzulässiger Schlüssel im Eingabesatz.	
009 x1) IO	DMS-Fehler.	
010 x1) IO	DMS-Fehler. Fehler beim FILE-Makro.	
014 - IO	Unzulässiger Dateiname für die Ausgabedatei erzeugt.	
023 u) IO y) Name von Bibliothek und Element	Fehler beim Öffnen eines Bibliothekselements	Abbruch
024 u) IO y) Name von Element	Fehler beim Schließen eines Bibliothekselements	
015 - PR Parameterzeile	Falsches Format für Datei- oder Bibliotheksname	Abbruch
016 - PR Parameterzeile	Falsches Format für Elementnamen	
017 - PR	CINPUT und SRCELEM angegeben	Die zuletzt eingebene Angabe gilt

Meldung nnn iiii cn xxx	Bedeutung	Auswirkung
016 - IO Eingabesatz	Eingabesatz länger als im Parameter RECLEN vorgegeben.	Der Satz wird abgeschnitten.
020 - PL	Zu viele Prozedurnamen (mehr als 300).	Primärprogramm verkleinern.
021 - PL	Zu viele @ENTR- und @PASS-Anweisungen.	Primärprogramm verkleinern.
022 - PL	Rekursiver Prozeduraufruf und Parameter FULPCLST=YES (Es wird eine Prozedurliste mit FULPCLST=NO erstellt).	
023 - PL	Mindestens eine @PASS-Anweisung steht außerhalb einer Prozedur.	Überprüfen des Primärprogramms, da Prozedurliste(n) möglicherweise fehlerhaft.
024 - PL	Mindestens 2 @ENTR-Anweisungen mit demselben Prozedurnamen treten auf. Kennzeichnung in der Prozedurliste: <Proz.-name> LN= **nn mit nn = Zeilennummer der 1. @ENTR-Anweisung	Überprüfen des Primärprogramms, da Prozedurliste(n) möglicherweise fehlerhaft.

Meldung nnn iiii cn xxx	Bedeutung	Auswirkung
025 - PL	Keine Prozedur auf "höchster" Stufe vorhanden (rekursiver Aufruf). Eine Prozedurliste wird erzeugt, die mit der 1. @ENTR-Anweisung beginnt.	
002 - SH	Fehler im String-Handling.	System-Kundendienst verständigen.
004 - SH	String-Speicher nicht ausreichend.	System-Kundendienst verständigen.

Erläuterung

```
         ERROR: nnn [iiii] COMPONENT: cn
        [PROGRAM IS TERMINATED]
        [xxxxxxxxxxx.........x]

nnn     Fehler-Nummer
iiii    zusätzliche Fehler-Bezeichnung des Systems
cn      Komponenten-Kurzbezeichnung
xxx     ergänzende Textzeile

x1)     DMS Code (siehe BS2000 Systemmeldungen, Beschreibung)
u)      PLAM/ILAM-Return-Code
y)      Bibliothekskennzeichen
           P   PLAM
           M   MLU, LMS (Version 1.0)
           C   COBLUR
           F   FMS
           U   undefiniert
```

Hinweis

Programmabschlußmeldungen, in denen auch mitgeteilt wird, ob Strukturfehler in dem strukturierten Quellprogramm aufgetreten sind, werden wie die Bedienungs- und Systemfehlermeldungen auf SYSOUT ausgegeben.

COLNUMA

Meldung	Bedeutung
NO CLIST PROGRAM IS TERMINATED	Keine Strukturliste zugewiesen und Programm nicht mit COLINDA aufbereitet
NO ASMLST PROGRAM IS TERMINATED	Keine Assemblerliste zugewiesen
WRONG NUMBERS IN COL. 73-80 INPUT FILE OR COLLIST PARAM LSTCOL NOT = 100 (STANDARD)	Eingabedatei nicht richtig numeriert oder Strukturliste zu breit oder zu schmal
END OF ASSEMBLER LISTING	Struktur- und Assemblerliste nicht vom selben Programm

10.5.2 Syntaxfehlermeldungen

Behandlung der Meldungen bei den einzelnen Dienstprogrammen

– COLLIST
  Die Meldungen über Strukturfehler werden bei COLLIST in der Liste an den Stellen dazwischengeschoben, an denen sie aufgetreten sind.

– COLNAS
  Bei Strukturfehlern innerhalb einer Prozedur wird von COLNAS für diese Prozedur kein Strukturdiagramm, sondern nur eine Fehlerliste ausgegeben. Warnungen werden nur ausgegeben, wenn in der aktuellen Prozedur auch ein Strukturfehler vorliegt.

– COLINDA
  Das Programm COLINDA gibt Strukturfehler und Warnungen als Kommentar-Zeilen an den Stellen, an denen sie auftreten, in das generierte eingerückte strukturierte Programm aus.

Format der Syntaxfehlermeldungen

```
⎧W⎫
⎩E⎭  aabb  zz....zz
```

W	Warning (Warnung)
E	Error (Fehler)
aa	Maximal zweistellige Zahl (führende Null unterdrückt), repräsentiert einen Strukturzustand, in dem nur bestimmte Schlüsselwörter zugelassen sind (Bedeutung siehe 10.5.3).
bb	Zweistellige Zahl, repräsentiert ein unzulässiges oder fehlendes Schlüsselwort: 01 in dem durch aa beschriebenen Zustand fehlt ein abschließendes @BEND oder @END (Hierarchiestufe nicht richtig beendet) 09-17 in dem durch aa beschriebenen Zustand tritt ein nicht zugelassenes Schlüsselwort auf.
zz....zz	Hinweis auf fehlerhafte Prozedur oder fehlerhaften Strukturblock.

Warnungen

Waabb	Zustand der Hierarchiestufe	Erwartete Schlüsselwörter auf gleicher Hierarchiestufe

Beispiel

W614	Zustand nach @CYCL Auf @CYCL folgt ein @BEND ohne zwischengestelltes @WHEN-@BREA (Abbruchbedingungen fehlen): Zählschleife oder Endlosschleife	@WHEN

10.5.3 Bedeutung von aabb in Syntaxfehlermeldungen

aa..	Zustand der Hierarchiestufe	Zugelassene Schlüsselwörter
1..	Anfangszustand bzw. Zustand zwischen @BEND und nächstem Strukturblockanfang	@BEGIN, @IF, @WHILE, @CASE, @CYCLE, @EXIT @PASS
2..	Zustand nach @ENTR	@END
3..	Zustand nach @BEGIN	@BEND, @WHEN
4..	Zustand nach @IF	@THEN
5..	Zustand nach @WHILE	@DO
6..	Zustand nach @CYCLE	@WHEN
7..	Zustand nach @CASE	@OF
8..	Zustand nach @THRU	@DO
9..	Zustand nach @ON	@DO
10..	Zustand nach @THEN	@ELSE, @BEND, @WHEN
11..	Zustand nach @ELSE	@BEND, @WHEN
12..	Zustand nach @DO	@BEND, @WHEN
13..	Zustand nach @WHEN	@BREAK
14..	Zustand nach @BREAK	@BEND, @WHEN
15..	Zustand nach @OF	@OF, @OFREST, @BEND
16..	Zustand nach @OFREST	@BEND

..bb	Fehlendes oder unzulässiges Schlüsselwort
..01	Abschließendes @BEND oder @END fehlt
	Unzulässige Schlüsselwörter (09-17)
..09	@THEN
..10	@ELSE
..11	@OF
..12	@OFREST
..13	@END
..14	@BEND
..15	@WHEN
..16	@BREAK
..17	@DO

10.6 Unterstützung von Monitorjobvariablen

Wird ein Dienstprogramm mit

/START-PROG ASSEMBH.COL... ,MONJV=jvname

aufgerufen,
wobei jvname der Name einer vom Anwender definierten Jobvariablen ist, so kann der Programmablauf überwacht werden, da die Dienstprogramme in Byte 4-7 der Jobvariablen einen Rückkehrcode hinterlegen.

Der Anwender kann also mit Jobvariablen Dialogprozeduren oder ENTER-Prozesse steuern. Die folgende Tabelle beschreibt den Zusammenhang zwischen Fehlergewicht und Rückkehrcode in der Jobvariablen. Es werden sowohl Bedienungsfehler als auch Fehler in der Presource (z.B. Strukturfehler) ausgewertet.

Belegung der Jobvariablen:

Fehlerklasse	Beendigung	Rückkehrcode in der Jobvariablen
ohne Fehler	normal	0000
Warnung	normal	1003
leichter Fehler	normal	2004
schwerer Fehler	sofort	2005

11 Anhang

11.1 Meldungen des ASSEMBH

Die Meldungen werden nach folgendem Schema dargestellt:

```
Message-Nummer  Flag Gewicht      1.Zeile
Message-Nummer  Text englisch     2.Zeile
Message-Nummer  Text deutsch      3.Zeile
-------------------------------------------------------------------
ASS0110  A10 - SIGNIFICANT ERROR
ASS0110  RELOCATABLE TERM IN PRODUCT OR DIVISION
ASS0110  PRODUKT ODER QUOTIENT ENTHAELT RELATIVEN ELEMENTARAUSDRUCK

ASS0111  ......
ASS0111  ......
ASS0111  ......

ASS0112  ......
  ...    ......
  ...    ......

   .
   .
   .
```

```
ASS0110    A10 - SIGNIFICANT ERROR
ASS0110    RELOCATABLE TERM NOT ALLOWED IN MULTIPLICATION OR DIVISION
ASS0110    RELATIVER AUSDRUCK IN MULTIPLIKATION ODER DIVISION UNZULAESSIG
```

Bedeutung

Argument in Multiplikation/Division ist ein Relativwert.

```
ASS0111    A11 - SIGNIFICANT ERROR
ASS0111    'EQU' EXPRESSION CANNOT BE EVALUATED
ASS0111    'EQU'-AUSDRUCK NICHT BERECHENBAR

ASS0112    A12 - SIGNIFICANT ERROR
ASS0112    'EQU' INSTRUCTION WITHIN XDSEC ILLEGAL
ASS0112    'EQU'-ANWEISUNG INNERHALB 'XDSEC' UNZULAESSIG

ASS0113    A13 - SIGNIFICANT ERROR
ASS0113    NEGATIVE RELOCATABLE ADDRESS
ASS0113    RELATIVE ADRESSE NEGATIV

ASS0114    A14 - SIGNIFICANT ERROR
ASS0114    ADDRESS OF A COMPLEX RELOCATABLE EXPRESSION CANNOT BE FOUND
ASS0114    ADRESSE EINES ZUSAMMENGESETZTEN RELATIVIERBAREN AUSDRUCKS NICHT AUFFINDBAR

ASS0115    A15 - SIGNIFICANT ERROR
ASS0115    UNRESOLVABLE EXPRESSION
ASS0115    AUSDRUCK UNAUFLOESBAR

ASS0116    A16 - SIGNIFICANT ERROR
ASS0116    EXPRESSION CANNOT BE EVALUATED
ASS0116    AUSDRUCK NICHT BERECHENBAR

ASS0117    A17 - SIGNIFICANT ERROR
ASS0117    EXPRESSION IS NOT RELOCATABLE
ASS0117    AUSDRUCK NICHT RELATIV

ASS0120    A20 - SIGNIFICANT ERROR
ASS0120    VALUE OF EXPRESSION GREATER THAN 2**31 - 1
ASS0120    WERT DES AUSDRUCKS GROESSER ALS 2**31 - 1

ASS0121    A21 - SIGNIFICANT ERROR
ASS0121    ILLEGAL NEGATIVE ADDRESS
ASS0121    NEGATIVE ADRESSE IST UNZULAESSIG

ASS0210    B10 - SIGNIFICANT ERROR
ASS0210    ILLEGAL OPERAND IN 'ICTL' OR 'ISEQ' INSTRUCTION
ASS0210    OPERAND IN 'ICTL'- ODER 'ISEQ'-ANWEISUNG UNZULAESSIG
```

```
ASS0211    B11 - SIGNIFICANT ERROR
ASS0211    'ICTL' MUST BE THE FIRST INSTRUCTION STATEMENT IN PROGRAM
ASS0211    'ICTL' MUSS ERSTE ANWEISUNG IM PROGRAMM SEIN

ASS0212    B12 - SIGNIFICANT ERROR
ASS0212    PRIMARY COLUMN IN 'ICTL' OPERAND MISSING
ASS0212    ANFANGSSPALTE IN 'ICTL'-OPERAND FEHLT

ASS0213    B13 - SIGNIFICANT ERROR
ASS0213    PRIMARY COLUMN IN 'ICTL' OPERAND IS NO DIRECT VALUE
ASS0213    ANFANGSSPALTE IN 'ICTL'-OPERAND KEIN DIREKTWERT

ASS0214    B14 - SIGNIFICANT ERROR
ASS0214    PRIMARY COLUMN IN 'ICTL' OPERAND IS WRONG
ASS0214    ANFANGSSPALTE IN 'ICTL'-OPERAND FEHLERHAFT

ASS0215    B15 - SIGNIFICANT ERROR
ASS0215    LAST COLUMN IN 'ICTL' OPERAND IS NO DIRECT VALUE
ASS0215    END-SPALTE IN 'ICTL'-OPERAND KEIN DIREKTWERT

ASS0216    B16 - SIGNIFICANT ERROR
ASS0216    LAST COLUMN IN 'ICTL' OPERAND IS WRONG
ASS0216    END-SPALTE IN 'ICTL'-OPERAND FEHLERHAFT

ASS0217    B17 - SIGNIFICANT ERROR
ASS0217    CONTINUE COLUMN IN 'ICTL' OPERAND IS WRONG
ASS0217    FORTSETZUNGSSPALTE IN 'ICTL'-OPERAND FEHLERHAFT

ASS0218    B18 - SIGNIFICANT ERROR
ASS0218    MAINTENANCE OPTION 'MONSYS-RECORDS' NOT GIVEN
ASS0218    MAINTENANCE-OPTION 'MONSYS-RECORDS' NICHT GESETZT

ASS0220    B20 - WARNING
ASS0220    ILLEGAL 'START' INSTRUCTION
ASS0220    'START'-ANWEISUNG UNZULAESSIG

ASS0221    B21 - SERIOUS ERROR
ASS0221    SECTION (&00) DOES NOT EXIST
ASS0221    SECTION (&00) NICHT VORHANDEN

ASS0230    B30 - SIGNIFICANT ERROR
ASS0230    ILLEGAL 'START' VALUE
ASS0230    'START'-WERT UNGUELTIG

ASS0231    B31 - SIGNIFICANT ERROR
ASS0231    ILLEGAL ATTRIBUTE (&00) IN 'CSECT' OR 'START' INSTRUCTION
ASS0231    MERKMAL (&00) IN 'CSECT'- ODER 'START'-ANWEISUNG UNZULAESSIG
```

```
ASS0233    B33 - SIGNIFICANT ERROR
ASS0233    ILLEGAL OPERAND IN 'END' INSTRUCTION
ASS0233    OPERAND IN 'END'-ANWEISUNG UNGUELTIG

ASS0234    B34 - WARNING
ASS0234    LENGTH OF ATTRIBUTED 'CSECT' (&00) IS ZERO; LINK PROBLEMS ARE POSSIBLE
ASS0234    'CSECT' (&00) MIT MERKMAL-ANGABE HAT LAENGE NULL; BINDERPROBLEME MOEGLICH
```

Bedeutung
Nachfolgende CSECT kann beim Laden die Merkmale der CSECT (&00) erhalten.

Maßnahme
CSECT oder Merkmale entfernen.

```
ASS0240    B40 - SIGNIFICANT ERROR
ASS0240    ILLEGAL OPCODE IN NAME OR OPERAND FIELD OF 'OPSYN' INSTRUCTION
ASS0240    OPERATIONS-CODE IM NAMENS- ODER OPERANDENFELD EINER 'OPSYN'-ANWEISUNG UNGUELTIG

ASS0241    B41 - SIGNIFICANT ERROR
ASS0241    'OPSYN' INSTRUCTION NOT ALLOWED IN MACROS
ASS0241    'OPSYN'-ANWEISUNG INNERHALB VON MAKROS UNZULAESSIG

ASS0242    B42 - SIGNIFICANT ERROR
ASS0242    'COPY' MEMBER NOT FOUND
ASS0242    'COPY'-ELEMENT NICHT GEFUNDEN
```

Maßnahme
Moegliche Massnahmen:
- COPY-Bibliothek in den Assembler-Options angeben;
- Elementname im COPY-Operand korrigieren.

```
ASS0243    B43 - SIGNIFICANT ERROR
ASS0243    NAME OF 'COPY' MEMBER INVALID
ASS0243    NAME DES 'COPY'-ELEMENTES FEHLERHAFT
```

Bedeutung
Fuer den Namen eines COPY-Elementes ist als erstes Zeichen ein Buchstabe erforderlich; als weitere Zeichen sind sowohl Buchstaben als auch Ziffern zulaessig. Maximallaenge fuer den Namen des COPY-Elementes: 64 Zeichen.

```
ASS0244    B44 - WARNING
ASS0244    MACRO NAME IN PROTOTYPE STATEMENT AND LIBRARY MEMBER NAME DIFFER
ASS0244    MAKRONAME IN MUSTERANWEISUNG UND BIBLIOTHEKSELEMENTNAME UNTERSCHIEDLICH
```

```
ASS0245   B45 - SIGNIFICANT ERROR
ASS0245   ILLEGAL OPERAND IN 'COPY' INSTRUCTION
ASS0245   'COPY'-OPERAND FEHLERHAFT
```

Bedeutung
Der Operand der COPY-Anweisung fehlt, ist syntaktisch falsch oder es wurde mehr als ein Operand angegeben. Das COPY-Element wird nicht eingefuegt.

Maßnahme
Operand korrigieren.

```
ASS0246   B46 - SIGNIFICANT ERROR
ASS0246   MAXIMUM 'COPY-LEVEL' (&00) EXCEEDED
ASS0246   MAXIMALER 'COPY-LEVEL' (&00) UEBERSCHRITTEN
```

```
ASS0247   B47 - FATAL ERROR
ASS0247   THE MAXIMUM MACRO LEVEL OF (&00) HAS BEEN REACHED
ASS0247   MAXIMALE MAKRO-VERSCHACHTELUNGSTIEFE VON (&00) ERREICHT
```

Bedeutung
Die Verschachtelungstiefe von Makroaufrufen hat die in der Assembler-Option (MAX-MACRO-NEST-LEVEL) angegebene Zahl erreicht (Standardwert: 255).

Maßnahme
Assembler-Option korrigieren bzw. Programm auf eine Endlosschleife bei Makroaufrufen pruefen.

```
ASS0248   B48 - NOTE
ASS0248   ATTENTION: SOURCE CONTAINS 'OPSYN' INSTRUCTIONS
ASS0248   VORSICHT: QUELLPROGRAMM ENTHAELT 'OPSYN'-ANWEISUNGEN
```

Bedeutung
Die Wirksamkeit der 'OPSYN'-Anweisung bezueglich Gueltigkeitsbereich und Dauer, vor allem im Zusammenwirken mit (Bibliotheks-)Makros ist zu beachten.
Siehe auch ASSEMBH-Beschreibung: Unterschiede ASSEMBH und ASSEMB V30.0A.

```
ASS0249   B49 - NOTE
ASS0249   'OPSYN' INACTIVATED
ASS0249   'OPSYN' INAKTIVIERT
```

```
ASS0250    B50 - WARNING
ASS0250    UNEXPECTED EOF BEFORE 'END' INSTRUCTION
ASS0250    EOF VOR 'END'-ANWEISUNG AUFGETRETEN
```

Bedeutung
Beim Einlesen der Source trat EOF auf, bevor die END-Anweisung gelesen wurde. Der ASSEMBH generiert eine END-Anweisung und setzt die Assemblierung fort.

```
ASS0251    B51 - NOTE
ASS0251    'MEND' INSTRUCTION MISSING
ASS0251    'MEND'-ANWEISUNG FEHLT
```

Bedeutung
MEND-Anweisung fehlt in Bibliotheksmakros. Folgefehler koennen nicht entstehen, die MEND-Anweisung wird generiert.

Maßnahme
MEND-Anweisung einfuegen.

```
ASS0252    B52 - WARNING
ASS0252    INPUT RECORD TOO LONG; MAXIMUM LENGTH = 256
ASS0252    EINGABESATZ ZU LANG; MAXIMALLAENGE = 256
```

```
ASS0254    B54 - WARNING
ASS0254    UNEXPECTED EOF
ASS0254    UNERWARTETES EOF
```

Bedeutung
Beim Einlesen einer Datei oder eines Bibliothekselementes trat EOF auf, bevor die Anweisung END oder MEND erkannt wurde.

Maßnahme
Fehlende END- bzw. MEND-Anweisung einfuegen.

```
ASS0255    B55 - SIGNIFICANT ERROR
ASS0255    'MEND' INSTRUCTION MISSING
ASS0255    'MEND'-ANWEISUNG FEHLT
```

Bedeutung
MEND-Anweisung in Source-Deck-Makros oder in Bibliotheksmakros mit inneren Makrodefinitionen fehlt. Dies kann dazu fuehren, dass die Generierung des Makros nicht bzw. falsch ausgefuehrt wird.

Maßnahme
MEND-Anweisung einfuegen.

```
ASS0256    B56 - NOTE
ASS0256    'END' INSTRUCTION IS GENERATED
ASS0256    'END'-ANWEISUNG GENERIERT

ASS0257    B57 - NOTE
ASS0257    'END' INSTRUCTION GENERATED BY MACRO EXPANSION
ASS0257    'END'-ANWEISUNG DURCH MAKRO-GENERIERUNG ERZEUGT

ASS0258    B58 - NOTE
ASS0258    THIS STATEMENT IS NO LONGER SUPPORTED
ASS0258    ANWEISUNG NICHT MEHR UNTERSTUETZT

ASS0259    B59 - NOTE
ASS0259    'CSECT' WITH NO NAME IS GENERATED
ASS0259    NAMENLOSE 'CSECT' WIRD GENERIERT

ASS0260    B60 - NOTE
ASS0260    THE 'MCALL'/'GSEQ' INSTRUCTIONS IN MACRO (&00) ARE NO LONGER NEEDED IN ASSEMBH
ASS0260    IM MAKRO (&00) AUFGETRETENE 'MCALL'/'GSEQ'-ANWEISUNG IM ASSEMBH NICHT MEHR
           BENOETIGT
```

Bedeutung
Die Anweisungen sind nicht mehr notwendig, sie werden ignoriert.

```
ASS0261    B61 - NOTE
ASS0261    INCOMPLETE PROGRAM; NO OBJECT GENERATION
ASS0261    PROGRAMM UNVOLLSTAENDIG; KEINE OBJEKTERZEUGUNG

ASS0262    B62 - NOTE
ASS0262    THE 'MCALL' OR 'GSEQ' INSTRUCTION IS NO LONGER NEEDED IN ASSEMBH
ASS0262    'MCALL'/'GSEQ'-ANWEISUNG IN ASSEMBH NICHT MEHR BENOETIGT

ASS0270    B70 - SIGNIFICANT ERROR
ASS0270    PROCEDURE NAME IN '$LSDL' STATEMENT MISSING
ASS0270    PROZEDUR-NAME BEI '$LSDL'-ANWEISUNG FEHLT

ASS0271    B71 - SIGNIFICANT ERROR
ASS0271    WRONG PROCEDURE NAME IN '$LSDLSAVE' STATEMENT
ASS0271    PROZEDUR-NAME BEI '$LSDLSAVE'-ANWEISUNG FEHLERHAFT

ASS0272    B72 - SIGNIFICANT ERROR
ASS0272    TYPE AND NAME IN '$LSDL' STATEMENT DO NOT CORRESPOND
ASS0272    TYP UND NAME BEI '$LSDL'-ANWEISUNG PASSEN NICHT ZUEINANDER

ASS0273    B73 - SIGNIFICANT ERROR
ASS0273    MORE THAN 3 OPERANDS IN '$LSDL' STATEMENT
ASS0273    MEHR ALS 3 OPERANDEN IN '$LSDL'-ANWEISUNG
```

```
ASS0274     B74 - SIGNIFICANT ERROR
ASS0274     TYPE IN '$LSDL' STATEMENT MISSING
ASS0274     TYP BEI '$LSDL'-ANWEISUNG FEHLT

ASS0275     B75 - WARNING
ASS0275     'CCW' FLAG BYTE WAS NOT CHECKED
ASS0275     'CCW'-FLAGBYTE NICHT GEPRUEFT

ASS0276     B76 - WARNING
ASS0276     CONSISTENCY CONSTANT IS GENERATED FOR EMPTY 'CSECT' SECTION
ASS0276     KONSISTENZ-KONSTANTE FUER LEERE 'CSECT' GENERIERT

ASS0311     C11 - SIGNIFICANT ERROR
ASS0311     ILLEGAL CONCATENATION
ASS0311     KONKATENIERUNG UNZULAESSIG
```

Bedeutung
Im Operandenfeld einer LCL-/GBL-Anweisung und im Namensfeld einer SET-Anweisung sind nur variable Parameter (auch generierte bzw. indizierte Parameter) zulaessig. Eine Konkatenierung ist unzulaessig.

```
ASS0312     C12 - SIGNIFICANT ERROR
ASS0312     ILLEGAL DIMENSION SPECIFIED
ASS0312     DIMENSIONSANGABE FEHLERHAFT
```

Bedeutung
Im Operandenfeld einer LCL- oder GBL-Instruktion muss die Dimension eine vorzeichenlose Dezimalzahl sein.

```
ASS0313     C13 - SIGNIFICANT ERROR
ASS0313     SYNTAX ERROR IN THE SUBSCRIPT OF A VARIABLE SYMBOL
ASS0313     SYNTAX-FEHLER IM INDEX EINES VARIABLEN PARAMETERS
```

Bedeutung
Der Index eines variablen Parameters muss ein SETA-Ausdruck sein.

```
ASS0321     C21 - SIGNIFICANT ERROR
ASS0321     OPERAND (&00) IS A SYMBOLIC PARAMETER IN A MACRO PROTOTYPE STATEMENT
ASS0321     OPERAND (&00) IST SYMBOLISCHER PARAMETER IN MUSTERANWEISUNG
```

Bedeutung
Ein variabler Parameter kann nicht ein symbolischer und zugleich ein 'SET'-Parameter sein.

```
ASS0322     C22 - SIGNIFICANT ERROR
ASS0322     ILLEGAL SYMBOLIC PARAMETER IN OPERAND FIELD OF 'LCL' OR 'GBL' INSTRUCTION
ASS0322     SYMBOLISCHE PARAMETER IM OPERANDENFELD EINER 'LCL'/'GBL'-ANWEISUNG UNZULAESSIG
```

```
ASS0335   C35 - SERIOUS ERROR
ASS0335   SERIOUS ERROR(S) FOR 'MACRO' OR PROTOTYPE STATEMENT OF A LIBRARY MACRO
ASS0335   SERIOUS ERROR(S) ZU 'MACRO' ODER MUSTERANWEISUNG EINES BIBLIOTHEKSMAKROS
```

Bedeutung
Einige der zu diesem Makroaufrufstatement ausgegebenen SERIOUS ERROR's betreffen die zugehoerige MACRO- oder Musteranweisung.

Maßnahme
MACRO- und Musteranweisung ueberpruefen.

```
ASS0336   C36 - NOTE
ASS0336   MACRO (&00) MULTIPLY DEFINED IN SOURCE
ASS0336   MAKRO (&00) IN DER SOURCE MEHRFACH DEFINIERT
```

Bedeutung
Hinweis bzgl. Inkompatibilitaet: Bei einem Aufruf wird immer der Makro generiert, dessen Definition als letzte durchlaufen wurde.

```
ASS0337   C37 - NOTE
ASS0337   NOTE(S) FOR 'MACRO' OR PROTOTYPE STATEMENT OF A LIBRARY MACRO
ASS0337   NOTE(S) ZU 'MACRO'- ODER MUSTERANWEISUNG EINES BIBLIOTHEKSMAKROS
```

Bedeutung
Einige der zu diesem Makroaufrufstatement ausgegebenen NOTE's betreffen die zugehoerige MACRO- oder Musteranweisung.

Maßnahme
MACRO- und Musteranweisung ueberpruefen.

```
ASS0338   C38 - WARNING
ASS0338   WARNING(S) FOR 'MACRO' OR PROTOTYPE STATEMENT OF A LIBRARY MACRO
ASS0338   WARNING(S) ZU 'MACRO'- ODER MUSTERANWEISUNG EINES BIBLIOTHEKSMAKROS
```

Bedeutung
Einige der zu diesem Makroaufrufstatement ausgegebenen WARNING's betreffen die zugehoerige MACRO- oder Musteranweisung.

Maßnahme
MACRO- und Musteranweisung ueberpruefen.

```
ASS0339    C39 - SIGNIFICANT ERROR
ASS0339    SIGNIFICANT ERROR(S) FOR 'MACRO' OR PROTOTYPE STATEMENT OF A LIBRARY MACRO
ASS0339    SIGNIFICANT ERROR(S) ZU 'MACRO'- ODER MUSTERANWEISUNG EINES BIBLIOTHEKSMAKROS
```

Bedeutung
Einige der zu diesem Makroaufrufstatement ausgegebenen SIGNIFICANT ERROR's betreffen die zugehoerige MACRO- oder Musteranweisung.

Maßnahme
MACRO- und Musteranweisung ueberpruefen.

```
ASS0340    C40 - SIGNIFICANT ERROR
ASS0340    MACRO PROTOTYPE STATEMENT HAS INVALID OPCODE
ASS0340    OPERATIONSCODE DER MUSTERANWEISUNG UNGUELTIG
```

Bedeutung
Der Makroname ist laenger als 64 Zeichen oder enthaelt ungueltige Zeichen. Der Makro wird nicht generiert.

```
ASS0341    C41 - SIGNIFICANT ERROR
ASS0341    MISSING OPCODE IN MACRO PROTOTYPE STATEMENT
ASS0341    OPERATIONSCODE IN MUSTERANWEISUNG FEHLT
```

Bedeutung
In der Musteranweisung eines Makros fehlt der Operationscode (=Makroname). Der Makro wird nicht generiert.

```
ASS0342    C42 - SIGNIFICANT ERROR
ASS0342    WRONG OPCODE IN FIRST STATEMENT OF LIBRARY MACRO
ASS0342    OPERATIONSCODE IN 1.ANWEISUNG EINES BIBLIOTHEKSMAKROS FEHLERHAFT
```

Bedeutung
In der ersten, in einem Bibliotheksmakro gefundenen Anweisung (nicht Leerzeile, Kommentar oder Makro-Kommentar), ist der Operationscode syntaktisch fehlerhaft oder er enthaelt variable Parameter.

Maßnahme
MACRO-Anweisung korrigieren/einfuegen.

```
ASS0343   C43 - SIGNIFICANT ERROR
ASS0343   MACRO DOES NOT CONTAIN ANY PROTOTYPE STATEMENT
ASS0343   MAKRO ENTHAELT KEINE MUSTERANWEISUNG
```

Bedeutung
Bei der Bearbeitung eines Makros wurde keine Musteranweisung gefunden (erste Anweisung nach der Makro-Anweisung, nicht Leerzeile, Kommentar oder Makro-Kommentar).

Maßnahme
Musteranweisung einfuegen bzw. korrigieren.

```
ASS0344   C44 - SIGNIFICANT ERROR
ASS0344   LIBRARY MACRO DOES NOT BEGIN WITH A 'MACRO' STATEMENT
ASS0344   BIBLIOTHEKSMAKRO BEGINNT NICHT MIT 'MACRO'-ANWEISUNG
```

```
ASS0346   C46 - SIGNIFICANT ERROR
ASS0346   MISSING OPCODE IN FIRST STATEMENT OF A LIBRARY MACRO
ASS0346   OPERATIONSCODE IN 1.ANWEISUNG EINES BIBLIOTHEKS-MAKROS FEHLT
```

Bedeutung
In der ersten, in dem Bibliotheksmakro gefundenen Anweisung (nicht Leerzeile, Kommentar oder Makro-Kommentar) fehlt der Opcode. Dieser muss stets MACRO sein.

Maßnahme
MACRO-Anweisung einfuegen bzw. korrigieren.

```
ASS0347   C47 - SIGNIFICANT ERROR
ASS0347   ERROR IN OPCODE OR OPERAND FIELD OF THE CORRESPONDING PROTOTYPE STATEMENT;
          MACRO WILL NOT BE GENERATED
ASS0347   OPCODE- ODER OPERANDENFELD DER ZUGEHOERIGEN MUSTERANWEISUNG FEHLERHAFT; MAKRO
          WIRD NICHT GENERIERT
```

```
ASS0348   C48 - SIGNIFICANT ERROR
ASS0348   MEND INSTRUCTION IS GENERATED
ASS0348   'MEND'-ANWEISUNG WURDE GENERIERT
```

```
ASS0349     C49 - NOTE
ASS0349     OPERAND FIELD OF THE PROTOTYPE STATEMENT ENDS WITH A COMMA
ASS0349     OPERANDENFELD DER MUSTERANWEISUNG ENDET MIT KOMMA
```

Bedeutung
Das Endkomma koennte darauf hindeuten, dass weitere Operanden folgen. Ist dies der Fall, beginnen sie in der falschen Spalte der Folgezeile und werden deshalb als Kommentar behandelt.

Maßnahme
Eventuell Folgezeile(n) auf korrekte Anfangsspalte ueberpruefen, ansonsten Komma entfernen.

```
ASS0351     C51 - SIGNIFICANT ERROR
ASS0351     SYMBOLIC PARAMETER (&00) OCCURS MORE THAN ONCE IN PROTOTYPE STATEMENT
ASS0351     SYMBOLISCHER PARAMETER (&00) TRITT IN MUSTERANWEISUNG MEHRFACH AUF

ASS0352     C52 - SIGNIFICANT ERROR
ASS0352     PRIMARY VALUE OF KEYWORD PARAMETER IN PROTOTYPE STATEMENT CANNOT BE GENERATED
ASS0352     GENERIERUNG DES ANFANGSWERTES EINES KENNWORT-OPERANDEN IN MUSTERANWEISUNG
            UNZULAESSIG

ASS0356     C56 - SIGNIFICANT ERROR
ASS0356     EMPTY PARAMETER IN PROTOTYPE STATEMENT
ASS0356     OPERAND IN MUSTERANWEISUNG LEER
```

Bedeutung
Ein leerer Operand in der Musteranweisung ist unzulaessig.

Maßnahme
Leeren bzw. fehlenden Operanden korrigieren.

```
ASS0381     C81 - WARNING
ASS0381     UNDEFINED KEYWORD PARAMETER (&00); OPERAND WAS INTERPRETED AS A POSITIONAL
            OPERAND
ASS0381     KENNWORT-OPERAND (&00) UNDEFINIERT; OPERAND WIRD ALS STELLUNGS-OPERAND
            INTERPRETIERT

ASS0401     D01 - SIGNIFICANT ERROR
ASS0401     INVALID CONSTANT TYPE
ASS0401     KONSTANTENTYP UNGUELTIG
```

Bedeutung
Bei einer DC- oder DS-Anweisung bzw. in einem Literal wurde ein falscher Konstantentyp angegeben.

Maßnahme
Konstantentyp in der Anweisung korrigieren.

```
ASS0402    D02 - SIGNIFICANT ERROR
ASS0402    LENGTH MODIFIER ERROR
ASS0402    LAENGENFAKTOR FEHLERHAFT
```

Bedeutung
Der Laengenfaktor einer DC-/DS-Anweisung oder eines Literals ist syntaktisch falsch bzw. sein Wert liegt ausserhalb des zulaessigen Bereichs.

Maßnahme
Syntax bzw. Wert des Laengenfaktors korrigieren.

```
ASS0403    D03 - SIGNIFICANT ERROR
ASS0403    CONSTANT OF TYPE S ILLEGAL IN A LITERAL STRING
ASS0403    S-KONSTANTE IN LITERALEN UNZULAESSIG

ASS0404    D04 - SIGNIFICANT ERROR
ASS0404    QUOTES NOT PAIRED OR ILLEGAL TERMINATION OF A QUOTED STRING
ASS0404    HOCHKOMMATA NICHT PAARWEISE ODER UNERLAUBTE BEENDIGUNG EINER ZEICHENKETTE

ASS0405    D05 - SIGNIFICANT ERROR
ASS0405    EMPTY OPERAND
ASS0405    OPERAND IST LEER

ASS0407    D07 - SIGNIFICANT ERROR
ASS0407    ALIGNMENT ERROR IN OPERAND (&00)
ASS0407    AUSRICHTUNGSFEHLER IN OPERAND (&00)
```

Bedeutung
Operand muss auf Halb-, Ganz- oder Doppelwortgrenze ausgerichtet sein.

```
ASS0408    D08 - SIGNIFICANT ERROR
ASS0408    UNPAIRED '&' IN CONSTANT VALUE OF A DC/DS OPERAND OR LITERAL
ASS0408    UNGEPAARTES '&' IM KONSTANTENWERT EINES DC/DS-OPERANDEN ODER LITERALS

ASS0409    D09 - SIGNIFICANT ERROR
ASS0409    DISPLACEMENT IN OPERAND (&00) NOT IN THE RANGE 0 TO 4095
ASS0409    DISTANZANGABE IN OPERAND (&00) NICHT 0 BIS 4095
```

Bedeutung
Distanz liegt nicht im Bereich 0 bis 4095 (einschliesslich).

```
ASS0411    D11 - SIGNIFICANT ERROR
ASS0411    LENGTH SPECIFICATION IN OPERAND (&00) NOT IN THE RANGE 1 TO 16
ASS0411    LAENGENANGABE IN OPERAND (&00) NICHT 1 BIS 16
```

```
ASS0412     D12 - SIGNIFICANT ERROR
ASS0412     DUPLICATION FACTOR ERROR
ASS0412     WIEDERHOLUNGSFAKTOR FEHLERHAFT
```

Bedeutung
Der Wiederholungsfaktor eines DC-/DS-Operanden bzw. Literals ist syntaktisch falsch bzw. sein Wert liegt ausserhalb des zulaessigen Bereichs.
Zulaessiger Bereich: 0 bis 2**24 - 1

Maßnahme
Syntax bzw. Wert des Wiederholungsfaktors korrigieren.

```
ASS0413     D13 - SIGNIFICANT ERROR
ASS0413     SCALE MODIFIER ERROR
ASS0413     SKALENFAKTOR FEHLERHAFT
```

Bedeutung
Der Skalenfaktor eines DC-/DS-Operanden bzw. Literals ist syntaktisch falsch bzw. sein Wert liegt ausserhalb des zulaessigen Bereichs. Der zulaessige Bereich ist vom Konstantentyp abhaengig.

Maßnahme
Syntax bzw. Wert des Skalenfaktors korrigieren.

```
ASS0414     D14 - SIGNIFICANT ERROR
ASS0414     EXPONENT MODIFIER ERROR
ASS0414     EXPONENTENFAKTOR FEHLERHAFT
```

Bedeutung
Der Exponentenfaktor eines DC-/DS-Operanden bzw. Literals ist syntaktisch falsch bzw. sein Wert liegt ausserhalb des zulaessigen Bereichs.
Zulaessiger Bereich: -85 bis +75.

Maßnahme
Syntax bzw. Wert des Exponentenfaktors korrigieren.

```
ASS0415     D15 - SIGNIFICANT ERROR
ASS0415     PRECISION LOST IN DC CONSTANT
ASS0415     GENAUIGKEITSVERLUST IN DC-KONSTANTE
```

Bedeutung
Gehen auf Grund der Angabe eines Skalenfaktors Stellen in der Konstante verloren, so fuehrt dies zur Ungenauigkeit der Konstante.

Maßnahme
Skalenfaktor korrekt angeben.

```
ASS0416    D16 - SIGNIFICANT ERROR
ASS0416    SELFDEFINING TERM (&00) TOO LARGE
ASS0416    SELBSTDEFINIERENDER WERT (&00) ZU GROSS

ASS0417    D17 - SIGNIFICANT ERROR
ASS0417    ARITHMETIC OVERFLOW
ASS0417    ARITHMETISCHER UEBERLAUF
```

Bedeutung
Das Endergebnis oder ein Zwischenergebnis bei der Berechnung eines arithmetischen Ausdrucks liegt nicht im Bereich zwischen 2**31-1 und -2**31.

Maßnahme
Arithmetischen Ausdruck aendern, so dass kein Ueberlauf mehr auftritt.

```
ASS0418    D18 - SIGNIFICANT ERROR
ASS0418    FLOATING-POINT CHARACTERISTIC OUT OF RANGE
ASS0418    GLEITPUNKTCHARAKTERISTIK AUSSERHALB DES ZULAESSIGEN BEREICHS
```

Bedeutung
Die Charakteristik, d.h. der hexadezimale Exponent einer Gleitpunktzahl (Typ E, D oder L) ist kleiner als -64 oder groesser als 64 und liegt somit ausserhalb des zulaessigen Bereichs.

Maßnahme
Konstante korrigieren.

```
ASS0419    D19 - SIGNIFICANT ERROR
ASS0419    INVALID CHARACTER IN CONSTANT VALUE OF A DC/DS OPERAND OR LITERAL
ASS0419    UNGUELTIGES ZEICHEN IM KONSTANTEN-WERT EINES DC-/DS-OPERANDEN ODER LITERALS
```

Bedeutung
Die Konstante enthaelt Zeichen, die in diesem Konstantentyp unzulaessig sind.

Maßnahme
Konstante korrigieren.

```
ASS0421    D21 - SIGNIFICANT ERROR
ASS0421    SYNTAX ERROR IN 'EQU' OPERAND
ASS0421    SYNTAX-FEHLER IN 'EQU'-OPERAND
```

```
ASS0422    D22 - SIGNIFICANT ERROR
ASS0422    INVALID LENGTH ATTRIBUTE IN 'EQU' OPERAND
ASS0422    LAENGENMERKMAL IN 'EQU'-OPERAND FEHLERHAFT
```

Bedeutung
Der Wert des Laengenmerkmals muss zwischen 0 und 2**24 - 1 liegen.

Maßnahme
Explizit angegebenes Laengenmerkmal korrigieren.

```
ASS0423    D23 - SIGNIFICANT ERROR
ASS0423    INVALID TYPE ATTRIBUTE IN 'EQU' OPERAND
ASS0423    TYPENMERKMAL IN 'EQU'-OPERAND FEHLERHAFT
```

Bedeutung
Das Typenmerkmal muss ein selbstdefinierender Wert (max. 1 byte lang) sein.

Maßnahme
Explizit angegebenes Typenmerkmal korrigieren.

```
ASS0424    D24 - SIGNIFICANT ERROR
ASS0424    LIMIT VALUES OF EXPONENT OUT OF RANGE
ASS0424    GRENZWERTE DER EXPONENTEN AUSSERHALB DES ZULAESSIGEN BEREICHS
```

Bedeutung
Die Summe aus innerem und aeusserem Exponenten einer DC-Konstanten ueber- oder unterschreitet die vorgegebenen Grenzwerte.

Maßnahme
Exponentenangabe korrigieren.

```
ASS0425    D25 - SIGNIFICANT ERROR
ASS0425    STRING VALUE (&00) CANNOT BE CONVERTED IN ARITHMETIC VALUE
ASS0425    ZEICHENWERT (&00) IN ARITHMETISCHEN WERT NICHT KONVERTIERBAR

ASS0427    D27 - SIGNIFICANT ERROR
ASS0427    ADDRESS CONSTANT CANNOT BE EVALUATED; NO GENERATION
ASS0427    ADRESSKONSTANTE NICHT BERECHENBAR. KEINE GENERIERUNG
```

```
ASS0428   D28 - SIGNIFICANT ERROR
ASS0428   CONSTANT VALUE OR EXPONENT OF A DC/DS OPERAND OR LITERAL OUT OF RANGE. DEFAULT
          VALUE 0 IS INSERTED
ASS0428   KONSTANTENWERT ODER EXPONENT EINES DC-/DS-OPERANDEN BZW. LITERALS AUSSERHALB
          DES ZULAESSIGEN BEREICHS; ERSATZWERT '0' WIRD VERWENDET
```

Bedeutung

Moegliche Ursachen:

- Der Konstantenwert ueberschreitet den durch den Konstantentyp festgelegten Wertebereich;
- die Summe aus Exponent und Exponentenfaktor liegt ausserhalb des zulaessigen Bereichs.

Zulaessiger Bereich: -85 bis +75.

```
ASS0429   D29 - SIGNIFICANT ERROR
ASS0429   ARITHMETIC OVERFLOW AFTER CONVERSION OF (&00)
ASS0429   ARITHMETISCHER UEBERLAUF NACH (&00) KONVERTIERUNG
```

Bedeutung

Der selbstdefinierende Wert kann in die interne (Binaer-) Darstellung nicht konvertiert werden, da durch die Konvertierung ein Binaer-Wert von mehr als 32 Bit erzeugt wuerde.
(&00): Zu konvertierender String.

Maßnahme

Selbstdefinierenden Wert so korrigieren, dass seine interne Darstellung in einem Wort (32 Bit) abgelegt werden kann.

```
ASS0430   D30 - SERIOUS ERROR
ASS0430   INVALID REGISTER SPECIFICATION IN OPERAND (&00); EVEN NUMBERED VALUE BETWEEN 0
          ANS 14 REQUIRED
ASS0430   REGISTER-ANGABE IN OPERAND (&00) UNGUELTIG; NUR GERADZAHLIGE NR. ZWISCHEN 0 UND
          14 ZULAESSIG

ASS0431   D31 - SERIOUS ERROR
ASS0431   INVALID FLOATING-POINT REGISTER SPECIFICATION IN OPERAND (&00); ONLY 0, 2, 4,
          OR 6 ALLOWED
ASS0431   GLEITPUNKTREGISTER-ANGABE IN OPERAND (&00) UNGUELTIG; NUR 0, 2, 4 ODER 6
          ZULAESSIG

ASS0432   D32 - SERIOUS ERROR
ASS0432   INVALID REGISTER SPECIFICATION; DIRECT VALUE (0 TO 15) EXPECTED
ASS0432   REGISTER-ANGABE UNGUELTIG; DIREKTWERT (0 BIS 15) WIRD ERWARTET
```

```
ASS0433    D33 - SIGNIFICANT ERROR
ASS0433    RELOCATABLE VALUE INVALID AS BASE REGISTER; VALUE MUST BE ABSOLUTE AND BETWEEN
           0 AND 15
ASS0433    RELATIVWERT ALS BASISREGISTER-ANGABE UNZULAESSIG. ABSOLUTWERT ZWISCHEN 0 UND 15
           ZULAESSIG

ASS0434    D34 - SERIOUS ERROR
ASS0434    INVALID PAIR NUMBER OF FLOATING-POINT REGISTER IN OPERAND (&00); VALUE 0 OR 4
           REQUIRED
ASS0434    GLEITPUNKTREGISTER-PAARNUMMER IN OPERAND (&00) UNGUELTIG; NUR 0 ODER 4
           ZULAESSIG

ASS0435    D35 - SIGNIFICANT ERROR
ASS0435    ILLEGAL SPECIFICATION OF A BASE REGISTER
ASS0435    BASISREGISTER-ANGABE UNZULAESSIG
```

Bedeutung
Ein Operand der Instruktion hat ein falsches Format. Statt Register bzw. Direktwert wurde Distanzadresse und Basisregister angegeben.

Maßnahme
Operandenformat korrigieren.

```
ASS0436    D36 - SIGNIFICANT ERROR
ASS0436    ILLEGAL SPECIFICATION OF A BASE REGISTER AND INDEX REGISTER OR LENGTH
ASS0436    BASISREGISTER-ANGABE UND INDEXREGISTER- BZW. LAENGENANGABE UNZULAESSIG
```

Bedeutung
Ein Operand der Instruktion hat ein falsches Format. Statt Register bzw. Direktwert wurde Distanzadresse, Basisregister und Indexregister oder Laenge angegeben.

Maßnahme
Operandenformat korrigieren.

```
ASS0437    D37 - SIGNIFICANT ERROR
ASS0437    ILLEGAL INDEX REGISTER OR LENGTH SPECIFICATION
ASS0437    INDEXREGISTER- BZW. LAENGENANGABE UNZULAESSIG
```

Bedeutung
Ein Operand der Anweisung hat ein falsches Format. Zusaetzlich zu Distanzadresse und Basisregister wurde ein Indexregister oder eine Laenge angegeben.

Maßnahme
Operandenformat korrigieren.

```
ASS0438    D38 - SERIOUS ERROR
ASS0438    ILLEGAL REGISTER SPECIFICATION IN OPERAND (&00)
ASS0438    REGISTERANGABE IN OPERAND (&00) UNGUELTIG
```

Bedeutung
Die Registernummer kann bei DUET-Befehlen nur bestimmte Werte annehmen.

```
ASS0439    D39 - SIGNIFICANT ERROR
ASS0439    ADDRESS VALUE IN OPERAND (&00) OUT OF RANGE
ASS0439    ADRESSWERT IN OPERAND (&00) AUSSERHALB DES ZULAESSIGEN BEREICHS
```

Bedeutung
Werte der Adressen ueberschreiten bei DUET-Befehlen zulaessige Grenzen.

```
ASS0441    D41 - SIGNIFICANT ERROR
ASS0441    QUOTES NOT PAIRED
ASS0441    UNGEPAARTE APOSTROPHE

ASS0442    D42 - SIGNIFICANT ERROR
ASS0442    SYNTAX ERROR IN DC/DS INSTRUCTION OR LITERAL
ASS0442    SYNTAX-FEHLER IN DC/DS-ANWEISUNG ODER LITERAL

ASS0443    D43 - SIGNIFICANT ERROR
ASS0443    SCALE OR EXPONENT MODIFIER ILLEGAL
ASS0443    SKALEN- BZW. EXPONENTENFAKTOR UNZULAESSIG
```

Bedeutung
Ein Skalen- und Exponentenfaktor ist in DC-Anweisungen und Literalen nur fuer Festpunkt- und Gleitpunktkonstanten zulaessig.

Maßnahme
Skalen- bzw. Exponentenfaktor weglassen oder Konstantentyp aendern.

```
ASS0445    D45 - SIGNIFICANT ERROR
ASS0445    INVALID LENGTH SPECIFIED IN OPERAND (&00); LENGTH MUST BE WITHIN THE RANGE 1 TO
           256
ASS0445    LAENGENANGABE IN OPERAND (&00) UNGUELTIG; LAENGE 1 BIS 256 ZULAESSIG

ASS0446    D46 - SIGNIFICANT ERROR
ASS0446    ILLEGAL SPECIFICATION OF ADDRESS; DISPLACEMENT WILL BE IGNORED
ASS0446    ADRESSANGABE FEHLERHAFT; DISTANZ WIRD IGNORIERT
```

Bedeutung
Adressangabe in Form eines Relativwerts bezogen auf ein Basisregister und die explizite Angabe eines Basisregisters sind unzulaessig.

```
ASS0447   D47 - SIGNIFICANT ERROR
ASS0447   ILLEGAL RELOCATABLE VALUE FOR LENGTH; LENGTH 0 WILL BE INSERTED
ASS0447   RELATIVWERT ALS LAENGENANGABE UNZULAESSIG. LAENGE '0' WIRD VERWENDET
```

Bedeutung
Die Laengenangabe muss ein Absolutwert sein.

```
ASS0448   D48 - SIGNIFICANT ERROR
ASS0448   SYNTAX ERROR IN CONSTANT OF A DC/DS OPERAND OR LITERAL
ASS0448   SYNTAX-FEHLER IM KONSTANTEN-WERT EINES DC/DS-OPERANDEN ODER LITERALS

ASS0449   D49 - NOTE
ASS0449   BASE REGISTER '0' IS USED
ASS0449   BASISREGISTER '0' WIRD VERWENDET

ASS0450   D50 - NOTE
ASS0450   NO OPERAND FIELD ENTRY ALLOWED
ASS0450   OPERANDENFELD-EINTRAG UNZULAESSIG

ASS0451   D51 - SIGNIFICANT ERROR
ASS0451   REQUIRED OPERAND FIELD ENTRY MISSING
ASS0451   ERFODERLICHER OPERANDENFELD-EINTRAG FEHLT

ASS0452   D52 - SIGNIFICANT ERROR
ASS0452   WRONG OPERAND TYPE IN OPERAND (&00). VALUE '0' IS INSERTED
ASS0452   FEHLERHAFTER OPERANDENTYP IN OPERAND (&00); ERSATZWERT '0' WIRD VERWENDET
```

Bedeutung
Es wird ein Absolutwert erwartet (z.B. Registerangabe).

```
ASS0453   D53 - WARNING
ASS0453   EMPTY OPERAND (&00)
ASS0453   OPERAND (&00) IST LEER
```

Bedeutung
Der genannte Operand ist leer, was fuer die bearbeitete Anweisung nicht sinnvoll ist.

```
ASS0454   D54 - SIGNIFICANT ERROR
ASS0454   REQUIRED NUMBER (&00) OF OPERANDS EXCEEDED; EXCESS OPERANDS WILL BE IGNORED
ASS0454   ERFODERLICHE OPERANDENANZAHL (&00) UEBERSCHRITTEN; UEBERZAEHLIGE OPERANDEN
          IGNORIERT

ASS0455   D55 - SERIOUS ERROR
ASS0455   REQUIRED OPERAND (&00) MISSING
ASS0455   ERFORDERLICHER OPERAND (&00) FEHLT
```

```
ASS0456    D56 - SIGNIFICANT ERROR
ASS0456    TOO MANY OPERANDS
ASS0456    ZU VIELE OPERANDEN

ASS0457    D57 - SIGNIFICANT ERROR
ASS0457    TOO FEW OPERANDS
ASS0457    ZU WENIG OPERANDEN

ASS0459    D59 - NOTE
ASS0459    OPERAND (&00) HAS NO EFFECT, SINCE THE OPTION 'PREFIX=EXCEPT' IS SET.
ASS0459    OPERAND (&00) UNWIRKSAM, DA OPTION 'PREFIX=EXCEPT' GESETZT

ASS0460    D60 - NOTE
ASS0460    DIVISION BY ZERO
ASS0460    DIVISION DURCH NULL

ASS0461    D61 - SIGNIFICANT ERROR
ASS0461    STRING DOES NOT BEGIN WITH A QUOTE
ASS0461    STRING BEGINNT NICHT MIT HOCHKOMMA

ASS0462    D62 - SIGNIFICANT ERROR
ASS0462    STRING IN '$DSDDI' OR '$DSDDR' OPERAND IS TOO LONG
ASS0462    STRING IN '$DSDDI'- BZW. '$DSDDR'-OPERAND ZU LANG
```

Bedeutung

Der String im ersten Operanden der $DSDDI- bzw. $DSDDR-Anweisung darf maximal 51, im zweiten Operand maximal 240 Zeichen lang sein.

```
ASS0463    D63 - SIGNIFICANT ERROR
ASS0463    REPRO OPERAND MISSING
ASS0463    'REPRO'-OPERAND FEHLT

ASS0464    D64 - NOTE
ASS0464    OPERAND (&00) IS NO LONGER SUPPORTED
ASS0464    OPERAND (&00) NICHT MEHR UNTERSTUETZT

ASS0504    E04 - SIGNIFICANT ERROR
ASS0504    WRONG CCW0/CCW1 OPCODE
ASS0504    CCW0/CCW1-OPERATIONSCODE FEHLERHAFT

ASS0505    E05 - SIGNIFICANT ERROR
ASS0505    WRONG CCW OPCODE
ASS0505    CCW-OPERATIONSCODE FEHLERHAFT

ASS0506    E06 - SIGNIFICANT ERROR
ASS0506    WRONG CCW BYTE COUNTER
ASS0506    CCW-BYTE-ZAEHLER FEHLERHAFT
```

```
ASS0507    E07 - SIGNIFICANT ERROR
ASS0507    WRONG CCW FLAG BYTE
ASS0507    CCW-FLAGBYTE FEHLERHAFT

ASS0508    E08 - SIGNIFICANT ERROR
ASS0508    WRONG CCW1 ADDRESS; A VALUE FROM 0 TO 2**31 - 1 IS PERMITTED
ASS0508    CCW1-ADRESSE UNGUELTIG; WERT 0 BIS 2**31 - 1 ZULAESSIG

ASS0509    E09 - SIGNIFICANT ERROR
ASS0509    WRONG CCW/CCW0 ADDRESS;                A VALUE FROM 0 TO 2**24 - 1 IS PERMITTED
ASS0509    CCW/CCW0-ADRESSE UNGUELTIG; WERT 0 BIS 2**24 - 1 ZULAESSIG

ASS0510    E10 - SIGNIFICANT ERROR
ASS0510    ILLEGAL CONTINUATION LINE
ASS0510    FORTSETZUNGSZEILE UNZULAESSIG

ASS0511    E11 - SIGNIFICANT ERROR
ASS0511    EOF WAS REACHED BEFORE CONTINUATION LINE
ASS0511    EOF VOR FORTSETZUNGSZEILE

ASS0512    E12 - SIGNIFICANT ERROR
ASS0512    LAST QUOTE MISSING IN OPERAND (&00)
ASS0512    ABSCHLIESSENDES HOCHKOMMA IN OPERAND (&00) FEHLT

ASS0513    E13 - SIGNIFICANT ERROR
ASS0513    NULL STRING ILLEGAL AS CONSTANT OF A DC/DS OPERAND OR LITERAL
ASS0513    NULLSTRING ALS KONSTANTENWERT EINES DC/DS-OPERANDEN ODER LITERALS UNZULAESSIG

ASS0517    E17 - SIGNIFICANT ERROR
ASS0517    SURPLUS 'STACK'/'UNSTK' INSTRUCTION (&00) (&01)
ASS0517    UEBERZAEHLIGE 'STACK'/'UNSTK'-ANWEISUNG (&00) (&01)
```

Bedeutung
(&01): Name der falsch gespeicherten bzw. freigegebenen Anweisung.

```
ASS0518    E18 - SIGNIFICANT ERROR
ASS0518    ILLEGAL CHARACTER BEFORE CONTINUATION LINE
ASS0518    UNZULAESSIGE(S) ZEICHEN VOR FORTSETZUNGSSPALTE

ASS0521    E21 - SIGNIFICANT ERROR
ASS0521    REQUIRED NAME FIELD ENTRY MISSING
ASS0521    ERFODERLICHER NAMENSFELD-EINTRAG FEHLT

ASS0524    E24 - SIGNIFICANT ERROR
ASS0524    SYMBOL NOT ALLOWED IN NAME FIELD
ASS0524    SYMBOL IM NAMENSFELD UNZULAESSIG
```

```
ASS0525    E25 - SIGNIFICANT ERROR
ASS0525    INVALID SYMBOL IN NAME FIELD
ASS0525    SYMBOL IM NAMENSFELD FEHLERHAFT

ASS0526    E26 - SIGNIFICANT ERROR
ASS0526    FIRST OPERAND IN 'AGO' INSTRUCTION IS EMPTY
ASS0526    ERSTER OPERAND IN 'AGO'-ANWEISUNG IST LEER
```

Bedeutung
Als erster Operand in einer AGO-Anweisung trat ein leerer Operand auf. Zulaessig sind ein Folgesymbol oder arithmetischer Ausdruck und Folgesymbol (Computed AGO).

```
ASS0527    E27 - SIGNIFICANT ERROR
ASS0527    INVALID SEQUENCE SYMBOL (&00) IN OPERAND (&01): NO BRANCH
ASS0527    FOLGESYMBOL (&00) IN OPERAND (&01) UNGUELTIG: KEIN SPRUNG

ASS0528    E28 - SIGNIFICANT ERROR
ASS0528    NAME OF SEQUENCE SYMBOL (&00) IS TOO LONG; MAXIMUM LENGTH = 64
ASS0528    FOLGESYMBOL (&00) ZU LANG; MAXIMALLAENGE = 64

ASS0529    E29 - SIGNIFICANT ERROR
ASS0529    OPERAND IS NOT A SEQUENCE SYMBOL
ASS0529    OPERAND IST KEIN FOLGESYMBOL

ASS0530    E30 - SIGNIFICANT ERROR
ASS0530    SYNTAX ERROR IN OPERAND (&00)
ASS0530    SYNTAX-FEHLER IN OPERAND (&00)

ASS0531    E31 - SIGNIFICANT ERROR
ASS0531    SEMANTIC ERROR IN OPERAND (&00)
ASS0531    SEMANTISCHER FEHLER IN OPERAND (&00)

ASS0532    E32 - NOTE
ASS0532    SYNTAX ERROR IN OPERAND (&00) OF THE 'SPACE' INSTRUCTION
ASS0532    SYNTAXFEHLER IN OPERAND (&00) DER 'SPACE'-ANWEISUNG
```

Bedeutung
Operand falsch oder Kommentar als Operand interpretiert.

```
ASS0533     E33 - SIGNIFICANT ERROR
ASS0533     ERROR IN OPERAND (&00)
ASS0533     FEHLER IN OPERAND (&00)
```

Bedeutung
Im bezeichneten Operanden trat ein syntaktischer bzw. semantischer Fehler auf, der im Regelfall durch eine weitere Fehlermeldung naeher erlaeutert wird.

Maßnahme
Operand korrigieren.

```
ASS0534     E34 - NOTE
ASS0534     DROP ISSUED FOR A RELEASED REGISTER OR ONE NOT ASSIGNED IN A 'USING'
            INSTRUCTION
ASS0534     ZU SPERRENDES REGISTER SCHON GESPERRT ODER NOCH NICHT DURCH 'USING'-ANWEISUNG
            ZUGEWIESEN

ASS0535     E35 - SIGNIFICANT ERROR
ASS0535     SEMANTIC ERROR
ASS0535     SEMANTISCHER FEHLER

ASS0538     E38 - SIGNIFICANT ERROR
ASS0538     ATTRIBUTE OF A SYMBOL CANNOT BE EVALUATED
ASS0538     MERKMAL EINES SYMBOLS NICHT BESTIMMBAR
```

Bedeutung
Das Symbol auf das Bezug genommen wird, ist undefiniert bzw. kann nicht bestimmt werden.

```
ASS0539     E39 - SIGNIFICANT ERROR
ASS0539     SYNTAX ERROR IN 'SETC' OPERAND OR IN TEXT REPLACEMENT
ASS0539     SYNTAXFEHLER IN 'SETC'-OPERAND ODER BEI TEXTERSETZUNG

ASS0540     E40 - SIGNIFICANT ERROR
ASS0540     LOGICAL EXPRESSION WRONG OR MISSING
ASS0540     LOGISCHER AUSDRUCK FEHLERHAFT BZW. NICHT VORHANDEN
```

Bedeutung
Logischer Ausdruck im AIF-Operanden fehlt bzw. ist nicht in Klammern eingeschlossen.

```
ASS0543    E43 - SIGNIFICANT ERROR
ASS0543    NAME OF THE SEQUENCE SYMBOL (&00) IS ILLEGAL
ASS0543    NAME DES FOLGESYMBOLS (&00) UNZULAESSIG
```

Bedeutung
Fuer ein Folgesymbol ist als erstes Zeichen ein Buchstabe erforderlich; als weitere Zeichen sind sowohl Buchstaben als auch Ziffern zulaessig.
Maximallaenge fuer ein Folgesymbol: 64 Zeichen.

```
ASS0546    E46 - SIGNIFICANT ERROR
ASS0546    INVALID SYMBOL REFERENCE
ASS0546    SYMBOLZUGRIFF FEHLERHAFT
```

```
ASS0550    E50 - SIGNIFICANT ERROR
ASS0550    PARENTHESIS ERROR
ASS0550    KLAMMERUNG FEHLERHAFT
```

Bedeutung
Eine rechte Klammer kann fehlen:
- nach Basisregister-Angabe im Operanden;
- bei einem geklammerten Term im Operanden;
- bei geklammertem Wiederholungs- bzw. Modifizierfaktor in einer DC- bzw. DS-Konstanten oder einem Literal.

```
ASS0552    E52 - SIGNIFICANT ERROR
ASS0552    PARENTHESIS ERROR IN OPERAND (&00)
ASS0552    KLAMMERUNG IN OPERAND (&00) FEHLERHAFT
```

```
ASS0553    E53 - SIGNIFICANT ERROR
ASS0553    ILLEGAL CHARACTER
ASS0553    UNZULAESSIGES ZEICHEN
```

```
ASS0554    E54 - SIGNIFICANT ERROR
ASS0554    ILLEGAL CHARACTER(S) IN OPERAND (&00)
ASS0554    OPERAND (&00) ENTHAELT UNZULAESSIGE(S) ZEICHEN
```

```
ASS0555    E55 - SIGNIFICANT ERROR
ASS0555    VALUE OF THE SECOND OPERAND IN SRP IS INVALID; VALUE '0' IS INSERTED
ASS0555    WERT DES 2. SRP-OPERANDEN UNGUELTIG; '0' WIRD VERWENDET
```

Bedeutung
Die Verschiebeinformation bewegt sich nicht in den moeglichen Grenzen.

```
ASS0556    E56 - SIGNIFICANT ERROR
ASS0556    SYNTAX ERROR IN OPERAND OF A 'SETA', 'AGO', OR 'SETC' INSTRUCTION
ASS0556    SYNTAX-FEHLER IN OPERAND EINER 'SETA'-, 'AGO'- ODER 'SETC'-ANWEISUNG
```

Bedeutung
Syntaxfehler im arithmetischen Ausdruck:
Bei SETA gesamtes Operandenfeld; bei 'berechnetem AGO' Nummer des Folgesymbols; bei SETC Wiederholungsfaktor und Argumente der Substring-Funktion (Anfangsposition und Laenge).

```
ASS0557    E57 - SIGNIFICANT ERROR
ASS0557    SYNTAX ERROR IN LOGICAL EXPRESSION
ASS0557    SYNTAX-FEHLER IN LOGISCHEM AUSDRUCK

ASS0566    E66 - SIGNIFICANT ERROR
ASS0566    OPERAND (&00) IS NOT A SYMBOLIC ADDRESS
ASS0566    OPERAND (&00) IST KEINE SYMBOLISCHE ADRESSE

ASS0571    E71 - SERIOUS ERROR
ASS0571    ILLEGAL LITERAL USE IN OPERAND (&00)
ASS0571    LITERAL IN OPERAND (&00) UNZULAESSIG

ASS0572    E72 - SIGNIFICANT ERROR
ASS0572    CONSTANT OF TYPE Q NOT ALLOWED WITHIN LITERALS
ASS0572    Q-KONSTANTE IN LITERALEN UNZULAESSIG

ASS0580    E80 - NOTE
ASS0580    'TITLE' TEXT EXCEEDS 97 CHARACTERS
ASS0580    'TITLE'-TEXT LAENGER ALS 97 ZEICHEN

ASS0581    E81 - SIGNIFICANT ERROR
ASS0581    'TITLE' TEXT MISSING
ASS0581    'TITLE'-TEXT FEHLT

ASS0582    E82 - NOTE
ASS0582    EXTERNAL SYMBOL IN OPERAND FIELD IS TRUNCATED TO 8 CHARACTERS
ASS0582    EXTERNES SYMBOL IM OPERANDENFELD AUF ZULAESSIGE 8 ZEICHEN GEKUERZT
```

Bedeutung
Der symbolische Name der externen Startadresse im END-Satz des Objektes ist auf 8 Zeichen begrenzt. Nur die ersten 8 Zeichen des angegebenen Namens werden verwendet.

```
ASS0593    E93 - SIGNIFICANT ERROR
ASS0593    ILLEGAL SEQUENCE SYMBOL IN NAME FIELD
ASS0593    FOLGESYMBOL IM NAMENSFELD UNZULAESSIG

ASS0594    E94 - NOTE
ASS0594    SYMBOL IN NAME FIELD IS TRUNCATED TO THE ALLOWED 8 CHARACTERS
ASS0594    SYMBOL IM NAMENSFELD AUF ZULAESSIGE 8 ZEICHEN GEKUERZT

ASS0595    E95 - SIGNIFICANT ERROR
ASS0595    SEQUENCE SYMBOL IS MISSING OR HAS A SYNTAX ERROR
ASS0595    FOLGESYMBOL FEHLT BZW. SYNTAKTISCH FALSCH

ASS0597    E97 - NOTE
ASS0597    NAME FOR OUTPUT TO ESD RECORD IS TRUNCATED TO 8 CHARACTERS
ASS0597    NAME FUER AUSGABE IN ESD-SATZ AUF 8 ZEICHEN GEKUERZT
```

Bedeutung
Der Name fuer Eintragungen in den ESD-Satz des Objektes ist auf 8 Zeichen begrenzt. Nur die ersten 8 Zeichen des Namens werden verwendet.

```
ASS0711    G11 - SIGNIFICANT ERROR
ASS0711    ILLEGAL 'MEND' OR 'MEXIT' INSTRUCTION
ASS0711    'MEND'- ODER 'MEXIT'-ANWEISUNG UNZULAESSIG
```

Bedeutung
Die Makroanweisungen MEND und MEXIT sind nur innerhalb einer Makrodefinition zulaessig.

```
ASS0712    G12 - WARNING
ASS0712    '.*' COMMENT IS ILLEGAL OUTSIDE OF MACRO DEFINITION
ASS0712    '.*'-KOMMENTAR AUSSERHALB VON MAKRODEFINITIONEN UNZULAESSIG

ASS0713    G13 - SIGNIFICANT ERROR
ASS0713    GENERATION OF A MACRO INSTRUCTION IS ILLEGAL
ASS0713    GENERIEREN EINER MAKRO-ANWEISUNG UNZULAESSIG

ASS0714    G14 - SIGNIFICANT ERROR
ASS0714    'MEND' AND 'MEXIT' INSTRUCTIONS ARE ONLY ALLOWED WITHIN MACRO DEFINITIONS
ASS0714    'MEND'- ODER 'MEXIT'-ANWEISUNG NUR IN MAKRODEFINITIONEN ZULAESSIG
```

Bedeutung
In der Source trat eine MEND- oder MEXIT-Anweisung auf. Dies ist nur in Makrodefinitionen zulaessig.

Maßnahme
Anweisung entfernen, ggf Verschachtelungstiefe innerer Makrodefinitionen kontrollieren.

```
ASS0724     G24 - SIGNIFICANT ERROR
ASS0724     OVERFLOW OF MAXIMUM COPY LEVEL (&00)
ASS0724     MAXIMALER COPY-LEVEL (&00) UEBERSCHRITTEN

ASS0730     G30 - SIGNIFICANT ERROR
ASS0730     GENERATED OPCODE (&00) IS NOT ALLOWED OR MUST NOT BE GENERATED
ASS0730     GENERIERTER OPERATIONS-CODE (&00) UNZULAESSIG BZW. DARF NICHT GENERIERT WERDEN
```

Bedeutung
Die Textersetzung ergab einen fehlerhaften Operations-Code bzw. einen solchen, der nicht generiert werden darf.

```
ASS0732     G32 - SIGNIFICANT ERROR
ASS0732     GENERATED OPCODE CONSISTS OF BLANKS
ASS0732     GENERIERTER OPERATIONS-CODE BESTEHT AUS LEERZEICHEN

ASS0734     G34 - WARNING
ASS0734     GENERATION OF '.*' COMMENTS IS NOT ALLOWED
ASS0734     GENERIEREN VON '.*'-KOMMENTAREN UNZULAESSIG
```

Bedeutung
Es sollte ein '.*'-Kommentar generiert werden.

```
ASS0736     G36 - SIGNIFICANT ERROR
ASS0736     ILLEGAL USE OF A NULL STRING WHEN A VARIABLE SYMBOL IS GENERATED
ASS0736     NULLSTRING BEI GENERIERUNG EINES VARIABLEN PARAMETERS UNZULAESSIG

ASS0737     G37 - SIGNIFICANT ERROR
ASS0737     GENERATED NAME/OPCODE FIELD CONSISTS OF MORE THAN ONE STRING
ASS0737     GENERIERTES NAMENS-/OPCODE-FELD BESTEHT AUS MEHR ALS EINEM STRING

ASS0740     G40 - SIGNIFICANT ERROR
ASS0740     ILLEGAL GENERATION OF A NULL STRING IN OPCODE FIELD
ASS0740     GENERIERUNG EINES NULLSTRINGS IN OPCODE-FELD UNZULAESSIG

ASS0811     H11 - SIGNIFICANT ERROR
ASS0811     DUMMY REGISTER EXCEEDS 4095 BYTES
ASS0811     PSEUDOREGISTER GROESSER ALS 4095 BYTES
```

Bedeutung
Die maximale Laenge eines DXD-Operanden (Wiederholungsfaktor * Laengenfaktor) darf 4095 byte nicht ueberschreiten.

```
ASS0910     I10 - SIGNIFICANT ERROR
ASS0910     INVALID DIRECT VALUE IN OPERAND (&00); VALUE MUST BE FROM 0 TO 255
ASS0910     DIREKTWERT IN OPERAND (&00) UNGUELTIG; WERT 0 BIS 255 ZULAESSIG
```

```
ASS0911    I11 - SIGNIFICANT ERROR
ASS0911    INVALID DIRECT VALUE IN OPERAND (&00); VALUE MUST BE FROM 0 TO 15
ASS0911    DIREKTWERT IN OPERAND (&00) UNGUELTIG; WERT 0 BIS 15 ZULAESSIG

ASS0912    I12 - SIGNIFICANT ERROR
ASS0912    INVALID ROUNDED VALUE IN OPERAND (&00); VALUE MUST BE FROM 0 TO 9
ASS0912    RUNDUNGSWERT IN OPERAND (&00) UNGUELTIG; WERT 0 BIS 9 ZULAESSIG

ASS0913    I13 - SIGNIFICANT ERROR
ASS0913    INVALID MASK SPECIFICATION IN OPERAND (&00); VALUE MUST BE FROM 0 TO 15
ASS0913    MASKENANGABE IN OPERAND (&00) UNGUELTIG; WERT 0 BIS 15 ZULAESSIG

ASS0920    I20 - SIGNIFICANT ERROR
ASS0920    INVALID SELF-DEFINING TERM
ASS0920    SELBSTDEFINIERENDER WERT FEHLERHAFT

ASS0921    I21 - SIGNIFICANT ERROR
ASS0921    ARITHMETIC VALUE (&00) CONTAINS ILLEGAL CHARACTERS
ASS0921    ARITHMETISCHER WERT (&00) ENTHAELT UNZULAESSIGE ZEICHEN

ASS1110    K10 - WARNING
ASS1110    SEQUENCE SYMBOL (&00) ALREADY DEFINED
ASS1110    FOLGESYMBOL (&00) BEREITS DEFINIERT

ASS1230    L30 - SIGNIFICANT ERROR
ASS1230    RELOCATABLE ADDRESS CONSTANT CONTAINS NAME FROM 'DSECT'
ASS1230    ZU RELATIVIERENDE ADRESSKONSTANTE ENTHAELT NAME AUS 'DSECT'
```

Bedeutung
Fuer eine Groesse aus einer DSECT kann keine RLD-Information erzeugt werden.

```
ASS1250    L50 - SERIOUS ERROR
ASS1250    OPERAND 2 OF THE 'CNOP' INSTRUCTION MUST BE '4' OR '8'
ASS1250    OPERAND 2 DER 'CNOP'-ANWEISUNG MUSS '4' ODER '8' SEIN

ASS1251    L51 - SERIOUS ERROR
ASS1251    OPERAND 1 OF THE 'CNOP' INSTRUCTION MUST BE '0', '2', '4' OR '6'
ASS1251    OPERAND 1 DER 'CNOP'-ANWEISUNG MUSS '0', '2', '4' ODER '6' SEIN

ASS1252    L52 - SERIOUS ERROR
ASS1252    OPERAND 1 OF THE 'CNOP' INSTRUCTION MUST BE '0' OR '2'
ASS1252    OPERAND 1 DER 'CNOP'-ANWEISUNG MUSS '0' ODER '2' SEIN

ASS1253    L53 - SERIOUS ERROR
ASS1253    'CNOP' OPERAND (&00) IS RELOCATABLE; IT MUST BE ABSOLUTE
ASS1253    'CNOP'-OPERAND (&00) IST RELATIV, MUSS JEDOCH ABSOLUT SEIN
```

```
ASS1254    L54 - SIGNIFICANT ERROR
ASS1254    MISSING 'CNOP' OPERAND
ASS1254    'CNOP'-OPERAND FEHLT

ASS1310    M10 - SIGNIFICANT ERROR
ASS1310    SYMBOL (&00) IS MULTIPLE DEFINED
ASS1310    SYMBOL (&00) MEHRFACH DEFINIERT

ASS1320    M20 - SIGNIFICANT ERROR
ASS1320    'ENTRY' NOT ALLOWED IN 'DSECT', 'COM', 'XDSEC', OR 'DXD'
ASS1320    'ENTRY' IN 'DSECT','COM','XDSEC','DXD' UNZULAESSIG

ASS1321    M21 - SIGNIFICANT ERROR
ASS1321    ENTRY (&00) IS IN 'DSECT' OR 'XDSEC'
ASS1321    'ENTRY' (&00) LIEGT IN 'DSECT'/'XDSEC'

ASS1324    M24 - SIGNIFICANT ERROR
ASS1324    'AMODE'/'RMODE' INCONSISTENCY
ASS1324    'AMODE'/'RMODE'-UNVERTRAEGLICHKEIT

ASS1325    M25 - SIGNIFICANT ERROR
ASS1325    'AMODE'/'RMODE' IS ILLEGAL FOR AN UNNAMED 'COMMON' CONTROL SECTION
ASS1325    'AMODE'/'RMODE' FUER UNBENANNTEN 'COM'-ABSCHNITT UNZULAESSIG

ASS1330    M30 - SIGNIFICANT ERROR
ASS1330    ADDRESS OR DIRECT VALUE NOT WITHIN THE RANGE 0 TO 2**31 - 1
ASS1330    ADRESSE ODER DIREKTWERT NICHT IM BEREICH ZWISCHEN 0 UND 2**31 - 1

ASS1332    M32 - SIGNIFICANT ERROR
ASS1332    VALUE IN THE OPERAND EXCEEDS 2**24-1
ASS1332    OPERANDENWERT UEBERSCHREITET 2**24-1

ASS1350    M50 - SIGNIFICANT ERROR
ASS1350    SYMBOL (&00) ALREADY DEFINED AS A 'CSECT', 'START', 'DSECT', 'COM', 'XDSEC' OR
           'DXD' NAME
ASS1350    SYMBOL (&00) BEREITS ALS 'CSECT'-, 'START'-, 'DSECT'-, 'COM'-, 'XDSEC'- ODER
           'DXD'-NAME DEFINIERT

ASS1351    M51 - SIGNIFICANT ERROR
ASS1351    SYMBOL (&00) ALREADY DEFINED AS NAME OF 'EXTRN' OR 'WXTRN'
ASS1351    SYMBOL (&00) BEREITS ALS 'EXTRN'- ODER 'WXTRN'-NAME DEFINIERT

ASS1352    M52 - SIGNIFICANT ERROR
ASS1352    'AMODE'/'RMODE' ALREADY PRESENT
ASS1352    'AMODE'/'RMODE' BEREITS VORHANDEN
```

```
ASS1353    M53 - SIGNIFICANT ERROR
ASS1353    SYMBOL (&00) WAS ALREADY REFERENCED AS A Q-CONSTANT
ASS1353    SYMBOL (&00) BEREITS ALS Q-KONSTANTE REFERENZIERT
```

Bedeutung
Das neu zu definierende Symbol wird ignoriert und kann zu Folgefehlern bei nachfolgenden Referenzen fuehren.

```
ASS1354    M54 - SIGNIFICANT ERROR
ASS1354    'XDSEC' ALREADY DEFINED. OPERAND DOES NOT EQUAL DEFINITION/REFERENCE
ASS1354    'XDSEC' BEREITS DEFINIERT. OPERAND NICHT GLEICH DEFINITION/REFERENZ

ASS1356    FAILURE
ASS1356    MISSING 'START' OR 'CSECT' INSTRUCTION
ASS1356    'START' BZW. 'CSECT'-ANWEISUNG FEHLT

ASS1357    M57 - SIGNIFICANT ERROR
ASS1357    'DSDD' INFORMATION ALREADY PRESENT
ASS1357    'DSDD'-INFORMATION BEREITS VORHANDEN

ASS1410    N10 - SIGNIFICANT ERROR
ASS1410    VARIABLE SYMBOL (&00) UNDEFINED AT TIME OF GENERATION
ASS1410    VARIABLER PARAMETER (&00) ZUM GENERIERUNGSZEITPUNKT UNDEFINIERT

ASS1420    N20 - SIGNIFICANT ERROR
ASS1420    UNDEFINED SEQUENCE SYMBOL IN OPERAND (&00): NO BRANCH
ASS1420    FOLGESYMBOL IN OPERAND (&00) UNDEFINIERT: KEIN SPRUNG

ASS1502    O02 - SIGNIFICANT ERROR
ASS1502    INVALID OPCODE
ASS1502    OPERATIONS-CODE UNGUELTIG

ASS1503    O03 - SIGNIFICANT ERROR
ASS1503    ERROR IN 'MACRO' OR MACRO PROTOTYPE STATEMENT: NO MACRO GENERATED.
ASS1503    FEHLER IN 'MACRO'- ODER MUSTERANWEISUNG: MAKRO WIRD NICHT GENERIERT
```

Bedeutung
Es traten Fehler in der MACRO-Anweisung oder im Operationscode/Operandenfeld der Musteranweisung des gerufenen Makros auf. Die Generierung unterbleibt.

Maßnahme
MACRO- und/oder Musteranweisung des Makros korrigieren.

```
ASS1504    O04 - SIGNIFICANT ERROR
ASS1504    MISSING OPCODE
ASS1504    OPERATIONS-CODE FEHLT
```

Bedeutung
Die Anweisung enthaelt keinen Operationscode (eventuell fehlendes Blank vor Operationscode).

Maßnahme
Operationscode (oder Blank) einfuegen.

```
ASS1505    O05 - SIGNIFICANT ERROR
ASS1505    OPCODE (&00) NOT FOUND
ASS1505    OPERATIONS-CODE (&00) NICHT GEFUNDEN

ASS1506    O06 - SIGNIFICANT ERROR
ASS1506    INVALID OPCODE IN OPERAND FIELD OF THE 'OPSYN' INSTRUCTION
ASS1506    OPERATIONS-CODE IN OPERANDENFELD DER 'OPSYN'-ANWEISUNG UNGUELTIG

ASS1522    O22 - SIGNIFICANT ERROR
ASS1522    SYMBOL (&00) CANNOT BE EVALUATED
ASS1522    SYMBOL (&00) NICHT BESTIMMBAR

ASS1601    P01 - WARNING
ASS1601    PRIVILEGED INSTRUCTION
ASS1601    PRIVILEGIERTER BEFEHL

ASS1711    Q11 - SIGNIFICANT ERROR
ASS1711    SYMBOL IN 'ORG' OPERAND DOES NOT BELONG TO THE CURRENT PROGRAM SECTION
ASS1711    SYMBOL IN 'ORG'-OPERAND LIEGT AUSSERHALB DES AKTUELLEN PROGRAMM-ABSCHNITTS

ASS1712    Q12 - SIGNIFICANT ERROR
ASS1712    OPERAND IS ABSOLUTE; MUST BE RELOCATABLE
ASS1712    OPERAND IST ABSOLUT; MUSS RELATIV SEIN

ASS1713    Q13 - SIGNIFICANT ERROR
ASS1713    VALUE IN 'ORG' OPERAND DOES NOT BELONG TO CURRENT PROGRAM SECTION
ASS1713    WERT IN 'ORG'-OPERAND LIEGT AUSSERHALB DES AKTUELLEN PROGRAMM-ABSCHNITTS

ASS1714    Q14 - SIGNIFICANT ERROR
ASS1714    SYMBOL IN ORG-OPERAND NOT PREVIOUSLY DEFINED
ASS1714    SYMBOL IM ORG-OPERAND NICHT VORHER DEFINIERT

ASS1721    Q21 - SIGNIFICANT ERROR
ASS1721    ILLEGAL SYMBOL REFERENCE IN OPERAND (&00)
ASS1721    SYMBOLZUGRIFF IN OPERAND (&00) FEHLERHAFT
```

```
ASS1901    S01 - SIGNIFICANT ERROR
ASS1901    SYMBOL TOO LONG; MAXIMUM LENGTH = 64
ASS1901    SYMBOL ZU LANG; MAXIMALLAENGE = 64
```

Bedeutung
Das Symbol im Namensfeld oder Operandenfeld der Assembler-Anweisung ist zu lang.

Maßnahme
Symbol kuerzen und Lauf wiederholen.

```
ASS1902    S02 - SIGNIFICANT ERROR
ASS1902    SEQUENCE SYMBOL NOT ALLOWED IN NAME FIELD
ASS1902    FOLGESYMBOL IM NAMENSFELD UNZULAESSIG
```

Bedeutung
Im Namensfeld tritt erstmalig ein Folgesymbol auf, das jedoch fuer diesen Operations-Code unzulaessig ist.

```
ASS1903    S03 - WARNING
ASS1903    NAME OF V OR Q CONSTANT TRUNCATED TO 8 CHARACTERS
ASS1903    NAME DER V- BZW. Q-KONSTANTE AUF 8 ZEICHEN GEKUERZT
```

Bedeutung
Der Name einer V- bzw. Q-Konstanten darf maximal 8 Zeichen lang sein. Es werden nur die ersten 8 Zeichen beruecksichtigt.

Maßnahme
Name der V- bzw. Q-Konstanten gegebenenfalls kuerzen.

```
ASS1910    S10 - SIGNIFICANT ERROR
ASS1910    ILLEGAL NAME FIELD ENTRY
ASS1910    NAMENSFELD-EINTRAG UNZULAESSIG

ASS1911    S11 - SIGNIFICANT ERROR
ASS1911    ILLEGAL GENERATION IN NAME FIELD
ASS1911    GENERIERUNG IM NAMENSFELD UNZULAESSIG

ASS1912    S12 - SIGNIFICANT ERROR
ASS1912    'TITLE' INSTRUCTION WITH NAME FIELD ENTRY IS NOT FIRST 'TITLE' INSTRUCTION
ASS1912    'TITLE'-ANWEISUNG MIT NAMENSFELD-EINTRAG IST NICHT ERSTE 'TITLE'-ANWEISUNG

ASS1913    S13 - SIGNIFICANT ERROR
ASS1913    ILLEGAL GENERATION OF A SEQUENCE SYMBOL IN NAME FIELD
ASS1913    GENERIERUNG EINES FOLGESYMBOLS IM NAMENSFELD UNZULAESSIG
```

Bedeutung
Das im Namensfeld generierte Folgesymbol wird ignoriert.

```
ASS1914     S14 - SIGNIFICANT ERROR
ASS1914     THE GENERATED NAME (&00) IS ILLEGAL; IT WILL BE IGNORED
ASS1914     GENERIERTER NAME (&00) UNGUELTIG; WIRD IGNORIERT

ASS1915     S15 - SIGNIFICANT ERROR
ASS1915     SEQUENCE SYMBOLS MUST NOT BE GENERATED IN THE NAME FIELD
ASS1915     GENERIEREN VON FOLGESYMBOLEN IM NAMENSFELD UNZULAESSIG

ASS1916     S16 - NOTE
ASS1916     'TITLE' NAME TRUNCATED TO 4 CHARACTERS
ASS1916     'TITLE'-NAME AUF 4 ZEICHEN GEKUERZT

ASS1917     S17 - NOTE
ASS1917     THE FIRST 'CSECT' IS UNNAMED; NO GENERATION OF 'AID'-INFORMATION
ASS1917     KEINE 'AID'-INFORMATION ERZEUGT, DA ERSTE 'CSECT' UNBENANNT
```

Bedeutung
Fuer AID ist der Name der ersten CSECT als Sourcemodulname bereitgestellt.

Maßnahme
Erste CSECT benennen.

```
ASS2010     T10 - SIGNIFICANT ERROR
ASS2010     CHARACTER VALUE TOO LONG; MAXIMUM LENGTH = 1020
ASS2010     ZEICHENWERT ZU LANG; MAXIMALLAENGE = 1020

ASS2013     T13 - SIGNIFICANT ERROR
ASS2013     ILLEGAL AMPERSAND GENERATED AFTER TEXT REPLACEMENT
ASS2013     '&'-ZEICHEN NACH TEXTERSETZUNG UNZULAESSIG

ASS2014     T14 - SIGNIFICANT ERROR
ASS2014     ILLEGAL AMPERSAND GENERATED IN OPERAND (&00) AFTER TEXT REPLACEMENT
ASS2014     '&'-ZEICHEN NACH TEXTERSETZUNG IN OPERAND (&00) UNZULAESSIG
```

Bedeutung
Nach der Textersetzung trat im genannten Operanden ein &-Zeichen, also ein erneutes Ersetzungskriterium auf. Dies ist unzulaessig.

Maßnahme
Die Werte der zu ersetzenden variablen Symbole ueberpruefen.

```
ASS2015     T15 - SIGNIFICANT ERROR
ASS2015     ILLEGAL CONCATENATION
ASS2015     KONKATENIERUNG UNZULAESSIG

ASS2110     U10 - SIGNIFICANT ERROR
ASS2110     SYMBOL (&00) IS UNDEFINED
ASS2110     SYMBOL (&00) UNDEFINIERT
```

```
ASS2111   U11 - SIGNIFICANT ERROR
ASS2111   UNDEFINED 'ENTRY' NAME
ASS2111   'ENTRY'-NAME UNDEFINIERT
```

Maßnahme
ENTRY-Namen ueberpruefen.

```
ASS2211   V11 - SIGNIFICANT ERROR
ASS2211   LOCAL VARIABLE SYMBOL IN OPERAND (&00) ALREADY DEFINED; FIRST DECLARATION IS
          VALID
ASS2211   LOKALER VARIABLER PARAMETER IN OPERAND (&00) BEREITS DEFINIERT; ERSTE
          DEKLARATION GILT

ASS2231   V31 - SIGNIFICANT ERROR
ASS2231   GLOBAL VARIABLE SYMBOL IN OPERAND (&00) ALREADY DEFINED. FIRST DECLARATION
          (&01) IS VALID
ASS2231   GLOBALER VARIABLER PARAMETER IN OPERAND (&00) BEREITS DEFINIERT; ERSTE
          DEKLARATION (&01) GILT
```

Bedeutung
(&01): STMT-Nr. der Erstdefinition.

```
ASS2232   V32 - SIGNIFICANT ERROR
ASS2232   TYPE INCONSISTENCY BY USE OF THE GLOBAL VARIABLE SYMBOL IN THE NAME FIELD.
          (&00)
ASS2232   TYP-UNVERTRAEGLICHKEIT BEI VERWENDUNG DES GLOBALEN VARIABLEN PARAMETERS IM
          NAMENSFELD. (&00)
```

Bedeutung
(&00): STMT-Nr. der Erstdefinition.

```
ASS2233   V33 - SIGNIFICANT ERROR
ASS2233   VARIABLE SYMBOL IN OPERAND (&00) OF THE 'GBL' INSTRUCTION IS ALREADY DEFINED AS
          A LOCAL VARIABLE SYMBOL
ASS2233   VARIABLER PARAMETER IN OPERAND (&00) DER 'GBL'-ANWEISUNG BEREITS ALS LOKALER
          VARIABLER PARAMETER DEKLARIERT

ASS2234   V34 - SIGNIFICANT ERROR
ASS2234   GLOBAL VARIABLE SYMBOL (&00) ALREADY DEFINED. (&01)
ASS2234   GLOBALER VARIABLER PARAMETER (&00) BEREITS DEFINIERT. (&01)
```

Bedeutung
(&01): STMT-Nr. der Erstdefinition.

```
ASS2241    V41 - SIGNIFICANT ERROR
ASS2241    SYSTEM VARIABLES NOT ALLOWED IN PROTOTYPE STATEMENT OR IN THE OPERAND FIELD OF
           'LCL''GBL' INSTRUCTIONS
ASS2241    SYSTEMVARIABLE IN MUSTERANWEISUNG BZW. IM OPERANDENFELD VON 'LCL'-/' 'GBL'-
           ANWEISUNGEN UNZULAESSIG

ASS2242    V42 - NOTE
ASS2242    SYSTEM VARIABLE SYMBOLS ILLEGAL IN NAME FIELD OF A PROTOTYPE STATEMENT
ASS2242    VARIABLE SYSTEMPARAMETER IM NAMENSFELD EINER MUSTERANWEISUNG UNZULAESSIG
```

Bedeutung
Im Namensfeld der Musteranweisung eines Makros trat ein variabler Systemparameter auf. Das Namensfeld wird ignoriert.

```
ASS2244    V44 - SIGNIFICANT ERROR
ASS2244    ILLEGAL USE OF SYSTEM VARIABLE SYMBOL IN SOURCE
ASS2244    VERWENDUNG DES VARIABLEN SYSTEMPARAMETERS IN DER SOURCE UNZULAESSIG
```

Bedeutung
Einige Systemvariable, z.B. &SYSVERM, &SYSECT, sind makrospezifisch und duerfen in der Source nicht verwendet werden.

Maßnahme
Unter Zuhilfenahme des Handbuches Informationen ueber die Verwendung der Systemvariablen einholen.

```
ASS2245    V45 - SIGNIFICANT ERROR
ASS2245    ILLEGAL REFERENCE TO THE SYSTEM VARIABLE SYMBOL '&SYSLIST'
ASS2245    ZUGRIFF AUF VARIABLEN SYSTEMPARAMETER '&SYSLIST' FEHLERHAFT
```

Bedeutung
Der Systemvariablen &SYSLIST ist kein Wert zugeordnet. Stellungoperanden (bzw. deren Unterlistenelemente) koennen nur ueber Unterlisten-Zugriffe (&SYSLIST(x,y,...)) referenziert werden.

Maßnahme
Nur Unterlistenzugriffe sind erlaubt.

```
ASS2246    V46 - SIGNIFICANT ERROR
ASS2246    ILLEGAL VALUE ASSIGNMENT TO SYSTEM VARIABLE SYMBOL
ASS2246    ZUWEISUNG AN VARIABLEN SYSTEMPARAMETER UNZULAESSIG

ASS2247    V47 - SIGNIFICANT ERROR
ASS2247    OPERAND (&00) IS A SYSTEM VARIABLE SYMBOL
ASS2247    OPERAND (&00) IST SYSTEMPARAMETER
```

Bedeutung
Deklaration von Systemparametern als SET-Parameter unzulaessig.

```
ASS2248    V48 - SIGNIFICANT ERROR
ASS2248    ONLY THE VALUE '24' OR '31' IS PERMITTED FOR THE SYSTEM VARIABLE SYMBOL
           '&SYSMOD'
ASS2248    FUER VARIABLEN SYSTEMPARAMETER '&SYSMOD' NUR WERT '24' ODER '31' ZULAESSIG

ASS2249    V49 - NOTE
ASS2249    OVERFLOW OF THE SYSTEM VARIABLE SYMBOL '&SYSNDX'
ASS2249    UEBERLAUF DES VARIABLEN SYSTEMPARAMETERS '&SYSNDX'
```

Bedeutung
Dieser Systemparameter ist ein Zaehler der variablen Parameter (maximal 10000 zulaessig).

```
ASS2250    V50 - SIGNIFICANT ERROR
ASS2250    INVALID SYMBOLIC PARAMETER NAME
ASS2250    PARAMETERNAME FEHLERHAFT
```

Bedeutung
Moegliche Fehlerursache:
- Der Parametername besteht nur aus dem &-Zeichen;
- der Parametername beginnt mit einer Ziffer;
- der Parametername enthaelt unzulaessige Zeichen.

```
ASS2251    V51 - WARNING
ASS2251    INVALID SUBLIST
ASS2251    UNGUELTIGE UNTERLISTE

ASS2252    V52 - SIGNIFICANT ERROR
ASS2252    SUBSCRIPTED GLOBAL VARIABLE SYMBOL (&00) IS REFERENCED WITHOUT SUBSCRIPT. (&01)
ASS2252    NICHT-INDIZIERTER ZUGRIFF AUF GLOBALEN INDIZIERTEN VARIABLEN PARAMETER (&00).
           (&01)
```

Bedeutung
Auf indiziert deklarierte variable Parameter darf nur indiziert zugegriffen werden.
(&01): STMT-Nr. der Erstdefinition.

```
ASS2253    V53 - SIGNIFICANT ERROR
ASS2253    ILLEGAL SUBSCRIPT OR SUBLIST REFERENCE TO NONSUBSCRIPTED GLOBAL VARIABLE SYMBOL
           (&00). (&01)
ASS2253    INDEX- ODER UNTERLISTENZUGRIFF AUF GLOBALEN NICHT-INDIZIERTEN VARIABLEN
           PARAMETER (&00) UNZULAESSIG. (&01)
```

Bedeutung
Nur auf indizierte variable Parameter kann indiziert zugegriffen werden.
(&01): STMT-Nr. der Erstdefinition.

```
ASS2254   V54 - SIGNIFICANT ERROR
ASS2254   ILLEGAL SUBSCRIPT OR SUBLIST REFERENCE TO NONSUBSCRIPTED LOCAL VARIABLE SYMBOL
          (&00).
ASS2254   INDEX- ODER UNTERLISTENZUGRIFF AUF LOKALEN NICHT-INDIZIERTEN VARIABLEN
          PARAMETER (&00) UNZULAESSIG
```

Bedeutung
Nur auf indizierte variable Parameter kann indiziert zugegriffen werden.

```
ASS2255   V55 - SIGNIFICANT ERROR
ASS2255   ILLEGAL SUBLIST REFERENCE TO THE LOCAL VARIABLE SYMBOL (&00)
ASS2255   UNTERLISTENZUGRIFF AUF LOKALEN VARIABLEN PARAMETER (&00) UNZULAESSIG
```

Bedeutung
Unterlistenzugriffe sind nur bei symbolischen Parametern zulaessig.

```
ASS2256   V56 - SIGNIFICANT ERROR
ASS2256   INVALID '&SYSLIST' REFERENCE TO NAME FIELD. SUBSTITUTION VALUE: NAME FIELD
          ENTRY
ASS2256   '&SYSLIST'-ZUGRIFF AUF NAMENSFELD FEHLERHAFT. ERSATZWERT: NAMENSFELD-EINTRAG
```

Bedeutung
Der Namensfeldeintrag kann nur ueber &SYSLIST(0) referenziert werden.

```
ASS2257   V57 - SIGNIFICANT ERROR
ASS2257   ILLEGAL SUBLIST REFERENCE TO THE GLOBAL VARIABLE SYMBOL (&00). (&01)
ASS2257   UNTERLISTENZUGRIFF AUF GLOBALEN VARIABLEN PARAMETER (&00) UNZULAESSIG. (&01)
```

Bedeutung
Unterlistenzugriffe sind nur bei symbolischen Parametern zulaessig.
(&01): STMT-Nr. der Erstdefinition.

```
ASS2258   V58 - SIGNIFICANT ERROR
ASS2258   THE SUBSCRIPTED LOCAL VARIABLE SYMBOL (&00) CANNOT BE REFERENCED WITHOUT A
          SUBSCRIPT.
ASS2258   NICHT-INDIZIERTER ZUGRIFF AUF LOKALEN INDIZIERTEN VARIABLEN PARAMETER (&00)
          UNZULAESSIG
```

Bedeutung
Auf einen indiziert deklarierten variablen Parameter darf nur indiziert zugegriffen werden.

```
ASS2259   V59 - NOTE
ASS2259   SYMBOLIC PARAMETERS AND 'SET' SYMBOLS MUST NOT BEGIN WITH 'SYS'
ASS2259   'SET'- UND SYMBOLISCHE PARAMETER DUERFEN NICHT MIT 'SYS' BEGINNEN
```

```
ASS2260   V60 - SIGNIFICANT ERROR
ASS2260   NAME FIELD CONTAINS ILLEGAL SET SYMBOL
ASS2260   'SET'-PARAMETER IM NAMENSFELD FEHLERHAFT
```

Bedeutung
Moegliche Fehlerursachen:
Der im Namensfeld angegebene SET-Parameter
- ist syntaktisch unzulaessig;
- ist ein nur lesend zu gebrauchender Systemparameter.

```
ASS2262   V62 - SIGNIFICANT ERROR
ASS2262   UNDEFINED VARIABLE SYMBOL IN NAME FIELD OF A 'SET' INSTRUCTION
ASS2262   VARIABLER PARAMETER IM NAMENSFELD DER 'SET'-ANWEISUNG UNDEFINIERT
```

Bedeutung
Im Namensfeld der SET-Anweisung trat ein variabler Parameter auf, der nicht bzw. doppelt definiert wurde.

Maßnahme
Moegliche Massnahmen:
- SET-Parameter vorher eindeutig deklarieren;
- SETx-Anweisung mit Typangabe verwenden.

```
ASS2263   V63 - SIGNIFICANT ERROR
ASS2263   ILLEGAL VARIABLE SYMBOL OR 'SETC' EXPRESSION IN THE NAME FIELD
ASS2263   VARIABLER PARAMETER BZW. 'SETC'-AUSDRUCK IM NAMENSFELD UNZULAESSIG
```

```
ASS2264   V64 - SIGNIFICANT ERROR
ASS2264   GENERATED 'SET' SYMBOL IN THE NAME FIELD OF A 'SET' INSTRUCTION IS ILLEGAL
ASS2264   GENERIERTER 'SET'-PARAMETER IM NAMENSFELD EINER 'SET'-ANWEISUNG UNZULAESSIG
```

Bedeutung
Im Namensfeld einer SET-Anweisung trat ein generiertes variables Symbol auf. Dies ist nur bei SETx-Anweisungen mit Typangabe zulaessig.

Maßnahme
Statt 'SET' 'SETx' mit Typangabe verwenden.

```
ASS2265   V65 - SIGNIFICANT ERROR
ASS2265   TYPE INCONSISTENCY BY USE OF THE LOCAL VARIABLE SYMBOL IN THE NAME FIELD
ASS2265   TYP-UNVERTRAEGLICHKEIT BEI VERWENDUNG DES LOKALEN VARIABLEN PARAMETERS IM
          NAMENSFELD
```

Bedeutung
Zugewiesener Wert entspricht nicht dem Deklarationstyp des variablen Parameters.

```
ASS2266    V66 - SIGNIFICANT ERROR
ASS2266    ILLEGAL IMPLICIT DEFINITION OF VARIABLE SYMBOL IN NAME FIELD OF A 'SET'
           INSTRUCTION
ASS2266    IMPLIZITE DEFINITION DES IM NAMENSFELD EINER 'SET'-ANWEISUNG STEHENDEN
           VARAIBLEN PARAMETERS UNZULAESSIG
```

Bedeutung
Eine implizite Definition ist nur mit SETA, SETB oder SETC zulaessig, nicht jedoch mit SET.

```
ASS2267    V67 - SIGNIFICANT ERROR
ASS2267    WRONG NAME FIELD ENTRY IN CORRESPONDING PROTOTYPE STATEMENT
ASS2267    NAMENSFELDEINTRAG IN ZUGEHOERIGER MUSTERANWEISUNG FEHLERHAFT

ASS2268    V68 - SIGNIFICANT ERROR
ASS2268    VARIABLE SYMBOL (&00) TOO LONG; MAXIMUM VALUE = 64
ASS2268    VARIABLER PARAMETER (&00) ZU LANG; MAXIMALLAENGE = 64

ASS2269    V69 - SIGNIFICANT ERROR
ASS2269    SYNTAX ERROR IN VARIABLE SYMBOL (&00)
ASS2269    VARIABLER PARAMETER (&00) SYNTAKTISCH FALSCH

ASS2270    V70 - SIGNIFICANT ERROR
ASS2270    BLANKS ARE NOT ALLOWED AS PART OF VARIABLE OR SEQUENCE SYMBOLS
ASS2270    BLANKS ALS BESTANDTEIL VARIABLER PARAMETER BZW. FOLGESYMBOLE UNZULAESSIG

ASS2271    V71 - SIGNIFICANT ERROR
ASS2271    SYNTAX ERROR IN PARAMETER (&00) OF THE PROTOTYPE STATEMENT
ASS2271    SYNTAX-FEHLER IN OPERAND (&00) DER MUSTERANWEISUNG

ASS2273    V73 - SIGNIFICANT ERROR
ASS2273    ILLEGAL VALUE ASSIGNMENTS TO SUBLIST ELEMENTS
ASS2273    WERTZUWEISUNGEN AN UNTERLISTENELEMENTE UNZULAESSIG

ASS2274    V74 - SIGNIFICANT ERROR
ASS2274    ILLEGAL VALUE ASSIGNMENT TO THE SYSTEM VARIABLE SYMBOL '&SYSLIST'
ASS2274    WERTZUWEISUNG AN VARIABLEN SYSTEMPARAMETER '&SYSLIST' UNZULAESSIG
```

Bedeutung
Es wurde weder in der Musteranweisung ein entspechender symbolischer Parameter uebergeben noch im Makroaufruf ein zugehoeriger Eintrag gemacht.

```
ASS2410    X10 - SIGNIFICANT ERROR
ASS2410    VALUE OF THE SETB EXPRESSION IS NEITHER '0' NOR '1'
ASS2410    WERT DES 'SETB'-AUSDRUCKS WEDER '0' NOCH '1'
```

Bedeutung
Ein logischer Ausdruck kann nur die Werte '0' oder '1' annehmen.

```
ASS2412    X12 - WARNING
ASS2412    VALUE OF THE ARITHMETIC EXPRESSION IN THE AGO INSTRUCTION IS NEGATIVE, '0', OR
           GREATER THAN THE NUMBER OF SUPPLIED SEQUENCE SYMBOLS: NO BRANCH
ASS2412    WERT DES ARITHMETISCHEN AUSDRUCKS IN 'AGO'-ANWEISUNG NEGATIV, '0' ODER GROESSER
           ALS ANZAHL DER MITGEGEBENEN FOLGESYMBOLE: KEIN SPRUNG

ASS2413    X13 - SIGNIFICANT ERROR
ASS2413    'ACTR' OPERAND IS NEGATIVE
ASS2413    'ACTR'-OPERAND NEGATIV
```

Bedeutung
Der Operand einer ACTR-Anweisung muss ein positiver arithmetischer Ausdruck sein.

```
ASS2414    X14 - SIGNIFICANT ERROR
ASS2414    STRING IN OPERAND TOO LONG; MAXIMUM VALUE = 1020
ASS2414    ZEICHENSTRING IN OPERAND ZU LANG; MAXIMALLAENGE = 1020

ASS2415    X15 - SIGNIFICANT ERROR
ASS2415    ILLEGAL STRING OR NULL STRING IN DUPLICATION FACTOR OF 'SETC' OPERAND
ASS2415    ZEICHENSTRING ODER NULLSTRING IM WIEDERHOLUNGSFAKTOR IN 'SETC'-OPERAND
           UNZULAESSIG

ASS2416    X16 - SIGNIFICANT ERROR
ASS2416    ILLEGAL STRING OR NULL STRING AS ARGUMENT OF THE SUBSTRING FUNCTION
ASS2416    ZEICHENSTRING ODER NULLSTRING ALS ARGUMENT DER SUBSTRING-FUNKTION UNZULAESSIG

ASS2417    X17 - SIGNIFICANT ERROR
ASS2417    ARGUMENT OF THE SUBSTRING FUNCTION IS '0' OR NEGATIVE
ASS2417    ARGUMENT DER SUBSTRING-FUNKTION '0' ODER NEGATIV

ASS2419    X19 - SIGNIFICANT ERROR
ASS2419    NEGATIVE DUPLICATION FACTOR IN 'SETC' OPERAND
ASS2419    WIEDERHOLUNGSFAKTOR IN 'SETC'-OPERAND NEGATIV
```

```
ASS2420    X20 - SIGNIFICANT ERROR
ASS2420    ILLEGAL ATTRIBUTE REFERENCE
ASS2420    MERKMAL-BEZUG FEHLERHAFT
```

Bedeutung
Dem Merkmal folgt kein Symbol/Parameter; Dem K-, N- Merkmal folgt ein Symbol (nur Parameter erlaubt).

```
ASS2421    X21 - SIGNIFICANT ERROR
ASS2421    ILLEGAL REFERENCE TO 'T' AND 'D' ATTRIBUTE IN A 'SETA' EXPRESSION
ASS2421    BEZUG AUF 'T'- UND 'D'-MERKMAL IM 'SETA'-AUSDRUCK UNZULAESSIG
```

```
ASS2422    X22 - SIGNIFICANT ERROR
ASS2422    ILLEGAL REFERENCE TO THE 'K' ATTRIBUTE FOR VARIABLE SYMBOL (&00)
ASS2422    BEZUG AUF 'K'-MERKMAL DES VARIABLEN PARAMETERS (&00) UNZULAESSIG
```

Bedeutung
Es erfolgte ein nicht-indizierter Zugriff auf das K-Merkmal einer indizierten SET-Parameters oder auf &SYSLIST.

```
ASS2423    X23 - SIGNIFICANT ERROR
ASS2423    ILLEGAL SUBSCRIPTED REFERENCE TO THE 'N' ATTRIBUTE FOR VARIABLE SYMBOL (&00)
ASS2423    BEZUG AUF 'N'-MERKMAL DES VARIABLEN PARAMETERS (&00) BEI INDIZIERTEM ZUGRIFF
           UNZULAESSIG
```

Bedeutung
Das N-Merkmal fuer indizierte SET-Parameter ist nur bei nicht indiziertem Zugriff definiert.

```
ASS2424    X24 - WARNING
ASS2424    ATTRIBUTE REFERENCE NOT DEFINED; DEFAULT VALUE INSERTED
ASS2424    MERKMAL-BEZUG FEHLERHAFT; ERSATZWERT WIRD VERWENDET
```

Bedeutung
Dem Merkmal folgt kein Symbol/Parameter bzw. der Symbol-/Parameter-Name ist syntaktisch falsch.

```
ASS2425    X25 - SIGNIFICANT ERROR
ASS2425    LENGTH OF THE SYMBOL (&00) CANNOT BE EVALUATED
ASS2425    LAENGE DES SYMBOLS (&00) NICHT BESTIMMBAR
```

```
ASS2427    X27 - SIGNIFICANT ERROR
ASS2427    ILLEGAL ATTRIBUTE REFERENCE: VALUE OF SYMBOLIC PARAMETER IS SYMBOL NAME
ASS2427    MERKMAL-BEZUG FEHLERHAFT: WERT DES SYMBOLISCHEN PARAMETERS IST SYMBOLNAME
```

```
ASS2433    X33 - SIGNIFICANT ERROR
ASS2433    SUBSCRIPT OF THE VARIABLE SYMBOL (&00) IS '0' OR NEGATIVE
ASS2433    INDEX DES VARIABLEN PARAMETERS (&00) '0' ODER NEGATIV
```

Bedeutung
Bei indizierten variablen Parameter muss der Index > 0 sein.

Maßnahme
Den Index durch einen solchen > 0 ersetzen.

```
ASS2434    X34 - SIGNIFICANT ERROR
ASS2434    ILLEGAL SUBLIST REFERENCE TO VARIABLE SYMBOL (&00) WITH SUBSCRIPT '0'
ASS2434    UNTERLISTENZUGRIFF AUF VARIABLEN PARAMETER (&00) MIT INDEX '0' UNZULAESSIG
```

Bedeutung
Der Index 0 ist nur fuer den Zugriff auf den Namensfeldeintrag ueber &SYSLIST(0) zulaessig.

Maßnahme
Den Index durch einen solchen > 0 ersetzen.

```
ASS2435    X35 - SIGNIFICANT ERROR
ASS2435    SUBSCRIPT OF THE VARIABLE SYMBOL (&00) IS '0' OR NEGATIVE
ASS2435    INDEX DES VARIABLEN PARAMETERS (&00) '0' ODER NEGATIV

ASS2436    X36 - SIGNIFICANT ERROR
ASS2436    SUBSCRIPTED USE OF PREDEFINED 'SET' SYMBOL IS ILLEGAL
ASS2436    INDIZIERTE VERWENDUNG VORDEFINIERTER 'SET'-PARAMETER UNZULAESSIG

ASS2437    X37 - WARNING
ASS2437    MULTIPLE ASSIGNMENT TO KEYWORD PARAMETER (&00); FIRST ASSIGNMENT IS VALID
ASS2437    MEHRFACH-ZUWEISUNG AN KENNWORTOPERANDEN (&00); ERSTE ZUWEISUNG GILT

ASS2439    X39 - SIGNIFICANT ERROR
ASS2439    SUBSCRIPTING ILLEGAL
ASS2439    INDIZIERUNG UNZULAESSIG

ASS2441    X41 - SIGNIFICANT ERROR
ASS2441    INVALID SEVERITY CODE (> 255) IN 'MNOTE' INSTRUCTION REPLACED BY ERROR CODE '0'
ASS2441    UNGUELTIGER FEHLERCODE (GROESSER 255) IN 'MNOTE'-ANWEISUNG DURCH FEHLERCODE '0'
           ERSETZT

ASS2448    X48 - SIGNIFICANT ERROR
ASS2448    ILLEGAL OPERAND FORMAT IN 'MNOTE' INSTRUCTION
ASS2448    OPERANDENFORMAT BEI 'MNOTE'-ANWEISUNG FEHLERHAFT
```

Bedeutung
Zulaessige Operanden fuer MNOTE sind Fehlercode und Fehlersatz in Hochkommas.

```
ASS2449    X49 - SIGNIFICANT ERROR
ASS2449    OPERAND NOT ENCLOSED WITHIN SINGLE QUOTES
ASS2449    OPERAND NICHT IN HOCHKOMMATA EINGESCHLOSSEN
```

Bedeutung
Der Meldungstext einer MNOTE- oder TITLE-Anweisung ist nicht in Hochkommata eingeschlossen (kein oder nur ein Hochkomma).

Maßnahme
Hochkomma(ta) einfuegen.

```
ASS2451    X51 - SIGNIFICANT ERROR
ASS2451    ONLY OPERAND 1 OF THE 'MNOTE' INSTRUCTION IS VALID
ASS2451    NUR OPERAND 1 BEI 'MNOTE'-ANWEISUNG GUELTIG
```

Bedeutung
In der 'MNOTE'-Anweisung ist der 1. Operand in Hochkommas eingeschlossen, die darauffolgenden weiteren Operanden werden ignoriert.

Maßnahme
Entfernen der ueberzaehlingen Operanden oder Hochkommata beim 1. Operanden.

```
ASS2452    X52 - SIGNIFICANT ERROR
ASS2452    INVALID SEVERITY CODE IN 'MNOTE'; THE INSTRUCTION WILL BE IGNORED
ASS2452    FEHLERCODE IN 'MNOTE'-ANWEISUNG UNGUELTIG; ANWEISUNG WIRD IGNORIERT

ASS2453    X53 - SIGNIFICANT ERROR
ASS2453    OPERANDS REQUIRED IN THE 'MNOTE' INSTRUCTION ARE MISSING OR EMPTY
ASS2453    ERFODERLICHE OPERANDEN IN 'MNOTE'-ANWEISUNG FEHLEN BZW. SIND LEER

ASS2454    X54 - WARNING
ASS2454    NO MORE THAN TWO OPERANDS ARE ALLOWED IN THE 'MNOTE' INSTRUCTION
ASS2454    MEHR ALS 2 OPERANDEN IN 'MNOTE'-ANWEISUNG UNZULAESSIG
```

Bedeutung
Uberfluessige Operanden werden wie Kommentare behandelt.

```
ASS2455    X55 - SIGNIFICANT ERROR
ASS2455    CHARACTER EXPRESSION NOT ALLOWED IN ARITHMETIC OR LOGICAL EXPRESSION
ASS2455    ZEICHENAUSDRUCK IM ARITHMETISCHEN ODER LOGISCHEN AUSDRUCK UNZULAESSIG
```

Bedeutung
Zeichenausdruecke in SETA-/SETB-Ausdruecken als Operand bzw. in arithmetischen oder logischen Verknuepfungen sind unzulaessig.

```
ASS2510    Y10 - SIGNIFICANT ERROR
ASS2510    REFERENCED ADDRESS NOT IN RANGE DEFINED BY 'USING' INSTRUCTION
ASS2510    ANGESPROCHENE ADRESSE AUSSERHALB DES DURCH 'USING'-ANWEISUNG ERFASSTEN BEREICHS
```

Bedeutung

Moegliche Fehlerursachen:

- Die angegebene Adresse wird nicht durch ein Basisregister abgedeckt;
- das Basisregister ist gesperrt.

```
ASS2511    Y11 - WARNING
ASS2511    MISSING USING INSTRUCTION; BASE REGISTER '0' USED
ASS2511    'USING'-ANWEISUNG FEHLT, BASISREGISTER '0' WIRD VERWENDET
```

```
ASS2640    Z40 - SIGNIFICANT ERROR
ASS2640    'ACTR' EXCEEDED WHEN PROCESSING A MACRO
ASS2640    'ACTR'-UEBERLAUF BEI BEARBEITUNG EINES MAKROS
```

Bedeutung

Die maximale Anzahl von AGO- und AIF-Anweisungen wurde ueberschritten. Diese Anzahl wird durch die ACTR-Anweisung festgelegt;
Voreinstellung: 4096. Die Makro-Bearbeitung wird abgebrochen.

Maßnahme

Moegliche Massnahmen:

- ACTR-Zaehler mit der ACTR-Anweisung hochsetzen;
- Programm auf eine Endlosschleife pruefen.

```
ASS2641    Z41 - FAILURE
ASS2641    'ACTR' EXCEEDED WHEN PROCESSING MACRO INSTRUCTIONS IN THE SOURCE
ASS2641    'ACTR'-UEBERLAUF BEI BEARBEITUNG VON MAKROANWEISUNGEN IN DER SOURCE
```

Bedeutung

Die maximale Anzahl von AGO- und AIF-Anweisungen wurde ueberschritten. Diese Anzahl wird durch die ACTR-Anweisung festgelegt;
Voreinstellung: 4096. Die Assemblierung wird abgebrochen.

Maßnahme

Moegliche Massnahmen:

- ACTR-Zaehler mit der ACTR-Anweisung hochsetzen;
- Programm auf eine Endlosschleife pruefen.

```
ASS2642    Z42 - SIGNIFICANT ERROR
ASS2642    PROGRAM COUNTER OVERFLOW
ASS2642    BEFEHLSZAEHLER UEBERLAUF
```

```
ASS6000    Z00 - FATAL ERROR
ASS6000    INTERNAL ERROR IN ASSEMBH: UNEXPECTED 'SPL4_RTS_GET_HEAP_RC' IN 'IARH850'
ASS6000    INTERNER FEHLER IM ASSEMBH: UNZULAESSIGER 'SPL4_RTS_GET_HAEP_RC' IN 'IARH850'
```

Maßnahme
Systemverwalter verstaendigen.

```
ASS6001    FATAL ERROR
ASS6001    INTERNAL ERROR IN ASSEMBH: UNEXPECTED RETURN CODE: (&00) IN MACRO (&01) IN
           'IARH850'
ASS6001    INTERNER FEHLER IM ASSEMBH: UNZULAESSIGER RETURN-CODE: (&00) IN MAKRO (&01) IN
           'IARH850'
```

Maßnahme
Systemverwalter verstaendigen.

```
ASS6002    Z02 - FAILURE
ASS6002    INTERNAL ERROR IN ASSEMBH: UNEXPECTED 'INSTRUCTION-SET' IN MODULE
           'IARH_OCTAB_COPY_700'
ASS6002    INTERNER FEHLER IM ASSEMBH: UNZULAESSIGER 'INSTRUCTION-SET' IM MODUL
           'IARH_OCTAB_COPY_700'
```

Bedeutung
Assembler-Abbruch wegen Systemfehler.

Maßnahme
Systemverwalter verstaendigen.

```
ASS6003    NO ERRORS
ASS6003    FILE CANNOT BE OPENED
ASS6003    DATEI KANN NICHT GEOEFFNET WERDEN

ASS6004    Z11 - FAILURE
ASS6004    OVERFLOW OF THE GENERATION BUFFER; MAX. SIZE IS 32K BYTE
ASS6004    MAXIMALE GROESSE DES GENERIERUNGSPUFFERS VON 32K BYTE UEBERSCHRITTEN

ASS6005    NO ERRORS
ASS6005    LISTING GENERATOR TIME FOR 'SAVLST'-CREATION: (&00) MSEC
ASS6005    ZEIT DES LISTEN-GENERATORS FUER 'SAVLST'-ERSTELLUNG: (&00) MSEC

ASS6006    NO ERRORS
ASS6006    LISTING GENERATOR TIME: (&00) MSEC
ASS6006    ZEIT DES LISTEN-GENERATORS: (&00) MSEC

ASS6007    NO ERRORS
ASS6007    TIME OF THE COMPONENT (&00): (&01) MSEC
ASS6007    ZEIT DER KOMPONENTE (&00): (&01) MSEC
```

```
ASS6008    NO ERRORS
ASS6008    ABNORMAL PROGRAM TERMINATION; ASSEMBH RETURN CODE: (&00)
ASS6008    ABNORMALE PROGRAMMBEENDIGUNG, ASSEMBH-RETURN-CODE: (&00)

ASS6009    NOTE
ASS6009    'MNOTE' WITH 'SEVERITY CODE' (&00)
ASS6009    'MNOTE' MIT 'SEVERITY CODE' (&00)
```

Bedeutung
Jede Zeile des MNOTE-XREF beginnt mit diesem Text.
(&00): Fehlercode 0 ... 255 der 'MNOTE'-Anweisung.

```
ASS6010    NO ERRORS
ASS6010    (&00) OF BS2000 ASSEMBH(&01) READY
ASS6010    (&00) DES BS2000 ASSEMBH(&01) READY

ASS6011    NO ERRORS
ASS6011    ASSEMBLY TIME: (&00) MSEC
ASS6011    ZEIT DER ASSEMBLIERUNG: (&00) MSEC

ASS6012    NO ERRORS
ASS6012    END OF ASSEMBH(&00)
ASS6012    ENDE ASSEMBH(&00)
```

Bedeutung
(&00): ASSEMBH-Leistungsumfang (XT, GA oder GEN).

```
ASS6013    Z13 - FAILURE
ASS6013    INTERNAL ERROR IN ASSEMBH: STXIT IN INSTRUCTION (&00). STXIT ACTIVATED
ASS6013    INTERNER FEHLER IM ASSEMBH: STXIT IN ANWEISUNG (&00). STXIT AKTIVIERT

ASS6014    Z14 - FATAL ERROR
ASS6014    FILE (&00) CANNOT BE CLOSED; RETURN CODE: (&01)
ASS6014    DATEI (&00) KANN NICHT GESCHLOSSEN WERDEN; RETURN-CODE: (&01)

ASS6017    Z17 - FATAL ERROR
ASS6017    INTERNAL ERROR IN ASSEMBH: FILE (&00) CANNOT BE OPENED; RETURN CODE = (&01)
ASS6017    INTERNER FEHLER IM ASSEMBH: OEFFNEN DER DATEI (&00) NICHT MOEGLICH;
           RETURN-CODE: (&01)
```

Bedeutung
Diese Meldung ist fuer die ASSEMBH-Entwicklung bestimmt.

Maßnahme
Systemverwalter verstaendigen.

```
ASS6018    NO ERRORS
ASS6018    (&00) FLAGS, (&01) PRIVILEGED FLAGS, (&02) MNOTES
ASS6018    (&00) FLAGS, (&01) PRIVILEGED FLAGS, (&02) MNOTES
```

Bedeutung
Statistikdaten.
(&00): Summe der aufgetretenen Flags;
(&01): Summe der aufgetretenen privilegierten Flags;
(&02): Summe der aufgetretenen Macronotes.

```
ASS6019    NO ERRORS
ASS6019    HIGHEST ERROR-WEIGHT: (&00)
ASS6019    HIGHEST ERROR-WEIGHT: (&00)
```

```
ASS6020    Z20 - FAILURE
ASS6020    INTERNAL ERROR IN ASSEMBH: ILLEGAL RECORD TYPE IN THE LOCATION COUNTER BASE
           CHAIN (PSTAB)
ASS6020    INTERNER FEHLER IM ASSEMBH: UNZULAESSIGER SATZTYP IN DER ADRESSPEGEL-BASISKETTE
           (PSTAB)
```

Bedeutung
Assembler-Abbruch.

Maßnahme
Systemverwalter verstaendigen.

```
ASS6021    Z21 - FAILURE
ASS6021    INTERNAL ERROR IN ASSEMBH: ILLEGAL OPCODE IN THE INTERMEDIATE LANGUAGE
ASS6021    INTERNER FEHLER IM ASSEMBH: UNZULAESSIGER OPERATIONS-CODE IN DER
           ZWISCHENSPRACHE
```

Bedeutung
Assembler-Abbruch.

Maßnahme
Systemverwalter verstaendigen.

```
ASS6022    Z22 - FAILURE
ASS6022    INTERNAL ERROR IN ASSEMBH: UNEXPECTED 'FILE-TYPE' IN THE 'FILE-DESCRIPTOR'
ASS6022    INTERNER FEHLER IM ASSEMBH: UNERWARTETER 'FILE-TYPE' IM 'FILE-DESCRIPTOR'
```

Maßnahme
Systemverwalter verstaendigen.

```
ASS6023   Z23 - FAILURE
ASS6023   INTERNAL ERROR IN ASSEMBH: ILLEGAL RETURN CODE OF THE PARSER
ASS6023   INTERNER FEHLER IM ASSEMBH: UNZULAESSIGER RETURN-CODE DES ZERTEILERS
```

Maßnahme
Systemverwalter verstaendigen.

```
ASS6024   Z24 - FAILURE
ASS6024   INTERNAL ERROR IN ASSEMBH: PARSER OVERFLOW
ASS6024   INTERNER FEHLER IM ASSEMBH: UEBERLAUF DES ZERTEILERS
```

Maßnahme
Systemverwalter verstaendigen.

```
ASS6025   Z25 - FAILURE
ASS6025   INTERNAL ERROR IN ASSEMBH: ERROR DURING ACCESS TO THE PARSER TABLE
ASS6025   INTERNER FEHLER IM ASSEMBH: FEHLER BEIM ZUGRIFF AUF ZERTEILER-TABELLE
```

Maßnahme
Systemverwalter verstaendigen.

```
ASS6026   Z26 - FAILURE
ASS6026   INTERNAL ERROR IN ASSEMBH: UNDERFLOW OF THE SEMANTIC STACK OF THE EXPRESSION
          ANALYSIS
ASS6026   INTERNER FEHLER IM ASSEMBH: UNTERLAUF DES SEMANTISCHEN STACKS DER
          AUSDRUCKSBEARBEITUNG
```

Maßnahme
Systemverwalter verstaendigen.

```
ASS6029   FATAL ERROR
ASS6029   DMS ERROR (&00) WHEN OPENING THE SOURCE. IN SYSTEM MODE: /HELP-MSG DMS(&00)
ASS6029   DVS-FEHLER '(&00)' BEIM OEFFNEN DER SOURCE. IM SYSTEMMODUS: /HELP-MSG DMS(&00)
```

Bedeutung
Beim Aufruf des ASSEMBH als Unterprogramm kann die Source nicht geoeffnet werden. Naehere Information ueber den DVS-Fehlerschluessel kann ueber /HELP-MSG im Systemmodus erfragt bzw. dem BS2000-Handbuch 'Systemmeldungen' oder einem der BS2000-DVS-Handbuecher entnommen werden.

Maßnahme
Source korrekt angeben.

```
ASS6030   Z30 - FATAL ERROR
ASS6030   INTERNAL ERROR IN ASSEMBH: SYSDTA OPEN ERROR (&00)
ASS6030   INTERNER FEHLER IM ASSEMBH: SYSDTA OPEN ERROR (&00)
```

Maßnahme
Systemverwalter verstaendigen.

```
ASS6031   Z31 - FATAL ERROR
ASS6031   INTERNAL ERROR IN ASSEMBH: OMF CLOSE ERROR (&00)
ASS6031   INTERNER FEHLER IM ASSEMBH: OMF CLOSE ERROR (&00)
```

Maßnahme
Systemverwalter verstaendigen.

```
ASS6032   Z32 - FAILURE
ASS6032   PLAM-LIB OPEN ERROR (&00) WHEN WRITING THE OBJECT MODULE
ASS6032   PLAM-LIB OPEN FEHLER (&00) BEI OBJEKTMODUL-AUSGABE
```

Bedeutung
Beim Oeffnen des Plam-Bibliothekselements trat ein Fehler auf.

```
ASS6033   Z33 - FATAL ERROR
ASS6033   EAM-OMF OPEN ERROR (&00)
ASS6033   EAM-OMF OPEN FEHLER (&00)
```

```
ASS6034   Z34 - FATAL ERROR
ASS6034   INTERNAL ASSEMBH ERROR. STREAM-STATUS-LIST IS NOT CORRECT (&00). STREAM COULD
          NOT BE OPENED
ASS6034   INTERNER FEHLER IM ASSEMBH: STROM-STATUS-LISTE NICHT KORREKT (&00). STROM
          KONNTE NICHT GEOEFFNET WERDEN
```

```
ASS6035   Z35 - FATAL ERROR
ASS6035   INTERNAL ERROR IN ASSEMBH: OPEN ERROR (&01) ON INPUT FILE (&00)
ASS6035   INTERNER FEHLER IM ASSEMBH: FEHLER (&01) BEIM EROEFFNEN DER EINGABEDATEI (&00)
```

Bedeutung
Diese Meldung ist fuer die ASSEMBH-Entwicklung bestimmt.

```
ASS6036   Z36 - FATAL ERROR
ASS6036   INTERNAL ERROR IN ASSEMBH: INPUT/OUTPUT NOT INITIALIZED
ASS6036   INTERNER FEHLER IM ASSEMBH: EIN-/AUSGABE NICHT INITIALISIERT
```

Maßnahme
Systemverwalter verstaendigen.

```
ASS6037    Z37 - FATAL ERROR
ASS6037    INTERNAL ERROR IN ASSEMBH: SYSDTA CLOSE ERROR (&00)
ASS6037    INTERNER FEHLER IM ASSEMBH: SYSDTA CLOSE ERROR (&00)
```

Bedeutung
Diese Meldung ist fuer die ASSEMBH-Entwicklung bestimmt.

Maßnahme
Systemverwalter verstaendigen.

```
ASS6038    Z38 - FATAL ERROR
ASS6038    INTERNAL ERROR IN ASSEMBH: WRONG FILE TYPE IN DATATAB
ASS6038    INTERNER FEHLER IM ASSEMBH: UNZULAESSIGER DATEITYP IN 'DATATAB'
```

Maßnahme
Systemverwalter verstaendigen.

```
ASS6040    Z01 - FAILURE
ASS6040    INTERNAL ASSEMBH ERROR DURING TEXT REPLACEMENT
ASS6040    INTERNER FEHLER IM ASSEMBH: FEHLER BEI TEXTERSETZUNG
```

Maßnahme
Systemverwalter verstaendigen.

```
ASS6041    Z10 - FAILURE
ASS6041    INTERNAL ERROR IN ASSEMBH: ILLEGAL SYMBOL TYPE FOR ENTRY PROCESSING
ASS6041    INTERNER FEHLER IM ASSEMBH: UNZULAESSIGER SYMBOLTYP IN ENTRY-BEARBEITUNG
```

Maßnahme
Systemverwalter verstaendigen.

```
ASS6042    NO ERRORS
ASS6042    ELAPSED TIME: (&00) SEC
ASS6042    VERBRAUCHTE ZEIT: (&00) SEC

ASS6043    NO ERRORS
ASS6043    OPTION '*INCREMENT' FOR READING LIBRARY ACCESS NOT ALLOWED
ASS6043    OPTION '*INCREMENT' BEI LESENDEM BIBLIOTHEKSZUGRIFF UNZULAESSIG

ASS6044    NO ERRORS
ASS6044    OPTION '*INCREMENT' POSSIBLE ONLY WITH LMS/PLAM V2.0A
ASS6044    OPTION '*INCREMENT' ERST AB LMS/PLAM V2.0A MOEGLICH

ASS6045    NO ERRORS
ASS6045    OPTION '*HIGHEST-EXISTING' POSSIBLE ONLY WITH LMS/PLAM V2.0A
ASS6045    OPTION '*HIGHEST-EXISTING' ERST AB LMS/PLAM V2.0A MOEGLICH
```

```
ASS6050    Z50 - FAILURE
ASS6050    INTERNAL ERROR IN ASSEMBH: ERROR (&00) IN MODULE (&01)
ASS6050    INTERNER FEHLER IM ASSEMBH: FEHLER (&00) IM MODUL (&01)
```

Bedeutung
Diese Meldung ist fuer die ASSEMBH-Entwicklung bestimmt.

Maßnahme
Systemverwalter verstaendigen.

```
ASS6051    FAILURE
ASS6051    INTERNAL ERROR IN ASSEMBH: INVALID 'SET' VALUE
ASS6051    INTERNER FEHLER IM ASSEMBH: 'SET'-WERT UNGUELTIG
```

Maßnahme
Systemverwalter verstaendigen.

```
ASS6052    Z15 - FAILURE
ASS6052    INTERNAL ERROR IN ASSEMBH: ERROR (&00) IN MODULE (&01) IN INCLUDE (&02)
ASS6052    INTERNER FEHLER IM ASSEMBH: FEHLER (&00) IN MODUL (&01) IM INCLUDE (&02)
```

Bedeutung
Diese Meldung ist fuer die ASSEMBH-Entwicklung bestimmt.
(&02): INCLUDE-Name.

Maßnahme
Systemverwalter verstaendigen.

```
ASS6060    FATAL ERROR
ASS6060    SDF SYNTAX FILE NOT CONTAINED IN CATALOG
ASS6060    SDF-SYNTAXDATEI IM KATALOG NICHT ENTHALTEN
```

Bedeutung
Die SDF-Syntaxdatei ist im BS2000 entweder nicht definiert oder nicht aktiviert.

Maßnahme
Systemverwalter verstaendigen.

```
ASS6061   FATAL ERROR
ASS6061   'ASSEMBH' NOT DEFINED IN SDF SYNTAX FILE
ASS6061   'ASSEMBH' NICHT IN SDF-SYNTAXDATEI DEFINIERT
```

Bedeutung
Der Name ASSEMBH ist in der SDF-Syntaxdatei nicht vorhanden.

Maßnahme
Systemverwalter verstaendigen.

```
ASS6062   FAILURE
ASS6062   INTERNAL ERROR IN ASSEMBH: SDF INPUT BUFFER TOO SMALL
ASS6062   INTERNER FEHLER IM ASSEMBH: SDF-EINGABEPUFFER ZU KLEIN
```

Maßnahme
Systemverwalter verstaendigen.

```
ASS6063   FATAL ERROR
ASS6063   SDF NOT LOADED IN BS2000
ASS6063   SDF IN BS2000 NICHT GELADEN
```

Maßnahme
Systemverwalter verstaendigen.

```
ASS6064   FATAL ERROR
ASS6064   SDF SYSTEM ERROR; UNEXPECTED RETURN CODE: (&00)
ASS6064   SDF-SYSTEMFEHLER, UNERWARTETER RETURN-CODE: (&00)
```

Maßnahme
Systemverwalter verstaendigen.

```
ASS6065   NOTE
ASS6065   INVALID VALUES FOR 'MARGINS'; DEFAULT VALUES ARE USED
ASS6065   'MARGINS'-WERTE UNGUELTIG; DURCH STANDARDWERTE ERSETZT
```

Maßnahme
Zulaessige Angaben im Benutzerhandbuch ueberpruefen.

```
ASS6066   Z66 - NOTE
ASS6066   SOURCE OPEN ERROR: (&00)
ASS6066   FEHLER BEIM OEFFNEN DER SOURCE: (&00)
```

Bedeutung
(&00): Fehlerursache

```
ASS6070    Z70 - FAILURE
ASS6070    INTERNAL ERROR IN ASSEMBH: WRONG 'SYMTAB' ENTRY FOR THE SYMBOL (&00)
ASS6070    INTERNER FEHLER IM ASSEMBH: FEHLERHAFTER 'SYMTAB'-EINTRAG FUER SYMBOL (&00)
```

Maßnahme
Systemverwalter verstaendigen.

```
ASS6071    Z71 - FAILURE
ASS6071    NO MEMBER NAME SPECIFIED FOR MODULE OUTPUT AND FIRST 'CSECT' UNNAMED; MODULE
           CANNOT BE OUTPUT
ASS6071    FUER MODULAUSGABE KEIN ELEMENTNAME ANGEGEBEN UND ERSTE 'CSECT' UNBENANNT; MODUL
           WIRD NICHT AUSGEGEBEN
```

Bedeutung
Ein Elementname kann nicht ermittelt werden, da 1.CSECT unbenannt und die Elementangabe in den Compileroptions fehlt. Die Uebersetzung wird abgebrochen.

```
ASS6072    Z72 - FAILURE
ASS6072    ERROR WHILE READING A SOURCE STATEMENT; RETURN CODE: (&00)
ASS6072    FEHLER BEIM LESEN DES SOURCE-STATEMENTS; RETURN-CODE: (&00)
```

```
ASS6073    Z73 - WARNING
ASS6073    ERROR ON OPENING THE MACRO/COPY LIBRARY (&00). LIBRARY IGNORED. ORIGIN ERROR:
           (&01)
ASS6073    FEHLER BEIM OEFFNEN DER MAKRO-/COPY-BIBLIOTHEK (&00). BIBLIOTHEK WIRD
           UEBERGANGEN.
           PRIMAERFEHLER: (&01)
```

Bedeutung
Die angegebene Makro- bzw. Copy-Bibliothek konnte nicht geoeffnet werden.
(&00): Bibliotheksname, Elementname
(&01): Fehlerursache.

```
ASS6074    Z74 - FAILURE
ASS6074    INTERNAL ERROR IN ASSEMBH: ERROR (&01) WHILE GENERATING A (&00) RECORD
ASS6074    INTERNER FEHLER IM ASSEMBH: FEHLER (&01) BEIM ERZEUGEN EINES (&00)-SATZES
```

Bedeutung
Diese Meldung ist fuer die ASSEMBH-Entwicklung bestimmt.

Maßnahme
Systemverwalter verstaendigen.

```
ASS6075    Z75 - FAILURE
ASS6075    INSUFFICIENT MEMORY WHEN GENERATING THE INTERNAL TABLE (&00)
ASS6075    SPEICHERMANGEL BEIM ERZEUGEN DER INTERNEN TABELLE (&00)
```

Maßnahme
Speicherberechtigung von der Systemverwaltung hochsetzen lassen.

```
ASS6076    Z76 - NOTE
ASS6076    SOURCE FILE OR SOURCE MEMBER HAS WRONG TYPE: (&00)
ASS6076    SOURCE-FILE BZW. SOURCE-ELEMENT HAT FALSCHEN TYP: (&00)

ASS6080    NOTE
ASS6080    INTERNAL ASSEMBH ERROR IN OPTION TREATMENT. DEFAULT OPTION VALUES USED
ASS6080    INTERNER FEHLER IM ASSEMBH: FEHLER IN DER OPTIONS-VERARBEITUNG; STANDARD-
           OPTIONS-WERTE VERWENDET

ASS6081    NOTE
ASS6081    INVALID OPTION KEYWORD (&00); OPTION WILL BE IGNORED UNTIL NEXT '*' OR ',' IS
           DETECTED
ASS6081    OPTION-SCHLUESSELWORT (&00) UNGUELTIG; OPTION WIRD BIS ZUM NAECHSTEN '*' ODER
           ',' IGNORIERT
```

Bedeutung
Nach *COMOPT wurde ein nicht identifizierbares Option-Schluesselwort angegeben.

```
ASS6082    NOTE
ASS6082    ONLY '*' OR ',' IS ALLOWED AS A DELIMITER BETWEEN OPTIONS. THE OPTIONS WILL BE
           IGNORED UNTIL THEY ARE ENCOUNTERED
ASS6082    ALS TRENNUNGSZEICHEN ZWISCHEN OPTIONEN NUR '*' ODER ',' ZULAESSIG; BIS ZU
           AUFTRETEN WERDEN OPTIONEN IGNORIERT

ASS6083    NOTE
ASS6083    SYNTAX ERROR IN '*END' OPTION
ASS6083    SYNTAX-FEHLER IN '*END'-OPTION

ASS6084    NOTE
ASS6084    SYNTAX ERROR IN 'INSTR-SET' OPTION. DEFAULT VALUE IS USED
ASS6084    SYNTAX-FEHLER IN 'INSTR-SET'-OPTION. STANDARDWERT WIRD VERWENDET

ASS6085    NOTE
ASS6085    SYNTAX ERROR IN 'ADIAG' OPTION. 'NOADIAG' IS SET
ASS6085    SYNTAX-FEHLER IN 'ADIAG'-OPTION. 'NOADIAG' WIRD GESETZT

ASS6086    NOTE
ASS6086    SYNTAX ERROR IN A NO LONGER SUPPORTED OPTION
ASS6086    SYNTAX-FEHLER IN NICHT MEHR UNTERSTUETZTER OPTION

ASS6087    NOTE
ASS6087    SYNTAX ERROR IN THE 'SYSPARM' OPTION. THE NULL STRING IS ASSIGNED TO '&SYSPARM'
ASS6087    SYNTAX-FEHLER IN DER 'SYSPARM'-OPTION. '&SYSPARM' WIRD NULLSTRING ZUGEWIESEN
```

```
ASS6088    NOTE
ASS6088    SYNTAX ERROR IN THE 'ERR' OPTION. DEFAULT VALUES ARE USED
ASS6088    SYNTAX-FEHLER IN 'ERR'-OPTION. STANDARDWERTE WERDEN VERWENDET

ASS6089    NOTE
ASS6089    SYNTAX ERROR IN 'ERRPR' OPTION. DEFAULT VALUE IS USED
ASS6089    SYNTAX-FEHLER IN 'ERRPR'-OPTION. STANDARDWERT WIRD VERWENDET

ASS6090    NOTE
ASS6090    SYNTAX ERROR IN 'LINECNT' OPTION. DEFAULT VALUE IS USED
ASS6090    SYNTAX-FEHLER IN 'LINECNT'-OPTION. STANDARDWERT WIRD VERWENDET

ASS6091    NOTE
ASS6091    SYNTAX ERROR IN 'PRTOFF' OPTION. DEFAULT VALUE IS USED
ASS6091    SYNTAX-FEHLER IN 'PRTOFF'-OPTION. STANDARDWERT WIRD VERWENDET

ASS6092    NOTE
ASS6092    SYNTAX ERROR IN 'SOURCE' OPTION. 'SOURCE=*' IS SET
ASS6092    SYNTAX-FEHLER IN 'SOURCE'-OPTION. 'SOURCE=*' WIRD GESETZT

ASS6093    NOTE
ASS6093    SYNTAX ERROR IN 'MODULE' OPTION. 'MODULE=*' IS SET
ASS6093    SYNTAX-FEHLER IN 'MODULE'-OPTION. 'MODULE=*' WIRD GESETZT

ASS6094    NOTE
ASS6094    ILLEGAL USE OF THE 'DUET' OPTION TOGETHER WITH 'INSTR=SET2' OR 'INSTR=SET3';
           THE 'DUET' OPTION WILL BE IGNORED
ASS6094    'DUET'-OPTION KOMBINIERT MIT 'INSTR=SET2'- BZW. 'INSTR=SET3'-OPTION
           UNZULAESSIG; 'DUET'-OPTION WIRD IGNORIERT

ASS6095    NOTE
ASS6095    THE 'ISD' AND 'ADIAG' OPTION IS NOT SUPPORTED IN ASSEMBH-BC.
ASS6095    'ISD'- UND 'ADIAG'-OPTION IM ASSEMBH-BC NICHT UNTERSTUETZT

ASS6096    NOTE
ASS6096    INVALID STRING LENGTH IN 'SYSPARM' OPTION; THE NULL STRING IS ASSIGNED TO
           '&SYSPARAM'
ASS6096    STRING-LAENGE BEI 'SYSPARM'-OPTION UNGUELTIG; '&SYSPARM' WIRD NULLSTRING
           ZUGEWIESEN

ASS6097    NOTE
ASS6097    INVALID VALUE IN 'ERRPR' OPTION; DEFAULT VALUE IS USED
ASS6097    WERT IN 'ERRPR'-OPTION UNGUELTIG; STANDARDWERT WIRD VERWENDET
```

```
ASS6098    NOTE
ASS6098    INVALID VALUE IN 'LINECNT' OPTION; DEFAULT VALUE IS USED
ASS6098    WERT IN 'LINECNT'-OPTION UNGUELTIG; STANDARDWERT WIRD VERWENDET

ASS6099    NOTE
ASS6099    INVALID VALUE IN THE OPTION 'ERR=N'; DEFAULT VALUE IS USED
ASS6099    WERT IN 'ERR=N'-OPTION UNGUELTIG; STANDARDWERT WIRD VERWENDET

ASS6100    NOTE
ASS6100    OPTION (&00) IS NO LONGER SUPPORTED IN ASSEMBH
ASS6100    OPTION (&00) IN ASSEMBH NICHT MEHR UNTERSTUETZT

ASS6101    NOTE
ASS6101    INVALID VALUE IN THE OPTION 'PRTOFF=N'; DEFAULT VALUE IS USED
ASS6101    WERT IN 'PRTOFF=N'-OPTION UNGUELTIG; STANDARDWERT WIRD VERWENDET

ASS6102    NOTE
ASS6102    IN THE OPTION 'PRTOFF=X1;X2...'; A INVALID CHARACTER IS GIVEN. IT WILL BE
           IGNORED
ASS6102    ZEICHEN IN 'PRTOFF=X1;X2...'-OPTION UNGUELTIG; ZEICHEN WIRD IGNORIERT

ASS6103    NOTE
ASS6103    ONLY ONE CHARACTER IS ALLOWED PER ENTRY IN THE 'PRTOFF=X1;X2...' OPTION; STRING
           ENTRIES ARE IGNORED
ASS6103    IN 'PRTOFF=X1;X2...'-OPTION JEWEILS NUR EIN ZEICHEN ZULAESSIG; STRING-ANGABE
           WIRD IGNORIERT

ASS6104    NOTE
ASS6104    LENGTH OF FILE NAME IN 'SOURCE' OPTION IS INVALID. 'SOURCE=*' IS SET
ASS6104    LAENGE DES DATEINAMENS IN 'SOURCE'-OPTION UNZULAESSIG; 'SOURCE=*' WIRD GESETZT

ASS6105    NOTE
ASS6105    LENGTH OF MEMBER NAME IN 'SOURCE' OPTION IS INVALID. 'SOURCE=*' IS SET
ASS6105    LAENGE DES ELEMENTNAMENS IN 'SOURCE'-OPTION UNZULAESSIG. 'SOURCE=*' WIRD
           GESETZT

ASS6106    NOTE
ASS6106    LENGTH OF VERSION IN 'SOURCE' OPTION IS INVALID. 'SOURCE=*' IS SET
ASS6106    LAENGE DER VERSIONSANGABE IN 'SOURCE'-OPTION UNZULAESSIG; 'SOURCE=*' WIRD
           GESETZT

ASS6107    NOTE
ASS6107    LENGTH OF LIBRARY NAME IN 'MODULE' OPTION IS INVALID. 'MODULE=*' IS SET
ASS6107    LAENGE DES BIBLIOTHEKNAMENS IN 'MODULE'-OPTION UNZULAESSIG; 'MODULE=*' WIRD
           GESETZT
```

```
ASS6108    NOTE
ASS6108    LENGTH OF MEMBER NAME IN 'MODULE' OPTION IS INVALID. 'MODULE=*' IS SET
ASS6108    LAENGE DES ELEMENTNAMENS IN 'MODULE'-OPTION UNZULAESSIG; 'MODULE=*' WIRD
           GESETZT

ASS6109    NOTE
ASS6109    LENGTH OF VERSION IN 'MODULE' OPTION IS INVALID. 'MODULE=*' IS SET
ASS6109    LAENGE DER VERSIONSANGABE IN 'MODULE'-OPTION UNZULAESSIG; 'MODULE=*' WIRD
           GESETZT

ASS6110    FATAL ERROR
ASS6110    SOURCE CANNOT BE OPENED; (&00); 'HALT' IS SET
ASS6110    SOURCE KANN NICHT GEOEFFNET WERDEN; (&00); 'HALT' WIRD GESETZT
```

Bedeutung
(&00): Fehlerursache (z.B. FILE NOT SHAREABLE).

```
ASS6111    NOTE
ASS6111    ONLY 'COMOPT', 'END', OR 'HALT' IS ALLOWED AFTER '*'; ALL OTHER ENTRIES ARE
           IGNORED
ASS6111    NACH '*' NUR 'COMOPT', 'END' ODER 'HALT' ZULAESSIG; ALLES ANDERE WIRD IGNORIERT

ASS6112    NOTE
ASS6112    UNEXPECTED EOF; 'END HALT' IS SET
ASS6112    UNERWARTETES EOF; 'END HALT' WURDE GESETZT

ASS6113    NOTE
ASS6113    UNEXPECTED EOF; 'HALT' IS SET
ASS6113    UNERWARTETES EOF; 'HALT' WURDE GESETZT

ASS6114    NOTE
ASS6114    LAST QUOTE IS MISSING IN THE 'SYSPARM' OPTION. THE NULL STRING IS ASSIGNED TO
           '&SYSPARM'
ASS6114    IN 'SYSPARM'-OPTION FEHLT ABSCHLIESSENDES APOSTROPH; '&SYSPARM' WIRD NULLSTRING
           ZUGEWIESEN

ASS6115    NOTE
ASS6115    THE 'DSDD' OR 'MONSYS RECORDS=YES' OPTION IS ONLY ALLOWED WHEN MODULE IS OUTPUT
           TO A PLAM LIBRARY; OPTION WILL BE IGNORED
ASS6115    'DSDD'- BZW. 'MONSYS-RECORDS=YES'-OPTION NUR BEI MODULAUSGABE IN PLAM-
           BIBLIOTHEK ZULAESSIG; OPTION WIRD IGNORIERT
```

```
ASS6117  NOTE
ASS6117  SYNTAX ERROR IN 'SEQ' OPTION; 'SEQ' OPTION IS IGNORED
ASS6117  SYNTAXFEHLER IN DER 'SEQ' OPTION; 'SEQ' OPTION WIRD IGNORIERT
```

Bedeutung
'SEQ' Option hat nicht die Form 'SEQ=(<nummer>[,<laenge>[,<id>]])' mit 4<= <laenge> <=8 und <id> <=4 Zeichen und Laenge von <id>+<laenge> <=8.

```
ASS6121  Z19 - WARNING
ASS6121  INTERNAL ERROR IN ASSEMBH: WARNING BY 'CIF' ACCESS ROUTINE; RETURN CODE: (&00)
ASS6121  INTERNER FEHLER IM ASSEMBH: WARNUNG DURCH 'CIF'-ZUGRIFFSROUTINE, RETURN-CODE:
         (&00)
```

Bedeutung
Diese Meldung ist fuer die ASSEMBH-Entwicklung bestimmt.

```
ASS6122  FAILURE
ASS6122  INTERNAL ERROR IN ASSEMBH: ERROR IN 'CIF' ACCESS ROUTINE
ASS6122  INTERNER FEHLER IM ASSEMBH: FEHLER IN 'CIF'-ZUGRIFFSROUTINE
```

Maßnahme
Systemverwalter verstaendigen.

```
ASS6123  SERIOUS ERROR
ASS6123  INTERNAL ERROR IN ASSEMBH: ERROR BY 'CIF' ACCESS ROUTINE; RETURN CODE: (&00)
ASS6123  INTERNER FEHLER IM ASSEMBH: FEHLER DURCH 'CIF'-ZUGRIFFSROUTINE, RETURN-CODE:
         (&00)
```

Bedeutung
Diese Meldung ist fuer die ASSEMBH-Entwicklung bestimmt.

Maßnahme
Systemverwalter verstaendigen.

```
ASS6124  SERIOUS ERROR
ASS6124  CC-DMS ERROR (&00) IN 'CIF' ACCESS ROUTINE
ASS6124  CC-DMS-FEHLER (&00) IN 'CIF'-ZUGRIFFSROUTINE

ASS6125  FAILURE
ASS6125  INTERNAL ERROR IN ASSEMBH: 'PIOM' TERMINATION CAUSED BY WRONG 'CIF'
ASS6125  INTERNER FEHLER IM ASSEMBH: 'PIOM'-ABBRUCH VERURSACHT DURCH FEHLERHAFTES 'CIF'
```

Maßnahme
Systemverwalter verstaendigen.

```
ASS6126   FAILURE
ASS6126   INTERNAL ERROR IN ASSEMBH: 'PIOM' TERMINATION CAUSED BY INCOMPATIBLE VERSIONS
          OF 'CIF' ACCESS ROUTINES
ASS6126   INTERNER FEHLER IM ASSEMBH: 'PIOM'-ABBRUCH VERURSACHT DURCH INKOMPATIBLE
          VERSIONEN DER 'CIF'-ZUGRIFFSROUTINEN
```

Maßnahme
Systemverwalter verstaendigen.

```
ASS6127   Z18 - FAILURE
ASS6127   INSUFFICIENT MEMORY FOR VIRTUAL CIF
ASS6127   NICHT AUSREICHEND SPEICHER FUER VIRTUELLEN 'CIF' VORHANDEN
```

Maßnahme
Speicherberechtigung von der Systemverwaltung hochsetzen lassen.

```
ASS6128   SERIOUS ERROR
ASS6128   "COMPILER INFORMATION FILE" IS NO PLAM LIBRARY
ASS6128   "COMPILER INFORMATION FILE" IST KEINE PLAM=BIBLIOTHEK
```

Bedeutung
Es wurde kein Listing erstellt.

Maßnahme
PLAM-Bibliothekselement fuer "Compiler Information File" angeben.

```
ASS6129   SERIOUS ERROR
ASS6129   THE PLAM LIBRARY MEMBER FOR "COMPILER INFORMATION FILE" IS LOCKED
ASS6129   PLAM-BIBLIOTHEKSELEMENT FUER "COMPILER INFORMATION FILE" IST GESPERRT
```

Bedeutung
Es wurde kein Listing erstellt.

Maßnahme
PLAM-Bibliothekselement fuer "Compiler Information File" freigeben.

```
ASS6132    Z29 - FAILURE
ASS6132    COMPILATION CANCELLED DUE TO TERMINATION CONDITION
ASS6132    ABBRUCHKRITERIUM ERREICHT, UEBERSETZUNG ABGEBROCHEN
```

Bedeutung
Moegliche Abbruchkriterien:
- maximale Fehleranzahl ueberschritten
- maximales Fehlergewicht erreicht
- maximale MACRO-, COPY-Verschachtelungstiefe ueberschritten
- ACTR-Überlauf

Maßnahme
Source korrigieren oder Grenzwerte mit der COMPILER-TERMINATION-Option bzw. ACTR-Anweisung hochsetzen.

```
ASS6140    FAILURE
ASS6140    INTERNAL ERROR IN ASSEMBH: ERROR IN LISTING GENERATION. TERMINATION OF THE
           ASSEMBH RUN WITH LG RETURN CODE: (&00)
ASS6140    INTERNER FEHLER IM ASSEMBH: FEHLER BEI DER LISTING-ERSTELLUNG. ABBRUCH DES
           ASSEMBH-LAUFS MIT LG-RETURNCODE: (&00)
```

Bedeutung
Diese Meldung ist fuer die ASSEMBH-Entwicklung bestimmt.
(&00): Listing Generator Returncode.

Maßnahme
Systemverwalter verstaendigen.

```
ASS6141    SIGNIFICANT ERROR
ASS6141    INTERNAL ERROR IN ASSEMBH: ERROR IN LISTING GENERATION. INCOMPLETE OR WRONG
           LISTING WAS GENERATED
ASS6141    INTERNER FEHLER IM ASSEMBH: FEHLER BEI DER LISTING-ERSTELLUNG. UNVOLLSTAENDIGES
           ODER FEHLERHAFTES LISTING WURDE ERZEUGT

ASS6142    NOTE
ASS6142    'AID' IS NOT SUPPORTED IN ASSEMBH-BC
ASS6142    'AID' IM ASSEMBH-BC NICHT UNTERSTUETZT

ASS6143    NO ERRORS
ASS6143    ASSDIAG COMMAND 'RERUN' AFTER ABORT OF ASSEMBH NOT ALLOWED; 'END' COMMAND
           ASSUMED
ASS6143    ASSDIAG-KOMMANDO 'RERUN' NACH ASSEMBH-ABBRUCH UNZULAESSIG; 'END'-KOMMANDO WIRD
           AUSGEFUEHRT

ASS6144    NO ERRORS
ASS6144    ERRORFILE GENERATION TIME: (&00) MSEC
ASS6144    ZEIT FUER ERRORFILE ERSTELLUNG: (&00) MSEC
```

```
ASS6145    FAILURE
ASS6145    ILLEGAL VERSION OF SYNTAXFILE FOR ASSEMBH
ASS6145    VERKEHRTE VERSION DES SYNTAX-FILES FUER DEN ASSEMBH
```

Maßnahme
Systemverwalter verstaendigen, damit er richtigen Syntax-File installiert

```
ASS6146    NOTE
ASS6146    UNEXPECTED EOF; '//END' IS SET
ASS6146    UNERWARTETES EOF; '//END' WURDE GESETZT
```

11.1.1 Meldungen des Assembler-Laufzeitsystems für die strukturierte Programmierung

```
ASS7001  INITIALIZATION OF THE ASSEMBLER RUNTIME SYSTEM NOT POSSIBLE
ASS7001  INITIALISIERUNG DES ASSEMBLER-LAUFZEITSYSTEMS NICHT MOEGLICH
```

Bedeutung
Wegen Speichermangel kann waehrend der Initialisierung des Laufzeitsystems der INITIAL-STACK fuer die Hauptprozedur nicht angelegt werden.

Maßnahme
Systemverwalter verstaendigen (Benutzeradressraum vergroessern).

```
ASS7002  FATAL ERROR
ASS7002  INSUFFICIENT MEMORY FOR THE 'INITIAL-STACK'
ASS7002  SPEICHERMANGEL BEI BESCHAFFUNG DES 'INITIAL-STACKS'
```

Bedeutung
Waehrend der Initialisierung des Laufzeitsystems kann der Bereich fuer Verwaltungsdaten nicht angelegt werden.

Maßnahme
Systemverwalter verstaendigen.

```
ASS7003  FATAL ERROR
ASS7003  INSUFFICIENT MEMORY TO INITIALIZE THE 'STACK' AS SPECIFIED BY 'STACK'-PARAMETER
         OF THE @ENTR-MACRO
ASS7003  SPEICHERMANGEL BEI BESCHAFFUNG DES 'STACK' GEMAESS 'STACK'-ANGABE IM '@ENTR'-
         MAKRO
```

Bedeutung
Waehrend der Initialisierung des Laufzeitsystems kann fuer die Hauptprozedur der STACK gemaess Benutzerangabe oder Standardwert nicht angelegt werden.

Maßnahme
Moegliche Massnahmen:
- STACK-Angabe reduzieren;
- Benutzeradressraum durch Systemverwalter vergroessern lassen.

```
ASS7005   STACK-POINTER DESTROYED; STACK-REGISTER 13 CONTAINS INVALID VALUE.
ASS7005   STACK-ZEIGER ZERSTOERT; STACK-REGISTER 13 ENTHAELT FEHLERHAFTEN WERT.
```

Bedeutung
Zu Beginn der Initialisierung des Laufzeitsystems aus Fremdprozeduren (FORTRAN, COBOL, ASSEMBLER) bzw. im Prozedurprolog zeigt das STACK-Register auf keine gueltige SAVE-AREA.

Maßnahme
Moegliche Massnahmen:
- STACK-Register vor der Initialisierung gueltig besetzen;
- STACK-Register innerhalb der Prozedurverschachtelung nicht veraendern.

```
ASS7006   NO MORE MEMORY AVAILABLE FOR THE 'STACK'
ASS7006   WEITERER SPEICHERPLATZ FUER 'STACK' NICHT VERFUEGBAR
```

Bedeutung
Im Prozedurprolog kann wegen Speichermangel die SAVE-AREA bzw. der Bereich fuer LOCAL-Daten nicht angelegt werden.

Maßnahme
Moegliche Massnahmen:
- Belegten Speicherplatz freigeben;
- Benutzeradressraum durch Systemverwalter vergroessern lassen.

```
ASS7007   NO MORE MEMORY AVAILABLE FOR THE 'AUTOMATIC' AREA
ASS7007   WEITERER SPEICHERPLATZ FUER 'AUTOMATIC'-BEREICH NICHT VERFUEGBAR
```

Bedeutung
Fuer eine Datenanforderung der Klasse AUTOMATIC ist kein STACK-Speicher mehr verfuegbar.

Maßnahme
Moegliche Massnahmen:
- Datenanforderung(en) reduzieren;
- Benutzeradressraum durch Systemverwalter vergroessern lassen.

```
ASS7008    NO MORE MEMORY AVAILABLE FOR THE 'CONTROLLED' AREA
ASS7008    WEITERER SPEICHERPLATZ FUER 'CONTROLLED'-BEREICH NICHT VERFUEGBAR
```

Bedeutung
Fuer eine Datenanforderung der Klasse CONTROLLED ist kein HEAP-Speicher mehr verfuegbar.

Maßnahme
Moegliche Massnahmen:
- Datenanforderung(en) reduzieren;
- nicht mehr benoetigte HEAP-Speicher freigeben;
- Benutzeradressraum durch Systemverwalter vergroessern lassen.

```
ASS7009    FATAL ERROR
ASS7009    ERROR IN RELEASING MEMORY OF THE 'CONTROLLED' AREA
ASS7009    FEHLER BEI FREIGABE EINES 'CONTROLLED'-BEREICHS
```

Bedeutung
Die angegebene Adresse zeigt auf keinen zugewiesenen Bereich im HEAP-Speicher.

Maßnahme
Gueltige Adresse angeben.

```
ASS7010    WARNING
ASS7010    INITIALIZATION ROUTINE 'IASSIN' WAS ALREADY CALLED
ASS7010    INITIALISIERUNGS-ROUTINE 'IASSIN' WURDE BEREITS AUFGERUFEN
```

Bedeutung
Mehrfacher Aufruf der Initialisierung des Laufzeitsystems aus Fremdprozeduren (FORTRAN, COBOL, ASSEMBLER).

Maßnahme
Mehrfache Initialisierung vermeiden.

```
ASS7011    INCONSISTENT AID-VERSION
ASS7011    INKONSISTENTE AID-VERSION
```

Bedeutung
Fehler während der Initialisierung des Laufzeitsystems, da im System eine inkonsistente AID-Version installiert ist.

11.1.2 Meldungen des Listengenerators

```
LGR0001    'CIF' ALREADY OPEN
LGR0001    'CIF' BEREITS GEOEFFNET

LGR0002    INFORMATION TABLES CLOSED IMPLICITLY
LGR0002    INFORMATIONS-TABELLEN IMPLIZIT GESCHLOSSEN

LGR0003    DUPLICATE KEYS EXIST
LGR0003    DUPLIKAT-SCHLUESSEL VORHANDEN

LGR0004    END OF PARTITION
LGR0004    ENDE DER PARTITION

LGR0005    INFORMATION TABLE CLOSED ABNORMALLY
LGR0005    INFORMATIONS-TABELLE ABNORMAL GESCHLOSSEN

LGR0006    PARTITION CLOSED IMPLICITLY
LGR0006    PARTITION IMPLIZIT GESCHLOSSEN

LGR0007    'CIF' ALREADY CLOSED
LGR0007    'CIF' BEREITS GESCHLOSSEN

LGR0101    'CIF' COULD NOT BE OPENED
LGR0101    'CIF' KONNTE NICHT GEOEFFNET WERDEN

LGR0102    'CIF' CURRENTLY LOCKED
LGR0102    'CIF' Z.Zt. GESPERRT

LGR0103    SPECIFIED 'CIF' DOES NOT EXIST
LGR0103    SPEZIFIZIERTER 'CIF' EXISTIERT NICHT

LGR0104    'CIF' NOT A LIBRARY
LGR0104    'CIF' KEINE BIBLIOTHEK

LGR0105    'CIF' IDENTIFIER INVALID
LGR0105    CIF-IDENTIFIKATOR UNGUELTIG

LGR0106    MAXIMUM NUMBER OF 'CIF'S EXCEEDED
LGR0106    MAXIMALE ANZAHL DER 'CIF'S UEBERSCHRITTEN

LGR0107    OPEN MODE ILLEGAL
LGR0107    OPEN-MODUS UNZULAESSIG

LGR0108    ENVIRONMENT UNSUITABLE
LGR0108    UMGEBUNG UNPASSEND

LGR0109    ACCESS TO INFORMATION TABLE NOT PERMITTED
LGR0109    ZUGRIFF AUF INFORMATIONS-TABELLE UNZULAESSIG
```

LGR0110 'IT' NAME INVALID
LGR0110 IT-NAME FEHLERHAFT

LGR0111 CLOSE MODE ILLEGAL
LGR0111 CLOSE-MODUS UNZULAESSIG

LGR0112 SORT ORDER NOT ASCENDING
LGR0112 SORTIERFOLGE NICHT AUFSTEIGEND

LGR0113 PARTITION INVALID
LGR0113 PARTITION FEHLERHAFT

LGR0114 SPECIFIED KEY NOT FOUND
LGR0114 SPEZIFIZIERTEN SCHLUESSEL NICHT GEFUNDEN

LGR0115 FUNCTION NOT SUPPORTED
LGR0115 FUNKTION NICHT UNTERSTUETZT

LGR0116 WRITE ACCESS ILLEGAL
LGR0116 SCHREIB-ZUGRIFF UNZULAESSIG

LGR0117 FIELD LENGTH INVALID
LGR0117 FELD-LAENGE FEHLERHAFT

LGR0118 SPACE OVERFLOW DURING ALLOCATION
LGR0118 SPEICHER-UEBERLAUF WAEHREND ALLOKATION

Bedeutung
Beim internen Versuch, Speicher zu beschaffen, trat ein Ueberlauf auf

Maßnahme
Bitte loeschen Sie nicht mehr benoetigten Speicher oder erhoehen Sie Ihre Speicherberechtigung und starten Sie das Programm neu

LGR0119 CC-DMS ERROR (&00) WHEN ACCESSING CIF
LGR0119 CC-DMS-FEHLER (&00) BEIM ZUGRIFF AUF CIF

Bedeutung
Naehere Information ueber den DVS-Fehlerschluessel kann ueber /HELP-MSG im Systemmodus erfragt bzw. dem BS2000-Handbuch 'Systemmeldungen' entnommen werden.

LGR0120 INFORMATION TABLE DOES NOT EXIST
LGR0120 INFORMATIONS-TABELLE EXISTIERT NICHT

LGR0121 INFORMATION TABLE ALREADY EXISTS
LGR0121 INFORMATIONS-TABELLE EXISTIERT BEREITS

LGR0122 READ ACCESS ILLEGAL
LGR0122 LESE-ZUGRIFF UNZULAESSIG

```
LGR0123    MANDATORY FIELD MISSING
LGR0123    MANDATORY FELD FEHLT

LGR0124    'CIF' TYPE INVALID
LGR0124    CIF-TYP FEHLERHAFT

LGR0125    ENVIRONMENT ILLEGAL
LGR0125    UMGEBUNG UNGUELTIG

LGR0126    INFORMATION TABLE NOT OPENED
LGR0126    INFORMATIONS-TABELLE NICHT GEOEFFNET

LGR0201    INCOMPATIBLE VERSION IDENTIFIERS
LGR0201    VERSIONS-KENNZEICHEN INKOMPATIBEL

LGR0202    'CIF' DESTROYED
LGR0202    'CIF' ZERSTOERT

LGR0203    INTERNAL ERROR: CHECK RETURN CODE
LGR0203    INTERNER FEHLER: RETURN-CODE PRUEFEN
```

Bedeutung
Fehlertext fuer ASSLG-Enwicklung: ret_code (stat):INTERNAL_ERROR

Maßnahme
Systemverwalter verstaendigen

```
LGR0299    INTERNAL ERROR: RETURN CODE UNKNOWN
LGR0299    INTERNER FEHLER: RETURN CODE NICHT INTERPRETIERBAR
```

Bedeutung
Fehlertext fuer ASSLG-Entwicklung: ret_code (stat): DEFAULT

Maßnahme
Systemverwalter verstaendigen

```
LGR0301    FILE IS LIBRARY
LGR0301    DATEI IST BIBLIOTHEK
```

Bedeutung
Der Ablageort des Assembler-Protokolls ist falsch angegeben

Maßnahme
Korrigieren der LISTING-Option und erneut starten

```
LGR0302    FILE IS PLAM LIBRARY
LGR0302    DATEI IST PLAM-BIBLIOTHEK
```

Bedeutung
Der Ablageort des Assembler-Protokolls ist falsch angegeben

Maßnahme
Korrigieren der LISTING-Option und erneut starten

LGR0303 UNEXPECTED 'EOF' DETECTED
LGR0303 UNERWARTETES 'EOF' AUFGETRETEN

LGR0304 FILE NOT A LIBRARY
LGR0304 DATEI KEINE BIBLIOTHEK

Bedeutung
Der Ablageort des Assembler-Protokolls ist falsch angegeben

Maßnahme
Korrigieren der LISTING-Option und erneut starten

LGR0305 FILE AN OSM LIBRARY
LGR0305 DATEI EINE OSM-BIBLIOTHEK

Bedeutung
Der Ablageort des Assembler-Protokolls ist falsch angegeben

Maßnahme
Korrigieren der LISTING-Option und erneut starten

LGR0306 FCB TYPE INVALID
LGR0306 FCB-TYP UNGUELTIG

LGR0307 NO FCB TYPE SPECIFIED
LGR0307 FCB-TYP-ANGABE FEHLT

LGR0308 WRITE NOT ALLOWED IN OSM LIBRARIES
LGR0308 SCHREIBEN IN OSM-BIBLIOTHEKEN NICHT ERLAUBT

LGR0309 FILE IS AN UNKNOWN LIBRARY
LGR0309 DATEI UNBEKANNTE BIBLIOTHEK

LGR0310 FILE EMPTY
LGR0310 DATEI LEER

LGR0311 FILE NOT CATALOGED
LGR0311 DATEI NICHT KATALOGISIERT

LGR0312 NO LINK OR FILE NAME FOUND
LGR0312 LINK- ODER DATEI-NAME NICHT GEFUNDEN

LGR0313 LIBRARY MEMBER NOT FOUND
LGR0313 BIBLIOTHEKS-ELEMENT NICHT GEFUNDEN

LGR0314 FILE LOCKED
LGR0314 DATEI GESPERRT

```
LGR0315    FILE NOT SHAREABLE
LGR0315    DATEI HAT NICHT EIGENSCHAFT 'SHARE'

LGR0316    PASSWORD MISSING
LGR0316    PASSWORT FEHLT

LGR0317    TYPE OF LIBRARY MEMBER INVALID
LGR0317    TYP DES BIBLIOTHEKS-ELEMENTES UNGUELTIG

LGR0318    NAME OF LIBRARY MEMBER INVALID
LGR0318    NAME DES BIBLIOTHEKS-ELEMENTES UNGUELTIG

LGR0319    VERSION OF LIBRARY MEMBER INVALID
LGR0319    VERSION DES BIBLIOTHEKS-ELEMENTES UNGUELTIG

LGR0320    MEMORY SPACE SATURATION
LGR0320    KEIN SPEICHERPLATZ MEHR VORHANDEN

LGR0321    LIBRARY MEMBER LOCKED
LGR0321    BIBLIOTHEKS-ELEMENT GESPERRT

LGR0322    VARIANT OF LIBRARY MEMBER NOT FOUND
LGR0322    VARIANTE DES BIBLIOTHEKS-ELEMENTES NICHT GEFUNDEN

LGR0323    PLAM NOT LOADED IN SYSTEM
LGR0323    PLAM IM SYSTEM NICHT VERFUEGBAR

LGR0324    FILE NAME INVALID
LGR0324    DATEI-NAME FEHLERHAFT

LGR0325    INSUFFICIENT MEMORY
LGR0325    SPEICHERPLATZ UNZUREICHEND

LGR0326    TOO MANY WILDCARDS
LGR0326    ZU VIELE WILDCARDS

LGR0327    DATE INVALID
LGR0327    DATUM FEHLERHAFT

LGR0328    FILE IS AN OML LIBRARY
LGR0328    DATEI IST EINE OML-BIBLIOTHEK

LGR0329    FILE IS A COBLUR LIBRARY
LGR0329    DATEI IST EINE COBLUR-BIBLIOTHEK

LGR0330    WRONG RETRIEVAL ADDRESS
LGR0330    FALSCHE WIEDERGEWINNUNGS-ADRESSE

LGR0331    OPTION *INCREMENT FOR READING LIBRARY ACCESS NOT ALLOWED
LGR0331    OPTION *INCREMENT BEI LESENDEM BILIOTHEKSZUGRIFF UNZULAESSIG
```

```
LGR0332    OPTION *INCREMENT POSSIBLE ONLY WITH LMS/PLAM V2.0
LGR0332    OPTION *INCREMENT ERST AB LMS/PLAM V2.0 MOEGLICH

LGR0333    OPTION *HIGHEST POSSIBLE ONLY WITH LMS/PLAM V2.0
LGR0333    OPTION *HIGHEST ERST AB LMS/PLAM V2.0 MOEGLICH

LGR0398    DMS ERROR (&00)
LGR0398    DMS-FEHLER (&00)
```

Bedeutung
Naehere Information ueber den DVS-Fehlerschluessel kann ueber /HELP-MSG im Systemmodus erfragt bzw. dem BS2000-Handbuch 'Systemmeldungen' entnommen werden.

```
LGR0399    INTERNAL ERROR: CC-DMS INTERFACE ERROR
LGR0399    INTERNER FEHLER: CC-DMS-SCHNITTSTELLEN-FEHLER

LGR1000    TIME FOR LIST GENERATION: (&00) SECONDS
LGR1000    DAUER DER LISTEN-ERSTELLUNG: (&00) SEKUNDEN

LGR1001    INTERNAL ERROR IN 'ASSLG' WHEN READING STATEMENT: UNRECOVERABLE SYSTEM ERROR
LGR1001    INTERNER FEHLER IM 'ASSLG' BEIM LESEN EINES STATEMENTS: NICHT KORRIGIERBARER
           SYSTEM-FEHLER
```

Bedeutung
Dieser Fehlertext ist fuer die ASSLG-Entwicklung bestimmt.

Maßnahme
Systemverwalter verstaendigen.

```
LGR1002    INTERNAL ERROR IN 'ASSLG' WHEN READING STATEMENT: OPERAND ERROR IN MACRO
           'RDSTMT'
LGR1002    INTERNER FEHLER IM 'ASSLG' BEIM LESEN EINES STATEMENTS: OPERANDEN-FEHLER BEIM
           LESEN VON MAKRO 'RDSTMT'
```

Bedeutung
Dieser Fehlertext ist fuer die ASSLG-Entwicklung bestimmt.

Maßnahme
Systemverwalter verstaendigen.

```
LGR1003    INTERNAL ERROR IN 'ASSLG' WHEN READING STATEMENT: TRANSFER AREA TOO SMALL
LGR1003    INTERNER FEHLER IM 'ASSLG' BEIM LESEN EINES STATEMENTS: TRANSFER-AREA ZU KLEIN
```

Bedeutung
Dieser Fehlertext ist fuer die ASSLG-Entwicklung bestimmt.

Maßnahme
Systemverwalter verstaendigen.

LGR1004 '//END' ASSUMED DUE TO 'EOF'
LGR1004 WEGEN 'EOF' WURDE '//END' ANGENOMMEN

LGR1005 'SDF' NOT LOADED
LGR1005 'SDF' NICHT GELADEN

Maßnahme
Systemverwalter verstaendigen.

LGR1006 SYNTAX FILE DOES NOT CONTAIN '//GENERATE STATEMENT'
LGR1006 SYNTAX-DATEI ENTHAELT KEIN '//GENERATE-STATEMENT'

Maßnahme
Systemverwalter verstaendigen.

LGR1007 INTERNAL ERROR IN 'ASSLG' WHEN READING STATEMENT: 'SDF-RTC=(&00)'
LGR1007 INTERNER FEHLER IM 'ASSLG' BEIN LESEN EINES STATEMENTS: 'SDF-RTC=(&00)'

Bedeutung
Dieser Fehlertext ist fuer die ASSLG-Entwicklung bestimmt.

Maßnahme
Systemverwalter verstaendigen.

LGR2000 MANDATORY FIELD '(&00)' NOT IN CURRENT ASPECT
LGR2000 MANDATORY FELD (&00) NICHT IM LAUFENDEN ASPECT ENTHALTEN

Bedeutung
Fehlertext fuer ASSLG-Entwicklung: Fehler im SCRIPT

Maßnahme
Systemverwalter verstaendigen

LGR2001 INVALID OPTION NAME ENCOUNTERED IN LINE (&00)
LGR2001 OPTIONS-NAME IN ZEILE (&00) FEHLERHAFT

LGR2002 INVALID DIRECTIVE NAME ENCOUNTERED IN LINE (&00)
LGR2002 DIREKTIVE-NAME IN ZEILE (&00) FEHLERHAFT

LGR2003 DIGIT EXPECTED IN LINE (&00)
LGR2003 ZAHL IN ZEILE (&00) ERWARTET

Bedeutung
Fehlertext fuer ASSLG-Entwicklung: Fehler im SCRIPT

Maßnahme
Systemverwalter verstaendigen

LGR2004 STRING TOO LONG IN LINE (&00)
LGR2004 STRING IN ZEILE (&00) ZU LANG

LGR2005 NAME TOO LONG IN LINE (&00)
LGR2005 NAME IN ZEILE (&00) ZU LANG

Bedeutung
Fehlertext fuer ASSLG-Entwicklung: Fehler im SCRIPT

Maßnahme
Systemverwalter verstaendigen

LGR2006 INVALID STRING IN LINE (&00)
LGR2006 STRING IN ZEILE (&00) FEHLERHAFT

Bedeutung
Fehlertext fuer ASSLG-Entwicklung: Fehler im SCRIPT

Maßnahme
Systemverwalter verstaendigen

LGR2007 KEYWORD IN LINE (&00) INVALID
LGR2007 SCHLUESSELWORT IN ZEILE (&00) FEHLERHAFT

Bedeutung
Fehlertext fuer ASSLG-Entwicklung: Fehler im SCRIPT

Maßnahme
Systemverwalter verstaendigen

LGR2008 TEMPLATE ID IN LINE (&00) INVALID
LGR2008 TEMPLATE-ID IN ZEILE (&00) FEHLERHAFT

Bedeutung
Fehlertext fuer ASSLG-Entwicklung: Fehler im SCRIPT

Maßnahme
Systemverwalter verstaendigen

LGR2009 TEMPLATE ID IN LINE (&00) ONLY SIGNIFICANT UP TO 4 CHARACTERS
LGR2009 TEMPLATE-ID IN ZEILE (&00) NUR BIS ZU 4 ZEICHEN SIGNIFIKANT

Bedeutung
Fehlertext fuer ASSLG-Entwicklung: Fehler im SCRIPT

Maßnahme
Systemverwalter verstaendigen

LGR2010 TEMPLATE IN LINE (&00) NOT DEFINED
LGR2010 TEMPLATE IN ZEILE (&00) NICHT DEFINIERT

Bedeutung
Fehlertext fuer ASSLG-Entwicklung: Fehler im SCRIPT

Maßnahme
Systemverwalter verstaendigen

```
LGR2011    SECTION SPECIFICATION EXPECTED IN LINE (&00)
LGR2011    SECTION-SPEZIFIKATION IN ZEILE (&00) ERWARTET
```

Bedeutung
Fehlertext fuer ASSLG-Entwicklung: Fehler in SCRIPT

Maßnahme
Systemverwalter verstaendigen

```
LGR2012    'DEF' OR 'ENDDEFS' EXPECTED IN LINE (&00)
LGR2012    'DEF' ODER 'ENDDEFS' IN ZEILE (&00) ERWARTET
```

Bedeutung
Fehlertext fuer ASSLG-Entwicklung: Fehler im SCRIPT

Maßnahme
Systemverwalter verstaendigen

```
LGR2013    'ASP' OR 'ENDASPS' EXPECTED IN LINE (&00)
LGR2013    'ASP' ODER 'ENDASPS' IN ZEILE (&00) ERWARTET
```

Bedeutung
Fehlertext fuer ASSLG-Entwicklung: Fehler im SCRIPT

Maßnahme
Systemverwalter verstaendigen

```
LGR2014    THIS ASP HAS MORE FIELDS THAN ORIGINALLY DEFINED (&00)
LGR2014    DIESER ASPECT HAT MEHR FELDER ALS URSPRUENGLICH DEFINIERT (&00)
```

Bedeutung
Fehlertext fuer ASSLG-Entwicklung: Fehler im SCRIPT

Maßnahme
Systemverwalter verstaendigen

```
LGR2015    NO MERGE FIELD SPECIFIED. ONLY LAST 'IT' OPENED (&00)
LGR2015    KEIN MERGE-FELD ANGEGEBEN. ES WURDE NUR DER LETZTE 'IT' GEOEFFNET (&00)
```

Bedeutung
Fehlertext fuer ASSLG-Entwicklung: Fehler im SCRIPT

Maßnahme
Systemverwalter verstaendigen

```
LGR2016    INTERNAL ERROR. REASON IN LINE (&00)
LGR2016    INTERNER FEHLER. GRUND IN ZEILE (&00)
```

Bedeutung
Fehlertext fuer ASSLG-Entwicklung: Fehler im SCRIPT

Maßnahme
Systemverwalter verstaendigen

```
LGR2017    DEFINITION OF TEMPLATE WITH INTERNAL CODE '(&00)' INVALID
LGR2017    TEMPLATE MIT INTERNEM CODE '(&00)' FEHLERHAFT DEFINIERT
```

Bedeutung
Fehlertext fuer ASSLG-Entwicklung: Fehler im SCRIPT

Maßnahme
Systemverwalter verstaendigen

```
LGR3000    IT NAME '(&00)' INVALID
LGR3000    IT-NAME '(&00)' FEHLERHAFT
```

Bedeutung
Fehlertext fuer ASSLG-Entwicklung: Fehler im SCRIPT

Maßnahme
Systemverwalter verstaendigen

```
LGR3001    IT NUMBER '(&00)' INVALID
LGR3001    IT-ANZAHL '(&00)' FEHLERHAFT
```

Bedeutung
Fehlertext fuer ASSLG-Entwicklung: Fehler im SCRIPT

Maßnahme
Systemverwalter verstaendigen

```
LGR3002    FIELD NAME '(&00)' INVALID
LGR3002    FELD-NAME '(&00)' UNGUELTIG
```

Bedeutung
Fehlertext fuer ASSLG-Entwicklung: Fehler im SCRIPT

Maßnahme
Systemverwalter verstaendigen

```
LGR3003    FIELD NUMBER '(&00)' INVALID
LGR3003    FELD-ANZAHL '(&00)' FEHLERHAFT
```

Bedeutung
Fehlertext fuer ASSLG-Entwicklung: Fehler im SCRIPT

Maßnahme
Systemverwalter verstaendigen

```
LGR3004    OPENING OF FILE '(&00)' NOT POSSIBLE
LGR3004    OEFFNEN DER DATEI '(&00)' NICHT MOEGLICH

LGR3005    LG INTERFACE VERSION NUMBER (&00) INVALID
LGR3005    LG-SCHNITTSTELLEN-VERSION (&00) FEHLERHAFT

LGR3006    EXCEPTION HANDLER '(&00)' MISSING
LGR3006    'EXCEPTION HANDLER (&00)' FEHLT

LGR3007    REQUIRED FIELD '(&00)' MISSING IN ASPECT
LGR3007    BENOETIGTES FELD '(&00)' FEHLT IN ASPECT
```

Bedeutung
Fehlertext fuer ASSLG-Entwicklung: Fehler im SCRIPT

Maßnahme
Systemverwalter verstaendigen

```
LGR4000    NO MORE MEMORY SPACE AVAILABLE
LGR4000    KEIN SPEICHER MEHR VERFUEGBAR

LGR4001    LG OPTIONS INVALID
LGR4001    LG-OPTIONS FEHLERHAFT

LGR4002    PUT-GET BUFFER NOT YET ALLOCATED
LGR4002    PUT-GET-PUFFER NOCH NICHT EINGERICHTET

LGR4999    INTERNAL LG ERROR
LGR4999    INTERNER LG-FEHLER
```

Bedeutung
Dieser Fehler-Text ist fuer die ASSLG-Entwicklung bestimmt.

Maßnahme
Systemverwalter verstaendigen.

11.2 Lookahead-Mechanismus

Der Lookahead-Mechanismus ist eine Funktion, die im Zusammenhang mit der Verwendung von Makrosprachelementen im Assembler-Quellprogrammtext auftritt. Unter Lookahead versteht man das Einlesen und Scannen von Quelltext-Instruktionen in eine interne und somit referenzfähige Datei. Lookahead beginnt ab dem Statement, das mindestens eines der folgenden Kriterien erfüllt bis zum Ende der Übersetzungseinheit:

(1) ein noch nicht definiertes Folgesymbol im Operandeneintrag einer AGO- oder AIF-Anweisung,

(2) eine Bezugnahme auf Merkmale von noch nicht definierten Namen in der Bedingung der AIF-Anweisung,

(3) eine Bezugnahme auf Merkmale von noch nicht definierten Namen im Operandeneintrag der SET-Anweisung,

(4) das erstmalige Auftreten eines Folgesymbols im Namenseintrag einer Instruktion (1-4 siehe "ASSEMBH", Beschreibung [1]).

Hinweis

Bei hohen Performance-Anforderungen zur Übersetzungszeit sollte man Quellprogramme so abfassen, daß kein Lookahead notwendig wird.

11.3 Format der Assemblerbefehle

In dieser Befehlsmenge sind die Befehle der Befehlssätze BS2000-NXS (SET1), BS2000-XS (SET3) und BS2000-ESA enthalten (die Assemblerbefehle sind in der Sprachbeschreibung "Assemblerbefehle (BS2000)" [11] beschrieben).
Der Befehlssatz BS2000-NXS unterstützt die Zentraleinheiten mit der 24-Bit-Adressierung (NXS bedeutet Nicht eXtended System).
Der Befehlssatz BS2000-XS unterstützt die XS-Anlagen mit der 31-Bit-Adressierung (XS bedeutet eXtended System).
Der Befehlssatz BS2000-ESA unterstützt die ESA-Anlagen, auf denen eine Erweiterung der virtuellen Adreßräume möglich ist (ESA bedeutet Enterprise Systems Architecture).
Im Befehlssatz BS2000-XS ist der Befehlssatz BS2000-NXS enthalten und beide sind im Befehlssatz BS2000-ESA enthalten.
In der Spalte NXS / XS / ESA ist jeweils mit dem Anfangsbuchstaben N, X oder E markiert, zu welchem Befehlssatz der Befehl gehört.
In folgender Darstellung sind die mit N markierten Befehle eine Befehlsgrundmenge und die mit X oder E markierten Befehle die zusätzliche Menge dieser Befehlssätze.

Mnem. Code	Befehlsname	NXS XS ESA	Masch. Code	Länge	Operandenformat
A	Addieren	N	5A	4	R1,D2(X2,B2)
AD	Addieren normalisiert lang	N	6A	4	R1,D2(X2,B2)
ADR	Addieren normalisiert lang	N	2A	2	R1,R2
AE	Addieren normalisiert kurz	N	7A	4	R1,D2(X2,B2)
AER	Addieren normalisiert kurz	N	3A	2	R1,R2
AH	Addieren Halbwort	N	4A	4	R1,D2(X2,B2)
AL	Addieren ohne Vorzeichen	N	5E	4	R1,D2(X2,B2)
ALR	Addieren ohne Vorzeichen	N	1E	2	R1,R2
AP	Addieren dezimal	N	FA	6	D1(L1,B1),D2(L2,B2)
AR	Addieren	N	1A	2	R1,R2
AU	Addieren nicht normalisiert kurz	N	7E	4	R1,D2(X2,B2)
AUR	Addieren nicht normalisiert kurz	N	3E	2	R1,R2
AW	Addieren nicht normalisiert lang	N	6E	4	R1,D2(X2,B2)
AWR	Addieren nicht normalisiert lang	N	2E	2	R1,R2
AXR	Addieren normalisiert mit erweiterter Länge	N	36	2	R1,R2
BAL	Springen und Speichern Rücksprungadresse	N	45	4	R1,D2(X2,B2)
BALR	Springen und Speichern Rücksprungadresse	N	05	2	R1,R2
BAS	Springen und Speichern Rücksprungadresse	N	4D	4	R1,D2(X2,B2)
BASR	Springen und Speichern Rücksprungadresse	N	0D	2	R1,R2
BASSM	Branch and Save and Set Mode	X	0C	2	R1,R2
BC	Springen bedingt	N	47	4	I,D2(X2,B2)
BCR	Springen bedingt	N	07	2	I,R2
BCT	Springen nach Zählen	N	46	4	R1,D2(X2,B2)
BCTR	Springen nach Zählen	N	06	2	R1,R2
BSM	Branch and Save	X	0B	2	R1,R2
BXH	Springen wenn Index größer	N	86	4	R1,R3,D2(B2)
BXLE	Springen wenn Index kleiner gleich	N	87	4	R1,R3,D2(B2)
C	Vergleichen algebraisch	N	59	4	R1,D2(X2,B2)
* CCPU	Prüfen Zentraleinheit	N	AC	4	D1(B1),I2
CCW	Define Channel Command Word	N		8	I1,I2,I3,I4
CCW0	Define Channel Command Word (Format 0)	X		8	I1,I2,I3,I4
CCW1	Define Channel Command Word (Format 1)	X		8	I1,I2,I3,I4
CD	Vergleichen lang	N	69	4	R1,D2(X2,B2)
CDR	Vergleichen lang	N	29	2	R1,R2
CDS	Vergleichen doppelt und austauschen	N	BB	4	R1,R3,D2(B2)
CE	Vergleichen kurz	N	79	4	R1,D2(X2,B2)
CER	Vergleichen kurz	N	39	2	R1,R2
CH	Vergleichen Halbwort	N	49	4	R1,D2(C2,B2)
* CIOC	Prüfen Ein-Ausgabesteuerung	N	AD	4	D1(B1),I2
* CKC	Prüfen Kanal	N	9F	4	D1(B1)
CL	Vergleichen logisch	N	55	4	R1,D2(X2,B2)
CLC	Vergleichen logisch	N	D5	6	D1(L,B1),D2(B2)
CLCL	Vergleichen logisch lang	N	0F	2	R1,R2

Mnem. Code	Befehlsname	NXS XS ESA	Masch. Code	Länge	Operandenformat
CLI	Vergleichen logisch	N	95	4	D1(B1),I2
CLM	Vergleichen logisch mit Maske	N	BD	4	R1,M3,D2(B2)
CLR	Vergleichen logisch	N	15	2	R1,R2
CP	Vergleichen dezimal	N	F9	6	D1(L1,B1),D2(L2,B2)
CPYA	Copy Access Register	E	B24D	4	R1,R2
CR	Vergleichen algebraisch	N	19	2	R1,R2
CS	Vergleichen und austauschen	N	BA	4	R1,R3,D2(B2)
* CSCH	Clear Subchannel	X	B230	4	kein Operand
CVB	Umwandeln in Binärform	N	4F	4	R1,D2(X2,B2)
CVD	Umwandeln in Dezimalform	N	4E	4	R1,D2(X2,B2)
D	Dividieren	N	5D	4	R1,D2(X2,B2)
DD	Dividieren lang	N	6D	4	R1,D2(X2,B2)
DDR	Dividieren lang	N	2D	2	R1,R2
DE	Dividieren kurz	N	7D	4	R1,D2(X2,B2)
DER	Dividieren kurz	N	3D	2	R1,R2
* DIG	Suchen Maschinenfehler	N	83	4	D1(B1)
DP	Dividieren dezimal	N	FD	6	D1(L1,B1),D2(L2,B2)
DR	Dividieren	N	1D	2	R1,R2
DXR	Divide extended	X	B22D	4	R1,R2
EAR	Extract Access Register	E	B24F	4	R1,R2
ED	Aufbereiten	N	DE	6	D1(L,B1),D2(B2)
EDMK	Aufbereiten und Markieren	N	DF	6	D1(L,B1),D2(B2)
** EPAR	Extract Primary ASN	X	B226	4	R1
** ESAR	Extract Secondary ASN	X	B227	4	R1
EX	Ausführen	N	44	4	R1,D2(X2,B2)
* FC	Ausführen Sonderfunktionen	N	9A	4	D1(B1),I2
* FCAL	Ausführen Sonderfunktionen	N	B7	4	D1(B1),I2
HDR	Halbieren lang	N	24	2	R1,R2
* HDV	Halten Gerät	N	9E	4	D1(B1)
HER	Halbieren kurz	N	34	2	R1,R2
* HSCH	Halt Subchannel	X	B231	4	kein Operand
** IAC	Insert Address Space Control	E	B224	4	R1
IC	Laden Zeichen	N	43	4	R1,D2(X2,B2)
ICM	Einsetzen Zeichen mit Maske	N	BF	4	R1,M3,D2(B2)
* IDL	Warten	N	80	4	I2
** IPK	Insert PSW Key	X	B20B	4	kein Operand
IPM	Einsetzen Programmaske	N	B222	4	R1
* ISK	Abfragen Speicherschlüssel	N	09	2	R1,R2
** IVSK	Insert Virtual Storage Key	X	B223	4	R1,R2
L	Laden	N	58	4	R1,D2(X2,B2)
LA	Laden Adresse	N	41	4	R1,D2(X2,B2)
LAE	Load Address Extended	E	51	4	R1,D2(X2,B2)
LAM	Load Access Multiple	E	9A	4	R1,R3,D2(B2)
LCDR	Laden Komplement lang	N	23	2	R1,R2
LCER	Laden Komplement kurz	N	33	2	R1,R2
LCR	Laden Komplement	N	13	2	R1,R2
LD	Laden lang	N	68	4	R1,D2(X2,B2)
LDR	Laden lang	N	28	2	R1,R2
LE	Laden kurz	N	78	4	R1,D2(X2,B2)
LER	Laden kurz	N	38	2	R1,R2
LH	Laden Halbwort	N	48	4	R1,D2(X2,B2)
LM	Laden mehrfach	N	98	4	R1,R3,D2(B2)

Mnem. Code	Befehlsname	NXS XS ESA	Masch. Code	Länge	Operandenformat
LNDR	Laden negativ lang	N	21	2	R1,R2
LNER	Laden negativ kurz	N	31	2	R1,R2
LNR	Laden negativ	N	11	2	R1,R2
LPDR	Laden positiv lang	N	20	2	R1,R2
LPER	Laden positiv kurz	N	30	2	R1,R2
LPR	Laden positiv	N	10	2	R1,R2
LR	Laden	N	18	2	R1,R2
LRDR	Laden und Runden mit erweiterter Länge, Ergebnis lang	N	25	2	R1,R2
LRER	Laden und Runden mit erweiterter Länge, Ergebnis kurz	N	35	2	R1,R2
* LSM	Laden Schattenspeicher	N	D9	6	D1(L,B1),D2(B2)
* LSP	Laden Zwischenspeicher	N	D8	6	D1(L,B1),D2(B2)
LTDR	Laden und Testen lang	N	22	2	R1,R2
LTER	Laden und Testen kurz	N	32	2	R1,R2
LTR	Laden und Testen	N	12	2	R1,R2
M	Multiplizieren	N	5C	4	R1,D2(X2,B2)
MD	Multiplizieren lang	N	6C	4	R1,D2(X2,B2)
MDR	Multiplizieren lang	N	2C	2	R1,R2
ME	Multiplizieren kurz	N	7C	4	R1,D2(X2,B2)
MER	Multiplizieren kurz	N	3C	2	R1,R2
MH	Multiplizieren Halbwort	N	4C	4	R1,D2(X2,B2)
MP	Multiplizieren dezimal	N	FC	6	D1(L1,B1),D2(L2,B2)
MR	Multiplizieren	N	1C	2	R1,R2
* MSCH	Modify Subchannel	X	B232	4	D2(B2)
MVC	Übertragen Zeichenfolge	N	D2	6	D1(L,B1),D2(B2)
MVCL	Übertragen Zeichenfolge lang	N	0E	2	R1,R2
** MVCP	Move to Primary	X	DA	6	D1(R1,B1),D2(B2),R3
** MVCS	Move to Secondary	X	DB	6	D1(R1,B1),D2(B2),R3
MVI	Ersetzen Zeichen	N	92	4	D1(B1),I2
MVN	Übertragen numerisch	N	D1	6	D1(L,B1),D2(B2)
MVO	Übertragen mit Versetzen	N	F1	6	D1(L1,B1),D2(L2,B2)
MVZ	Übertragen Zonen	N	D3	6	D1(L,B1),D2(B2)
MXD	Multiplizieren mit erweiterter Länge, Ergebnis lang	N	67	4	R1,D2(X2,B2)
MXDR	Multiplizieren mit erweiterter Länge, Ergebnis lang	N	27	2	R1,R2
MXR	Multiplizieren mit erweiterter Länge, Ergebnis kurz	N	26	2	R1,R2
N	UND	N	54	4	R1,D2(X2,B2)
NC	UND	N	D4	6	D1(L,B1),D2(B2)
NI	UND	N	94	4	D1(B1),I2
NR	UND	N	14	2	R1,R2
O	ODER	N	56	4	R1,D2(X2,B2)
OC	ODER	N	D6	6	D1(L,B1),D2(B2)
OI	ODER	N	96	4	D1(B1),I2
OR	ODER	N	16	2	R1,R2
PACK	Packen	N	F2	6	D1(L1,B1),D2(L2,B2)
** PC	Wechseln Funktionszustand	X	B218	4	D2(B2)
** PT	Program Transfer	X	B228	4	R1,R2
* RCHP	Reset Channel Path	X	B23B	4	kein Operand
* RDD	Lesen direkt	N	85	4	D1(B1),I2

Mnem. Code	Befehlsname	NXS XS ESA	Masch. Code	Länge	Operandenformat
* RSCH	Resume Subchannel	X	B238	4	kein Operand
S	Subtrahieren	N	5B	4	R1,D2(X2,B2)
SAC	Set Address Space Control	E	B219	4	D2(B2)
* SAL	Set Address Limit	X	B237	4	kein Operand
SAR	Set Access Register	E	B24E	4	R1,R2
* SCHM	Set Channel Monitor	X	B23C	4	kein Operand
SD	Subtrahieren normalisiert lang	N	6B	4	R1,D2(X2,B2)
SDR	Subtrahieren normalisiert lang	N	2B	2	R1,R2
* SDV	Starten Gerät	N	9C	4	D1(B1)
SE	Subtrahieren normalisiert kurz	N	7B	4	R1,D2(X2,B2)
SER	Subtrahieren normalisiert kurz	N	3B	2	R1,R2
SH	Subtrahieren Halbwort	N	4B	4	R1,D2(X2,B2)
SL	Subtrahieren ohne Vorzeichen	N	5F	4	R1,D2(X2,B2)
SLA	Verschieben links	N	8B	4	R1,D2(B2)
SLDA	Verschieben links doppelt	N	8F	4	R1,D2(B2)
SLDL	Verschieben links doppelt logisch	N	8D	4	R1,D2(B2)
SLL	Verschieben links logisch	N	89	4	R1,D2(B2)
SLR	Subtrahieren ohne Überlauf	N	1F	2	R1,R2
SP	Subtrahieren dezimal	N	FB	6	D1(L1,B1),D2(L2,B2)
** SPKA	Set PSW Key from Address	X	B20A	4	D2(B2)
SPM	Setzen Programmaske	N	04	2	R1
SR	Subtrahieren	N	1B	2	R1,R2
SRA	Verschieben rechts	N	8A	4	R1,D2(B2)
SRDA	Verschieben rechts doppelt	N	8E	4	R1,D2(B2)
SRDL	Verschieben rechts doppelt logisch	N	8C	4	R1,D2(B2)
SRL	Verschieben rechts logisch	N	88	4	R1,D2(B2)
SRP	Verschieben und Runden dezimal	N	F0	6	D1(L1,B1),D2(B2),I3
* SSCH	Start Subchannel	X	B233	4	D2(B2)
* SSK	Setzen Speicherschlüssel	N	08	2	R1,R2
* SSM	Speichern aus Schattenspeicher	N	DA	6	D1(L,B1),D2(B2)
* SSP	Speichern aus Zwischenspeicher	N	D0	6	D1(L,B1),D2(B2)
ST	Speichern	N	50	4	R1,D2(X2,B2)
STAM	Store Access Multiple	E	9B	4	R1,R3,D2(B2)
STC	Speichern Zeichen	N	42	4	R1,D2(X2,B2)
STCK	Speichern Uhrzeit	N	B2	4	D1(B1)
STCM	Speichern Zeichen mit Maske	N	BE	4	R1,M3,D2(B2)
* STCPS	Store Channel Path Status	X	B23A	4	D2(B2)
* STCRW	Store Channel Report Word	X	B239	4	D2(B2)
STD	Speichern lang	N	60	4	R1,D2(X2,B2)
STE	Speichern kurz	N	70	4	R1,D2(X2,B2)
STH	Speichern Halbwort	N	40	4	R1,D2(X2,B2)
STM	Speichern mehrfach	N	90	4	R1,R3,D2(B2)
* STSCH	Store Subchannel	X	B234	4	D2(B2)
SU	Subtrahieren nicht normalisiert kurz	N	7F	4	R1,D2(X2,B2)
SUR	Subtrahieren nicht normalisiert kurz	N	3F	2	R1,R2
SVC	Aufrufen Organisationsprogramm	N	0A	2	I

Mnem. Code	Befehlsname	NXS XS ESA	Masch. Code	Länge	Operandenformat
SW	Subtrahieren nicht normalisiert lang	N	6F	4	R1,D2(X2,B2)
SWR	Subtrahieren nicht normalisiert lang	N	2F	2	R1,R2
SXR	Subtrahieren normalisiert mit erweiterter Länge	N	37	2	R1,R2
TAR	Test Access Register	E	B24C	4	R1,R2
* TDV	Prüfen Gerät	N	9D	4	D1(B1)
TM	Testen mit Maske	N	91	4	D1(B1),I2
* TPI	Test Pending Interruption	X	B236	4	D2(B2)
TR	Umsetzen Code	N	DC	6	D1(L,B1),D2(B2)
* TRACE	Trace	X	99	4	R1,R3,D2(B2)
TRT	Umsetzen und Testen	N	DD	6	D1(L,B1),D2(B2)
TS	Testen und Setzen	N	93	4	D1(B1)
* TSCH	Test Subchannel	X	B235	4	D2(B2)
UNPK	Entpacken	N	F3	6	D1(L1,B1),D2(L2,B2)
* WRD	Schreiben direkt	N	84	4	D1(B1),I2
X	Ausschließendes ODER	N	57	4	R1,D2(X2,B2)
XC	Ausschließendes ODER	N	D7	6	D1(L,B1),D2(B2)
XI	Ausschließendes ODER	N	97	4	D1(B1),I2
XR	Ausschließendes ODER	N	17	2	R1,R2
ZAP	Löschen und Addieren dezimal	N	F8	6	D1(L1,B1),D2(L2,B2)

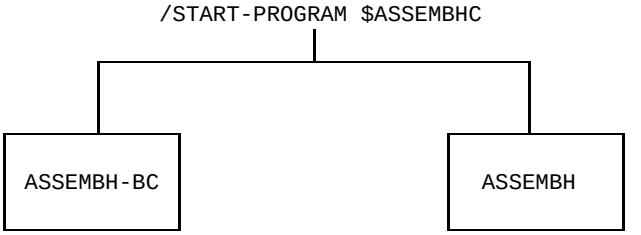
* Privilegierte Befehle

** Semiprivilegierte Befehle

11.4 *COMOPT-Anweisungen

Die bisherige Steuerung über *COMOPT ist aus Kompatibilitätsgründen mit dem Assembler ASSEMBH noch möglich (neue Leistungen werden von *COMOPT jedoch nicht unterstützt).

Der Assembler **ASSEMBH-BC** und der Assembler **ASSEMBH** werden für die *COMOPT-Steuerung wie folgt gestartet:



*COMOPT-Anweisungen werden von SYSDTA gelesen:

- am Anfang nach dem Laden des Übersetzers,
- bei jedem Restart

Eine *COMOPT-Anweisung beginnt mit *COMOPT, gefolgt von einer oder mehreren durch Kommas getrennten Optionen.
Die Fortsetzung einer Option über das Zeilenende hinaus in der Folgezeile ist zugelassen. Zeilenunterbrechung darf nur an Stellen erfolgen, wo auch ein Leerzeichen (Blank) stehen darf, d.h. Worte dürfen nicht getrennt werden.

z.B. *COMOPT SOURCE = AN falsch
TONY

*COMOPT SOURCE = ␣␣ richtig
ANTONY

Die Eingabe der Optionen ist formatfrei (z.B. SOURCE␣=␣A␣,␣). Die Eingabe der *COMOPT-Anweisungen wird durch *END abgeschlossen (*END HALT und *HALT siehe weiter unten).

Die wirksam gewordenen *COMOPT-Anweisungen werden im SDF-Format protokolliert. Fehler werden nach SYSOUT und ins Listing ausgegeben und können mit einer weiteren *COMOPT-Anweisung korrigiert werden. Beim ASSEMBH mit *COMOPT-Steuerung wird immer ein F-Assembler kompatibler Modul und ein F-Assembler kompatibles Listing erzeugt.

Da der Assembler bei der Eingabe von *COMOPTs über SYSDTA und Einlesen des Quellprogramms von Datei bzw. Bibliothek nach Beenden der Assemblierung erneut *COMOPTs für die nächste Assemblierung anfordert, stehen neben der EOF-Bedingung zwei weitere Möglichkeiten zur Beendigung des Assembler zur Verfügung:

- Durch den Operanden HALT in der *END-Anweisung (siehe 11.4.1, Beendigung der Option-Eingabe). Dadurch wird der Assembler nach der folgenden Assemblierung beendet.
- Durch die *HALT-Anweisung anstelle einer *COMOPT- oder *END-Anweisung wird der Assembler sofort beendet.

Wird statt der ersten *COMOPT-Anweisung aus Versehen etwas Falsches eingegeben, so interpretiert der Assembler diese Eingabe als erste Quellprogrammzeile. Mit der Eingabe ␣END kann eine Übersetzung dieser falschen Eingabe gestartet werden (auch hier wird standardmäßig ein Listing erzeugt). Danach können die richtigen Optionen für die nächste Übersetzung eingegeben werden.

11.4.1 Tabelle der *COMOPT-Anweisungen

*COMOPT	Bedeutung
ADIAG=n	Es wird eine Diagnosedatei erzeugt (s. COMOPT SAVLST). Nach der Übersetzung wird beim Auftreten von Fehlern mit dem Wert n (siehe Kapitel 8) bzw. MNOTES mit einem entsprechenden Severity-Code implizit das Programm $ASSDIAG gestartet. 0 ≤ n ≤ 3
ALTLIB[n]	Zuweisen einer bzw. der n-ten Makrobibliothek (2 ≤ n ≤ 5)
NOALTLIB[n]	Standardzuweisung (nur SYSLIB)
ATXREF	In der Querverweisliste werden die Referenzen mit einem Attribut versehen, das auf die Zugriffsart hinweist. W Schreibender Zugriff R Lesender Zugriff durch Befehle A Adresszugriff E EQU/ORG-Anweisungen Leerzeichen Sonstige Assembler-Anweisungen
NOATXREF	Standardzuweisung. Kein Attributen-XREF.
DUET	Zulassen der TRANSDATA 960-Befehle
NODUET	Standardzuweisung

*COMOPT	Bedeutung
ERR=n	Die Übersetzung wird bei mehr als n Fehlern mit TERM UNIT=STEP,MODE=ABNORMAL abgebrochen. 0 ≤ n ≤ 255
ERR=Sm	Die Übersetzung wird beim Auftreten von Fehlern mit dem Gewicht ≥ m, bzw. bei MNOTE's mit einem Severity-Code der Klasse ≥ m mit TERM UNIT=STEP,MODE=ABNORMAL abgebrochen. 0 ≤ m ≤ 3 Zuordnung Gewicht | Fehlerklasse 0 | Warning 1 | Significant 2 | Serious 3 | Fatal / Failure
ERRFIL	Ausgabe der Fehlerliste in eine Fehlerdatei mit dem Linknamen ERRLINK. Ist die Datei zur Übersetzungszeit nicht bekannt, so wird sie mit dem Namen ERRFIL.ASSEMBH.{<tsn> | <CSECT-name>} angelegt. Die Datei wird nur erzeugt, falls das in der COMOPT-Anweisung ERRPR eingestellte Gewicht überschritten wird. Geben Sie ERRFIL und SAVLST gleichzeitig an, wird ERRFIL ignoriert.
ERRPR=n	Für die Endemeldung des Assembler und für die COMOPT ERR werden nur die Fehlerflags ausgewertet, deren Gewicht nicht kleiner als n ist. 0 ≤ n ≤ 3 Standardvoreinstellung: n=1
FLGLST	Bei Anweisungen mit Fehlerflags werden die Fehlertypen (max. 3) an den linken Rand der Zeile gedruckt, sowie die entsprechenden Fehlertexte im Anschluß an die Zeile ausgegeben.
NOFLGLST	Bei Anweisungen mit Fehlerflags werden die Fehlertypen (max. 3) an den linken Rand der Zeile gedruckt.
HWTST	Für spezielle Tests kann das CCW-Flag-Byte unverändert (unverkürzt) übernommen werden. Dies wird jedoch als Flag angezeigt.
NOHWTST	Keine unveränderte Übernahme des CCW-Flag-Byte.

*COMOPT	Bedeutung
INSTR={SET1 / SET3}	Angabe, welcher Befehlssatz generiert werden soll. SET1 Befehlssatz BS2000-NXS (siehe 11.3) SET3 Befehlssatz BS2000-XS (siehe 11.3)
ISD NOISD	Der Assembler gibt AID-Information mit in den Objektmodul aus. ISD-Karten werden nicht mehr erzeugt. Standardzuweisung
LINECNT=n	Steuerung der Zeilen pro Druckseite, einschl. der Kopfzeile (15 ≤ n ≤ 255). Standardzuweisung: n = 60
LIST NOLIST	Standardzuweisung. Das Übersetzungsprotokoll wird nach SYSLST ausgegeben. Es werden nur die fehlerhaften Anweisungen ausgegeben.
MLPRNT NOMLPRNT	Die Makroidentifikationszeile, bestehend aus den Daten Versionsnummer, Erstellungsdatum und Linkname der Makrobibliothek, wird ins Übersetzungsprotokoll ausgegeben. Die Versionsnummer besteht aus Leerzeichen, wenn der Makro mit dem Dienstprogramm MLU in die Makrobibliothek eingebracht wurde. Standardzuweisung
MODULE= spezifikation	Angabe, wohin der Objektmodul ausgegeben werden soll. Wird diese Option nicht benutzt, wird der Objektmodul in die EAM-Datei ausgegeben. Vollständige Beschreibung dieser Option siehe Anhang 11.4.3

*COMOPT	Bedeutung
NDLIST	Es wird ein laserdruckerspezifisches Assemblerlisting erzeugt.
NONDLIST	Standardzuweisung
PRTALL	Erzeugt ein vollständiges Assemblerlisting. Die Optionen der PRINT-Anweisung NOGEN, OFF und NOCOPY werden unterdrückt.
NOPRTALL	Standardzuweisung
PRTIT	Die Wirkung der durch Makros erzeugten TITLE-Anweisungen bleibt erhalten auch wenn das Ausdrucken der TITLE-Anweisungen durch PRINT NOGEN unterdrückt wird.
NOPRTIT	Die Wirkung der durch Makros erzeugten TITLE-Anweisungen wird bei PRINT NOGEN unterdrückt.
PRTOFF= { n / X1[;X2]...[;X5] }	Abhängig von Makrostufe bzw. erstem Zeichen des Makronamens, werden die durch Makros erzeugten Anweisungen gedruckt oder nicht. n Durch Makros erzeugte Anweisungen werden ab der n-ten Makrostufe nicht gedruckt. 1 ≤ n ≤ 250 X1[;X2]...[;X5] Durch Makros erzeugte Anweisungen werden grundsätzlich nicht gedruckt, wenn das erste Zeichen des Makronamens in der Liste X1 bis X5 angegeben ist (unabhängig von einer evtl. mit PRTOFF=n eingestellten Makrostufe). In dieser Liste können bis max. 5 erste Zeichen von Makronamen stehen; alle Makros, deren Namen mit einem dieser Zeichen beginnen, sind davon betroffen.

*COMOPT	Bedeutung
SAVLST	Das Diagnoselisting für den ASSDIAG wird in eine Diagnosedatei mit dem Linknamen SAVLINK ausgegeben. Ist die Datei zur Übersetzungszeit nicht bekannt, so wird sie mit dem Namen SAVLST.ASSEMBH.{<tsn> | <CSECT-name>} angelegt. Bei unbenannter CSECT-Anweisung wird der Name mit <tsn>, bei benannter CSECT-Anweisung mit <CSECT-name> angelegt. Wird in einem Job der Assembler mehrmals mit der Option SAVLST gestartet, so muß vor jedem weiteren Start des Assemblers ein /RELEASE SAVLINK gegeben werden, um Überschreibungen zu vermeiden. Bei Anforderung einer SAVLST wird standardmäßig über LIST ein Listing erzeugt. Der Listen-Generator des ASSEMBH kann die beiden Medien SAVLST und Listing nicht gleichzeitig, sondern nur hintereinander erstellen. Aus Performancegründen wird deshalb bei SAVLST die Option NOLIST empfohlen: *COMOPT NOLIST,SAVLST Dadurch wird nur noch die SAVLST erstellt und kein Listing mehr ausgegeben.
NOSAVLST	Standardzuweisung
SEQ=(nummer [,länge[,id]])	Angaben zum Identifikationsfeld (Spalte 73 bis Spalte 80) im Übersetzungsprotokoll: nummer = Anfangsnumerierung mit einer (impliziten) Schrittweite von 100. Führende Nullen können weggelassen werden. länge = Stellenanzahl der Numerierung, rechtsbündig im Identifikationsfeld. 4 ≤ länge ≤ 8 (Voreinstellung = 8) id = alphanumerisches Kennzeichen, das ab Spalte 73 in das Identifikationsfeld übernommen wird (max. 4 Zeichen). Bei Expansion von COPY-Elementen bzw. Makros und bei generierten Literalen findet keine Numerierung statt.
SOURCE= spezifikation	Angabe, von wo das Quellprogramm gelesen werden soll. Wird diese Option nicht benutzt, so wird von SYSDTA gelesen. Vollständige Beschreibung dieser Option siehe siehe 11.4.2

*COMOPT	Bedeutung
SYSPARM= 'max. 8 Zeichen'	Der Systemparameter &SYSPARM (eine 8 Byte lange Zeichenvariable; siehe "ASSEMBH", Beschreibung [1]) wird mit den angegebenen Zeichen besetzt und kann während der Makroverarbeitung ausgewertet werden.
XREF	Die Querverweisliste wird ausgegeben.
NOXREF	Standardzuweisung. Keine Ausgabe der Querverweisliste.

Hinweise

- Folgende *COMOPT-Anweisungen werden nicht mehr unterstützt:
  COPYMAC, MCALL, MDIAG, OUTPUT, PROCOM, UPD und SOURCE = +
- Bei Verwendung der gestrichenen Anweisungen wird eine Informationsmeldung ausgegeben.

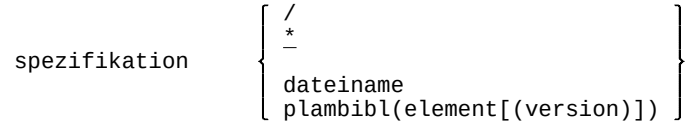
Beendigung der Option-Eingabe

	Bedeutung
*END	Eingabeende der *COMOPT-Anweisungen und Start der Assemblierung. Anforderung neuer Optionen nach der Assemblierung.
*END HALT	Wie *END, jedoch mit Beendigung des Assemblers nach der Assemblierung.
*HALT	Sofortige Beendigung des Assemblers. Kein Start der Assemblierung.

11.4.2 SOURCE-Option

Mit der SOURCE-Option kann angegeben werden, von wo das Quellprogramm eingelesen werden soll. Bei Weglassen der SOURCE-Option wird das Quellprogramm von SYSDTA gelesen.

SOURCE = spezifikation



Wenn keine Angabe für spezifikation gemacht wird, wird das Quellprogramm von SYSDTA gelesen.

/ Es erfolgt nach dem Einlesen der Optionen eine Unterbrechung. Über SYSCMD kann mit dem /SYSFILE-Kommando SYSDTA zugeordnet werden. Das Quellprogramm wird dann über SYSDTA eingelesen. Die neue Zuordnung von SYSDTA wird aber erst nach dem Abarbeiten aller Optionen wirksam.

*
Das Quellprogramm wird von SYSDTA gelesen. Voreinstellung, wenn keine Angabe zu SOURCE gemacht wird.

dateiname Name einer katalogisierten Datei, in der das Quellprogramm steht. Der Name darf max. 54 Zeichen lang sein, alphanumerisch mit Punkt und Bindestrich.

plambibl Name einer nach LMS-Konventionen eingerichteten Programmbibliothek (siehe "LMS", Benutzerhandbuch [8]), in der das Quellprogramm steht. (Element vom Typ=S).
Der Name kann, entsprechend einer BS2000-Datei, maximal 54 Zeichen lang sein.

element Name des Elements vom Typ=S, in dem das Quellprogramm steht. element darf max. 54 Zeichen lang sein.

version Versionsbezeichnung des Elements.
version darf max. 24 Zeichen lang sein.
Fehlt die Versionsangabe, wird das Element (vom Typ=S) mit der höchsten vorhandenen Version verwendet.

Hinweise

- Die Angaben in der SOURCE-Option (Bibliotheksname, Elementname und Version) werden nur auf die zulässige Länge, aber nicht auf syntaktische Richtigkeit gemäß LMS-Konventionen (siehe "LMS", Benutzerhandbuch [8]) überprüft.
- Zu Bibliotheken
  Zusätzlich zu den PLAM-Bibliotheken sind auch noch OSM-Quellprogrammbibliotheken zugelassen: bibl(name)

11.4.3 MODULE-Option

Mit dieser Option kann die Ausgabe des Objektmoduls gesteuert werden. Bei Weglassen der Option wird der Objektmodul in die EAM-Datei ausgegeben.

MODULE = spezifikation

```
                ⎧ *                                      ⎫
spezifikation   ⎨            ⎧*      ⎫                   ⎬
                ⎩ plambibl[( ⎩element⎭[(version)])]      ⎭
```

*	Der Objektmodul wird in die EAM-Datei ausgegeben.
plambibl	Name einer nach LMS-Konventionen eingerichteten Programmbibliothek (siehe "LMS", Benutzerhandbuch [8]). Der Objektmodul wird als Element vom Typ=R (Modul) abgelegt. Wenn die Programmbibliothek noch nicht existiert, wird sie vom Assembler eingerichtet.
element	Elementname des Objektmoduls. Der Elementname muß den "Richtlinien für Elementbezeichnungen in Programmbibliotheken" (siehe "LMS", Benutzerhandbuch [8]) entsprechen. Unter diesem Namen (max. 54 Zeichen) wird das Element (vom Typ=R=Modul) in der Programmbibliothek abgelegt.
*	Wird * (Stern) angegeben, erhält das Element den Namen des ersten Programmabschnitts (CSECT-Name) des Objektmoduls. Bei unbenanntem ersten Programmabschnitt kann das Element *nicht* in die Programmbibliothek ausgegeben werden. Es wird eine Meldung ausgegeben, jedoch kein Objektmodul erzeugt. Enthält das Programm keine CSECT-Anweisung (oder START-Anweisung) so wird der erste DSECT- oder COM-Name als Name des Objektelementes verwendet. Gleiches gilt, wenn nur der Bibliotheksname angegeben wurde.
version	Versionsbezeichnung des Elements. version darf max. 24 Zeichen lang sein. *Von LMS unterstützter Zeichenvorrat:* Buchstaben: A-Z Ziffern: 0-9 Sonderzeichen: '.', '-', '@' Fehlt diese Angabe, erhält das Element die höchste Version (in der Programmbibliothek durch ein Zeichen '@' dargestellt). Ab PLAM V1.4 ist '@' als Version nicht mehr möglich. Existiert bereits ein Element mit der gleichen Version, so wird diese überschrieben.

Hinweise

- Die Angaben in der MODULE-Option werden nicht auf syntaktische Richtigkeit überprüft (siehe auch Hinweise zur SOURCE-Option).
- Der Binder verarbeitet zur Zeit nur Elementnamen mit max. 8 Zeichen.

11.4.4 Gegenüberstellung der *COMOPT- und COMPILE-Anweisungen

*COMOPT	//COMPILE
ADIAG=n	CORRECTION-CYCLE=YES
ALTLIB[n]	MACRO-LIBRARY= COPY-LIBRARY=
ATXREF	[LISTING] CROSS-REFERENCE=(SYMBOL=YES)
DUET	[SOURCE-PROPERTIES] INSTRUCTION-SET=DUET
ERR=n	[COMPILER-TERMINATION] MAX-ERROR-NUMBER=n Standardzuweisung : n = 32767; oder n = 0..32767
ERR=Sm	MAX-ERROR-WEIGHT={WARNING / SIGNIFICANT / SERIOUS / FATAL}
ERRFIL	
ERRPR=n	[LISTING] MIN-MESSAGE-WEIGHT={NOTE / WARNING / SIGNIFICANT / SERIOUS / FATAL}
FLGLST	[LISTING] MESSAGE-PLACEMENT=INSERTED
HWTST	[MAINTENANCE-OPTIONS] CHANNEL-INSTRUCTIONS=YES
INSTR={SET1 / SET3}	[SOURCE-PROPERTIES] INSTRUCTION-SET={HOST-STD / BS2000-NXS / BS2000-XS}
ISD	TEST-SUPPORT=YES

*COMOPT	//COMPILE
LINECNT=n	[LISTING] LAYOUT=(LINES-PER-PAGE=n) Standardzuweisung: n = 60; oder n = 15..255
NOLIST	[LISTING] SOURCE-PRINT=ERRORS-ONLY
MLPRNT	[LISTING, MACRO-PRINT] MACRO-ORIGIN-INFO=INSERTED
MODULE= spezifikation	MODULE-LIBRARY=
NDLIST	[LISTING] LAYOUT=(LASER-PRINTER=ND2)
PRTALL	[LISTING] SOURCE-PRINT=WITH-OBJECT-CODE(PRINT-STATEMENTS=IGNORED)
NOPRTIT	[LISTING, MACRO-PRINT] TITLE-STATEMENTS=IGNORED
PRTOFF= ⎡n ⎤ ⎪ ⎪ ⎨ ⎬ ⎪ ⎪ ⎣X1[;X2]...[;X5]⎦	[LISTING, MACRO-PRINT] NOPRINT-NEST-LEVEL=n Standardzuweisung : n = 255; oder n = 1..255 PREFIX-EXCEPTION=(A,B,C,...) Eine Liste bis zu 256 Makronamen-Anfangsbuchstaben ist möglich.
SAVLST	
SEQ=(nummer [,länge[,id]])	[LISTING] SOURCE-PRINT=(LINE-NUMBERING=YES)
SOURCE= spezifikation SOURCE=/	SOURCE= SOURCE=*SYSDTA-AFTER-BREAK
SYSPARM= ’max. 8 Zeichen’	[SOURCE-PROPERTIES] SYSPARM=C’ABC...’ oder ’ABC...’ Maximal sind 255 Zeichen möglich.
XREF	[LISTING] CROSS-REFERENCE=(WITH-ATTRIBUTES=NO)

12 Handbuchergänzungen

Diese Kapitel aktualisiert das vorliegende Handbuch auf den Stand ASSEMBH V1.2D.

12.1 Steuerung des ASSEMBH, der Stand-Alone-Listengenerator ASSLG und *COMOPT-Anweisungen

Abschnitt 2.2 Steuerung des ASSEMBH (S. 8),

Abschnitt 2.5 Der Stand-Alone-Listengenerator ASSLG (S. 55) und

Abschnitt 11.4 *COMOPT-Anweisungen (S. 340)

Der ASSEMBH und der ASSLG haben ab V1.2 ein eigenes Start-Kommando das der SDF-Domain "UTILITIES" zugeordnet ist:

KOMMANDO: START-ASSEMBH

oder: START-ASSLG

oder: START-ASSEMBHC

OPERANDEN : CPU-LIMIT=*JOB-REST,MONJV=*NONE

CPU-LIMIT = *JOB-REST oder <integer 1..32767>

Maximaler CPU-Zeit-Bedarf in Sekunden für den Programmlauf

MONJV = *NONE oder <full-filename 1..54 without-gen-vers>

Name der Job-Variablen, die den Programmlauf überwachen soll

12.2 COMPILATION-INFO-Option

Abschnitt 2.4.3, COMPILATION-INFO-Option (ab S. 37)

VERSION = *INCREMENT wird nicht mehr unterstützt

Das Jahr wird 4-stellig ausgegeben, sonst alles wie bei &SYSDATE.

12.3 LISTING-Option

Abschnitt 2.4.4 LISTING-Option (ab S. 42)

Bei SOURCE-FORMAT = STD muss es richtig heissen:

Es wird ein Standard Listing erzeugt.

Bei TITLE-STATEMENTS = ... ist der Defaultwert IGNORED und die Beschreibung muss lauten:

TITLE-Anweisungen, die durch Makros generiert werden, werden ausgeführt oder, wenn die PRINT-Anweisung NOGEN angegeben ist, ignoriert.

12.4 OUTPUT-Option

Abschnitt 2.5.1, GENERATE-Anweisung (ab S. 55)

Bei der Option OUTPUT=.. der GENERATE-Anweisung wird der Wert *SAVLST nicht mehr unterstützt.

12.5 ESD-Liste

Abschnitt 6.1.2 ESD-Liste (ab S. 97)

In der ESD-Liste ist für die Symbolnamen generell Platz für 32 Zeichen vorgesehen (wie unter Abschnitt 6.6.2 für das LLM-Format beschrieben).

12.6 Die strukturierte Liste

Abschnitt 6.5 Die strukturierte Liste (ab S. 117): 3. Abschnitt

Voraussetzung für die Erstellung einer strukturierten Liste ist die Anwendung der vordefinierten Makros (folgend auch Strukturmakros genannt) der Strukturierten Programmierung (siehe "ASSEMBH, Beschreibung" [1]).

Diese müssen auch in der Liste vollständig protokolliert werden, damit ein Strukturblock (siehe Abschnitt 6.5.2) korrekt aufbereitet werden kann. Werden nicht alle Stukturmakros protokolliert, kann die Aufbereitung eines Strukturblockes unvollständig oder fehlerhaft sein.

12.7 Parameter ENV=C und LOADR12

Abschnitt 7.2.1 Verknüpfung von strukturierten ASSEMBLER-Programmen mit C-Programmen (S.143, hinter dem letzten Absatz)

Die Parameter ENV=C und LOADR12 koennen nur für C-Programme verwendet werden, die V1-kompatibel sind.

Für Programme, die im CPLUSPLUS-Modus erzeugt wurden, muss immer der Parameter ILCS=YES verwendet werden.

12.8 Bedienung des Dienstprogramms COLNUMA

Abschnitt 10.4.1 Erweiterung der Strukturliste (S.241)

Zulässige Elementtypen sind P, M, S, D, X.

Abschnitt 10.4.2 Ergänzung der Strukturliste eines von COLINDA aufbereiteten Programms (S. 243)

Zulässige Elementtypen sind P, M, S, D, X.

12.9 Meldungen der Dienstprogramme

Abschnitt 10.5 Meldungen der Dienstprogramme (S. 248)

Es fehlt die Meldung:

```
015 – IO   Bedeutung: Mehr als 32767 Sätze in der Eingabe.
```

Auswirkung: Die überzähligen Sätze werden ausgegeben, aber nicht mehr strukturgerecht, sondern nur als Text.

12.10 Meldungen

12.10.1 Noch nicht aufgenommene Meldungen

Ergänzungen und Änderungen von Meldungen des ASSEMBH im Abschnitt 11.1 Meldungen des ASSEMBH (S. 257).:

```
ASS0217    B17 - SIGNIFICANT ERROR
ASS0217    CONTINUATION COLUMN IN 'ICTL' OPERAND IS WRONG
ASS0217    FORTSETZUNGSSPALTE IN 'ICTL'-OPERAND FEHLERHAFT

ASS0336    C36 - NOTE
ASS0336    MACRO (&00): MULTIPLE DEFINITIONS IN SOURCE
ASS0336    MAKRO (&00) IN DER SOURCE MEHRFACH DEFINIERT

ASS0597    E97 - WARNING
ASS0597    EXTERNAL NAMES TRUNCATED TO 8 CHARACTERS
ASS0597    EXTERNE NAMEN WERDEN AUF 8 ZEICHEN VERKUERZT
```

Bedeutung
Der Name für Eintragungen in den ESD-Satz des Objektes ist auf 8 Zeichen begrenzt. Nur die ersten 8 Zeichen des Namens werden verwendet.

```
ASS0598    E98 - WARNING
ASS0598    EXTERNAL NAMES TRUNCATED TO 32 CHARACTERS
ASS0598    EXTERNE NAMEN WERDEN AUF 32 ZEICHEN VERKUERZT

ASS1310    M10 - SIGNIFICANT ERROR
ASS1310    SYMBOL (&00) MEHRFACH DEFINIERT
ASS1310    SYMBOL (&00): MULTIPLE DEFINITIONS

ASS1918    S18 - SIGNIFICANT ERROR
ASS1918    LSDV OF (&00) (&01) TOO BIG; AID INFORMATION INCOMPLETE
ASS1918    LSDV VON (&00) (&01) ZU GROSS; AID INFORMATIONEN UNVOLLSTAENDIG
```

Bedeutung
Die Anzahl der Elemente in einer Struktur ist zu groß
(&00): Typ der Struktur ('DSECT' oder 'COM' oder 'XDSEC')
(&01): Name der Struktur

Maßnahme
Keine; Testen ohne AID Information ist möglich

```
ASS6012    NO ERROR
ASS6012    END OF ASSEMBH(&00)
ASS6012    ENDE ASSEMBH(&00)

ASS6102    NOTE
ASS6102    ZEICHEN IN 'PRTOFF=X1;X2...'-OPTION UNGUELTIG; ZEICHEN WIRD IGNORIERT
ASS6102    IN THE OPTION 'PRTOFF=X1;X2...'; AN INVALID CHARACTER IS GIVEN. IT WILL
           BE IGNORED

ASS6118    NO ERROR
ASS6118    MODULE FORMAT LLM ONLY POSSIBLE WHEN MODULE IS PUT TO A PLAM LIBRARY;
           MODUL PUT TO THE STANDARD LIBRARY 'SYS.PROG.LIB'
ASS6118    MODUL-AUSGABE IM LLM-FORMAT NUR IN PLAM-BIBLIOTHEKEN MOEGLICH;
           AUSGABE ERFOLGT IN STANDARD-BIBLIOTHEK 'SYS.PROG.LIB'.

ASS6119    WARNING
ASS6119    OPTION 'SOURCE-FORMAT=STRUCTURED' NOT POSSIBLE WITH OPTION 'OUTPUT=*SAVLST'
ASS6119    STRUKTURIERTES LISTING BEI OUTPUT=*SAVLST NICHT MOEGLICH
```

Bedeutung
Alle Angaben zum strukturierten Listing werden ignoriert

```
ASS6128    SERIOUS ERROR
ASS6128    "COMPILER INFORMATION FILE" IST KEINE PLAM-BIBLIOTHEK
ASS6128    'COMPILER INFORMATION FILE' IS NO PLAM LIBRARY
```

Bedeutung
Es wurde kein Listing erstellt.

Maßnahme
PLAM-Bibliothek für "Compiler Information File" angeben.

```
ASS6129    SERIOUS ERROR
ASS6129    PLAM-BIBLIOTHEKSELEMENT FUER "COMPILER INFORMATION FILE" IST GESPERRT
ASS6129    THE PLAM LIBRARY MEMBER FOR 'COMPILER INFORMATION FILE' IS LOCKED
```

Bedeutung
Es wurde kein Listing erstellt.

Maßnahme
PLAM-Bibliothekselement fuer 'Compiler Information File' freigeben.

```
ASS6200    SIGNIFICANT ERROR
ASS6200    INTERNAL LLM ACCESS ERROR. FUNCTION START-OUTPUT.
ASS6200    INTERNER FEHLER BEI LLM-ZUGRIFF. FUNKTION START-OUTPUT.
```

Bedeutung
Fehler bei LLM-Zugriff mit Funktion START-OUTPUT.

Maßnahme
Systemverwalter verständigen.

```
ASS6201   WARNING
ASS6201   INTERNAL LLM ACCESS WARNING. SUB_RC_=(&00).
ASS6201   INTERNE WARNUNG BEI LLM-ZUGRIFF. SUB_RC_=(&00).
```

Bedeutung
Warnung bei LLM-Zugriff.

Maßnahme
Systemverwalter verständigen.

```
ASS6202   SIGNIFICANT ERROR
ASS6202   INTERNAL LLM ACCESS ERROR: FUNCTION= (&00), MAIN-CODE= (&01), SUB-CODE= (&02)
ASS6202   INTERNER LLM-FEHLER: FUNKTION= (&00), MAIN-CODE= (&01), SUB-CODE= (&02)
```

Bedeutung
Fehler bei LLM-Zugriff. Fehlerart entsprechend dem MAIN-Retcode und dem SUB-Retcode.

Maßnahme
Systemverwalter verständigen.

```
ASS6203   SIGNIFICANT ERROR
ASS6203   INTERNAL LLM ACCESS ERROR. NO LLM GENERATED BECAUSE OF PRECEDING
ASS6203   ERROR(S) INTERNER FEHLER BEI LLM-ZUGRIFF. MODUL KONNTE WEGEN
          VORANGEGANGENER FEHLER NICHT IM LLM-FORMAT ERZEUGT WERDEN.

ASS6204   FAILURE
ASS6204   ILLEGAL VERSION OF LLMAM FOR ASSEMBH
ASS6204   VERKEHRTE VERSION VON LLMAM FUER DEN ASSEMBH FUER DIE LLM-AUSGABE
```

Maßnahme
Systemverwalter verständigen, damit er richtige LLMAM-Version installiert.

```
ASS6205   FAILURE
ASS6205   NO LLM GENERATION POSSIBLE DUE TO PRECEDING ERROR
ASS6205   LLM-GENERIERUNG WEGEN EINES VORANGEGANGENEN FEHLERS NICHT MOEGLICH

ASS6206   FAILURE
ASS6206   MEMBER NAME FOR MODULE INVALID: NO LLM GENERATION POSSIBLE
ASS6206   UNGUELTIGER ELEMENTNAME FUER DIE MODULAUSGABE: LLM-GENERIERUNG NICHT
          MOEGLICH

ASS6207   WARNING
ASS6207   'TEST-SUPPORT=AID' AND 'MODULE-FORMAT=LLM' FOR THE PRESENT ONLY
          POSSIBLE WITH 'EXTERNAL-NAMES=TRUNCATED'
ASS6207   'TEST-SUPPORT=AID' BEI LLM-MODULAUSGABE ZUNAECHST NUR MIT OPTION
          'LLM(EXTERNAL-NAMES=TRUNCATED)'

LGR0008   STRUCTURE NOT CLOSED
LGR0008   STRUKTUR NICHT ABGESCHLOSSEN
```

```
LGR0119    DMS-ERROR (&00) WHEN ACCESSING CIF
LGR0119    DMS-FEHLER (&00) BEIM ZUGRIFF AUF DEN CIF
LGR2100    INTERNAL ERROR WHEN PROCESSING CIF PARTITIONS, CODE: (&00)
LGR2100    INTERNER FEHLER BEIM BEARBEITEN VON CIF-EINTRAEGEN, CODE: (&00)
```

Bedeutung
Fehlertext fuer ASSLG-Entwicklung.

Maßnahme
Systemverwalter verständigen.

12.10.2 Neue bzw. erweiterte Meldungen des ASSEMBH-Laufzeitsystems

Ergänzungen und Korrekturen zum Abschnitt 11.1.1 (S. 319).

```
ASS7001    FATAL ERROR
ASS7001    INITIALIZATION OF THE ASSEMBLER RUNTIME SYSTEM NOT POSSIBLE; ILCS-RTC=(&00)
ASS7001    INITIALISIERUNG DES ASSEMBLER-LAUFZEITSYSTEMS NICHT MOEGLICH; ILCS-RTC=(&00)
```

Bedeutung
Mögliche Ursachen: RTC= 2: Die BS2000-Version wird nicht unterstützt
= 3: Versionsunverträglichkeit IT0SL# und IT0SL@
= 4: Speichermangel bei Initial. Stackverwaltung
= 5: Speichermangel bei Initial. Heapverwaltung
= 6: Standard Event Handler nicht initialisierbar
= 7: Eine IxxSINI Routine meldete Fehler in Reg.15

Maßnahme
Systemverwalter verständigen.

```
ASS7012    FATAL ERROR
ASS7012    PROGRAM TERMINATION WITH ERROR; ILCS-RTC=(&00)
ASS7012    TERMINIERUNG DES HAUPTPROGRAMMES MIT FEHLER; ILCS-RTC=(&00)
```

Bedeutung
Mögliche Ursachen: RTC= 2: ILCS ist nicht initialisiert
= 3: Rekursiver Aufruf einer Initialisierungs- oder Terminierungsroutine
= 4: Aufruf von IT0TERM aus einer Server-Koroutine

Maßnahme
Systemverwalter verständigen.

```
ASS7013    FATAL ERROR
ASS7013    NO MORE HEAP AVAILABLE DURING INITIALIZATION OF THE RUNTIME SYSTEM.
ASS7013    KEIN HEAPSPEICHER VERFUEGBAR WAEHREND DER INITIALISIERUNG DES LAUFZEITSYSTEMS.
```

Bedeutung
Für Verwaltungsinformation im Laufzeitsystem kann kein HEAP-Speicher besorgt werden

Maßnahme
- Datenanforderung(en) reduzieren;
- nicht mehr benötigte HEAP-Speicher freigeben;
- Benutzeradressraum durch Systemverwalter vergrößern lassen.

Literatur

[1] **ASSEMBH**
Beschreibung

[2] **AID (BS2000)**
Testen von ASSEMBH-Programmen
Benutzerhandbuch

[3] **AID** (BS2000)
Advanced Interactive Debugger

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

[5] **SDF** (BS2000)
Einführung in die Dialogschnittstelle SDF
Benutzerhandbuch

[6] **BS2000/OSD-BC**
Kommandos Band 1-7

[7] **JV** (BS2000)
Jobvariablen
Benutzerhandbuch

[8] **LMS** (BS2000)
SDF-Format
Benutzerhandbuch

[9] **BS2000/OSD-BC**
Bindelader-Starter
Benutzerhandbuch

[10] **BINDER**
(BS2000/OSD)
Benutzerhandbuch

[11] **BS2000**
Assemblerbefehle
Beschreibung

[12] **BS2000/OSD-BC**
Makroaufrufe an den Ablaufteil
Benutzerhandbuch

Stichwörter

D

M

Inhalt

FUJITSU

ASSEMBH

Benutzerhandbuch

Stand der Beschreibung:
ASSEMBH V1.2
Mit Ergänzungkapitel zu ASSEMBH V1.2D

Ausgabe Juni 2010

Kritik… Anregungen… Korrekturen…

Die Redaktion ist interessiert an Ihren Kommentaren zu diesem Handbuch. Ihre Rückmeldungen helfen uns, die Dokumentation zu optimieren und auf Ihre Wünsche und Bedürfnisse abzustimmen.

Sie können uns Ihre Kommentare per E-Mail an manuals@ts.fujitsu.com senden.

Zertifizierte Dokumentation nach DIN EN ISO 9001:2000

Um eine gleichbleibend hohe Qualität und Anwenderfreundlichkeit zu gewährleisten, wurde diese Dokumentation nach den Vorgaben eines Qualitätsmanagementsystems erstellt, welches die Forderungen der DIN EN ISO 9001:2000 erfüllt.

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Copyright und Handelsmarken

Zum 1. April 2009 ist Fujitsu Siemens Computers in den alleinigen Besitz von Fujitsu übergegangen. Diese neue Tochtergesellschaft von Fujitsu trägt seitdem den Namen Fujitsu Technology Solutions.
Das vorliegende Dokument ist eine fachlich ergänzte Neuausgabe eines früheren Handbuchs zu einer bereits vor längerer Zeit freigegebene Produktversion.
Bitte beachten Sie, dass alle Firmenbezüge und Copyrights im vorliegenden Dokument rechtlich auf Fujitsu Technology Solutions übergegangen sind.
Kontakt- und Supportadressen werden nun von Fujitsu Technology Solutions angeboten und haben die Form …@ts.fujitsu.com.
Die Internetseiten von Fujitsu Technology Solutions finden Sie unter http://de.ts.fujitsu.com/