AID V3.2A

Basishandbuch

Ausgabe November 2006

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.

Für Ihre Kommentare stehen Ihnen Fax-Formulare auf den letzten Seiten dieses Handbuchs zur Verfügung.

Dort finden Sie auch die Adressen der zuständigen Redaktion.

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

Inhalt

1 Einleitung

Mit AID (Advanced Interactive Debugger) steht im Betriebssystem BS2000 eine leistungsstarke Dialog-Testhilfe zur Verfügung. Fehlerdiagnose, Test und vorläufige Fehlerbehebung aller im BS2000 erstellten Programme können Sie mit AID wesentlich schneller und mit weniger Aufwand durchführen als mit anderen Mitteln, wie z.B. dem Einfügen von Testhilfe-Anweisungen im Programm. AID ist permanent verfügbar und besitzt eine hohe Anpassungsfähigkeit an die jeweilige Programmiersprache. Ein Programm, das Sie mit AID getestet haben, muss nicht immer erneut übersetzt werden, sondern kann sofort in den produktiven Einsatz gehen. Der Funktionsumfang von AID und seine Testsprache, die AID-Kommandos, sind primär auf die Dialoganwendung zugeschnitten. AID kann aber ebenso gut im Batch-Betrieb eingesetzt werden. AID bietet Ihnen vielfältige Möglichkeiten zur Ablaufüberwachung, Ablaufsteuerung, Ausgabe und Änderung von Speicherinhalten. Außerdem können Sie Informationen über den Programmablauf und zur Handhabung von AID abfragen.

Mit AID können Sie sowohl auf der symbolischen Ebene der jeweiligen Programmiersprache als auch auf Maschinencode-Ebene testen. Beim „symbolischen Testen“ eines Programms können Sie Daten, Anweisungen und Programmteile mit den im Quellcode vereinbarten Namen ansprechen und die Anweisungen ohne Namen mit der vom Compiler erzeugten Source-Referenz.

AID V3.2A können Sie ab BS2000/OSD V5.0 oder OSD/XC V1.0 einsetzen.

1.1 Zielgruppe

AID wendet sich an alle Software-Entwickler, die im BS2000 mit den Programmiersprachen COBOL, Fortran, C, C++, PL/I oder ASSEMBH arbeiten oder die Programme auf Maschinencode-Ebene testen oder korrigieren wollen.

1.2 Konzept der AID-Handbücher

Die Dokumentation von AID besteht aus einem Basishandbuch und den sprachspezifischen Handbüchern für das symbolische Testen sowie dem Handbuch für das Testen auf Maschinencode-Ebene. Zusätzlich liegt für den geübten AID-Anwender als Nachschlagewerk ein Tabellenheft [7] vor, in dem die Syntax aller Kommandos und die Operanden mit kurzen Erläuterungen aufgeführt sind. Außerdem zeigt das Tabellenheft die %SET-Tabellen. Zusammen mit dem Basishandbuch enthält das Handbuch für die von Ihnen gewählte Sprache alle Informationen, die Sie zum Testen brauchen. Das Handbuch für das Testen auf Maschinencode-Ebene kann statt oder zusätzlich zu einem der sprachspezifischen Handbücher eingesetzt werden.

AID - Basishandbuch
Im Basishandbuch finden Sie einen Überblick über AID und die Beschreibung der Sachverhalte und Operanden, die für alle Programmiersprachen gleich sind. Im Überblick wird die BS2000-Umgebung beschrieben, es werden die grundlegenden Begriffe erläutert und der AID-Kommandovorrat vorgestellt. Die anderen Kapitel beschreiben die Testvorbereitung, die Kommandoeingabe, das Subkommando, die Adressierung bei AID, den Operanden Medium-und-Menge, die AID-Literale und die Schlüsselwörter. Das Handbuch enthält außerdem die Meldungen, die in Kommandofolgen unzulässigen BS2000-Kommandos und die nur noch in dieser Version unterstützten Operanden.

AID-Benutzerhandbücher
In den Benutzerhandbüchern finden Sie die Kommandos in alphabetischer Reihenfolge. Alle einfachen Speicherreferenzen sind in diesen Handbüchern beschrieben.
AID - Testen von COBOL-Programmen [2]
AID - Testen von FORTRAN-Programmen [3]
AID - Testen von PL/I-Programmen [4]
AID - Testen von ASSEMBH-Programmen [5]
AID - Testen von C/C++-Programmen [6]
In diesen sprachspezifischen Handbüchern ist die Beschreibung der Operanden auf die jeweilige Programmiersprache zugeschnitten. Es wird vorausgesetzt, dass Ihnen der Sprachumfang und die Handhabung des entsprechenden Compilers geläufig sind.

Die zusätzlichen Möglichkeiten des „maschinennahen Testens“ sind beschrieben in „**AID - Testen auf Maschinencode-Ebene**“ **[1]**.
Das Handbuch für das Testen auf Maschinencode-Ebene können Sie für Programme einsetzen, zu denen keine LSD-Sätze vorhanden sind oder für die die Informationen aus dem symbolischen Testen zur Diagnose nicht ausreichen. Beim Testen auf Maschinencode-Ebene sind Sie bei der Anwendung der AID-Kommandos unabhängig von der Programmiersprache, in der das Programm erstellt wurde.

1.3 Readme-Datei

Funktionelle Änderungen und Nachträge der aktuellen Produktversion zu diesem Handbuch entnehmen Sie gegebenenfalls der produktspezifischen Readme-Datei. Die Readme-Datei heißt `SYSRME.produkt.version.D`. Sie finden diese Datei auf Ihrem BS2000-Rechner. Die Benutzerkennung, unter der sich die Readme-Datei befindet, erfragen Sie bitte bei Ihrem zuständigen Systemverwalter. Die Readme-Datei können Sie mit dem Kommando `/SHOW-FILE` oder mit einem Editor ansehen oder auf einem Standarddrucker mit dem folgenden Kommando ausdrucken:

```
/PRINT-DOC FROM-FILE=dateiname,-
/DOCUMENT-FORMAT=*TEXT(LINE-SPACING=*BY-EBCDIC-CONTR)
```

1.4 Änderungen gegenüber AID V2.1A

Die Readme-Datei zu AID V3.1 wurde in das Manual eingearbeitet:

– Indexangabe bei Arrays

– Erweiterungen/Änderungen bei den Kommandos %AID, %CONTROLn, %STOP, %TRACE

AID V3.2 unterstützt Unicode. Dies beinhaltet die folgenden Erweiterungen:

– Datentyp %UTF16 zur Darstellung von Strings, deren Zeichen eine 2-Byte-UTF16-Codierung besitzen

– Konvertierungsfunktionen %UTF16() und %C()

– UTFE-Literal U'..'

1.5 Verwendete Darstellungsmittel

kursiv Im Fließtext werden Operanden in *kursiven kleinbuchstaben* geschrieben.

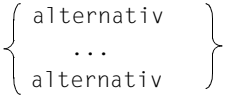
Mit diesem Symbol werden Stellen im Text gekennzeichnet, die Sie besonders beachten sollten.

2 Metasyntax

Für die Darstellung von Kommandos und deren Operanden wird folgende Metasyntax verwendet. Die Aufstellung zeigt die verwendeten Symbole und beschreibt ihre Bedeutung.

GROSSBUCHSTABEN
: Zeichenfolge, die Sie unverändert übernehmen müssen, wenn Sie eine Funktion auswählen.

kleinbuchstaben
: Zeichenfolge, die eine Variable bezeichnet. An ihre Stelle müssen Sie einen der zugelassenen Operandenwerte setzen.



`{alternativ | ... | alternativ}`
: Alternativen, unter denen Sie eine auswählen müssen. Die beiden Formate sind gleichbedeutend.

`[wahlweise]`
: Die in eckige Klammern eingeschlossenen Angaben können entfallen. Bei AID-Kommandonamen kann der in eckigen Klammern stehende Teil nur komplett entfallen, andere Abkürzungen ergeben einen Syntaxfehler.

`[...]`
: Wiederholbarkeit einer wahlfreien syntaktischen Einheit. Muss vor jede Wiederholung ein Trennzeichen, z.B. Komma gesetzt werden, steht es vor den Wiederholungspunkten.

`{...}`
: Wiederholbarkeit einer syntaktischen Einheit, die einmal angegeben werden muss. Muss vor jede Wiederholung ein Trennzeichen, z.B. Komma gesetzt werden, steht es vor den Wiederholungspunkten.

Unterstreichung

Die Unterstreichung kennzeichnet den Standardwert, den AID einsetzt, wenn Sie für einen Operanden keinen Wert angeben.

- Der dickere Punkt trennt Qualifikationen oder er steht für eine *vorqualifikation* (siehe %QUALIFY) oder er ist der Operator für einen Adressversatz, oder er ist Teil des Durchlaufzählers bzw. Subkommandonamens. Eingegeben wird der dickere Punkt mit dem Punkt, der auf der Tastatur ist. Er wurde nur der besseren Lesbarkeit wegen dicker dargestellt.

3 BS2000-Umgebung, Grundlagen und Kommandovorrat

3.1 AID im BS2000

Das Testsystem AID besteht aus zwei Komponenten
– der Anwenderschnittstelle AID und
– der Systemschnittstelle AIDSYS.

Durch diese Aufteilung ist die Anwenderschnittstelle AID unabhängig von den BS2000-Versionen. Alle erforderlichen Systemfunktionen werden über AIDSYS abgewickelt. Diese Unabhängigkeit von BS2000-Versionen ist von Bedeutung, wenn Sie sich z.B. aus einem Dump, der auf einem anderen Rechner mit einer anderen BS2000-Version erzeugt wurde, eine der möglichen Systemtabellen aufbereitet ausgeben lassen. Die versionsabhängige Aufbereitung erfolgt über AIDSYS, das in der Lage ist festzustellen, mit welcher BS2000-Version der Speicherabzug erzeugt wurde, und dann die angeforderte Systemtabelle entsprechend aufbereitet an AID zur Ausgabe übergibt.

Die Eingabe von AID-Kommandos und die Ausgabe von AID-Meldungen erfolgt über die Systemdateien SYSCMD und SYSOUT wie bei BS2000-Kommandos (siehe Kommandos Band 1 - 5 [8]).

3.1.1 Laden von AID

AID wird nicht durch den einzelnen Anwender, sondern durch den Systemverwalter mit dem Kommando /START-SUBSYSTEM AID (siehe Handbuch „Systemverwalter-Kommandos (SDF-Format)“ [10]) unter der Benutzerkennung TSOS geladen. Danach steht AID jedem Anwender zur Verfügung, ohne dass er weitere Vorkehrungen treffen muss.

3.1.2 Anwenden von AID

Im folgenden sind zwei Arten beschrieben, wie eine Testsitzung mit AID beginnen kann:

1. Sie laden und starten Ihr Programm. Wenn der Programmablauf durch einen Fehler unterbrochen wird, laden Sie, um symbolisch testen zu können, die LSD-Sätze mit dem %SYMLIB-Kommando zu der Übersetzungseinheit, in der der Fehler aufgetreten ist. Nun können Sie AID-Kommandos eingeben.

```
/START-EXECUTABLE-PROGRAM FROM-FILE=...
...
%  IDAON51 PROGRAM INTERRUPT AT LOCATION '000B62 (MOBS), (CDUMP), EC=58'
%  IDAON45 DUMP DESIRED? REPLY (Y = USER/AREA DUMP; Y,SYSTEM = SYSTEM DUMP;
N = NO)
...
/%SYMLIB ...
/%SDUMP %NEST
...
```

2. Sie laden Ihr Programm. Wenn Sie symbolisch testen wollen, geben Sie den Parameter an, der die LSD-Sätze gleich mitlädt. Sie geben AID-Kommandos zur Überwachung ein und starten das Programm mit einem AID-Kommando.

```
/LOAD-EXECUTABLE-PROGRAM FROM-FILE=..., TEST-OPTIONS=AID
/%INSERT ...
/%R
...
```

3.1.3 AID und der BS2000-Kommandointerpreter

Die AID-Funktionen rufen Sie mit AID-Kommandos auf. Ein AID-Kommando beginnt mit einem %-Zeichen, unmittelbar gefolgt vom Kommandonamen:

```
%DISPLAY ARRAY1
```

Immer, wenn sich Ihre Task im Kommandomodus befindet, können Sie auch AID-Kommandos eingeben. Die AID-Kommandos werden wie die BS2000-Kommandos vom Kommandointerpreter des BS2000 übernommen. Der identifiziert sie anhand des %-Zeichens als AID-Kommandos und übergibt sie dann an AID.

AID-Kommandos können Sie im Dialog oder in Prozedurdateien eingeben. Über den Makro CMD können Sie auch aus dem Programm AID-Kommandos aufrufen (siehe „Makroaufrufe an den Ablaufteil“ [11]).
In Kommandofolgen können Sie sowohl AID-Kommandos als auch BS2000-Kommandos verwenden.

3.1.4 AID und SDF

SDF (System Dialog Facility) ist die Dialogschnittstelle zum BS2000. Zu SDF gehört eine eigene Kommandosprache, die die bisherige Kommandosprache im ISP-Format (Interactive String Processor) ablöst. Im AID-Basishandbuch und in den sprachspezifischen Handbüchern werden BS2000-Kommandos generell im SDF-Format und zwar in der EXPERT-Form angegeben (siehe „Einführung in die Dialogschnittstelle SDF“ [16]). In einigen Fällen finden Sie eine Gegenüberstellung mit den entsprechenden ISP-Kommandos im Anhang. Darauf wird dann jeweils hingewiesen.

Für AID-Kommandos gibt es keine SDF-Notation.

3.1.5 AID-Linknamen

Dump-Dateien können mit AID über die Linknamen D0 bis D7 angesprochen werden.

Datenausgaben, TRACE-Protokolle und REPs können Sie in Ausgabedateien schreiben lassen. AID-Ausgabedateien haben das Format `FCBTYPE=SAM, RECFORM=V, OPEN=EXTEND`. AID-Ausgabedateien weisen Sie über die Linknamen F0 bis F7 zu.

3.1.6 Programme auf XS-Anlagen

In allen BS2000/OSD-Versionen können Programme den erweiterten Adressraum von über 16 Megabyte bis zu 2 Gigabyte nutzen. Entsprechend können Sie mit AID auch Programme im erweiterten Adressraum testen.

AID stellt sich automatisch auf den Adressierungsmodus des Testobjekts ein und kann sowohl mit 24-Bit-Adressen (unterer Adressraum) als auch mit 31-Bit-Adressen (erweiterter Adressraum) arbeiten.

Für den Fall, dass Sie z.B. Programmverknüpfungen von Modulen mit unterschiedlichem A-Modus testen, bietet AID folgende Funktionen an (siehe „AID - Testen auf Maschinencode-Ebene“ [1]):

– Schlüsselwort für die Systeminformation AMODE (%AMODE)
– Anzeigen des aktuellen A-Modus (%DISPLAY)
– Umschalten des A-Modus für das Testobjekt (%MOVE)
– Umschalten der Adressinterpretation bei indirekter Adressierung (%AINT)

3.1.7 Programme auf ESA-Anlagen

Ab BS2000/OSD V1.0 können Anwenderprogramme auf ESA-Anlagen (Enterprise System Architecture) neben dem Programmraum (program space), der dem bisherigen Adressraum entspricht, weitere Adressräume für Daten, die Datenräume (data spaces) nutzen. Datenräume können nur Daten enthalten, Programmcode kann in einem Datenraum nicht ausgeführt werden. Ihre eindeutige Ansprache ist über die SPID (space identification) oder über einen bzw. mehrere ALETs (access list entry token) möglich. Für die Adressierung mit ALETs wurden die Zugriffsregister als weiterer Registersatz parallel zu den Mehrzweckregistern eingeführt. In den Zugriffsregistern sind die ALETs enthalten. Wenn der AR-Modus (access register mode) eingeschaltet ist, werden bei der Adressumsetzung in einem Maschinenbefehl die Zugriffsregister mit ausgewertet und so Daten in einem Datenraum adressiert.

Nur Programme, die auf ESA-Anlagen mit einer BS2000/OSD-Version ≥ 1.0 laufen und die ESA-Befehle einsetzen, können Daten in einem solchen Datenraum ablegen (siehe Handbuch „Makroaufrufe an den Ablaufteil“ [11).

Zur ESA-Unterstützung bietet AID folgende Funktionen an (siehe Handbuch „AID - Testen auf Maschinencode-Ebene“ [1]):

– Schlüsselwort für die Systeminformation ASC-Modus (%ASC) zur Abfrage des AR-Modus.
– Schlüsselwörter für die Zugriffsregister (%nAR, %AR)
– ALET- und SPID-Qualifikation zur eindeutigen Ansprache von virtuellen Adressen in Datenräumen
– Schlüsselwort für die Systeminformation über die aktiven Datenräume (%DS[(ALET/SPID-qua)])
– Kennzeichnung von virtuellen Adressen aus Datenräumen mit einem Stern "*" bei der Ausgabe mit %DISPLAY und im %TRACE-Protokoll.

Daten in Datenräumen können nur über ihre virtuellen Adressen angesprochen werden. Wenn die Daten symbolisch aufbereitet werden sollen, können Sie dies über eine abschließende Typmodifikation erreichen.

3.1.8 Test-Privilegierung

Es ist notwendig, sicherzustellen, dass ein AID-Anwender nicht auf beliebige Datenbestände und Speicherbereiche innerhalb des Systems zugreifen oder sie sogar verändern kann. Für die Steuerung der Zugriffsrechte gibt es in jedem Benutzereintrag in der JOIN-Datei die Angabe „Test-Privilegierung“ für die Lese- und Schreib-Zugriffsberechtigung. Diese wird vom Systemverwalter eingetragen.

Beim Start einer Task werden immer die niedrigsten Privilegierungen (1,1) vergeben. Damit können Sie den gesamten Leistungsumfang von AID nutzen, wie er in der Dokumentation beschrieben ist.

Wenn Sie Dateien oder Bibliotheken mit READ- oder EXEC-PASSWORD geschützt haben, können Sie auch mit AID nur zugreifen, wenn Sie das Kennwort eingegeben haben.
Wenn Sie auf Speicherbereiche zugreifen wollen, die eine höhere Privilegierung benötigen, können Sie mit dem Kommando /MODIFY-TEST-OPTIONS Ihre Privilegierung verändern, sofern es der Eintrag in der JOIN-Datei erlaubt. Diesen Eintrag können Sie sich mit dem Kommando SHOW-USER-ATTRIBUTES anschauen (siehe „Kommandos Band 1 - 5“ [8]).

Außerdem bietet Ihnen AID Schlüsselwörter an, mit denen Sie auf geschützte Bereiche, z.B. Register, zugreifen können.

3.2 Grundlegende Begriffe

3.2.1 Das Testobjekt

Das Programm, das Sie mit AID bearbeiten wollen, wird als Testobjekt bezeichnet. Es kann unter Ihrer Benutzerkennung geladen sein oder als Speicherabzug in einer Dump-Datei vorliegen. Innerhalb einer Testsitzung können Sie zwischen diesen Möglichkeiten wechseln, um z.B. Daten im geladenen Programm mit Daten in einer Dump-Datei oder um Speicherabzüge aus unterschiedlichen Versionen desselben Objekts zu vergleichen.

Das Programm können Sie immer auf Maschinencode-Ebene testen und, wenn bei der Übersetzung LSD-Sätze erzeugt wurden, auch auf symbolischer Ebene. Hierzu brauchen Sie das Programm nicht erneut zu übersetzen oder zu binden. Da Sie das Programm ohne die symbolischen Informationen laden können, sind nach einem fehlerfreien Testablauf keine weiteren Übersetzungs- oder Binderläufe nötig. Das Programm können Sie sofort in den produktiven Einsatz übernehmen.

3.2.2 Objekt-Strukturliste und LSD

AID arbeitet mit zwei Listen, aus denen es Informationen über das Programm entnimmt:

Die Objekt-Strukturliste wird beim Binden mit BINDER standardmäßig aus dem ESV (External Symbols Vector) erzeugt. Beim Binden mit TSOSLNK heißt das entsprechende Verzeichnis ESD (External Symbolic Dictionary). In der Objekt-Strukturliste sind unter anderem Informationen über die CSECTs, DSECTs und COMMONs eines Programms enthalten. Ist diese Liste vorhanden, können Sie über den Namen einer CSECT oder eines COMMONs auf die zugehörige Adresse, den Inhalt und die Länge zugreifen.

Die LSD (List for Symbolic Debugging) ist das Verzeichnis der im Modul definierten Datennamen, Anweisungsnamen und Namen von Programmteilen. Ebenso enthält sie die vom Compiler erzeugten Source-Referenzen. LSD-Sätze werden vom Compiler erzeugt, wenn Sie die entsprechende Option beim Übersetzen angeben. Wurden LSD-Sätze erzeugt, können Sie über die im Quellprogramm definierten und die vom Compiler erzeugten Namen auf Adresse, Inhalt, Länge und Typ der entsprechenden Speicherobjekte zugreifen.

Der Klasse-5-Speicherbedarf eines Programms mit LSD-Sätzen steigt je nach der Menge der symbolischen Definitionen auf ein Vielfaches der reinen Programmgröße. Speichern Sie die Bindemodule in einer PLAM-Bibliothek ab, dann können Sie das Programm ohne LSD-Sätze laden und starten und die PLAM-Bibliothek, die die LSD-Sätze enthält, mit dem Kommando %SYMLIB öffnen. Bei Bedarf lädt AID dann die LSD-Sätze aus der zugewiesenen Bibliothek nach.

3.2.3 Symbolisch - maschinennah

AID kennt zwei Testebenen: Auf symbolischer Ebene werden die vom Compiler erzeugten symbolischen Adressen aus den LSD-Sätzen benutzt. Speicherstellen sprechen Sie über die Namen an, die im Quellprogramm vergeben wurden. AID-Ausgaben enthalten die Programm-, Daten- , Anweisungsnamen und Source-Referenzen. Das Schlüsselwort %HLLOC (High-Level-Location) als Operand des AID-Kommandos %DISPLAY gibt die symbolischen Lokalisierungsinformationen aus. Sie bestehen aus dem Kontextnamen, dem Namen der Übersetzungseinheit und des aktuellen Haupt- oder Unterprogramms sowie der Source-Referenz, der die Adresse zugeordnet ist. Die AID-Kommandos %JUMP, %SDUMP und %SYMLIB können Sie nur auf symbolischer Ebene einsetzen, also wenn LSD-Sätze zu dem angesprochenen Programmteil vorhanden sind. Bei den Kommandos %CONTROL*n* und %TRACE entscheidet das Schlüsselwort für *kriterium*, ob die Überwachung oder Verfolgung auf symbolischer Ebene stattfindet.

Wenn Sie eine Adresse von AID errechnen lassen (komplexe Speicherreferenz), können Sie jederzeit von der symbolischen zur maschinennahen Ebene wechseln. Mit Adressselektion und Pointer-Operator (%@(name)->) können Sie das angesprochene Speicherobjekt mit allen seinen maschinennahen Eigenschaften benutzen.

Auf Maschinencode-Ebene werden nur die CSECT- und COMMON-Informationen aus der Objekt-Strukturliste benutzt. AID-Ausgaben enthalten virtuelle Adressen und CSECT- bzw. COMMON-Namen. Das Schlüsselwort %LOC (Low-Level-Location) als Operand des AID-Kommandos %DISPLAY gibt die maschinennahen Lokalisierungsinformationen aus. Sie bestehen aus dem Kontextnamen, den Namen der Ladeeinheit, des Objektmoduls, der CSECT bzw. des COMMON sowie aus der CSECT- bzw. COMMON-relativen Adresse. Bei den Kommandos %CONTROL*n* und %TRACE entscheidet das Schlüsselwort für *kriterium*, ob die Überwachung oder Verfolgung auf Maschinencode-Ebene stattfindet.
In einer komplexen Speicherreferenz können Sie symbolische Adressen und Elemente der maschinennahen Adressierung verbinden und dabei alle Eigenschaften der symbolischen Adressierung weiterhin benutzen.

3.2.4 Der AID-Arbeitsbereich

Der AID-Arbeitsbereich ist der Adressraum, in dem Sie Speicherobjekte ohne Angabe einer Basisqualifikation ansprechen können. Er umfasst den nicht-privilegierten Teil des virtuellen Speichers in Ihrer Task, der vom Programm zusammen mit seinen angeschlossenen Subsystemen belegt ist oder den entsprechenden Bereich in einem Speicherabzug. Ob Sie im geladenen Programm oder in einem Speicherabzug testen, können Sie durch das Kommando %BASE festlegen. Ohne Angabe von %BASE liegt der AID-Arbeitsbereich im geladenen Programm. Dies wird als AID-Standard-Arbeitsbereich bezeichnet. Sie können auch innerhalb eines Kommandos vom aktuell eingestellten Arbeitsbereich abweichen, indem Sie in einem Adressoperanden eine entsprechende Basisqualifikation {E=VM|Dn} angeben.

Liegt der AID-Arbeitsbereich in einer Dump-Datei, können Sie die Kommandos zur Ablaufüberwachung und zur Ablaufsteuerung nicht einsetzen. Außerdem können Sie eine Dump-Datei nicht mit AID-Kommandos modifizieren. Sie können sich jedoch aus einer Dump-Datei Daten ausgeben lassen, die Aufrufhierarchie zum Zeitpunkt der Programmunterbrechung zurückverfolgen, Maschinencode in symbolische Assembler-Notation rückübersetzen, und Sie können Zeichenfolgen in einer Dump-Datei suchen. Außerdem können Speicherinhalte eines geladenen Programms mit Daten aus einer Dump-Datei überschrieben werden.

3.2.5 Speicherobjekte und Speicherreferenzen

Als Speicherobjekt wird eine bestimmte Anzahl von Bytes bezeichnet, die zusammenhängend ab einer bestimmten Adresse im Speicherbereich des Programms liegen. Das sind zum einen die Daten eines Programms und zum anderen der Befehlscode. Die außerhalb des Programmspeichers liegenden Register und der Befehlszähler gehören ebenfalls zu den Speicherobjekten. Sie werden von AID über Schlüsselwörter angesprochen.

Konstanten werden nicht als Speicherobjekte bezeichnet. Zu den Konstanten zählen alle im Programm definierten Konstanten, die Anweisungsnamen, Source-Referenzen, die Ergebnisse von Adressselektion, Längenselektion und Längenfunktion und die AID-Literale. Sie repräsentieren einen Wert, der nicht verändert werden kann. Sie besitzen kein Adressattribut.

Mit einer Speicherreferenz sprechen Sie ein Speicherobjekt an. Es gibt einfache Speicherreferenzen und komplexe Speicherreferenzen. Einfache Speicherreferenzen sind virtuelle Adressen, Namen, zu denen sich AID die Adresse aus den LSD-Sätzen holen kann, und Schlüsselwörter.
In einer komplexen Speicherreferenz errechnet AID eine Adresse gemäß Ihren Angaben, denen AID gleichzeitig Typ und Länge des durch diese Adresse bezeichneten Speicherobjekts entnimmt. Folgende Operationen können in einer komplexen Speicherreferenz vorkommen: Adressversatz, indirekte Adressierung, Typmodifikation, Längenmodifikation und Adressselektion.

Liegt eine Speicherreferenz nicht im gerade gültigen AID-Arbeitsbereichs oder außerhalb des aktuellen Haupt- oder Unterprogramms oder ist sie da drin nicht eindeutig, können Sie mit Qualifikationen den Pfad zu der gewünschten Speicherreferenz beschreiben.

3.2.6 Namenskonventionen bei AID

Für alle Namen, die Sie in AID-Kommandos einsetzen, um Programme oder Programmteile, Daten oder Anweisungen anzusprechen oder um Subkommandos zu benennen, gilt unabhängig von der verwendeten Programmiersprache der folgende Zeichenvorrat:
a-z, A-Z, 0-9, $, #, @, Unterstrich "_" oder Bindestrich "-".

Der Bindestrich ist nicht als erstes Zeichen zugelassen und ist auch nur dann als Teil des Namens erlaubt ist, wenn SYMCHARS=STD eingestellt ist (Kommando %AID).
Wenn der Bindestrich das letzte Zeichen eines Namens ist, können Sie diesen Namen nur mit N'...' angeben.

Generell müssen alle Namen, die Sonderzeichen enthalten oder für AID zweideutig sein können, in N'...' gesetzt werden. Marken mit Sonderzeichen sowie Marken als Ausgangsadresse für eine komplexe Speicherreferenz schreiben Sie in L'...'.

Um AID zu veranlassen, zwischen Groß-/Kleinschreibung zu unterscheiden, müssen Sie zunächst das Kommando %AID LOW[=ON] eingeben.
Bei BLS-Namen, also den Namen, die dem Binder-Lader-Starter bekannt sind wie z.B. Namen von CSECTs, COMMONs oder Entrys sowie bei den Namen von Übersetzungseinheiten (bei Fortran: Programmeinheiten) wird die Groß-/Kleinschreibung jedoch nur dann berücksichtigt, wenn das Kommando %AID LOW=ALL eingegeben wurde.

Erlaubte Länge ist für BLS-Namen und Namen von Übersetzungseinheiten 32 Zeichen, Namen von Daten und Programmteilen wie Funktionen, Prozeduren oder Unterprogrammen dürfen bis zu 255 Zeichen lang sein. Namen von Subkommandos können einschließlich der vorangestellten Zeichen "%•" bis zu 32 Zeichen lang sein.

Übersicht

Namen	Länge (max.)	%AID LOW=ON wirksam	%AID LOW=ALL wirksam
BLS-Namen und Namen von Übersetzungseinheiten	32	nein	ja
Daten- und Programmnamen	255	ja	ja
Namen von Marken	255	ja	ja
Subkommandonamen	32 einschl. %•	nein	nein

3.2.7 Zeichendarstellung durch UTF16 / UTFE

Mit Unterstützung von Unicode gibt es in AID den neuen Datentyp %UTF16 zur Darstellung von Strings. Bei diesem Datentyp hat jedes Zeichen eine 2-Byte-Verschlüsselung. Der bisher von AID unterstützte Datentyp zur Darstellung von Strings hat eine 1-Byte-EBCDIC-Codierung.

AID unterstützt für Eingabe- und Ausgabemedien die UTFE-Zeichencodierung. Diese Codierung ist die EBCDIC-Variante von UTF8, die eine Mehrbyte-Codierung in variabler Bytelänge unterstützt.

Einstellung einer EBCDIC-Codiertabelle über %AID

Das %AID-Kommando wurde um den Operanden EBCDIC erweitert. Damit kann die EBCDIC-Codierung eines C-Strings spezifiziert werden, die AID verwendet, wenn eine Konvertierung zwischen einem UTFE-/%UTF16-String und einem 1-Byte-C-String durchgeführt werden muss.

AID unterstützt alle 1-Byte-EBCDIC-Codierungen, die das Subsystem XHCS-SYS anbietet. Die aktuellen Namen der Codiertabellen können Sie sich mit dem Kommando %SHOW %CCSN ausgeben lassen.

Mit den neuen Funktionen %C(...) und %UTF16(...) können Sie z.B. die Literal-Codierung in eine andere Codierung umwandeln.

3.3 Die AID-Kommandos

AID verfügt über eine Vielzahl von Funktionen, die Sie über AID-Kommandos aufrufen. Dieses Kapitel gibt Ihnen einen Überblick über den Leistungsumfang von AID. Die vollständigen Beschreibungen der Kommandos finden Sie in den sprachspezifischen Handbüchern oder dem Handbuch für das Testen auf Maschinencode-Ebene. Den AID-Kommandovorrat kann man in vier Funktionsgruppen einteilen. In der folgenden Übersicht sind alle AID-Kommandos mit ihren Operanden aufgeführt. Kommandos, die nur für das Testen auf symbolischer Ebene verwendet werden können, sind in der zweiten Spalte mit 'SY' gekennzeichnet.

Ablaufüberwachung

Kommandoname		Operanden
%C[ONTROL]*n*		[kriterium][,...] [IN control-bereich] <subkdo>
%IN[SERT]		testpunkt [<subkdo>] [steuerung]
%ON		{write-ereignis | ereignis} [<subkdo>]
%REM[OVE]		ziel

Ablaufsteuerung und Ablaufprotokollierung

Kommandoname		Operanden
%CONT[INUE]		
%JUMP	SY	fortsetzung
%R[ESUME]		
%STOP		
%T[RACE]		[anzahl] [kriterium][,...] [IN trace-bereich]

Ausgabe und Modifikation von Speicherinhalten

Kommandoname		Operanden
%D[IS]A[SSEMBLE]		[anzahl] [FROM start]
%D[ISPLAY]		daten {,...} [medium-u-menge][,...]
%F[IND]		[[ALL] suchbegriff] [IN find-bereich] [alignment]
%M[OVE]		sender INTO empfänger [REP]
%SD[UMP]	SY	[dump-bereich][,...] [medium-u-menge][,...]
%SET		sender INTO empfänger

Verwaltung

Kommandoname		Operanden
%AID		[CHECK] [REP] [SYMCHARS] [OV] [LOW] [DELIM] [LANG] [EBCDIC]
%AINT		[aid-mode] [,...]
%BASE		[basis]
%D[UMP]F[ILE]		[link [= datei]]
%H[ELP]		[info-ziel] [medium-u-menge][,...]
%OUT		[ziel-kommando [medium-u-menge][,...]]
%OUTFILE		[link [=datei]]
%Q[UALIFY]		[vorqualifikation]
%SYMLIB	SY	[qualifikation-u-lib][,...]
%SHOW		[show-ziel]
%TITLE		[seitenkopf]

3.3.1 Ablaufüberwachung

Die Kommandos %CONTROL*n*, %INSERT und %ON dienen der dynamischen Überwachung des Programmablaufs. Mit ihnen vereinbaren Sie Überwachungsbedingungen und Subkommandos (siehe Kapitel „Subkommando“ auf Seite 47). Tritt die Überwachungsbedingung ein, wird das zugehörige Subkommando bearbeitet. Mit jedem der drei Kommandos legen Sie eine andere Art von Überwachungsbedingung fest, die Sie mit %REMOVE wieder aufheben können.

%CONTROL*n kriterium*
Mit *kriterium* bezeichnen Sie den Typ der zu überwachenden Anweisungen oder Maschinenbefehle.

%INSERT *testpunkt*
Mit *testpunkt* bezeichnen Sie eine Adresse im ausführbaren Teil des Programms.

%ON {*write-ereignis* | *ereignis*}
Mit *write-ereignis* schalten Sie die Schreibüberwachung ein. Mit *ereignis* beschreiben Sie ein Ereignis im Progammablauf, z.B. Adressierungsfehler, Systemaufruf (SVC) oder Nachladen eines Moduls.

%REMOVE *ziel*
Mit dem Kommando %REMOVE können Sie Überwachungsvereinbarungen wieder aufheben. Mit *ziel* geben Sie an, welche Vereinbarung Sie aufheben.

Je nachdem, was Sie in Ihrem Test überwachen wollen, wählen Sie das entsprechende Kommando aus. Mit dem zugehörigen Subkommando können Sie gezielte Eingriffe in den Programmablauf vornehmen. Mit einem Subkommando vereinbaren Sie ein Kommando oder eine Kommandofolge und eventuell eine Bedingung. Außerdem können Sie dem Subkommando einen Namen geben, mit dem Sie seinen Durchlaufzähler ansprechen oder das Subkommando löschen können. Mit den entsprechenden AID-Kommandos können Sie im Subkommando steuern, ob das Programm unterbrochen werden oder ob es weiterlaufen soll. So ist es möglich, einen automatischen Testablauf vorzubereiten. Allerdings müssen Sie bereits bei der Eingabe des Überwachungskommandos über alle erforderlichen Informationen verfügen, die bei Eintreten der Überwachungsbedingung für das weitere Vorgehen von Bedeutung sind. Sie müssen also nicht mehr mit der Eingabe von Kommandos am Terminal in den Testablauf eingreifen, um z.B. aktuelle Speicher- oder Registerstände zu ändern, wenn Sie die entsprechenden Kommandos im Subkommando hinterlegen.

3.3.2 Ablaufsteuerung

Mit den Kommandos %CONTINUE, %RESUME, %STOP und %TRACE verändern Sie den Zustand eines geladenen Programms. Mit %JUMP können Sie für FOR1- und COBOL-Programme eine Fortsetzungsadresse vereinbaren, die vom codierten Programmablauf abweicht. Ein geladenes Programm kann sich in einem von drei Programmzuständen befinden:

1. Das Programm steht.
   %STOP, K2-Taste oder Ende eines %TRACE unterbrechen ein laufendes Programm. Die Task befindet sich im Kommandomodus. Sie können Kommandos eingeben.
2. Das Programm läuft ohne Ablaufverfolgung.
   %RESUME startet ein Programm oder setzt es fort. %CONTINUE bewirkt dasselbe; ist allerdings ein %TRACE noch nicht ganz abgearbeitet, so setzt %CONTINUE das Programm mit Ablaufverfolgung fort.
3. Das Programm läuft mit Ablaufverfolgung.
   %TRACE startet ein Programm oder setzt es fort. Der Programmablauf wird entsprechend der Vereinbarungen im %TRACE protokolliert. %CONTINUE bewirkt dasselbe, wenn noch ein %TRACE aktiv ist.

Wenn Sie keine andere Fortsetzungsadresse vereinbart haben, wird der Programmablauf an der Unterbrechungsstelle fortgesetzt. Mit %JUMP (nur FOR1 und COBOL) oder durch Änderung des Befehlszählers (%PC) mit %MOVE oder %SET können Sie eine andere Fortsetzungsadresse festlegen. Für beide Eingriffe in den Programmablauf gilt, dass Sie immer selbst dafür sorgen müssen, dass Speicherinhalte, Registerstände, Datei-Status und -Inhalte auch zur vereinbarten Fortsetzungsadresse passen.

%CONTINUE und %RESUME starten ein geladenes Programm oder setzen es fort. Der Unterschied zwischen den beiden Kommandos besteht darin, dass %RESUME einen noch aktiven %TRACE löscht, %CONTINUE hingegen führt den %TRACE fort.

%STOP hält das Programm an und gibt eine Meldung aus (STOP-Meldung), die Informationen über die aktuelle Unterbrechungsstelle enthält.

%TRACE schaltet die Ablaufverfolgung ein. Das Programm läuft, und die ausgewählten Befehle werden protokolliert. Beendet wird ein %TRACE, wenn die angegebene Anzahl von Befehlen protokolliert ist oder wenn nach einer Unterbrechung das Programm mit %RESUME fortgesetzt wird. Ist der %TRACE nur unterbrochen, weil ein Subkommando ausgeführt wurde, das ein %STOP enthielt, oder ein *steuerungs*-Operand KEEP oder STOP ausgeführt wurde, oder weil die K2-Taste gedrückt wurde, so kann der %TRACE mit %CONTINUE fortgesetzt werden.
Das %TRACE Kommando wurde um den Operanden *fortsetzung* erweitert. Mit diesem können Sie steuern, ob das Programm nach Beendigung des %TRACE anhält oder ohne Protokollierung weiterläuft.

3.3.3 Ausgabe und Modifikation von Speicherinhalten

Mit den Kommandos %DISPLAY, %SDUMP und %DISASSEMBLE können Sie sich Speicherinhalte und Informationen zum Programm ausgeben lassen.
Mit den Kommandos %MOVE und %SET können Sie Speicherinhalte im geladenen Programm verändern.
Mit %FIND können Sie Zeichenfolgen suchen.

%DISPLAY gibt Ihnen den aktuellen Inhalt von Speicherobjekten, deren Adressen bzw. Längen oder die Werte von Konstanten, Anweisungsnamen und Source-Referenzen aus. Sie können mit %DISPLAY auch Systeminformationen abfragen, den Vorschub nach SYSLST steuern oder AID-Literale ausgeben lassen, z.B. um damit den Testverlauf zu kommentieren. Die Ausgabe erfolgt über SYSOUT, SYSLST oder in eine katalogisierte Datei.
Wenn Sie ein Speicherobjekt mit seinem Namen ansprechen, gibt AID es in dem Datentyp und der Länge aus, die im Quellprogramm festgelegt wurde. Mit der Typ- und/oder Längenmodifikation können Sie eine andere Aufbereitung vereinbaren.

%SDUMP gibt einen symbolischen Dump aus. Sie können sich entweder Daten der aktuellen Aufrufhierarchie oder die Aufrufhierarchie selbst ausgeben lassen. Die aktuelle Aufrufhierarchie reicht von der Unterprogrammebene, in der das Programm unterbrochen wurde, über die durch CALL-Anweisungen aufgerufenen Unterprogramme bis zum Hauptprogramm. %SDUMP %NEST gibt die Namen aller Programmteile der aktuellen Aufrufhierarchie aus, soweit AID die Verknüpfungskonventionen bekannt sind. Aus den Programmteilen dieser Hierarchie können Sie dann Daten oder Datenbereiche ausgeben lassen.

Mit %DISASSEMBLE können Sie Speicherinhalte im ausführbaren Teil eines Programms rückübersetzen lassen. AID gibt die Speicherinhalte in symbolischer Assembler-Notation aufbereitet aus. Für Speicherinhalt, der nicht als Befehl interpretiert werden kann, wird eine Ausgabezeile erzeugt, die die sedezimale Darstellung des Speicherinhalts und den Hinweis INVALID OPCODE enthält.

Mit %MOVE verändern Sie Speicherinhalte im geladenen Programm. %MOVE überträgt einen Sender auf einen Empfänger ohne zu überprüfen, ob die Speichertypen von Sender und Empfänger miteinander verträglich sind und ohne Typanpassung. Die Übertragung wird in der Länge von Sender durchgeführt. Sender wird linksbündig in Empfänger übertragen. AID überprüft nur, dass nicht über die rechte Bereichsgrenze (=End-adresse) von Empfänger geschrieben wird.
Zu %MOVE können Sie den Änderungsdialog einschalten oder REP-Sätze erzeugen lassen.

Mit %SET verändern Sie Speicherinhalte im geladenen Programm. %SET überträgt einen Sender auf einen Empfänger und überprüft vor der Übertragung, ob die Speichertypen von Sender und Empfänger miteinander verträglich sind. Die Übertragung wird in der Länge von Empfänger durchgeführt. Sender wird typgerecht in Empfänger übertragen, bei Bedarf

wird abgeschnitten, aufgefüllt oder eine Typ-Anpassung durchgeführt. Die Regeln für die Übertragung mit %SET lehnen sich eng an die jeweilige Programmiersprache an. Die %SET-Beschreibung in den sprachspezifischen Handbüchern enthält eine Tabelle der zulässigen Speichertyp-Kombinationen.
Zu %SET können Sie den Änderungs-Dialog einschalten.

Mit %FIND können Sie in Daten oder im gesamten Benutzer-Adressraum des geladenen Programms bzw. in einer Dump-Datei eine Zeichenfolge suchen und die Treffer auf Terminal (SYSOUT) ausgeben lassen.
Bei einem Treffer gibt AID die Adresse aus, an der die Zeichenfolge gefunden wurde und, wenn möglich, den Namen der zugehörigen CSECT oder des COMMON und die Distanz zur Anfangsadresse von CSECT oder COMMON. Dazu wird der Speicherinhalt ab der Trefferadresse bis zum Ende des Suchbereichs, höchstens aber in der Länge von 12 Bytes, ausgegeben. Zusätzlich speichert AID die Trefferadresse im AID-Register %0G und die Fortsetzungsadresse (Trefferadresse + Suchstringlänge) im AID-Register %1G ab.

3.3.4 Verwaltungsfunktionen

Mit den Kommandos %DUMPFILE, %SYMLIB und %OUTFILE verwalten Sie AID-Eingabe- und AID-Ausgabedateien. Sie weisen ihnen Linknamen zu und öffnen oder schließen sie.
Mit %OUT steuern Sie die Ausgaben von AID, mit %TITLE legen Sie eine Kopfzeile für die Ausgabe nach SYSLST fest.
Mit %AID, %AINT, %BASE und %QUALIFY vereinbaren Sie globale Voreinstellungen.
Mit %HELP können Sie sich Hilfe-Texte ausgeben lassen.
Mit %SHOW können Sie sich über die aktuell gültigen Voreinstellungen und über die AID-Kommandos informieren lassen, die Sie im bisherigen Testverlauf eingegeben haben und die noch aktiv sind.

Mit %DUMPFILE verwalten Sie Dump-Dateien. Sie werden zugewiesen über die AID-Linknamen D0 bis D7. Sie können AID veranlassen, eine Dump-Datei zu öffnen oder zu schließen. In den Dump-Dateien befinden sich Speicherabzüge, die Sie in Ihrem Test verwenden wollen.

Mit %SYMLIB können Sie PLAM-Bibliotheken öffnen, in denen Sie die OMs oder LLMs Ihres Programms mit den LSD-Sätzen abgespeichert haben.
Auf geöffnete PLAM-Bibliotheken greift AID zu, wenn Sie in einem Kommando symbolische Namen ansprechen, die in einer Übersetzungseinheit (bei Fortran: Programmeinheit) liegen, für die keine LSD-Sätze geladen wurden. Mit einer Basisqualifikation können Sie die PLAM-Bibliothek einem bestimmten AID-Arbeitsbereich zuordnen.
Bei der Anmeldung mit %SYMLIB stellt AID nur fest, ob die angegebene Bibliothek geöffnet werden kann; AID überprüft nicht, ob der Inhalt einer Bibliothek zu dem Programm passt, das gerade bearbeitet wird. Dadurch ist es möglich, vorbereitend die Bibliotheken anzumelden, die Sie während eines Testlaufs eventuell benötigen. Sind für eine Basisqualifikation

mehrere Bibliotheken angemeldet, so durchsucht AID sie in der Reihenfolge, in der sie im %SYMLIB-Kommando angegeben wurden.
AID kann bis zu 14 PLAM-Bibliotheken parallel verwalten.

Mit %OUTFILE verwalten Sie AID-Ausgabedateien, in die die Ausgaben der Kommandos %DISASSEMBLE, %DISPLAY, %HELP, %SDUMP und %TRACE oder die REPs des %MOVE-Kommandos geschrieben werden. Sie werden den AID-Linknamen F0 bis F7 zugewiesen. Falls eine Datei noch nicht existiert, wird sie durch AID katalogisiert und geöffnet. Außerdem können Sie veranlassen, dass offene Ausgabedateien wieder geschlossen werden.

Mit %OUT legen Sie für die Ausgabekommandos %DISASSEMBLE, %DISPLAY, %HELP, %SDUMP und %TRACE fest, welche Ausgabemedien verwendet werden und ob Zusatzinformationen ausgegeben werden sollen. Ausgabemedien sind das Terminal (SYSOUT), SYSLST oder eine AID-Ausgabedatei, die zuvor mit %OUTFILE zugewiesen werden kann. Bei den Kommandos %DISPLAY, %HELP und %SDUMP können Sie lokal einen eigenen *medium-u-menge*-Operanden angeben, der für diese Kommandos die Vereinbarungen des %OUT-Kommandos vorübergehend außer Kraft setzt. Die Kommandos %DISASSEMBLE und %TRACE bieten diesen Operanden nicht an.

Mit %TITLE können Sie für die Ausgabe nach SYSLST einen eigenen Seitenkopftext definieren und den Seitenzähler steuern. Die Ausgabe nach SYSLST wird mit %OUT oder dem *medium-u-menge*-Operanden eines Ausgabekommandos vereinbart.

Mit %AID können Sie den Änderungsdialog einschalten, REPs zu den Speicheränderungen mit %MOVE erstellen lassen, die Interpretation des Bindestrichs in Namen steuern, die Überlagerungsstruktur eines Programms (Overlay) berücksichtigen lassen, Klein- und Großbuchstabenunterscheidung aus Benutzereingaben einschalten, andere Begrenzer für Ausgaben vom Typ Character festlegen, einen EBCDIC-Zeichensatz für Konversionen von oder nach UTF16/UTFE oder für die Interpretation und Darstellung von Zeichen festlegen oder auf die englische Ausgabe von Hilfe-Texten umschalten.

Mit %AINT können Sie die Interpretation von indirekten Adressangaben in AID-Kommandos umschalten. Sie legen damit fest, ob AID eine Adresse vor einem Pointer-Operator (->) als 24-Bit-Adresse oder als 31-Bit-Adresse interpretieren soll. Der Adressierungsmodus des Testobjekts wird damit nicht beeinflusst. Mit einer Basisqualifikation können Sie den Bereich vereinbaren, für den die Angabe zur Adressinterpretation gelten soll.

Mit %BASE vereinbaren Sie die Basisqualifikation. Sie gilt für alle nachfolgenden Kommandos, in denen Sie keine explizite Basisqualifikation angeben. Mit %BASE legen Sie fest, ob sich der AID-Arbeitsbereich im geladenen Programm oder in einer Dump-Datei befinden soll. Wenn Sie als Basisqualifikation eine Dump-Datei angeben, muss diese zuvor mit %DUMPFILE zugewiesen werden.

Mit %QUALIFY bezeichnen Sie Qualifikationen oder eine Adresse, auf die Sie sich im Adressoperanden eines anderen Kommandos durch Voran Stellen eines Punktes beziehen möchten. Diese verkürzte Schreibweise ist immer dann sinnvoll, wenn Sie mehrfach Adressen ansprechen wollen, die die gleichen Qualifikationen erfordern oder die von der gleichen Ausgangsadresse aus berechnet werden.

Mit %HELP können Sie sich über die AID-Kommandos informieren. Auf das gewählte Ausgabemedium werden ausgegeben: entweder alle AID-Kommandos oder das gewählte Kommando und seine Operanden. Der HELP-Informations-Pool umfasst Informationen für das symbolische Testen sowie für das Testen auf Maschinencode-Ebene.

Mit %SHOW können Sie sich über die AID-Kommandos des bisherigen Testverlaufs, über die aktuell gültigen globalen Einstellungen oder über die aktuell gültigen Operandenwerte der Kommandos informieren.
%SHOW ohne Operand zeigt das zuletzt eingegebene AID-Kommando an.
Zu den Kommandos der Ablaufüberwachung (%CONTROL*n*, %INSERT und %ON) gibt %SHOW eine Liste der Original-Eingabestrings aller aktiven %CONTROL*n* oder %ON oder die Liste der gesetzten Testpunkte mit Kontext, virtueller Adresse, CSECT oder COMMON und Distanz zum CSECT- oder COMMON-Anfang aus. Den Original-Eingabestring zu einem bestimmten Testpunkt erhalten Sie mit %SHOW %INSERT *testpunkt*. Sind am Testpunkt mehrere %INSERTs aktiv, so werden die Kommandos in der umgekehrten Reihenfolge ausgegeben, d.h. der zuletzt eingegebene %INSERT wird als erster aufgelistet.
Zu %SHOW %TRACE gibt AID das zuletzt eingegebene %TRACE-Kommando, die abgearbeiteten %TRACE-Schritte und die aktuell gültigen Operandenwerte aus.
Zu %SHOW %DISASSEMBLE erhalten Sie die aktuell gültigen Operandenwerte, zu %SHOW %FIND das zuletzt eingegebene Kommando und den letzten Treffer.
Zu den Kommandos, die AID-Ein- oder Ausgabedateien verwalten, gibt %SHOW alle explizit oder implizit geöffneten Dateien sowie verschiedene Zusatzinformationen aus.
%SHOW %OUT zeigt die aktuellen Ausgabe-Vereinbarungen der Kommandos an, deren Ausgabe über %OUT gesteuert wird. Zu %SHOW %AID, %SHOW %BASE oder %SHOW %QUALIFY werden die mit diesen Kommandos getroffenen Vereinbarungen ausgegeben.

3.3.5 Übersicht über die Geltungsdauer der Kommandos

Beim Arbeiten mit AID ist es wichtig zu wissen, welche Kommandos, Operandenwerte oder Vereinbarungen bis zur Eingabe des nächsten Kommandos gleichen Typs, bis zum Ende des Programms oder Tasks gültig sind. Dazu dient die folgende Übersicht:

Kommando	Operand	Gültigkeitsdauer
%AID	alle	gültig bis zur Eingabe eines neuen %AID mit entsprechendem Operanden oder bis /EXIT-JOB.

Tabelle 1: Übersicht über die Geltungsdauer der Kommandos

Kommando	Operand	Gültigkeitsdauer
%AINT	aid-mode	gültig bis zur Eingabe eines neuen %AINT zur selben Basisqualifikation oder bis /EXIT-JOB oder bis Schließen der zugehörigen Dump-Datei.
%BASE	basis	gültig bis zur Eingabe eines neuen %BASE oder bis /EXIT-JOB oder bis zum Schließen der als basis vereinbarten Dump-Datei.
%CONTROLn	kriterium/ control-bereich	kann vom nächsten %CONTROLn übernommen werden, ansonsten gültig bis zum Löschen des %CONTROLn oder Programmende.
	subkdo	muss stets angegeben werden.
%DISASSEMBLE	anzahl	kann von einem neuen %DISASSEMBLE übernommen werden; diese Möglichkeit besteht bis Programmende.
	start	Die an den zuletzt rückübersetzten Befehl anschließende Adresse kann als start-Wert übernommen werden; diese Möglichkeit besteht bis Programmende.
%DISPLAY	alle	Alle Operanden müssen stets angegeben werden.
%DUMPFILE	link=datei	Die über link zugewiesene Datei bleibt geöffnet bis /EXIT-JOB, falls sie nicht explizit geschlossen wird.
%FIND	suchbegriff	kann aus einem vorherigen %FIND übernommen werden, bis find-bereich vollends durchsucht ist. Diese Möglichkeit besteht bis /EXIT-JOB.
	find-bereich/ alignment	Ist kein suchbegriff angegeben, wird find-bereich/ alignment aus dem vorherigen %FIND übernommen, bis find-bereich vollends durchsucht ist.
%INSERT	testpunkt	Der Testpunkt bleibt eingetragen, bis er mit %REMOVE gelöscht wurde, bis alle %INSERTs gelöscht wurden oder Programmende. Bei Programmen, die als Overlays gebunden sind, bleibt der Testpunkt im Modul eingetragen, in dem er gesetzt wurde, auch wenn an gleicher Stelle inzwischen ein anderer Modul geladen ist.
	subkdo	Das Subkommando bleibt eingetragen, bis es mit %REMOVE gelöscht wird, bis der zugehörige Testpunkt gelöscht wird oder bis Programmende.
%MOVE	alle	Alle Operanden müssen stets angegeben werden.

Tabelle 1: Übersicht über die Geltungsdauer der Kommandos

Kommando	Operand	Gültigkeitsdauer
%ON	ereignis/ write-ereignis	Das Ereignis bleibt eingetragen, bis es mit %REMOVE gelöscht wurde, bis alle %ON gelöscht werden oder bis Programmende. Eine Ausnahme bildet %ON %WRITE: ein neues write-Ereignis überschreibt ein schon eingetragenes Ereignis.
	subkdo	Das Subkommando bleibt eingetragen, bis es mit %REMOVE gelöscht wird, bis das zugehörige ereignis gelöscht wird, bis alle %ON gelöscht werden oder bis Programmende.
%OUT	alle	gültig bis zur Eingabe eines neuen %OUT mit entsprechendem Operanden oder bis /EXIT-JOB.
%OUTFILE	link=datei	Wenn Datei nicht explizit geschlossen wird, bleibt sie geöffnet bis /EXIT-JOB.
%QUALIFY	vorqualifikation	Vorqualifikation gilt, bis sie durch einen neuen %QUALIFY überschrieben wird, durch einen %QUALIFY ohne Operanden aufgehoben wird oder bis /EXIT-JOB.
%REMOVE	ziel	ziel muss stets angegeben werden.
%SDUMP	alle	Es kann nichts aus einem vorherigen %SDUMP übernommen werden.
%SET	alle	Alle Operanden müssen stets angegeben werden.
%SHOW	info-ziel	Es kann nichts aus einem vorherigen %SHOW übernommen werden.
%SYMLIB	qual-u-lib	Eine Bibliothek bleibt angemeldet bis zum nächsten %SYMLIB zur selben Basisqualifikation, bis zum nächsten %SYMLIB ohne Operanden, bis /EXIT-JOB oder bis zum Schließen der zugehörigen Dump-Datei.
%TITLE	seitenkopf	gültig, bis zum nächsten %TITLE oder Programmende.
%TRACE	alle	Alle Operanden gelten solange, bis sie durch entsprechende Angaben in einem neuen %TRACE überschrieben werden oder bis Programmende.trace-bereich wird nicht übernommen, wenn ein %TRACE ohne Trace-Bereich eingegeben wird und die Unterbrechungsstelle nicht im Trace-Bereich liegt.

Tabelle 1: Übersicht über die Geltungsdauer der Kommandos

4 Voraussetzungen zum Testen mit AID

Beim Testen mit AID wird unterschieden zwischen dem symbolischen Testen, wobei die im Quellprogramm vergebenen Namen zur Adressierung verwendet werden und dem maschinennahen Testen, wenn mit virtuellen Adressen gearbeitet wird. Um symbolisch testen zu können, müssen Sie beim Übersetzen den Compiler veranlassen, LSD-Sätze zu generieren. Beim Binden und Laden können Sie über Operanden steuern, dass die LSD-Sätze miteingebunden bzw. mitgeladen werden. Wenn jedoch die LSD-Sätze beim Binden und Laden nicht berücksichtigt wurden, können Sie sie von AID aus einer PLAM-Bibliothek nachladen lassen.
Zum Testen auf Maschinencode-Ebene bedarf es dagegen keiner besonderen Maßnahmen.

Wenn CSECTs mit der LMS-Anweisung MODIFY-ELEMENT (Subanweisung RENAME-SYMBOLS) oder mit der TSOSLNK-Anweisung RENAME umbenannt werden, kann nicht mehr symbolisch getestet werden. Wurde ein LLM vom Compiler direkt erzeugt und enthält der LLM die LSD-Sätze, können Sie CSECTs mit BINDER umbenennen (Anweisung MODIFY-MODULE-ATTRIBUTES). Für welche Compiler-Versionen diese Möglichkeit besteht, können Sie im Benutzerhandbuch des jeweiligen Compilers nachlesen.
Das maschinennahe Testen wird durch das Umbenennen von CSECTs nicht beeinträchtigt. Die CSECTs Ihres Programms können Sie mit den neuen Namen ansprechen.

4.1 Testen auf Maschinencode-Ebene

Für das Testen im Maschinencode brauchen Sie beim Übersetzen, Binden und Laden keine speziellen Operanden angeben, um später alle Funktionen ausnutzen zu können, wie sie im Handbuch für das „Testen auf Maschinencode-Ebene“ [1] beschrieben sind.

Beim Binden wird standardmäßig die Objekt-Strukturliste aus dem Verzeichnis der Externbezüge (siehe „Bindelader-Starter in BS2000/OSD“ [15]) erzeugt. Dies ist beim BINDER das ESV (External Symbols Vector) und beim TSOSLNK das ESD (External Symbolic Dictionary).

Keine Objekt-Strukturliste wird jedoch erzeugt, wenn Sie beim Binden eines Programms mit BINDER in der Anweisung SAVE-LLM den Operanden SYMBOL-DICTIONARY=NO schreiben, bzw. beim Binden mit TSOSLNK den PROGRAM-Operanden SYMTEST=NO angeben. Dann können Sie die folgenden Funktionen nicht ausführen:

- Ausgeben einer Liste aller CSECTs und COMMONs des Benutzerprogramms (%D %SORTEDMAP oder %D %MAP)
- Ausgeben der Lokalisierungsinformationen einer Speicherreferenz (%D %LOC(speicherref))
- Angeben einer CSECT-/COMMON-Qualifikation in einer Speicherreferenz
- Ablaufüberwachung durch die Kommandos %CONTROL*n* und %TRACE, wenn sie implizit oder explizit auf eine CSECT beschränkt sein sollen
- Erzeugen von REPs für Änderungen

Außerdem kann AID in diesem Fall bei %TRACE, %DISASSEMBLE, %FIND und in der STOP-Meldung keine CSECT-/COMMON-relativen Adressen ausgeben.

Vorsicht bei LLMs oder Kontexten, die gleichnamige CSECTs enthalten: in diesem Fall ist nicht vorhersehbar, welche CSECT mit AID angesprochen wird.

4.2 Symbolisches Testen

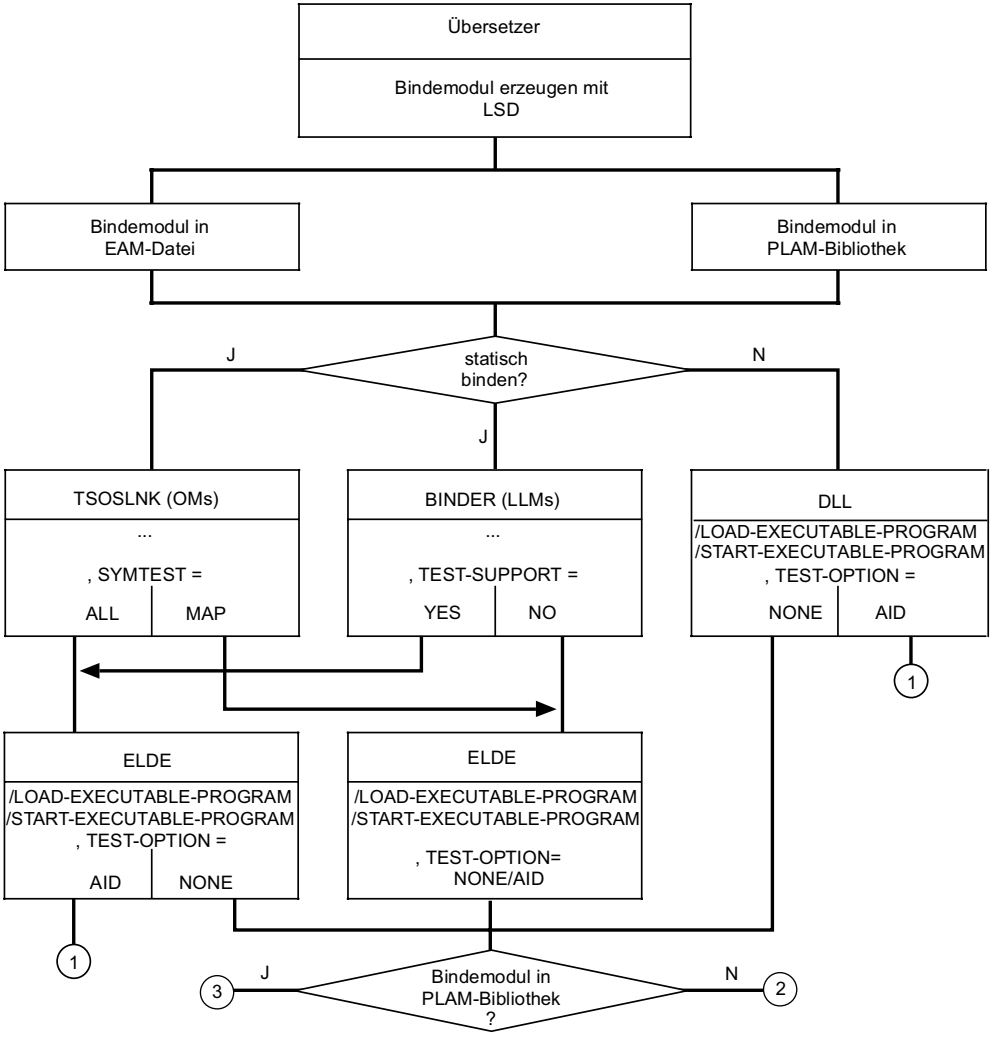
Beim symbolischen Testen mit AID können Sie Daten mit den von Ihnen definierten Namen aus dem Quellprogramm ansprechen oder sich auf Anweisungen durch die Angabe von Anweisungsnamen oder Source-Referenzen beziehen. Dafür benötigt AID Informationen über die verwendeten Namen aus den Quellprogrammen, aus denen das zu testende Programm entstanden ist. Diese Information besteht aus zwei Teilen:

1. LSD (List for Symbolic Debugging), Verzeichnis der im Modul definierten Namen und Source-Referenzen.
2. ESD (External Symbolic Dictionary), Verzeichnis der Externbezüge eines Moduls beim Binden mit TSOSLNK bzw. ESV (External Symbols Vector) beim Binden mit BINDER.

In den folgenden Abschnitten werden die verschiedenen Möglichkeiten, ESD/ESV- und LSD-Sätze zu behandeln, für jeden der folgenden Schritte im Programm-Entwicklungsprozess beschrieben:
– Übersetzen des Quellprogramms
– Binden und Laden mit DBL (DLL bis BS2000 V9.5) oder
– Binden mit TSOSLNK/BINDER und
  Laden mit ELDE

Außerdem bietet AID noch das Kommando %SYMLIB, mit dem PLAM-Bibliotheken (siehe Handbuch „LMS (BS2000/OSD)“ [12]) geöffnet werden, aus denen AID bei Bedarf die fehlenden LSD-Sätze nachlädt.

Die folgende Graphik gibt einen Überblick über die Möglichkeiten, die vom Compiler erzeugten LSD-Sätze beim Binden und Laden mitzuführen oder nicht.



(1) Das Programm kann uneingeschränkt symbolisch getestet werden.
(2) Das Programm kann nur eingeschränkt symbolisch getestet werden, d.h. Namen von Programmteilen können angesprochen und Aufrufhierarchien rückverfolgt werden.
(3) Das Programm kann erst symbolisch getestet werden, wenn mit dem Kommando %SYMLIB die PLAM-Bibliothek mit den OMs oder LLMs zugewiesen wurde.

Es gibt also verschiedene Wege, wie AID die LSD-Informationen erhält. Voraussetzung ist immer, dass beim Übersetzen die LSD-Sätze an den erzeugten Bindemodul(OM) oder Bindelademodul (LLM) übergeben werden. Speichern Sie den OM oder LLM in einer PLAM-Bibliothek ab, können Sie die LSD-Sätze entweder beim Binden und Laden einbeziehen oder sie erst bei Bedarf durch AID nachladen lassen.

Das Nachladen der LSD-Sätze ist besonders für Programme sinnvoll, die aus mehreren Übersetzungsläufen entstanden sind und von denen nur einzelne Module symbolisch getestet werden sollen. Die nachzuladende LSD muss im selben Übersetzungslauf erzeugt worden sein wie der Modul.

Aus der temporären Bindemodul-Datei (*OMF-Datei) kann AID nicht nachladen.

4.2.1 Übersetzen

Über eine Compiler-Option steuern Sie die Erzeugung der LSD-Sätze durch den Compiler. Die genaue Beschreibung der Operanden finden Sie in den sprachspezifischen Handbüchern zu AID (siehe [2] - [6]). Prinzipiell wird zwischen zwei Möglichkeiten unterschieden:

– Es werden nur ESD/ESV-Sätze, aber keine LSD-Sätze erzeugt (Standardwert). Das Programm kann nur maschinennah getestet werden.
– Vom Compiler werden ESD/ESV- und LSD-Sätze erzeugt. Das Programm kann mit AID symbolisch getestet werden.

4.2.2 Binden mit BINDER

Beim Binden mit BINDER können bei allen Bindevorgängen LSD-Sätze miteingebunden werden.
In den BINDER-Anweisungen, die das Anlegen, Ändern oder Abspeichern eines LLM steuern, bestimmt der Operand TEST-SUPPORT, ob die LSD-Sätze aus Bindelademodulen (LLMs) miteingebunden werden oder nicht (siehe Handbuch „Binder in BS2000/OSD“ [14]). Anweisungen, die die gleichen Operandenwerte verlangen, werden in der folgenden Tabelle zusammengefasst und mit der Syntax des TEST-SUPPORT-Operanden aufgeführt. Die Beschreibung der Operandenwerte schließt sich an die Tabelle der Anweisungen und der zugehörigen TEST-SUPPORT-Syntax an.

Anweisung	Bedeutung	TEST-SUPPORT-Operandenwerte
START-LLM-CREATION	Erzeugen eines LLM	TEST-SUPPORT= { *YES / *NO }
START-LLM-UPDATE MODIFY-LLM-ATTRIBUTES	Ändern eines LLM Ändern der physischen Struktur eines LLM	TEST-SUPPORT= { *UNCHANGED / *YES / *NO }
MODIFY-MODULE-ATTRIBUTES	Ändern der logischen Struktur eines LLM	TEST-SUPPORT= { *UNCHANGED / *INCLUSION-DEFAULT / *YES / *NO }
SAVE-LLM	Speichern eines LLM	TEST-SUPPORT= { *LAST-SAVE / *YES / *NO }
INCLUDE-MODULES REPLACE-MODULES RESOLVE-BY-AUTO-LINK	Einfügen von Modulen Ersetzen von Modulen Befriedigen von Externverweisen durch Autolink	TEST-SUPPORT= { *INCLUSION-DEFAULT / *YES / *NO }

*YES Die LSD-Sätze werden übernommen. Der Binder überprüft jedoch nicht, ob der verarbeitete Bindemodul (OM) oder Bindelademodul (LLM) tatsächlich LSD-Sätze enthält.

*NO Die LSD-Sätze werden nicht übernommen. Falls in der SAVE-LLM-Anweisung jedoch SYMBOL-DICTIONARY=YES gesetzt wurde, d.h. das ESV wurde mitgeführt, können Aufrufhierarchien rückverfolgt werden. Sind zusätzlich bei der Übersetzung LSD-Sätze erzeugt und mit dem OM oder LLM in eine PLAM-Bibliothek geschrieben worden, können Sie zum symbolischen Testen bei Bedarf die LSD nachladen.

*UNCHANGED Die LSD-Sätze werden, falls in den verarbeiteten Modulen vorhanden, in den aktuellen LLM übernommen.

*INCLUSION-DEFAULT
: Es werden die Werte des Operanden INCLUSION-DEFAULTS aus den Anweisungen START-LLM-CREATION, START-LLM-UPDATE, oder MODIFY-LLM-ATTRIBUTES desselben Edit-Laufs übernommen.

*LAST-SAVE
: Der Binder übernimmt die Werte aus der letzten SAVE-LLM-Anweisung desselben Edit-Laufs. Wurde bisher noch kein SAVE-LLM angegeben, setzt der Binder YES ein.

Der BINDER erlaubt, gleichnamige CSECTs mehrfach in einen LLM einzubinden. Beim Testen mit AID führt dies jedoch zu unvorhersehbaren Ergebnissen.

4.2.3 Binden und Laden mit DBL oder Laden mit ELDE

Ein Programm, das getestet werden soll, wird mit dem BS2000-Kommando LOAD-EXECUTABLE-PROGRAM aufgerufen, damit anschließend die AID-Kommandos eingegeben werden können. Ein Programm, das nur bei Auftreten eines Fehlers noch mit AID bearbeitet werden soll, kann mit START-EXECUTABLE-PROGRAM geladen und gestartet werden. Auch eine mit dem Makroaufruf BIND eingebundene Ladeeinheit kann mit AID getestet werden.

Ein Programm in Form von Bindemodulen (OMs) oder Bindelademodulen (LLMs) wird durch den dynamischen Bindelader DBL, ein mit TSOSLNK gebundenes Programm (Ladeeinheit) durch den Lader ELDE geladen (siehe Handbuch „Bindelader-Starter in BS2000/OSD“ [15]).

– Laden bzw. Laden und Starten mit dem DBL aufgerufen durch SDF-Kommandos:

```
/LOAD-EXECUTABLE-PROGRAM  }                          { NONE }
                          } ,...,TEST-OPTIONS = {    {      }
/START-EXECUTABLE-PROGRAM }                          { AID  }
```

NONE
: Das Programm wird ohne LSD-Sätze geladen. Symbolisches Testen ist nur möglich, wenn AID zum Nachladen der LSD-Sätze die PLAM-Bibliothek zur Verfügung gestellt wird, in der die zugehörigen OMs oder LLMs stehen.

AID
: Das Programm wird mit den LSD-Sätzen geladen. Es wird auch dann geladen, wenn es keine LSD-Sätze enthält. Wenn gleichzeitig DBL-PARAMETERS:LOADING angegeben wurde, ist darauf zu achten, dass der zugehörige Operand LOAD-INFORMATION, der das Laden des ESV steuert, auf DEFINITIONS (Standardwert) oder auf REFERENCES gesetzt ist.

– Einbinden einer weiteren Ladeeinheit mit dem DBL über den Makroaufruf BIND:

```
BIND       .....,TSTOPT = ⎧ N[ONE] ⎫
                          ⎨       ⎬
                          ⎩ A[ID] ⎭
```

NONE Bedeutung wie oben

AID Das Programm wird mit den LSD-Sätzen geladen. Es wird auch dann geladen, wenn es keine LSD-Sätze enthält. Gleichzeitig muss der Operand LDINFO auf DEF oder REF gesetzt sein, damit das ESV geladen wird.

– Laden bzw. Laden und Starten mit ELDE aufgerufen durch SDF-Kommandos:

```
⎧/LOAD-EXECUTABLE-PROGRAM  ⎫                    ⎧ NONE  ⎫
⎨                          ⎬ .....,TEST-OPTIONS = ⎨       ⎬
⎩/START-EXECUTABLE-PROGRAM ⎭                    ⎩ AID   ⎭
```

NONE Das Programm wird ohne LSD-Sätze geladen. Symbolisches Testen ist nur möglich, wenn AID zum Nachladen der LSD-Sätze die PLAM-Bibliothek zur Verfügung gestellt wird, in der die zugehörigen OMs stehen.

AID Das Programm wird mit den LSD-Sätzen geladen. Es wird auch dann geladen, wenn es keine LSD-Sätze enthält.

Beispiele

1. `/LOAD-EXECUTABLE-PROGRAM FROM-FILE=*OMF,TEST-OPTIONS=AID`

   Aus der *OMF-Datei lädt der dynamische Bindelader einen Bindemodul mit LSD-Sätzen.

2. `/LOAD-EXECUTABLE-PROGRAM FROM-FILE=IDEAL,TEST-OPTIONS=AID`

   Das gebundene Programm IDEAL wird mit den LSD-Sätzen durch ELDE geladen.

3. `/LOAD-EXECUTABLE-PROGRAM FROM-FILE=*LIBRARY-ELEMENT(LIBRARY=`
   `PROGRAMLIB,ELEMENT-OR-SYMBOL=ROOTMOD)`

   Aus der PLAM-Bibliothek PROGRAMLIB wird das gebundene Programm ROOTMOD ohne LSD-Sätze geladen.

Die Beispiele gelten entsprechend für das /START-EXECUTABLE-PROGRAM-Kommando.

4.2.4 Nachladen von LSD-Sätzen durch AID

AID kann aus PLAM-Bibliotheken LSD-Sätze für ein Programm nachladen, wenn in der Bibliothek die zugehörigen OMs oder LLMs mit den LSD-Sätzen stehen. Mit dem Kommando %SYMLIB veranlassen Sie AID, die angegebene Bibliothek zu öffnen. Stellt AID bei der Bearbeitung eines Kommandos mit symbolischen Operanden fest, dass die zugehörigen LSD-Sätze nicht im Speicher zur Verfügung stehen, greift es auf mit %SYMLIB zugewiesene und geöffnete Bibliotheken zurück. AID überprüft, ob die nachgeladenen LSD-Sätze aus derselben Übersetzung stammen wie der Modul, zu dem sie nachgeladen werden.

Ist keine Bibliothek zugewiesen oder enthalten die zugewiesenen Bibliotheken nicht den gesuchten OM oder LLM oder sind darin keine LSD-Sätze enthalten, meldet AID, dass die LSD-Sätze fehlen. Die erforderliche Bibliothek können Sie mit einem neuen %SYMLIB zuweisen. Wiederholen Sie dann das AID-Kommando, für das die LSD-Sätze fehlten, kann es von AID bearbeitet werden.

Bei LLMs ist es möglich, darin enthaltene CSECTs mit der BINDER-Anweisung MODIFY-SYMBOL-VISIBILITY zu maskieren. Zu solchen CSECTs kann AID keine LSD-Informationen nachladen. Das gleiche gilt für CSECTs, für die der Operand RUN-TIME-VISIBILITY auf YES gesetzt wurde, da dies die Maskierung der CSECT einschließt. Dieser Operand kann bei den folgenden BINDER-Anweisungen angegeben werden:

- INCLUDE-MODULES
- MODIFY-MODULE-ATTRIBUTES
- REPLACE-MODULES
- RESOLVE-BY-AUTOLINK

Programme, die maskierte CSECTs enthalten, können mit AID nur dann symbolisch getestet werden, wenn die LSD gleich zusammen mit dem Programm geladen wird. Nachladen ist nur möglich, wenn in einem gesonderten BINDER-Lauf die Maskierungen zurückgesetzt wurden.

Es wird darauf hingewiesen, dass AID die LSD-Suche in einem LLM abbricht, wenn es die erste CSECT des gesuchten Namens findet, selbst wenn die LSD zu dieser CSECT inkonsistent ist, weil z.B. die LSD nicht aus der gleichen Übersetzung hervorgegangen ist wie die CSECT. Es nützt also nichts, wenn im selben LLM eine weitere CSECT gleichen Namens mit konsistenter LSD enthalten ist.

Beispiele

1. `/LOAD-EXECUTABLE-PROGRAM FROM-FILE=*LIBRARY-ELEMENT(LIBRARY=`
   `PROGRAMLIB,ELEMENT-OR-SYMBOL=ROOTMOD)`

   Das gebundene FORTRAN-Programm ROOTMOD wird ohne LSD-Sätze aus der PLAM-Bibliothek PROGRAMLIB geladen.

   Wenn Sie nun das folgende AID-Kommando eingeben, mit dem Sie die Anweisung mit der Anweisungsmarke 10 ansprechen:
   `%INSERT L'10'`

   erhalten Sie die Fehlermeldung:

   ```
   AID0378   Symbolinformation fehlt
   ```

   Enthält die Bibliothek PROGRAMLIB den zum Programm gehörenden Bindemodul mit den LSD-Sätzen, können Sie mit dem Kommando

   ```
   %SYMLIB   PROGRAMLIB
   ```

   die erforderliche PLAM-Bibliothek zuweisen und dann das %INSERT-Kommando wiederholen, das nun von AID bearbeitet werden kann. Bindemodul und Ladeeinheit können in verschiedenen Bibliotheken stehen, aber die Ladeeinheit muss aus der Bindemodul-Version gebunden worden sein, aus der die LSD-Sätze nachgeladen werden. Im Fehlerfall gibt AID die folgende Meldung aus:

   ```
   AID0377   Inkonsistente Symbolinformation fuer (&00)
             &00 = Programmname
   ```

2. `%SYMLIB E=D1.BINDEMODUL.LIB1,E=D1.BINDEMODUL.LIB2`

   Für die Dump-Datei mit dem Linknamen D1 stehen die zwei PLAM-Bibliotheken BINDEMODUL.LIB1 und BINDEMODUL.LIB2 zum Nachladen der LSD-Sätze zur Verfügung.

3. ```
   %QUALIFY E=D2
   %SYMLIB E=D3.BIB1,.BIB2,BIB3
   ```

   Für die Dump-Datei mit dem Linknamen D3 wird die PLAM-Bibliothek BIB1 angemeldet und geöffnet.
   Für die Dump-Datei mit dem Linknamen D2 wird die Bibliothek BIB2 angemeldet und geöffnet.

   Die PLAM-Bibliothek BIB3 wird für den aktuellen AID-Arbeitsbereich angemeldet und geöffnet.

   Wurde bisher kein %BASE-Kommando eingegeben, ist der aktuelle AID-Arbeitsbereich immer der virtuelle Speicherbereich des geladenen Programms. Nach einem %BASE-Kommando ist der AID-Arbeitsbereich der mit %BASE bezeichnete Bereich.
   ```

5 Kommandoeingabe

5.1 Kommandoformat

Jedes AID-Kommando beginnt mit dem Prozentzeichen "%", unmittelbar gefolgt vom Kommandonamen.
Nach mindestens einem Leerzeichen können Operanden folgen.
Folgen Operanden und/oder Schlüsselwörter unmittelbar aufeinander ohne ein vorgegebenes Trennzeichen, dann muss mindestens ein Leerzeichen eingefügt werden.
Die Operanden müssen Sie in der Reihenfolge angeben, in der diese im Format aufgeführt sind.

Namen für Kommandos

Ein AID-Kommando kann wie ein BS2000-Kommando mit einem Namen versehen werden. Der Name besteht aus bis zu 255 Zeichen:
– 1. Zeichen: A-Z, $, # oder @
– alle folgenden Zeichen: A-Z, 0-9, $, #, @ oder -,
wobei der Bindestrich "-" nicht letztes Zeichen des Namens sein darf.

Namen, die mit SKIP-COMMANDS angesprungen werden, beginnen mit einem Punkt; Namen aus S-Prozeduren, zu denen mit GOTO verzweigt wird, werden mit einem Doppelpunkt abgeschlossen.
Der Name folgt auf den Schrägstrich, der im Dialog vom System vorgegeben oder in Prozedurdateien von Ihnen eingegeben wird. Zwischen dem Namen und dem %-Zeichen des AID-Kommandos muss mindestens ein Leerzeichen stehen.
Beispiel: `/.START %AID CHECK=NO`

Die BS2000-Kommandonamen dienen als Sprungziele in Prozeduren; für das Testen mit AID sind sie jedoch ohne Bedeutung.

Fortsetzung von Eingabezeilen

Wenn Sie ein AID-Kommando in die nächste Zeile fortsetzen müssen, so gilt dafür derselbe Fortsetzungsmechanismus, wie er auch für BS2000-Kommandos vorgesehen ist. Im Dialog können Sie eine Eingabe einfach über mehrere Zeilen hinwegschreiben; Sie haben aber auch die Möglichkeit, die Zeile mit einem Bindestrich abzuschließen und abzusenden. Dann beginnt die Folgezeile nach dem vom System gesendeten Schrägstrich.

In Prozedurdateien muss die Folgezeile durch einen Bindestrich angekündigt werden, dem nur noch Leerzeichen bis zum Zeilenende folgen dürfen.
Die Folgezeile muss mit einem Schrägstrich beginnen.

Ein AID-Kommandos darf nicht länger als 1000 Zeichen sein. Da im Speicher nur ein begrenzter Bereich für die Interpretation eines Kommandos zur Verfügung steht, ist die Anzahl der Operanden in einem Kommando beschränkt. Wieviele Operanden jeweils angegeben werden können, finden Sie in der Beschreibung der einzelnen Kommandos.

Verwendung von Leerzeichen und Kommentaren

Um Ihre AID-Kommandos übersichtlicher und verständlicher zu gestalten, können Sie Leerzeichen und Kommentare einsetzen. Kommentare müssen wie in der BS2000-Kommandosprache zwischen Anführungszeichen (") gesetzt werden. Leerzeichen und Kommentare können Sie überall dort einfügen, wo eins der folgenden Zeichen steht:

```
␣          Leerzeichen
.          Punkt
,          Komma
=          Gleichheitszeichen
'...'      Hochkomma
(...)      öffnende und schließende runde Klammern
<...>      öffnende und schließende spitze Klammern
[...]      öffnende und schließende eckige Klammern
;          Semikolon
+ - * /    arithmetische Operatoren
->         Pointer-Operator
```

Bei der Verwendung des Minuszeichens/Bindestrichs "-" ist die Voreinstellung des Operanden *SYMCHARS* im Kommando %AID zu berücksichtigen.

Beispiel

```
%CONTROL1      %CALL        "SORT-AUFRUF"       <%DISPLAY 'CALL'; %STOP>
```

5.2 Einzelkommandos

AID-Kommandos können Sie im BS2000-Kommandomodus eingeben oder auch über die CMD-Makro-Schnittstelle aufrufen. AID-Kommandos werden wie BS2000-Kommandos zuerst vom BS2000 übernommen, anhand des %-Zeichens als AID-Kommandos identifiziert und dann an AID übergeben. Stellt der BS2000-Kommandointerpreter bei der Eingabe des AID-Kommandos fest, dass das Kommando zu lang ist, wird es mit einer Fehlermeldung abgewiesen, und Sie können das korrigierte Kommando erneut eingeben.

AID überprüft Syntax und Semantik eines Kommandos und stellt fest, ob die Operandenwerte in der aktuellen Testsituation bearbeitet werden können. Es führt z.B. zu einer Fehlermeldung, wenn Sie eine symbolische Adresse ansprechen wollen, die in den verfügbaren LSD-Sätzen (siehe Abschnitt „Grundlegende Begriffe“ auf Seite 16) nicht verzeichnet ist.

Ist ein Kommando syntaktisch falsch, schreibt AID eine entsprechende Fehlermeldung und markiert die Stelle, an der es den Fehler entdeckt hat. Anschließend können Sie das korrigierte Kommando erneut eingeben.

Hat AID ein Kommando übernommen und ausgeführt, dann hängt es vom Typ des Kommandos ab, ob das Programm gestartet wird oder ob Sie weitere Kommandos eingeben können.

5.3 Kommandofolgen und Subkommandos

In Kommandofolgen können Sie mehrere AID- und BS2000-Kommandos zusammenfassen. Aufeinanderfolgende Kommandos müssen Sie durch Semikolon trennen. Auch eine Kommandofolge darf nicht länger sein als 1000 Zeichen.
Kommandofolgen werden unmittelbar ausgeführt. Sie werden von links nach rechts abgearbeitet.

Alle Kommandos einer Kommandofolge, die mit % beginnen, werden von AID als AID-Kommandos identifiziert und sofort auf Fehler untersucht. Erkennt AID einen Syntaxfehler, so wird die gesamte Kommandofolge schon bei der Eingabe abgewiesen. Kommandos ohne führendes Prozentzeichen interpretiert AID als BS2000-Kommandos und übernimmt sie ohne weitere Prüfung. Fehlerhafte oder unzulässige BS2000-Kommandos werden daher erst bei der Ausführung erkannt und führen zum Abbruch der Kommandofolge. Auch bei schwerwiegenden Fehlern in AID-Kommandos, wie z.B. Adressenüberlauf, wird die Bearbeitung der Kommandofolge abgebrochen. Danach befinden Sie sich wieder im Kommandomodus und können weitere Kommandos eingeben.

Kann ein AID-Kommando nicht ausgeführt werden, weil ein angegebener Name nicht in den LSD-Sätzen verzeichnet ist oder weil keine LSD-Sätze geladen sind, so gibt AID für dieses Kommando eine entsprechende Fehlermeldung aus und fährt in der Bearbeitung eventuell noch folgender Kommandos fort.
Da nach bestimmten Fehlern die gesamte Kommandofolge erneut eingegeben werden muss, bietet sich die Verwendung längerer Kommandofolgen nur für ausgetestete Prozedurdateien an.

In Kommandofolgen sind alle BS2000-Kommandos zugelassen, die Sie auch im CMD-Makro (siehe Handbuch „Makroaufrufe an den Ablaufteil“ [11]) angeben können und fast alle AID-Kommandos.

Nicht zugelassen sind dagegen die folgenden Kommandos:

AID-Kommandos: %AID, %ALIAS, %BASE, %DUMPFILE, %HELP, %OUT, %QUALIFY, %?

BS2000-Kommandos: Die Liste der Kommandos finden Sie im Anhang.

Auch SDF-P-Kontrollflusskommandos dürfen in Kommandofolgen nicht verwendet werden. Außerdem ist bei der Verwendung einiger BS2000-Kommandos zu berücksichtigen, dass diese zwar in Kommandofolgen erlaubt sind, dass sie aber ein geladenes Programm abbrechen und Sie danach das Programm nicht mehr mit AID-Kommandos bearbeiten können. Das gilt für die nachfolgenden BS2000-Kommandos (siehe „Handbuch Kommandos Band 1 - 5“ [8]):

Kommando	**Funktion**
CALL-PROCEDURE	Prozedur aufrufen
EXIT-JOB	Auftrag (Job) beenden
HELP-SDF	Anleitung zum Aufruf von SDF-Kommandos
LOAD-PROGRAM	Programm laden
LOAD-EXECUTABLE-PROGRAM	Programm laden
LOGOFF	Auftrag (Job) beenden
START-PROGRAM	Programm laden und starten
START-EXECUTABLE-PROGRAM	Programm laden und starten

Eine Liste der entsprechenden Kommandos im ISP-Format finden Sie im Anhang.

Das geladene Programm wird auch bei Aufruf aller mittels SDF-A definierten und durch Kommando-Prozeduren implementierten (benutzer)eigenen Kommandos beendet (siehe Handbuch „Makroaufrufe an den Ablaufteil“ [11]).

Die Kommandos %TRACE, %RESUME, %CONTINUE oder %STOP brechen eine Kommandofolge ab. Während Sie nach einem %STOP-Kommando wieder im Kommandomodus sind, wird durch die Kommandos %TRACE, %RESUME und %CONTINUE das Programm gestartet bzw. fortgesetzt. Diese Kommandos sind also nur als letztes Kommando einer Kommandofolge sinnvoll.

Ein Subkommando ist kein eigenständiges Kommando, sondern ein Operand der Überwachungskommandos %CONTROL*n*, %INSERT und %ON. Das Subkommando wird erst dann bearbeitet, wenn die Überwachungsbedingung eingetreten ist.

Von folgenden Besonderheiten abgesehen gelten für den Kommandoteil der Subkommandos dieselben Regeln wie für Kommandofolgen:

– Das Überwachungskommando mit dem Subkommando zusammen darf nicht länger sein als 1000 Zeichen.
– Zusätzlich zu %CONTINUE, %RESUME, %STOP und %TRACE ist auch ein %REMOVE auf das gerade ausgeführte Subkommando nur als letztes Kommando sinnvoll, da nachfolgende Kommandos des Subkommandos nicht mehr ausgeführt werden.
– Im Subkommando eines %CONTROL*n* dürfen Sie kein weiteres %CONTROL*n*-Kommando, keinen %INSERT, %JUMP (COBOL, FOR1) oder %ON angeben.

Beispiele

1. `%INSERT S'20' <%DISPLAY A,B;%SET A INTO B;SET-FILE-LINK...;%REM %•>`

   Wenn der Programmablauf an die Anweisung mit der Nummer 20 kommt, gibt AID den Inhalt der Variablen A und B aus, weist der Variablen B den Wert von A zu und ruft das SDF-Kommando SET-FILE-LINK auf. Da das Subkommando auch ein %REMOVE %• enthält, wird es nach seiner Ausführung gleich wieder gelöscht.

2. `%ON %LPOV <%DISPLAY %LINK>`

   Immer wenn während des Programmablaufs ein Modul nachgeladen wird, gibt AID seinen Namen aus. Der Programmablauf wird fortgesetzt.

5.4 Kommandodateien

AID-Kommandos können auch in BS2000-Prozedurdateien stehen. Soll ein Eingabesatz mit einem AID-Kommando beginnen, so muss als erstes Zeichen ein Schrägstrich stehen, dem dann das %-Zeichen des AID-Kommandos folgt. Eine Marke für /SKIP-COMMANDS bzw. /GOTO muss jedoch noch vor dem %-Zeichen stehen.

Steht in einer BS2000-Prozedur ein AID-Kommando, das eine Quittung verlangt (z.B. /%AID CHECK=ALL.../%SET...), so setzt AID im Stapelbetrieb als Antwort Y ein.

Beispiel

```
/LOAD-EXECUTABLE-PROGRAM FROM-FILE=*LIB-ELEM(LIB=TESTLIB,LIB-OR-SYM=TESTLLM),
                                                                    TEST-OPT=AID
/ANF: %SET 17 INTO SLF
/%INSERT S'71'  <%DISPLAY I,J,K>
.
.
.
/GOTO ANF
/END: EXIT-PROC
```

6 Subkommando

6.1 Beschreibung

Ein Subkommando ist ein Operand der Überwachungskommandos %CONTROL*n*, %INSERT und %ON. Mit diesen Kommandos legen Sie eine Überwachungsbedingung fest, die erfüllt sein muss, damit das zugehörige Subkommando bearbeitet wird. Sie haben also die Möglichkeit, steuernd in den Testablauf einzugreifen. Sie können automatisierte Testabläufe aufbauen, in denen an vorbestimmten Stellen im Programmablauf z.B. aktuelle Datenstände in Protokoll-Dateien ausgegeben oder Datenfeld- oder Registerinhalte modifiziert werden.

```
subkdo-OPERAND  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

                                    ⎧AID-kommando    ⎫
<[subkdoname:] [(bedingung):] [     ⎨                ⎬ {;...}]>
                                    ⎩BS2000-kommando ⎭

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Über den Subkommandonamen können Sie im weiteren Testverlauf das Subkommando ansprechen, um sich z.B. die Anzahl der Durchläufe des Subkommandos ausgeben zu lassen oder um das Subkommando wieder zu löschen. Die Ausführung des Subkommandos kann von einer Bedingung abhängen, die zwischen dem Subkommandonamen und dem Kommandoteil stehen muss. Der Kommandoteil kann aus einem einzelnen Kommando oder aus einer Kommandofolge bestehen und AID- und BS2000-Kommandos enthalten (siehe Abschnitt „Kommandofolgen und Subkommandos“ auf Seite 43).

Geben Sie in einem %INSERT, %CONTROL*n* oder %ON kein Subkommando an, setzt AID das Subkommando <%STOP> ein.
Wenn Sie jedoch *subkdoname* oder *bedingung* angeben und den Kommandoteil weglassen, ergänzt AID kein %STOP, sondern verhält sich am Testpunkt bzw. beim Eintreten des vereinbarten Ereignisses so, als ob ein %CONTINUE ergänzt würde:

– der Durchlaufzähler wird erhöht und kann gegebenenfalls über %•*subkdoname* abgefragt werden,
– das Programm läuft weiter,
– ein %TRACE wird fortgesetzt.

Adressoperanden im Kommandoteil eines Subkommandos, die keine vollständige explizite Qualifikation enthalten, werden bei der Eingabe entsprechend den aktuellen Vereinbarungen zur Basisqualifikation (siehe %BASE) bzw. zur *vorqualifikation* (siehe %QUALIFY) ergänzt.
Bei der Eingabe wird ein Subkommando nur syntaktisch geprüft. Ob die angegebenen symbolischen Adressen in den LSD-Sätzen verzeichnet sind oder ob zu einem Programmteil, der über eine Qualifikation angesprochen wird, überhaupt LSD-Sätze geladen sind, prüft AID erst bei der Ausführung des Subkommandos. Die entsprechenden LSD-Sätze müssen also bei der Eingabe des Subkommandos noch nicht geladen sein. Ebenso wird bei der Verwendung von Qualifikationen bei der Eingabe des Subkommandos noch nicht geprüft, ob die damit bezeichneten Programmteile existieren oder geladen sind.

Die Subkommandos der Kommandos %INSERT oder %ON können Sie ketten, und zwar nach dem LIFO-Prinzip. Sie können also entsprechend dem Testverlauf durch spätere Eingaben Subkommandos modifizieren bzw. aktualisieren. Dazu finden Sie im Abschnitt „Ketten“ auf Seite 58 ausführliche Informationen. In den Subkommandos zu %INSERT und %ON können Sie weitere %INSERT- und %ON-Kommandos definieren. Dies ist im Abschnitt „Schachteln“ auf Seite 61 beschrieben. Es gibt Kommandos, die in Subkommandos nicht zugelassen sind und/oder die zum Abbruch des Subkommandos, des Programms oder gar der Task führen. Alle Informationen darüber finden Sie im Kapitel „Kommandoeingabe“ auf Seite 41. Dort steht auch, wie Fehler in Subkommandos behandelt werden.

Beispiele

1. `%CONTROL1 %STMT IN (S'20':S'27') <%DISPLAY A_ARR>`

   Jeweils vor der Ausführung der Anweisungen 20 bis 27 wird der Inhalt aller Elemente des Feldes A_ARR ausgegeben.

2. `%INSERT INPUT <%DISPLAY INDAT;%SET KEY INTO I-KEY>`

   Jedes Mal, wenn der Programmablauf den Paragraphen mit dem Namen INPUT erreicht, wird der Eingabesatz INDAT ausgegeben und der Inhalt des Schlüsselfeldes KEY nach I-KEY übertragen.

3. `%ON %LPOV <%SDUMP %NEST>`

   Nach jedem Laden eines neuen Segmentes wird die aktuelle Aufrufhierarchie ausgegeben.

4. `%INSERT V'2C' <%SET #'2C' INTO %5; %DISPLAY 'INS_2C!!!'>`

   Vor der Ausführung des Befehls mit der Adresse V’2C’ lädt AID das Register 5 mit dem Wert #’2C’ und gibt den Text "INS_2C!!!" aus.

5. `%ON %SVC <%C1 %INSTR <%R>; %C2 %INSTR <%REM %C>; %T 2 %INSTR>; %R`

   Alle SVCs ab der Eingabe des obigen %ON-Kommandos werden protokolliert. Zunächst wird mit %ON %SVC festgelegt, dass vor der Ausführung eines SVC das anschließende Subkommando ausgeführt wird. Das Subkommando enthält zwei %CONTROL-Kommandos und einen %TRACE, jeweils mit dem Kriterium %INSTR. Der %TRACE führt den nächsten Befehl aus, eben den SVC und protokolliert ihn. Die Ausführung des nächsten Befehls löst die Bearbeitung der beiden Subkommandos zu %CONTROL1 und %CONTROL2 aus: mit %RESUME wird der Programmablauf fortgesetzt, bis der nächste SVC erkannt wird; mit dem zweiten Subkommando (%REMOVE %CONTROL) wird der %CONTROL sofort wieder zurückgesetzt, da sonst zu jedem nachfolgenden Befehl ein %RESUME durchgeführt würde, was den Programmablauf stark verlangsamen würde.

6.2 Name und Durchlaufzähler

Der Subkommandoname besteht aus bis zu 30 Zeichen; das erste Zeichen kann A-Z, $, @ oder Unterstrich "_" sein. Die folgenden Zeichen können zusätzlich die Ziffern 0-9 und den Bindestrich "-" enthalten. Zu beachten ist, dass der Bindestrich am Ende einer Kommandozeile im Dialog stets als Fortsetzungszeichen interpretiert wird. Abgeschlossen wird der Subkommandoname mit einem Doppelpunkt, der jedoch nicht Teil des Namens ist.
Der Name muss nach der öffnenden spitzen Klammer stehen. Er muss eindeutig sein; gleiche Subkommandonamen weist AID mit einer Fehlermeldung ab. Bei geschachtelten Subkommandos prüft AID den Namen nicht bei der Eingabe, sondern erst, wenn das Subkommando ausgeführt wird. Insbesondere können innere Subkommandos aus einer Schachtelung, die mehrmals durchlaufen werden, nur dann einen Namen erhalten, wenn sie jedes Mal nach ihrer Ausführung explizit gelöscht werden.
Sie können bis zu 256 verschiedene Subkommandonamen vergeben.
Enthält das Subkommando ein %STOP, so wird der Subkommandoname in der STOP-Meldung mit ausgegeben.
Um den Subkommandonamen als allgemeinen AID-Operanden verwenden zu können, müssen Sie die Zeichen %• voran Stellen. So entsteht ein AID-Schlüsselwort %•*subkdoname*, mit dem Sie im weiteren Testverlauf das Subkommando und den zugehörigen Durchlaufzähler ansprechen können.

Auf alle Subkommandos, auch wenn sie keinen Namen haben, können Sie innerhalb des Subkommandos mit der Kurzform %• Bezug nehmen. Außerhalb des Subkommandos können Sie Durchlaufzähler und Subkommando nicht mit der Kurzform %• ansprechen.

Der Durchlaufzähler hängt unmittelbar mit dem Subkommandonamen zusammen, da er über diesen angesprochen werden kann. Aber auch für Subkommandos, die keinen Namen haben, führt AID einen Durchlaufzähler, der dann nur innerhalb des zugehörigen Subkommandos mit der Kurzform %• abgefragt werden kann.
Der Durchlaufzähler ist ein numerischer Wert, der bei jeder Bearbeitung des Subkommandos um eins erhöht wird. Der Zähler wird auch dann erhöht, wenn das Subkommando eine Bedingung enthält und das Ergebnis der Bedingung FALSE ist und deshalb der zugehörige Kommandoteil nicht ausgeführt wird. Sie können selbst mit %MOVE oder %SET den Stand des Durchlaufzählers verändern. Der Durchlaufzähler kann auch einen negativen Wert annehmen.
Den aktuellen Stand des Durchlaufzählers erfahren Sie mit %DISPLAY %•*subkdoname* bzw. mit %DISPLAY %• für das gerade auszuführende Subkommando.

Beispiele

1. `%CONTROL1 %IO <IO: %CONTINUE>`

   Bei jeder Ein-/Ausgabeoperation des Programms wird der Durchlaufzähler %•IO um 1 erhöht.

2. `%IN L'200' <L200: %DISPLAY %•IO; %STOP>`

   Das Programm hält an, wenn die Anweisungsmarke 200 erreicht ist. AID gibt den Stand des Durchlaufzählers aus, der zum Subkommando mit dem Namen %•IO aus Beispiel 1 gehört.

3. `%CONTROL2 %CALL <PAR: %D %•,PAR1,PAR2,PAR3 P=MAX>`

   Das obige Kommando überwacht die Unterprogramm-Aufrufe. Dem Durchlaufzähler können Sie entnehmen, um die wievielte CALL-Anweisung es sich handelt. Zusätzlich zum jeweiligen Stand des Durchlaufzählers werden die Parameter PAR1, PAR2 und PAR3 nach SYSLST protokolliert (P=MAX).
   Da das Subkommando den Namen PAR hat, können Sie sich jederzeit im weiteren Testverlauf mit %DISPLAY %•PAR ausgeben lassen, wie viele CALL-Aufrufe das Programm bis dahin durchlaufen hat.

6.3 Bedingte Ausführung

AID bietet Ihnen die Möglichkeit, die Ausführung eines Subkommandos von einer Bedingung abhängig zu machen. Die Bedingung muss in runde Klammern gesetzt und von einem Doppelpunkt abgeschlossen werden. Sie steht unmittelbar vor dem Kommandoteil des Subkommandos.
AID prüft die Bedingung und weist ihr den Wert TRUE oder FALSE zu. Nur bei TRUE wird der Kommandoteil ausgeführt, bei FALSE wird er übersprungen.

```
bedingung-OPERAND - - - - - - - - - - - - - - - - - - - - - - - - - - - -

                                 ⎧AND⎫
([NOT]vergleich1 [ ⎨OR ⎬[NOT]vergleich2] [...]):
                                 ⎩XOR⎭

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Für *vergleich*$_n$ können Operanden gemäß der folgenden Syntax gebildet und eingesetzt werden:

```
vergleich-OPERAND - - - - - - - - - - - - - - - - - - - - - - - - - - - -

⎧     ⎧datenname          ⎫⎫            ⎧     ⎧datenname          ⎫⎫
⎪     ⎪anweisungsname     ⎪⎪            ⎪     ⎪anweisungsname     ⎪⎪
⎪     ⎪S'...'             ⎪⎪            ⎪     ⎪S'...'             ⎪⎪
⎪     ⎪kompl-speicherref  ⎪⎪            ⎪     ⎪kompl-speicherref  ⎪⎪
⎪[qua]⎨                   ⎬⎪ ⎧EQ | NE⎫ ⎪[qua]⎨                   ⎬⎪
⎪     ⎪V'f...f'           ⎪⎪ ⎪GE | LE⎪ ⎪     ⎪V'f...f'           ⎪⎪
⎪     ⎪C=csect            ⎪⎪ ⎨       ⎬ ⎪     ⎪C=csect            ⎪⎪
⎪     ⎪COM=common         ⎪⎪ ⎪GT | LT⎪ ⎪     ⎪COM=common         ⎪⎪
⎪     ⎩schlüsselwort      ⎭⎪ ⎩NG | NL⎭ ⎪     ⎩schlüsselwort      ⎭⎪
⎪%@(...)                  ⎪⎪            ⎪%@(...)                  ⎪⎪
⎪%L(...)                  ⎪⎪            ⎪%L(...)                  ⎪⎪
⎪%L=(ausdruck)            ⎪⎪            ⎪%L=(ausdruck)            ⎪⎪
⎩AID-literal              ⎭⎭            ⎩AID-literal              ⎭⎭

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Zusammenstellung der Vergleichs- und der Booleschen Operatoren:

Vergleichsoperatoren		**Boolesche Operatoren**	
EQ	gleich	NOT	logische Negation
NE	nicht gleich	AND	logisches UND
LE	kleiner oder gleich	OR	logisches ODER (inklusiv)
LT	kleiner	XOR	logisches ODER (exklusiv)
NL	nicht kleiner		
GE	größer oder gleich		
GT	größer		
NG	nicht größer		

Die Vergleichsoperatoren sind untereinander gleichrangig und werden vor den Booleschen Operatoren abgearbeitet.

Bei den Booleschen Operatoren gilt die folgende Rangfolge:

NOT	höchster Rang
AND	zweithöchster Rang
OR/XOR	niedrigster Rang

Gleichrangige Operatoren werden von links nach rechts abgearbeitet. Um zu erreichen, dass die Operatoren in anderer Reihenfolge abgearbeitet werden, als durch die Rangfolge vorgegeben ist, müssen Sie entsprechende Klammern setzen.

Beispiele

```
(I EQ J AND VAR EQ 'A' OR VAR EQ 'B'):
entspricht
(((I EQ J) AND (VAR EQ 'A')) OR (VAR EQ 'B')):

(NOT VAR EQ 'A' OR ADR NE %OG):
entspricht
((NOT (VAR EQ 'A')) OR (ADR NE %OG)):
```

Klammern sind auch dann zu setzen, wenn die Vergleichsoperatoren oder Boolschen Operatoren mit Variablennamen verwechselt werden können und dadurch eventuell als fehlerhafte Syntax abgewiesen werden.

Der einzige unäre Operator ist das NOT, d.h. er bezieht sich auf nur einen Operanden. Alle anderen Booleschen Operatoren und alle Vergleichsoperatoren sind binäre Operatoren, sie verknüpfen zwei Operanden miteinander.

Mit den Vergleichsoperatoren können stets nur genau zwei Operanden verglichen werden; eine Kettung ist hier also nicht möglich. Geben Sie z.B. als Bedingung
`(A EQ B EQ C):` an, so wird dies von AID schon bei der Eingabe mit der Meldung
`AID0271 Syntaxfehler` abgelehnt. Die Bedingung müsste vielmehr durch
`(A EQ B AND B EQ C):` dargestellt werden.

Als Operand von Vergleichsoperatoren ist jede Speicherreferenz zugelassen, die Sie bei AID bilden können (siehe Abschnitt „Speicherreferenzen“ auf Seite 71). Wenn Sie Datenelemente aus Ihrem Programm verwenden, werden diese den folgenden Speichertypen zugeordnet:
– Binärstring (≙ %X)
– Character (≙ %C)
– numerisch (≙ %A, %F, %P, %D).

Speicherinhalte vom Typ Character können bis maximal 1000 Bytes in einer Bedingung verglichen werden. Logische Variablen können Sie vergleichen, wenn Sie mit einer Typmodifikation einen anderen Speichertyp festlegen (z.B. `(ALOG%X EQ X'FF'):`). Von den Schlüsselwörtern können die Durchlaufzähler der Subkommandos, die AID-Register, alle Programmregister sowie der Befehlszähler (Program Counter) %PC für Vergleiche herangezogen werden.

Die AID-Literale sind ebenfalls als Vergleichsoperanden zugelassen. Bei Character-Literalen (C'x...x') verwendet AID stets den Code des Eingabemediums, das heißt den Coded-Character-Set des Terminals oder der Prozedurdatei mit den AID-Kommandos. Wenn Sie Speicherinhalte vom Typ Character in ASCII vergleichen wollen, müssen Sie den Vergleichstext in ein Sedezimal-Literal übersetzen, so hat z.B. C'Hugo' in ASCII den sedezimalen Wert X'4875676F'.

Beim Schreiben einer Bedingung müssen Sie darauf achten, dass die Operanden-Typen verträglich sind. Welche Vergleiche zulässig sind und wie der Vergleich durchgeführt wird, ersehen Sie aus der Tabelle auf der nächsten Seite.
Die Zulässigkeit des Vergleichs wird erst geprüft, wenn das Überwachungsereignis eingetreten ist. Im Fehlerfall gibt AID eine entsprechende Meldung aus und setzt das Ergebnis des Vergleichs auf FALSE, d.h. der Kommandoteil des Subkommandos wird nicht ausgeführt.

AID unterscheidet zwischen Binär-, Character- und numerischem Vergleich. Die Art des Vergleichs leitet AID aus dem Typ der beteiligten Operanden ab. Die Operanden einer Bedingung werden von AID in einer bestimmten, von der jeweiligen Programmiersprache unabhängigen Systematik konvertiert und verglichen, da Sie ja in einem Vergleich Datenelemente aus Modulen verknüpfen können, die in unterschiedlichen Programmiersprachen geschrieben sind. Daraus ergibt sich, dass das Ergebnis eines AID-Vergleichs nicht unbedingt mit dem Ergebnis eines entsprechenden Vergleichs in einer bestimmten Programmiersprache übereinstimmen muss:

– Beim Character-Vergleich wird der kürzere Operand logisch mit Leerzeichen aufgefüllt, und AID vergleicht zwei gleichlange Operanden, während z.B. Fortran das Ergebnis des Vergleichs stets mit FALSE bewertet, wenn die beteiligten Operanden in der Länge nicht übereinstimmen.
– Beim Binärvergleich wird rechts mit X'00' aufgefüllt und dann genauso verfahren wie beim Character-Vergleich.
– Bei numerischen Vergleichen können unterschiedliche Ergebnisse dadurch zustande kommen, dass AID bei der Konvertierung der Operanden mit einer anderen Genauigkeit arbeitet wie die jeweilige Programmiersprache.
– COBOL ordnet die numerisch druckaufbereiteten Variablen den numerischen Operanden zu, für AID gehören diese Variablen zum Speichertyp Character.

Insbesondere müssen Sie diesen Sachverhalt in Betracht ziehen, wenn Sie Operanden aus verschiedenen Programmiersprachen vergleichen wollen.

Die Vergleiche, die mit den Vergleichsoperatoren gebildet werden, dienen als Operanden der Booleschen Operatoren. Logische Variablen, wie sie z.B. Fortran kennt, können nicht eingesetzt werden.
Mit den Booleschen Operatoren können Sie auch mehr als zwei Vergleiche miteinander verknüpfen. Die Obergrenze wird bestimmt durch die Komplexität der Vergleichsoperanden und die Größe des AID-internen Eingabepuffers.

Der folgenden Tabelle können Sie entnehmen, wie die verschiedenen Operanden-Typen miteinander verglichen werden, bzw. welche Vergleiche nicht zulässig sind:

Speicher-typ 1.Operand	Speichertyp 2. Operand					
	%X X' f...f' B' b...b'	numerisch	%C C'x...x' U'x...x'	%UTF16/ NATIONAL	num. Literal	Pointer
%X X' f...f' B' b...b'	bin	bin	bin	bin	–	bin
numerisch	bin	num	num(1)	numchar UTF16-chr	num	–
%C C'x...x' U'x...x'	bin	num(1)	char	UTF16-conv UTF16-chr	num(1)	–
%UTF16/ NATIONAL	bin	numchar UTF16-chr	UTF16-conv UTF16-chr	UTF16-chr	numchar UTF16-chr	–
num. Literal	–	num	num(1)	numchar UTF16-chr	num	–
Pointer	bin	–	–	–	–	bin

bin Binär-Vergleich
Es wird Bit für Bit von links nach rechts verglichen. Der kürzere Operand wird rechts mit Nullen (B'0') aufgefüllt.

char Character-Vergleich
Es wird Byte für Byte von links nach rechts verglichen. Der kürzere Operand wird rechts mit Blanks (X'40') aufgefüllt.

num numerischer Vergleich
Der arithmetische Wert der beiden Operanden wird verglichen.

numchar
Beim Vergleich wird von dem ganzzahligen, numerischen Feld die abdruckbare Ziffernfolge in der UTF16-Codierung verwendet.

UTF16-chr
: Character-Vergleich in UTF16-Codierung
Es wird Byte für Byte von links nach rechts verglichen. Das Auffüllen und Kürzen wird jedoch mit UTF16-Zeichen durchgeführt (Leerzeichen in 2 Byte-Codierung).

i Ordnungsrelationen, die bei Vergleichen in EBCDIC-Codierung verwendet werden, werden durch den byteweisen UTF16-Vergleich von AID nicht mehr unterstützt.

UTF16-conv
: Wenn ein Operand vom Typ %UTF16 ist und der zu vergleichende Operand vom Typ %C oder ein C-/U-Literal ist, erfolgt durch UTF16-conv die Konvertierung nach %UTF16. Anschließend kann der Vergleich wie bei UTF16-chr durchgeführt werden.

num(1)
: Enthält ein Operand vom Typ Character nur Ziffern und ist er höchstens 19 Stellen lang, so wird er numerisch verglichen, falls der zweite Operand vom Typ numerisch ist.
Alle übrigen Operanden vom Typ Character können mit numerischen Speichertypen oder numerischen Literalen nicht verglichen werden.

\-
: kein Vergleich möglich
Der Vergleich wird mit einer Fehlermeldung abgewiesen. Das Ergebnis des Vergleichs wird auf FALSE gesetzt.

Numerische Speichertypen:

%A, %Y (entspricht %AL2)	Ganzzahl ohne Vorzeichen
%F, %H (entspricht %FL2)	Ganzzahl mit Vorzeichen
%P	gepackte Zahl
%D	Gleitpunktzahl
%PC	Befehlszähler (Program Counter)
alle Register	
%•[*subkdoname*]	Durchlaufzähler

sowie alle symbolisch adressierten Datenelemente vom Typ numerisch.

i Nicht alle Datenelemente, die in den einzelnen Programmiersprachen numerisch behandelt werden, haben auch für AID einen numerischen Speichertyp; siehe hierzu die einzelnen sprachspezifischen Handbücher (%SET-Tabelle).

Numerische Literale:

```
[{±}]n                          Ganzzahl
#' f...f'                       Sedezimalzahl
[{±}]n.m                        Dezimalpunktzahl
[{±}]mantisseE[{±}]exponent     Gleitpunktzahl
```

Die Speichertypen %S bzw. %SX sind in Vergleichen nur bedingt sinnvoll und daher in obiger Tabelle nicht aufgeführt. %S wird wie %XL2 und %SX wie %XL4 behandelt.

Weitere Informationen zu Speichertypen und Literalen finden Sie in dem Kapitel „AID-Literale“ auf Seite 103 und dem Kapitel „Schlüsselwörter“ auf Seite 111.

Beispiele

1. `%IN S'18' <(%• LT 10): %D I,J; %MOVE X'58' INTO V'348'>`

   Wenn das Programm die Anweisung mit der Nummer 18 erreicht, unterbricht AID den Programmablauf und prüft die Bedingung des Subkommandos. Die ersten 9 Mal wird der Kommandoteil ausgeführt, d.h. AID gibt den Wert der Variablen I und J am Bildschirm aus und setzt den Inhalt der virtuellen Adresse V’348’ auf X’58’. Da das Subkommando kein %STOP enthält, wird der Programmablauf bei der Anweisung mit der Nummer 18 fortgesetzt.
   Bei jedem weiteren Durchlaufen des Testpunkts S’18’ wird der Kommandoteil des Subkommandos übersprungen.

2. `%IN S'25' <INS25: (ACHAR EQ 'END'): %D ISUM,JSUM; %STOP>`

   Das Subkommando des Testpunkts S’25’ wird solange nicht ausgeführt, bis das Feld mit der symbolischen Adresse ACHAR den Inhalt ’END’ hat. Dann zeigt AID den Inhalt der Summenfelder ISUM und JSUM am Bildschirm an, und das Programm wechselt in den Kommandomodus.

3. `%CONTROL1 %IO <OUTPUT: (SLF NE 200): %D %•, OUTDAT, SLF P=MAX>`

   Beim Testen eines Programmes, das Sätze mit der Satzlänge 200 ausgeben sollte, stellen Sie fest, dass die Satzlänge nicht immer stimmt. Mit dem obigen Kommando %CONTROL1 können Sie das Satzlängenfeld SLF überwachen. Jedes Mal wenn ein Satz ausgegeben wird, aber SLF nicht den Wert 200 enthält, protokolliert AID den Inhalt des Durchlaufzählers sowie der Variablen OUTDAT und SLF nach SYSLST.

4. `%INSERT S'18' <IN1:(I EQ 15): %SET 1 INTO J; %D ARRAY(K),K;%STOP>`

   Auf die Anweisung mit der Nummer 18 wird ein Testpunkt gesetzt. Zu diesem Testpunkt wird ein Subkommando mit dem Namen IN1 eingetragen: Immer dann, wenn der Index I den Wert 15 hat, wird Index J auf 1 gesetzt und das Vektorelement ARRAY(K) mit dem zugehörigen Index K ausgegeben. Danach hält das Programm an.

5.
```
%SET 0 INTO %0G
%INSERT S'25' <ZL1: (CNO NE '3'): %SET %L=(1 + %0G) INTO %0G;-
               %D %•, %0G, CNO, OUTDAT P=MAX>
```

   Der %SET setzt das AID-Register %0G auf 0. Mit dem %INSERT wird auf die Anweisung mit der Nummer 25 ein Testpunkt gesetzt und das Subkommando mit dem Namen ZL1 definiert. Bei jedem Durchlaufen des Testpunkts erhöht AID den Zähler %•ZL1 um 1. Nur wenn die Kennziffer CNO am Testpunkt nicht auf ’3’ steht, wird auch Register %0G um 1 erhöht, und AID protokolliert die Zählerstände, den Inhalt der Kennziffer CNO und den Ausgabesatz OUTDAT nach SYSLST (P=MAX).
   Der Inhalt von %•ZL1 gibt also an, wie oft das Programm Anweisung 25 durchläuft, während in %0G gezählt wird, wie oft die Kennziffer CNO am Testpunkt nicht auf 3 steht.

6.
```
%IN PROC <(%3 GT 4096 AND %5->%L1 EQ 'A'): %SET %L=(%5 + 2) INTO %5>
```

   Vor der Ausführung der Anweisung mit dem Namen PROC prüft AID, ob Register %3 einen Wert > 4096 enthält und ob gleichzeitig an der Speicherstelle, auf die Register %5 zeigt, ein ’A’ steht. Trifft dies zu, wird der Inhalt von Register %5 um 2 erhöht und der Programmablauf fortgesetzt.

6.4 Ketten

Wenn Sie mehrere %INSERT für denselben *testpunkt* oder %ON für dasselbe *ereignis* schreiben, kettet AID das zuletzt eingegebene Subkommando vor das vorige (LIFO-Prinzip). Eine Ausnahme bildet das *write-ereignis* beim %ON. Hier kann nicht gekettet werden, sondern das zuletzt eingegebene Kommando überschreibt das vorherige. AID macht Sie mit der Warnung AID0496 darauf aufmerksam.
Gekettet wird auch, wenn das neuere Kommando kein explizites Subkommando hat. In diesem Fall wird dann das implizit erzeugte Subkommando <%STOP> vor das bereits eingetragene gesetzt, und dieses ältere Subkommando wird nie mehr ausgeführt. Sinnvoller ist allerdings, das nicht mehr benötigte Subkommando zuerst zu löschen, da AID sonst im weiteren Testverlauf einen Eintrag verwalten muss, der nie mehr bearbeitet wird.

Enthält ein gekettetes Subkommando eine Bedingung, so gilt diese Bedingung nur für den dazugehörigen Kommandoteil. Für die Behandlung der Subkommandos, die in der Kette nachfolgen, gibt es zwei Möglichkeiten:

– Der von der Bedingung abhängige Kommandoteil ist mit %CONTINUE, %RESUME, %TRACE oder %STOP abgeschlossen. Dann werden nachfolgende Subkommandos nur bearbeitet, wenn das Ergebnis der Bedingung FALSE ist.
– Der von der Bedingung abhängige Kommandoteil enthält kein %CONTINUE, %RESUME, %TRACE oder %STOP: Nachfolgende Subkommandos werden stets abgearbeitet unabhängig davon, ob die Bedingung zutrifft oder nicht.

Zu einem %CONTROL*n* können Sie dagegen keine Subkommandos ketten. Ein neu eingegebener %CONTROL*n* überschreibt alle Operandenwerte eines früheren %CONTROL*n* zu derselben Nummer *n* mit den Angaben aus dem neuen Kommando.

Beispiele

1. Im Testverlauf geben Sie folgende Kommandos ein:

```
%ON %LPOV(SUBTOT)
       .
       .
       .
%ON %LPOV(SUBTOT)    <%DISPLAY S=B1@.PROC=B1.CHAR_DAT>
```

   Beim ersten %ON ergänzt AID als Subkommando <%STOP>, da explizit kein Subkommando angegeben wurde. Durch Kettung entsteht nach der Eingabe des zweiten %ON für das Ereignis %LPOV(SUBTOT), d.h. nach dem Laden des Moduls SUBTOT das folgende Subkommando:

```
<%DISPLAY S=B1@.PROC=B1.CHAR_DAT; %STOP>
```

2. Wie sich die Kettung in Verbindung mit dem standardmäßig eingesetzten <%STOP> auswirkt, soll nun gezeigt werden:

```
%INSERT ST4  <%D TEXTDAT>
                     .
                     .
                     .
%INSERT ST4
```

   Der zweite %INSERT enthält kein Subkommando. Dafür setzt AID ein <%STOP> ein. Da das zweite %INSERT-Kommando denselben Testpunkt bezeichnet wie das vorhergehende, kommt es zur Kettung, bei der folgendes Subkommando entsteht:

```
<%STOP; %DISPLAY TEXTDAT>
```

   Die Ausführung eines Subkommandos wird aber durch %STOP abgebrochen. `%DISPLAY TEXTDAT` wird also nie mehr ausgeführt, bleibt jedoch als Subkommando zum Testpunkt ST4 eingetragen und kann aus der Kette auch nicht herausgelöscht werden, da es keinen Namen hat. Am besten geben Sie jedem Subkommando einen Namen, so dass Ihnen bei falscher Kettung stets die Möglichkeit bleibt, das Subkommando über seinen Namen aus der Kette zu löschen.
   Richtiger wäre im obigen Beispiel gewesen, wenn Sie zunächst mit

```
%REMOVE ST4
```

   den ersten %INSERT gelöscht und dann

```
%INSERT ST4
```

   geschrieben hätten.

3. Die beiden folgenden %INSERTs können Sie in einer Prozedur einsetzen, um ein beliebiges Character-Literal zu suchen. Im Trefferfall steht die gefundene Adresse im AID-Register %0G und die Länge des gesuchten Strings in %2G (siehe %FIND).

```
%INSERT V'1648' <(%0G NE -1): %SET %L=(%1G - %0G) INTO %2G>
%INSERT V'1648' <%FIND C'x...x'>
```

   Die Kettung wird notwendig, da Sie eine Bedingung nur am Anfang des Subkommandos schreiben können. Erst durch Kettung entsteht zum Testpunkt V'1648' das gewünschte Subkommando:

```
<%FIND C'x...x'; (%0G NE -1): %SET %L=(%1G - %0G) INTO %2G>
```

4.
```
%INSERT S'50' <%D NO,INDAT; %STOP>
%INSERT S'50' <(ISW EQ X'FF'): %SET X'00' INTO ISW; %CONT>
```

   Durch die beiden %INSERTs zur selben Source-Referenz, nämlich der Anweisung mit der Nummer 50, entsteht am Testpunkt S'50' das folgende bedingte Subkommando mit THEN- und ELSE-Zweig (zur Verdeutlichung sind die verschiedenen Teile der Konstruktion mit IF, THEN und ELSE gekennzeichnet):

```
(ISW EQ X'FF'): %S X'00' INTO ISW; %CONT; %D NO, INDAT; %STOP
↑               ↑                         ↑
IF              THEN                      ELSE
```

   Jeweils vor der Ausführung der Anweisung S'50' unterbricht AID den Programmablauf und prüft den Inhalt von Schalter ISW. Enthält der Schalter den Wert X'FF', wird er auf X'00' zurückgesetzt und der Programmablauf fortgesetzt. Andernfalls gibt AID den Inhalt von NO und INDAT aus und hält das Programm an.

6.5 Schachteln

Im Subkommando eines %INSERT oder %ON kann wieder ein %INSERT oder %ON stehen. Das wird als Schachtelung von Subkommandos bezeichnet. AID unterstützt diese Schachtelung über mehrere Subkommando-Stufen. Wie tief Sie Subkommandos schachteln können, hängt von deren Komplexität und von der Größe des AID-internen Eingabepuffers ab.
Geschachtelte Subkommandos werden schrittweise wirksam. Während die Überwachungsbedingung der ersten Generation (äußere Schachtel) sofort von AID eingetragen wird und damit im nachfolgenden Programmablauf auch schon eine Unterbrechung erzeugen kann, trägt AID den *testpunkt* (%INSERT) oder das *write-ereignis* bzw. *ereignis* (%ON) in jüngeren Subkommandos erst dann ein, wenn die Überwachungsbedingung der unmittelbar vorhergehenden Generation zur Unterbrechung geführt hat. Tritt in einer Schachtelung eines %INSERT- oder eines %ON-Kommandos für eine schon eingetragene Überwachungsbedingung ein weiteres anderes Kommando auf, so kommt es zusätzlich zur Kettung des neuen Subkommandos vor das ältere (LIFO-Prinzip). Nicht gekettet wird dagegen ein Subkommando zu einem Testpunkt oder einem Ereignis einer inneren Schachtel, wenn der Testpunkt der nächsthöheren Stufe der Schachtelung auf Grund einer Programmschleife mehrmals durchlaufen wird bzw. wenn das Ereignis der äußeren Schachtel mehrmals eintritt.

Subkommandos zu einem %CONTROL*n* können Sie nicht schachteln, daher sind im Subkommando eines %CONTROL*n* zusätzlich zu den Kommandos, die in allen Subkommandos nicht erlaubt sind, auch die Kommandos %CONTROL*n*, %INSERT und %ON nicht zugelassen. In Subkommandos eines %CONTROL*n* darf außerdem kein %JUMP (COBOL, FOR1) angegeben werden.

Beispiele

1. `%IN ST3 <%DISPLAY 'INSERT1', TEXTDAT;%IN OUTPUT <%D 'INSERT2', I,J,K,-`
   `NUM-TAB; %ON %SVC(186) <%D 'OPEN DAT1',I,J>>>`

   Das Beispiel bezieht sich auf ein COBOL-Programm. Mit %INSERT ST3 wird als Testpunkt der Paragraph ST3 vereinbart. Dieser %INSERT enthält einen darin geschachtelten %INSERT, der wiederum ein %ON-Kommando enthält. Der darin angegebene Testpunkt OUTPUT sowie das angegebene Ereignis %SVC(186) (≙OPEN) haben noch keine Auswirkung auf den Ablauf. Sie können erst aktiv werden, wenn der Testpunkt des %INSERT erreicht wird, in dessen Subkommando sie definiert sind. Kommt der Programmablauf zur symbolischen Adresse ST3, wird das zugehörige *subkdo* ausgeführt, d.h. das Literal 'INSERT1' und der Inhalt des Ausgabesatzes TEXTDAT werden ausgegeben und der Testpunkt OUTPUT wird gesetzt. Das Subkommando zum Testpunkt OUTPUT ist noch nicht wirksam. Im zu testenden Programm sind also bis zu dieser Stelle des Programmablaufs die Testpunkte ST3 und OUTPUT gesetzt.

   Da das Subkommando zum Testpunkt ST3 kein %STOP-Kommando enthält, wird das Programm fortgesetzt. Erreicht der Programmablauf die Adresse OUTPUT, wird nun der `%DISPLAY 'INSERT2',I,J,K, NUM-TAB` ausgeführt. Neben diesem Kommando enthält das Subkommando noch einen %ON für das Ereignis %SVC(186). Erkennt AID im weiteren Programmablauf einen SVC zum Öffnen einer Datei, führt es das Subkommando aus, das mit dem %ON vereinbart wurde: das Literal 'OPEN DAT1' und der Inhalt der Indizes I und J werden ausgegeben.

2. `%IN ST4 <%D TEXTDAT>`
   `%ON %LPOV (SUBTOT) <%REMOVE ST4; %IN ST4 <%D 'SUBTOT LOADED'; %STOP >>`
   `%RESUME`

   Immer wenn der Testpunkt ST4 erreicht wird, gibt AID den Speicherinhalt des Datenfeldes TEXTDAT aus. Tritt das vereinbarte Ereignis %LPOV (SUBTOT) ein, d.h. der Modul SUBTOT wird geladen, führt AID das Subkommando im %ON-Kommando aus. Der Testpunkt ST4 wird gelöscht, aber es wird sofort wieder zu diesem Testpunkt ein neues Subkommando eingetragen:

   `<%DISPLAY 'SUBTOT LOADED'; %STOP>`

   Wenn ST4 das nächste Mal durchlaufen wird, gibt AID den Text 'SUBTOT LOADED' aus und unterbricht den Programmablauf. Sie können wieder neue Kommandos eingeben.

6.6 Löschen

Zum Löschen von Subkommandos steht das Kommando %REMOVE zur Verfügung. Implizit löschen Sie ein Subkommando, indem Sie das zugehörige Überwachungskommando oder bei %INSERT den zugehörigen Testpunkt oder bei %ON das zugehörige Ereignis löschen.
Explizit können Sie ein Subkommando über seinen Namen löschen. Diese Möglichkeit besteht nur für Subkommandos eines %CONTROL*n* oder eines %INSERT. Da zu einem %CONTROL*n* Subkommandos nicht gekettet werden können, bewirkt ein %REMOVE %•*subkdoname* dasselbe wie %REMOVE %CONTROL*n*. Mit %INSERT können jedoch zu einem bestimmten Testpunkt eine ganze Reihe von Subkommandos hintereinander gekettet werden. In diesem Fall löscht %REMOVE %.*subkdoname* ein einzelnes Subkommando über seinen Namen aus der Kette heraus. Wenn das Subkommando keinen Namen hat, kann es nur mit dem gesamten Testpunkt gelöscht werden. Es empfiehlt sich daher, Subkommandos stets einen Namen zu geben.

Bei geschachtelten Subkommandos können innere Subkommandos, die noch nicht am zugehörigen Testpunkt eingetragen wurden, nicht aus der Schachtelung herausgelöscht werden, auch nicht über ihren Namen. Diese Subkommandos können Sie nur zusammen mit dem gesamten %INSERT (%REMOVE %INSERT) oder mit dem Testpunkt (%REMOVE *testpunkt*) löschen.

Das aktuelle Subkommando können Sie unmittelbar nach seiner Ausführung wieder löschen, indem Sie als letztes Kommando des Kommmandoteils %REMOVE %• schreiben. Der %REMOVE %• wird sofort ausgeführt, was zur Folge hat, dass eventuell noch nachfolgende Kommandos ebenfalls gelöscht werden und somit nicht mehr ausgeführt werden können.

7 Adressierung in AID

Die Kommandos zur Ablaufüberwachung, das Kommando %JUMP (Festlegen einer Fortsetzungsadresse) und die Kommandos zur Ausgabe und Modifikation von Speicherinhalten verlangen Operanden, die eine Adresse oder einen Bereich im Speicher bezeichnen. Eine Adresse im ausführbaren Teil des Programms müssen Sie bei %DISASSEMBLE, %INSERT, %JUMP und %REMOVE angeben. %CONTROL*n* und %TRACE verlangen als Operanden einen Speicherbereich im ausführbaren Teil des Programms, während bei den Kommandos %DISPLAY, %FIND, %MOVE und %SET der anzugebende Speicherbereich auch im Datenteil des Programms liegen kann.

Eine Adresse bezeichnen Sie bei AID durch eine Adresskonstante oder durch eine komplexe Speicherreferenz. Einen Speicherbereich können Sie abhängig vom Kommando durch eine Qualifikation oder eine Speicherreferenz angeben oder Sie beschreiben den Bereich durch zwei Adressen, der Bereich liegt dann zwischen der ersten und der zweiten Adresse. Ausführlich beschrieben finden Sie die anzugebenden Operanden bei den Kommandobeschreibungen in den sprachspezifischen Handbüchern und im Handbuch für das Testen auf Maschinencode-Ebene.
Das Kommando %SDUMP nimmt eine Sonderstellung ein, der zugehörige Operand *dump-bereich* bezeichnet entweder einen Namensraum, der mit einer Qualifikation angegeben werden kann, oder ein einzelnes Datum.
In den folgenden Abschnitten finden Sie alle Begriffe, mit denen Sie bei AID eine Adresse bezeichnen können, die verschiedenen Qualifikationen, die einfachen und die komplexen Speicherreferenzen, beschrieben.

7.1 Qualifikationen

Mit Qualifikationen beschreiben Sie den Pfad zu einem Speicherobjekt, das außerhalb des gerade gültigen AID-Arbeitsbereiches oder nicht im aktuellen Haupt- oder Unterprogramm liegt oder dort nicht eindeutig ist. In einigen Fällen kann die Adressierung mit einer Qualifikation enden, Sie sprechen dann mit der Qualifikation das Speicherobjekt selbst an. Es gibt die Basisqualifikation und Bereichsqualifikationen. Qualifikationen geben Sie immer in der Reihenfolge der Übergeordneten zur Untergeordneten an und nur soweit sie zur eindeutigen Pfadbeschreibung nötig sind. Überflüssige Qualifikationen ignoriert AID.

Aufeinanderfolgende Qualifikationen werden durch einen Punkt getrennt. Außerdem muss zwischen der letzten Qualifikation und dem anschließenden Adressteil ein Punkt stehen. Mit %QUALIFY können Sie Qualifikationen vordefinieren. In einem Adressoperanden können Sie die mit %QUALIFY vereinbarten Qualifikationen durch Voran Stellen eines Punktes übernehmen.

7.1.1 Basisqualifikation

Die Basisqualifikation bezeichnet die Umgebung (Environment), sie legt fest, ob eine anschließende Adresse im virtuellen Speicher oder in einer Dump-Datei liegen soll. Die Basisqualifikation wird beim symbolischen und maschinennahen Testen gleich verwendet.

E=VM ist der Standardwert und bezeichnet den virtuellen Speicherbereich des geladenen Programms.

E=Dn bezeichnet einen Speicherabzug in einer Dump-Datei mit einem Linknamen D0 - D7; die Dump-Datei muss zuvor mit %DUMPFILE zugewiesen werden.

Die Basisqualifikation können Sie global mit %BASE vereinbaren oder im Adressoperanden für eine einzelne Speicherreferenz angeben. Als alleiniger Operand ist die Basisqualifikation in den Kommandos %BASE, %QUALIFY und %SDUMP zugelassen. In allen übrigen Adress- oder Bereichsoperanden müssen Sie an eine Basisqualifikation einen der folgenden Begriffe anschließen:

- Bereichsqualifikation
- Datenname
- Anweisungsname
- Source-Referenz
- virtuelle Adresse
- Schlüsselwort

7.1.2 Bereichsqualifikationen

Mit einer Bereichsqualifikation bezeichnen Sie einen Teilbereich eines Programms. Beim Programmieren, Übersetzen oder Binden eines Programms legen Sie die Teilbereiche fest und/oder benennen sie. Bereichsqualifikationen geben Sie an, wenn eine Adresse nicht in dem Programmteil liegt, der gerade durchlaufen wird. Es gibt unterschiedliche Bereichsqualifikationen für das maschinennahe Testen und für die jeweilige Programmiersprache. Die Bereichsqualifikationen für das symbolische Testen werden von der Struktur der Programmiersprache bestimmt. Im Kapitel „Voraussetzungen zum Testen mit AID“ auf Seite 31 der sprachspezifischen Handbücher ist beschrieben, welche Programmteile durch welche Qualifikationen angesprochen werden.

SPID=X’f...f’	ESA-Anlagen maschinennah
ALET={X’f...f’ |%nAR|%nG}	ESA-Anlagen maschinennah
CTX=kontext	symbolisch und maschinennah
L=ladeeinheit	maschinennah
O=objektmodul	maschinennah
C=csect/segmentname/sharename	maschinennah/COBOL
COM=common	symbolisch und maschinennah
S=srcname	alle Programmiersprachen
PROC=name	alle Programmiersprachen
PROG=name	Assembler, COBOL, Fortran
ONUNIT=’onunitname’	PL/I
BLK=' blkname'	C++/C, PL/I

Die Bereichsqualifikationen geben Sie im Adressoperanden zu einer Speicherreferenz an. Dort werden sie zur Pfadbeschreibung eingesetzt. Sie müssen nur die Qualifikationen angeben, die zur eindeutigen Ansprache benötigt werden. Liegt die Unterbrechungsstelle jedoch in den Routinen des Laufzeitsystems, können Sie Daten und Anweisungen im eigenen Programm nur mit der vollen Qualifikation ansprechen.
In den Kommandos, die einen Bereichsoperanden erfordern, können Sie alle Bereichsqualifikationen außer SPID und ALET einsetzen, um damit einen Bereich zu bezeichnen. C=*csect* und COM=*common* können Sie zusätzlich auch als Anfangsadresse einsetzen. Alle Bereichsqualifikationen können Sie mit %QUALIFY als Vorqualifikation vereinbaren.
In einer komplexen Speicherreferenz müssen Sie darauf achten, dass Sie mit den nachfolgenden Operationen die Bereichsgrenzen nicht überschreiten. Obwohl Sie auf das Attribut Länge einer Bereichsqualifikation nicht zugreifen können, wird dennoch überprüft, ob das Ergebnis eines Adressversatzes oder einer Längenmodifikation noch innerhalb des Bereichs liegt, der mit der Qualifikation angegeben wurde.

Die **ALET- und SPID-Qualifikation** können Sie nur auf ESA-Anlagen verwenden. Sie bezeichnen einen Datenraum und können nur vor einer virtuellen Adresse oder einer komplexen Speicherreferenz, die ohne symbolische Komponenten gebildet wird, verwendet werden.

Die **Kontextqualifikation** bezeichnet den Kontext, in dem die mit nachfolgenden Qualifikationen oder Adressangaben angesprochenen Speicherbereiche oder Adressen liegen sollen. Sie ist nur erforderlich, wenn eine CSECT, ein COMMON oder eine Übersetzungseinheit in mehreren Kontexten enthalten ist und die aktuelle Unterbrechungsstelle nicht in der CSECT, dem COMMON oder der Übersetzungseinheit liegt, in dem das mit dem Adressoperanden ausgewählte Speicherobjekt enthalten ist.

Angegeben wird die Kontextqualifikation mit CTX=kontext. Dabei ist *kontext* ist der explizit im BIND-Makro mit dem Operand LNKCTX[@] vergebene Name oder der implizite Name LOCAL#DEFAULT, falls LNKCTX[@] nicht angegeben wurde. Dynamisch mit dem DBL geladene Programme erhalten den gleichen standardmäßig vergebenen Kontextnamen LOCAL#DEFAULT. Statisch mit TSOSLNK gebundenen Programmen wird der Kontext CTXPHASE zugewiesen. Weitere Kontexte eines Programms können durch Konnektieren an ein Shared-Code-Programm (z.B. an ein DSSM-Subsystem oder an ein Programm in einem COMMON-MEMORY-POOL) entstehen.
Die Operanden *vorqualifikation* des Kommandos %QUALIFY und *dump-bereich* im %SDUMP können mit CTX=*kontext* enden. In allen übrigen Adress- oder Bereichsoperanden müssen Sie an eine CTX-Qualifikation einen der folgenden Begriffe anschließen:
– eine weitere Bereichsqualifikation
– Datenname
– Anweisungsname
– Source-Referenz

Die **L- und O-Qualifikation** geben Sie an, wenn Sie den Pfad zu einer von mehreren gleichnamigen CSECTs oder COMMONs beschreiben müssen. Sie müssen nur die L- und/oder O-Qualifikation angeben, die zur eindeutigen Ansprache genügt. Auf L- und/oder O-Qualifikation muss stets eine C- oder COM-Qualifikation folgen.

Die **C- und COM-Qualifikation** können Sie in den Kommandos %CONTROL*n* und %TRACE als Bereichsangabe verwenden. Wenn Sie mit der C-Qualifikation eine CSECT bzw. mit der COM-Qualifikation einen COMMON bezeichnen, können Sie nur ein maschinennahes Kriterium angeben. Die Angabe C=*sharename/segmentname*, die Sie beim Testen von COBOL-Programmen einsetzen können, darf nur mit einem symbolischen Kriterium kombiniert werden. Auf eine C-Qualifikation kann jedoch nie eine symbolische Speicherreferenz folgen, auch nicht in COBOL.
In den Kommandos %DISASSEMBLE, %INSERT und %REMOVE wird mit der C-/COM-Qualifikation die Anfangsadresse der CSECT/des COMMON angegeben. Auch in den Kommandos %DISPLAY, %FIND, %MOVE, %ON %WRITE(...) und %SET kann der Adressoperand auf C=*csect*/COM=*common* enden. Es wird damit die gesamte CSECT bzw. der gesamte COMMON angesprochen. Die CSECT bzw. der COMMON wird in diesen Kommandos als maschinennahe Speicherreferenz verwendet. Ebenso können Sie innerhalb einer komplexen Speicherreferenz die C-/COM-Qualifikation als Speicherreferenz einsetzen (siehe Abschnitt „Maschinennahe Speicherreferenzen“ auf Seite 73).

Die **S-, PROC-, BLK-, ONUNIT- und PROG-Qualifikation** können Sie zur Bezeichnung eines Speicherbereichs in %CONTROL*n* und %TRACE oder des Namensraumes bei %SDUMP angeben. Sie bezeichnen damit den gesamten angegebenen Programmteil. Als Speicherreferenz können Sie diese Qualifikationen nicht verwenden.

Auf eine S-Qualifikation kann folgen:
– eine PROC-, BLK- oder ONUNIT-Qualifikation
– ein Datenname
– ein Anweisungsname
– eine Source-Referenz

Auf eine PROC-, BLK- oder ONUNIT-Qualifikation kann folgen:
– ein Datenname
– ein Anweisungsname
– eine Source-Referenz

Die **PROG-Qualifikation** ist eine Zusammenfassung für S=srcname•PROC=name, falls *srcname* und *name* gleich sind. Sie kann in Assembler, COBOL und Fortran eingesetzt werden.

Beispiele

1. ```
   %BASE E=VM
   %DUMPFILE D1=M.DUMP
   %DISPLAY V'10A', E=D1.V'10A'
   ```
   Mit %BASE vereinbaren Sie den virtuellen Speicherbereich des geladenen Programms als Basisqualifikation. Mit %DUMPFILE weisen Sie dem Linknamen D1 die Datei M.DUMP zu.
   Da die Basisqualifikation E=VM gilt, gibt AID für die erste Angabe im %DISPLAY-Kommando vier Bytes ab Adresse V'10A' des geladenen Programms aus und für die zweite Angabe vier Bytes ab Adresse V'10A' aus einem Speicherabzug aus. Dieser Speicherauszug steht in einer Datei, die mit %DUMPFILE dem Linknamen D1 zugewiesen wurde.
   4 Bytes ist die implizite Länge einer V-Adresse.

2. ```
   %DISPLAY S=COMPUTE@.PROC=COMPUTE.SUM
   ```
   Aus der Programmeinheit COMPUTE@ eines Fortran-Programms gibt AID den Inhalt der Variablen SUM aus. S- und PROC-Qualifikation sind nötig, wenn das Programm in einer anderen Programmeinheit unterbrochen wurde.

3. ```
   %QUALIFY E=D1.S=COMPUTE@.PROC=COMPUTE
   %DISPLAY .SUM
   ```
   Vor den Punkt vor SUM setzt AID die vereinbarte Vorqualifikation. So entsteht das Kommando: `%DISPLAY E=D1.S=COMPUTE@.PROC=COMPUTE.SUM`
   AID gibt aus der Dump-Datei, die dem Linknamen D1 zugewiesen ist, aus der Programmeinheit COMPUTE@ das Datenfeld SUM aus.
   ```

4. `%DISPLAY E=D2.S=TEST@.BLK='23'.var`
   In der Dump-Datei mit dem Linknamen D2, die den Speicherabzug eines C-Programms enthält, und dort in der Übersetzungseinheit mit dem Codemodulnamen TEST@ liegt im Block, der in Zeile 23 beginnt, die lokale Variable var. Deren Inhalt gibt AID aus.

5. `%MOVE L=LAD1.C=CS1.(%L(L=LAD1.C=CS1) - 4) INTO %2G`
   AID übertragt die letzten 4 Bytes der CSECT CS1 aus der Ladeeinheit LAD1 in das AID-Register %2G. Die L-Qualifikation ist nötig, da CS1 nicht die aktuelle CSECT ist und der Name CS1 nicht eindeutig innerhalb des Programmsystems ist.

6. `%INSERT S=COMPUTE@.PROC=COMPUTE.S'16' <%DISPLAY %•>`
   AID setzt einen Testpunkt auf die Anweisung mit der Nummer 16 in der Programmeinheit COMPUTE@. Kommt der Programmablauf an diesen Testpunkt, wird der Durchlaufzähler ausgegeben und das Programm fortgesetzt.

7.2 Speicherreferenzen

Mit einer Speicherreferenz können Sie in einem AID-Kommando ein Speicherobjekt ansprechen. Beschreibt die Speicherreferenz eine Zeichenfolge, z.B. vom Typ %C oder %UTF16, können die Zeichenkonvertierungsfunktionen %C() oder %UTF16() auf die Speicherreferenz angewendet werden.
AID unterscheidet zwischen einfachen und komplexen Speicherreferenzen.
Beispiele für einfache Speicherreferenzen sind:

- virtuelle Adressen: V'f...f'
- Datennamen: VAR1, FELD(I)
- Schlüsselwörter: %14, %2D, %PC, %CLASS6
- C-Qualifikationen: C=CS1
- COM-Qualifikationen: COM=CB
- Anweisungsnamen: L'20', COMPUTE1
- Source-Referenzen: S'133', S'44ADD'

Beispiele für komplexe Speicherreferenzen sind:

- %@(VAR1)->.(%L=(I+5))%XL20
- C=CS1.#'100'%SX->%CL8

Die einfachen Speicherreferenzen (das sind die maschinennahen und die symbolischen Speicherreferenzen und die Schlüsselwörter), sind mit ihren Attributen und Eigenschaften im Abschnitt „Maschinennahe Speicherreferenzen“ auf Seite 73 beschrieben.

Eine komplexe Speicherreferenz ist eine Vorschrift, nach der AID eine Adresse errechnet bzw. nach der Sie die Attribute eines Speicherobjekts modifizieren können. In einer komplexen Speicherreferenz können Sie symbolische und maschinennahe Speicherreferenzen, Schlüsselwörter, Konstanten und AID-Literale in den Operationen Adressversatz, indirekte Adressierung, Typ- und Längenmodifikation und Adressselektion verwenden, um die in einer Testsituation benötigte Adresse von AID errechnen zu lassen bzw. um Typ und Länge einer Speicherreferenz festzulegen. Informationen zur komplexen Speicherreferenz und den zugehörigen Operationen stehen im Abschnitt „Symbolische Speicherreferenzen“ auf Seite 75.

Attribute

Attribute beschreiben die Eigenschaften eines Speicherobjekts oder einer Konstanten. Speicherobjekte haben bis zu sechs Attribute:

– Name (optional)
– Adresse
– Inhalt
– Länge
– Speichertyp
– Ausgabetyp

Auf die Attribute Adresse, Länge und Speichertyp können Sie mit Selektoren zugreifen. Mit der Modifikation können Sie Länge und Speichertyp verändern. Das Attribut Länge bestimmt zugleich die Bereichsgrenzen: Adresse bis Adresse + (Länge - 1). Die Bereichsgrenzen werden bei den Operationen Längenmodifikation und Adressversatz überprüft. Bei der Übertragung mit %MOVE wird überprüft, ob *sender* innerhalb der Bereichsgrenzen von *empfänger* Platz hat. Eine Ausnahme bildet die virtuelle Adresse, der als Bereich der gesamte Benutzer-Adressraum zugeordnet ist, obwohl das Längenattribut nur 4 Bytes beträgt.

Wie AID im jeweiligen Kommando die Attribute berücksichtigt und welchen Prüfungen sie unterzogen werden, ist bei den einzelnen Kommandos in den sprachspezifischen Handbüchern bzw. in dem Handbuch für das Testen auf Maschinencode-Ebene beschrieben.
Wie AID innerhalb der Berechnung einer komplexen Speicherreferenz diese Attribute berücksichtigt, ist bei den Operationen der komplexen Speicherreferenz beschrieben.

Konstanten haben kein Adressattribut. Sie können deshalb nur in bestimmten Zusammenhängen verwendet werden. Insbesondere kann die Adressselektion nicht auf sie angewendet werden.

7.2.1 Maschinennahe Speicherreferenzen

Maschinennahe Speicherreferenzen sind die CSECTs, die COMMONs und die virtuellen Adressen.

Die **CSECTs und COMMONs** werden in Form der C- bzw. COM-Qualifikation mit C=*csect* bzw. COM=*common* angegeben. Da die CSECTs/COMMONs wie Qualifikationen angegeben werden und in einigen Kommandos auch wie Qualifikationen verwendet werden können, sind sie auch im Abschnitt „Bereichsqualifikationen“ auf Seite 67 beschrieben.
Als Speicherreferenz besitzt die C-/COM-Qualifikation die folgenden Attribute:

```
Name
Adresse
Inhalt
Länge (Länge der CSECT/des COMMON)
Speichertyp (%X)
Ausgabetyp (Dump)
```

Die Bereichsgrenzen werden durch die Anfangsadresse und die Länge der CSECT/des COMMON festgelegt.

Die folgenden Operationen können auf eine C-/COM-Qualifikation angewandt werden:
– Adressselektor
– Längenselektor
– Adressversatz
– Typmodifikation
– Längenmodifikation

Eine **virtuelle Adresse** geben Sie in folgender Form an:
V’f...f’, wobei *’f...f’* eine maximal 8stellige Sedezimalzahl zwischen ’0’ und ’7FFFFFFF’ ist.
Mit einer virtuellen Adresse sprechen Sie direkt eine Speicherstelle im geladenen Programm oder einer Dump-Datei an. Auf ESA-Anlagen kann damit auch eine Speicherstelle in einem Datenraum angesprochen werden. Hierzu ist die Angabe einer ALET-/SPID-Qualifikation vor der virtuellen Adresse nötig. Ansonsten ist vor einer virtuellen Adresse nur die Angabe einer Basisqualifikation sinnvoll.
Das Ergebnis eines Adressversatzes oder einer indirekten Adressierung ist ebenfalls eine virtuelle Adresse und hat dann auch deren Attribute.

Attribute einer virtuellen Adresse sind:

```
Adresse (f...f)
Inhalt
Länge (4 Bytes)
Speichertyp (%X)
Ausgabetyp (Dump)
```

Die Bereichsgrenzen reichen von V'0' bis V'7FFFFFFF'.
Im Gegensatz zu allen anderen Speicherobjekten, bei denen die Adresse und die Länge zugleich die Bereichsgrenzen festlegt, gelten bei der virtuellen Adresse andere Bereichsgrenzen. Der gesamte Benutzer-Adressraum ist für Operationen mit virtuellen Adressen möglich und nur die niedrigste Adresse V'0' und die höchste mögliche Adresse V'7FFFFFFF' dürfen nicht überschritten werden.

Auf eine virtuelle Adresse kann folgen:
– Adressversatz (•)
– indirekte Adressierung (->)
– Typmodifikation
– Längenmodifikation

Beispiele

1. `%DISPLAY V'100'->->->%C`
   Die 4 Bytes ab Adresse V'100' haben den Inhalt X'00000A1A'. Adresse V'A1A' hat den Inhalt X'0000000F' (erster Pointer-Operator). Adresse V'F' hat den Inhalt X'0000B001' (zweiter Pointer-Operator). Adresse V'B001' hat den Inhalt X'F1F2F3F4' (dritter Pointer-Operator). Den interpretiert AID als Character und gibt '1234' aus.

2. `%MOVE E=D1.V'206'.(%1)->.(%2-5) INTO %2G`
   In der Dump-Datei D1 soll ab Adresse V'206' um den Inhalt in Register %1 (X'00000004') weiterpositioniert werden. Der dort (V'20A') stehende Speicherinhalt (X'0000B111') wird als Adresse für eine Pointer-Operation verwendet. Ab dieser neuen Adresse (V'B111') wird um Inhalt von Register %2 (X'00000008') minus 5 weiterpositioniert und 4 Bytes ab dieser Adresse (V'B114') in das AID-Register %2G übertragen.

7.2.2 Symbolische Speicherreferenzen

Mit symbolischen Speicherreferenzen werden die symbolischen Adressen bezeichnet, die der Compiler bei der Übersetzung in den LSD-Sätzen hinterlegt. Dazu gehören die von Ihnen im Programm vergebenen Namen von Daten und Anweisungen, also Marken, Entrys oder Funktionsnamen sowie die vom Compiler erzeugten Source-Referenzen, mit denen Sie jede ausführbare Anweisung Ihres Programms ansprechen können, unabhängig davon, ob die Anweisung eine Marke hat oder nicht. Wenn also LSD-Sätze erzeugt wurden und zur Verfügung stehen, kann AID über Daten- oder Anweisungsnamen oder über Source-Referenzen auf die zugehörigen Adressen und die mit der Adresse verbundenen Attribute zugreifen.
Anweisungsnamen und Source-Referenzen sind Adresskonstanten und werden erst mit einem nachfolgenden Pointer-Operator zur Speicherreferenz. Ohne Pointer-Operator können sie nur in den Kommandos verwendet werden, die eine Adresse als Operanden verlangen. Wenn Sie jedoch den Befehlscode ansprechen wollen, der an der entsprechenden Adresse im Speicher steht, müssen Sie den Pointer-Operator anschließen.

7.2.2.1 Datennamen

Datennamen sind Namen von Variablen, Datenstrukturen, Feldern, Matrizen oder Vektoren, je nachdem was die jeweilige Programmiersprache anbietet und wie sie es benennt. In Tabellen oder Strukturen können Sie mit AID auf die einzelnen Elemente wie in einer Anweisung der Programmiersprache zugreifen. Sie setzen also erforderliche Kennzeichnungen, Indizes oder Subskripte hinter den Datennamen. Die wenigen Abweichungen sind in den Kommandobeschreibungen der sprachspezifischen Handbücher angegeben.

Im Quellprogramm definierte Konstanten gehören ebenfalls zu den Datennamen. Sie werden z.B. mit EQU (Assembler), *literal* und *symbolic character* im SPECIAL NAMES-Paragraphen (COBOL) oder mit PARAMETER (Fortran) vereinbart. Da sie keinen Speicherplatz belegen, können sie jedoch nicht wie alle anderen Daten verwendet werden. Sie haben kein Adressattribut. AID steht nur der Wert der Konstanten zur Verfügung. Die übrigen Attribute können Sie nicht verwenden.

Die Attribute von Datennamen werden im Quellprogramm festgelegt. Nur den Ausgabetyp legt AID selbst analog der Speichertyp-Ausgabetyp-Zuordnung fest (siehe Abschnitt „Allgemeine Speichertypen“ auf Seite 111). Datennamen haben die Attribute:

```
Name
Adresse
Inhalt
Länge
Speichertyp
Ausgabetyp
```

Die Bereichsgrenzen werden durch Adresse und Länge bestimmt.

Datennamen können Sie in allen Kommandos, die den Datenteil adressieren, angeben. Mit Selektoren können Sie auf die Attribute Adresse, Länge und Speichertyp zugreifen, um ihre Ergebnisse ausgeben oder übertragen zu lassen, um sie zu modifizieren oder um auf die Maschinencode-Ebene zu wechseln.

Auf einen Datennamen kann angewandt werden oder folgen:
– Adressselektor
– Längenselektor
– Typselektor
– Zeichenkonvertierungsfunktion
– Adressversatz (•)
– Längenmodifikation
– Typmodifikation
– indirekte Adressierung (->), aber nur wenn der Datenname vom Typ %A ist.

Mit einer Typmodifikation können Sie den Speichertyp oder den damit verbundenen Ausgabetyp verändern. Für die Kommandos %DISPLAY und %SET müssen der Speicherinhalt und der mit der Modifikation vereinbarte Speichertyp zusammen passen.

Mit der Längenmodifikation können Sie von der mit einem Datennamen verbundenen Länge abweichen. Der Datentyp bleibt nicht erhalten, AID nimmt den Speichertyp %X an. Mit der Längenmodifikation dürfen Sie die Bereichsgrenzen nicht überschreiten, d.h. die modifizierte Länge darf nicht größer als die implizite Länge aus dem Längenattribut sein.

Wenn Sie an die impliziten Attribute eines Datennamens nicht mehr gebunden sein wollen, wenden Sie den Adressselektor auf den Datennamen an und setzen dahinter einen Pointer-Operator. Mit %@(datenname)-> sprechen Sie dann die virtuelle Adresse eines Datennamens an, und es gelten die Attribute einer virtuellen Adresse.

Indizes und Subskripte

Ist ein Datenname der Name einer tabellarischen Struktur, kann er wie in einer Anweisung der Programmiersprache indiziert werden. COBOL unterscheidet zwischen Indizierung und Subskribierung, wobei die Subskribierung der Indizierung bei anderen Programmiersprachen entspricht. Besonderheiten bei der Behandlung des COBOL-Indizes durch AID sind im Handbuch „Testen von COBOL-Programmen“ beschrieben.

Der Index kann folgendermaßen angegeben werden:

```
{n                    }
{datenname            }
{arithmetischer ausdruck}
```

n ist eine Ganzzahl mit einem Wert $-2^{31} \leq n \leq 2^{31}-1$.

datenname
: bezeichnet den zu dem Vektor definierten Index oder eine numerische Variable, die im selben Programmteil wie der Vektor liegen muss; d.h. für den Index wird die Qualifikation des Vektors übernommen.

arithmetischer Ausdruck
: Der Wert für *index* wird von AID errechnet. Erlaubt sind die arithmetischen Operatoren (+, –, /, ∗) und die oben aufgeführten Operanden *n* und *datenname*.
Für COBOL gilt, dass in einem arithmetischen Ausdruck nur das Subskript, nicht aber der Index verwendet werden kann.

Es kann auch ein Bereich von Indizes angegeben werden. Dies erfolgt in der Form:

index1:*index2*
: Damit wird der Bereich zwischen *index1* und *index2* bezeichnet. Beide müssen innerhalb der Indexgrenzen liegen und *index1* muss kleiner oder gleich *index2* sein.

Beispiele

1. `%DISPLAY V'10A'%T(SYMBOL)`
   Der Speicherinhalt ab Adresse V’10A’ wird mit dem Speichertyp von SYMBOL interpretiert und in dem zugehörigen Ausgabetyp und der Länge von SYMBOL ausgegeben. AID prüft, ob der Speicherinhalt von V’10A’ und der Speichertyp von SYMBOL vereinbar sind.

2. `%DISPLAY %@(DATAREC)->.4%T(INPUT)`
   Für DATAREC gibt es keine Datenstruktur-Beschreibung; aber die Struktur von INPUT entspricht der von DATAREC mit dem Unterschied, dass in DATAREC eine 4 Bytes lange Nummer voransteht. Mit Adressselektor und nachfolgendem Pointer-Operator wird die virtuelle Adresse von DATAREC verwendet. Damit haben die Bereichsgrenzen keine Bedeutung mehr. Mit dem Adressversatz werden die ersten 4 Bytes von DATAREC übersprungen. Mit dem Typselektor wird der Speichertyp und die Länge von INPUT ausgewählt und der Speicherinhalt ab der errechneten Adresse entsprechend ausgegeben.

7.2.2.2 Anweisungsnamen und Source-Referenzen

Anweisungsnamen sind die Namen, die im Quellprogramm für Marken, Kapitel, Paragraphen oder Marken-/Eingangskonstanten bzw. -variablen vergeben wurden, je nachdem was die jeweilige Programmiersprache anbietet und wie sie es benennt. Die Anweisungsnamen stehen für die Adresse des zu der ersten Anweisung nach der Marke generierten Befehlscodes.
Anweisungsnamen geben Sie in folgender Form an:

L'nummer'	Marke (Fortran)
L'name'	Marke (alle Programmiersprachen); in %DISASSMBLE, %INSERT, %REMOVE ohne L' ...' möglich, falls keine Adressrechnung anschließt.
name	Funktion (C++/C), Programmname (COBOL, Fortran, Assembler; kann nur in den Kommandos %DISASSEMBLE, %INSERT und %REMOVE zur Bezeichnung des Programmanfangs verwendet werden), Entrykonstante oder -variable (PL/I)

Source-Referenzen sind die vom Compiler erzeugten Nummern oder Namen von Anweisungen, die in den LSD-Sätzen hinterlegt sind und mit denen Sie die Anweisungen ansprechen können, die weder am Anfang eines Haupt- oder Unterprogramms stehen noch eine Marke haben. Wie der jeweilige Compiler die Anweisungen benennt und was er dazu in der Übersetzungsliste eingeträgt, ist in den sprachspezifischen Handbüchern beschrieben. Die Source-Referenzen stehen für die Anfangsadresse des zu einer Anweisung generierten Befehlscodes.
Source-Referenzen geben Sie in folgender Form an:
S'nummer/name'

Eigenschaften

Source-Referenzen und Anweisungsnamen sind Adresskonstanten. Sie belegen keinen Speicherplatz, haben also kein Adressattribut und können somit auch nicht verändert werden. Eine Ausnahme bilden Entry- und Markenvariablen bei PL/I, für die wie für alle anderen Daten Speicherplatz angelegt wird und die Sie mit %SET überschreiben können.

Wenn LSD-Sätze erzeugt wurden und zur Verfügung stehen, kann AID über Anweisungsnamen und Source-Referenzen auf den Befehlscode zugreifen.

Als einfache Speicherreferenzen können Sie Anweisungsnamen und Source-Referenzen in den Kommandos %DISASSEMBLE, %JUMP, %INSERT und %REMOVE verwenden. Mit Source-Referenzen können Sie außerdem in den Kommandos %CONTROL*n* und

%TRACE einen Speicherbereich angeben. Bei %DISPLAY, %MOVE und %SET sprechen Sie den Wert der Adresskonstanten an, den Sie allerdings nur als *sender* verwenden können.

Eine Ausnahme bilden Kapitel- und Paragraphennamen bei COBOL. AID kennt von ihnen nicht nur die Adresse des ersten Befehls, sondern auch das Ende des Kapitels oder Paragraphen. Sie können sie deshalb zusätzlich bei den Kommandos %CONTROL*n* und %TRACE zur Bereichsangabe verwenden.

Ansonsten können Sie mit Anweisungsnamen und Source-Referenzen nur dann eine Speicherstelle ansprechen, wenn ihnen ein Pointer-Operator folgt. AID steht nur die Adresskonstante zur Verfügung. Die übrigen Attribute können Sie nicht verwenden.

Auf einen Anweisungsnamen kann nur die indirekte Adressierung (->) folgen.

Beispiele

1. ```
   %DISPLAY L'SUMMIEREN'
   %DISPLAY L'SUMMIEREN'->
   ```
   Mit dem ersten %DISPLAY gibt AID die Adresse des Befehlscodes aus, der zu der ersten Anweisung nach der Marke SUMMIEREN generiert wurde.
   Mit dem zweiten %DISPLAY gibt AID 4 Bytes Speicherinhalt ab dieser Adresse aus.

2. ```
   %INSERT S'123'
   %INSERT S'123'->.(-6)
   ```
   Im ersten %INSERT setzen Sie einen Testpunkt auf die Adresse des Befehlscodes, der zu der Anweisung 123 generiert wurde. Sie benutzen die Source-Referenz S'123' als einfache Speicherreferenz.

   Im zweiten %INSERT setzen Sie einen Testpunkt auf die Adresse des Befehlscodes, der 6 Bytes vor dem Testpunkt aus dem ersten %INSERT liegt. Sie benutzen die Source-Referenz S'123' in einer komplexen Speicherreferenz und müssen deshalb ihre Eigenschaften als Adresskonstante beachten und den Pointer-Operator dahinter schreiben.

3. ```
   %MOVE L'123' INTO %2G
   %MOVE X'D2' INTO L'123'->
   ```
   Zu der Marke 123 in einem Fortran-Programm sei die Adresse V'A1A' in den LSD-Sätzen hinterlegt. Mit dem ersten %MOVE überträgt AID diese Adresse in das AID-Register %2G. Mit dem zweiten %MOVE überträgt AID das Sedezimal-Literal X'D2' auf die Speicherstelle mit der Adresse V'A1A'.
   ```

7.2.3 Schlüsselwörter

Speicherobjekte, die außerhalb des Programmspeichers liegen, aber vom Programm oder von AID benutzt werden, kann AID über Schlüsselwörter ansprechen. Das sind die Mehrzweckregister %0-%15, die Gleitpunktregister %nE, %nD und %nQ, die Zugriffsregister %nAR, der Befehlszähler %PC, die AID-Register %0G-%15G und %nGD sowie der Durchlaufzähler %•*subkdoname*. Außerdem können Sie den Klasse-5- und Klasse-6-Speicher mit den Schlüsselwörtern %CLASS5, -ABOVE und -BELOW und %CLASS6, -ABOVE und -BELOW ansprechen. Alle anderen Schlüsselwörter können nicht als Speicherreferenz verwendet werden. Vor einem Schlüsselwort kann nur eine Basisqualifikation angegeben werden.
Alle Schlüsselwörter, die Sie bei AID verwenden können, sind im Kapitel „Schlüsselwörter“ auf Seite 111 beschrieben.

Schlüsselwörter haben die Attribute:

```
Name (%name)
Adresse
Inhalt
Länge
Speichertyp
Ausgabetyp
```

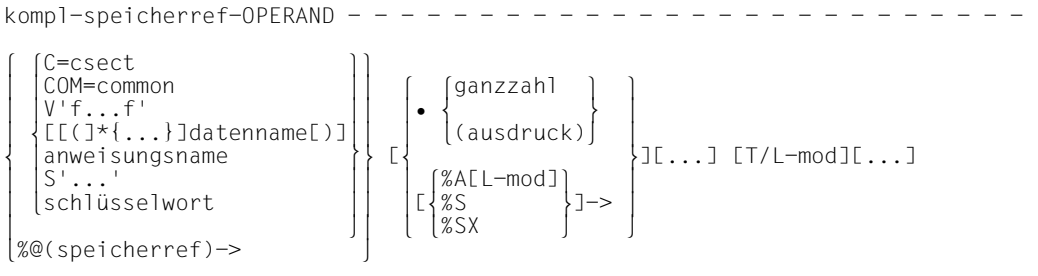
Die Bereichsgrenzen werden durch die Anfangsadresse und die Länge bestimmt.

Auf ein Schlüsselwort kann angewandt werden oder folgen:
– Adressselektor (das Ergebnis ist nicht benutzbar, wenn die Adresse außerhalb des Benutzerbereichs liegt)
– Längenselektor
– Adressversatz (•)
– indirekte Adressierung (->)
– Längenmodifikation
– Typmodifikation

Bei Adressversatz und Längenmodifikation überprüft AID die Bereichsgrenzen. Mehrzweckregister können Sie ohne Typmodifikation vor einem Pointer-Operator verwenden, obwohl sie vom Typ %F sind.

7.2.4 Komplexe Speicherreferenzen

In einer komplexen Speicherreferenz wird, von einer symbolischen oder maschinennahen Speicherreferenz oder von einem Schlüsselwort ausgehend, eine Adressrechnung durchgeführt. Das Ergebnis einer komplexen Speicherreferenz ist ohne abschließende Typ- und Längenmodifikation eine virtuelle Adresse mit Speichertyp %XL4. Mit einer Typ- oder einer Längenmodifikation oder beiden Modifikationen können Sie der errechneten Adresse jedoch den von Ihnen gewünschten Speichertyp zuordnen.



C=csect	C-Qualifikation
COM=common	Common-Qualifikation
V'f...f'	virtuelle Adresse
datenname	Datenname
anweisungsname	Anweisungsname
S'...'	Source-Referenz
schlüsselwort	Schlüsselwort
%@(speicherref)	Adressselektor
T/L-mod	Typ- und/oder Längenmodifikation
%A, %S, %SX	Speichertypen zur Adressinterpretation
• {...}	Adressversatz (Byte-Offset)
->	indirekte Adressierung (Pointer-Operator)
*	indirekte Adressierung (Inhaltsoperator, nur C++/C)
(ausdruck)	arithmetischer Ausdruck

Adressversatz, indirekte Adressierung, Typ- und Längenmodifikation, arithmetischer Ausdruck und der Adressselektor sind in den unmittelbar folgenden Abschnitten beschrieben. Alle übrigen Begriffe finden Sie am Anfang dieses Kapitels beschrieben.

7.2.4.1 Adressversatz "."

Mit dem Adressversatz oder Byte-Offset können Sie in Byte-Schritten von einer Adresse aus vorwärts oder zurück gehen. Das Ergebnis eines Adressversatzes ist immer eine virtuelle Adresse. Beim Adressversatz dürfen die Bereichsgrenzen des betroffenen Speicherobjekts nicht überschritten werden.

- Offset-Operator

speicherreferenz
kann jede, wie auch immer angesprochene Speicherstelle sein: virtuelle Adresse, Datenname, Schlüsselwort, C-Qualifikation oder komplexe Speicherreferenz.

zahl
ist eine positive ganze Dezimal- oder Sedezimalzahl zwischen 0 und 2^{31}-1.

ausdruck
ist ein von AID errechneter Wert zwischen -2^{31} und 2^{31}-1. *ausdruck* ist in Abschnitt „Adress-, Typ- und Längenselektor“ auf Seite 95 beschrieben. Er wird gebildet aus Zahlen, numerischen Inhalten von Speicherreferenzen, dem Ergebnis von Adress-und Längenselektor sowie Längenfunktion und den arithmetischen Operatoren (+ – ∗ /).

Der Adressversatz kann nur innerhalb der Bereichsgrenzen des betroffenen Speicherobjekts durchgeführt werden. Das Ergebnis eines Adressversatzes ist eine virtuelle Adresse mit der Länge 4. Diese 4 Bytes müssen innerhalb der Bereichsgrenzen des Speicherobjekts Platz haben. Werden die Bereichsgrenzen überschritten, gibt AID eine Fehlermeldung aus.

Außer bei den virtuellen Adressen werden die Bereichsgrenzen durch die Anfangsadresse und das Längenattribut bestimmt. Für eine virtuelle Adresse ist der gesamte virtuelle Speicherbereich (V’0’ bis V’7FFFFFFF’) als Bereich zugeordnet. Bei Datennamen, den Schlüsselwörtern %CLASS6, -ABOVE, -BELOW und der C-Qualifikation können Sie mit Adressselektion und nachfolgendem Pointer-Operator, %@(...)->, die symbolische Ebene verlassen, dann gelten die Bereichsgrenzen einer virtuellen Adresse.

Auf Adressversatz kann folgen:
– Adressversatz (•)
– indirekte Adressierung (->)
– Längenmodifikation
– Typmodifikation

Beispiele

1. `%DISPLAY SYMBOL.10`
   `%DISPLAY %@(SYMBOL)->.10`
   SYMBOL hat die Länge 10. Ein Adressversatz um 10 Bytes kann AID nicht durchführen, da dadurch die ersten 4 Bytes nach SYMBOL angesprochen und somit die Bereichsgrenzen von SYMBOL überschritten würden. AID gibt eine Fehlermeldung aus. Wenn Sie mit Adressselektion und nachfolgendem Pointer-Operator auf die Maschinencode-Ebene wechseln, gelten die Bereichsgrenzen einer virtuellen Adresse. Mit dem Adressversatz positionieren Sie wie im ersten %DISPLAY auf das erste Byte nach SYMBOL. Diesen %DISPLAY kann AID jetzt ausführen und gibt die ersten 4 Bytes nach SYMBOL aus.

2. `%D %@(VAR)->.(%L(ELEM(1))*5)%T(ELEM(1))`
   In einem COBOL-Programm sei ein Vektor ELEM mit 10 Elementen ELEM(1) bis ELEM(10) enthalten. Die Variable VAR soll als Vektor in der Struktur von ELEM redefiniert werden, und das 6. Element davon soll ausgegeben werden. AID geht ab der Anfangsadresse von VAR mit einem Adressversatz um den Wert vorwärts, der sich aus der Länge von ELEM multipliziert mit 5 ergibt. Durch die anschließende Typmodifikation erreichen Sie, dass der Inhalt an der errechneten Adresse in Typ und Länge eines Elementes von ELEM ausgegeben wird.
   Hätten Sie ELEM ohne Index angegeben, dann hätte AID Typ und Länge des gesamten Vektors ELEM genommen.

3. `%D %5->.(%L(INDEX)*%L(ADRESSE))%CL=(%L(ADRESSE)+50)`
   Der Inhalt von Register 5 (X'00000A00') wird als Adresse eingesetzt. Ab Adresse V'A00' wird um die Länge von Index (2) multipliziert mit der Länge von ADRESSE (7) weiterpositioniert. Der Speicherinhalt an der Adresse V'A0E' (#'A00' + (2*7) = #'A0E') wird im Character-Format in der Länge 57 (7+50) ausgegeben.

4. `%D S'123COMP'->.8%S->%L10`
   Für die Source-Referenz S'123COMP' (COBOL) ist die Adresse V'1B0' in den LSD-Sätzen hinterlegt. Mit dem Pointer-Operator wird auf die Stelle im Speicher mit der Adresse V'1B0' positioniert. Mit dem Adressversatz wird um 8 Bytes weitergegangen. Der Inhalt X'600F0130' an dieser Stelle, das ist an Adresse V'1B8', wird mit %S interpretiert. Aus Basisregister 6, Inhalt X'000B010', und Distanz #'00F' ergibt sich die Adresse V'B01F', die mit dem Pointer-Operator angesprochen wird. Ab dieser Adresse werden 10 Bytes im Dump-Format ausgegeben.

5. `%D C=CS1.(%L(C=CS1))`
   `%D C=CS1.(%L(C=CS1)-4)`
   Der erste %DISPLAY wird abgewiesen, da ein Adressversatz in der Länge von CS1 das erste Byte hinter CS1 anspricht und somit die Bereichsgrenzen von CS1 überschritten würden.
   Beim zweiten %DISPLAY ist der Adressversatz um 4 Bytes kürzer. Die Bereichsgrenzen von CS1 werden nicht verletzt und AID gibt die letzten 4 Bytes von CS1 aus.

6. `%D V'4'.(-5).4`
   `%D V'4'.(4-5)`
   `%D V'4'.4.(-5)`
   Der Adressversatz aus dem ersten %DISPLAY wird abgelehnt. Mit (-5) wird die untere Bereichsgrenze von virtuellen Adressen V'0' verletzt, auch wenn das Endergebnis durch den zweiten Adressversatz innerhalb des zulässigen Bereichs liegen würde. Im zweiten und dritten %DISPLAY werden bei jedem Adressversatz die Bereichsgrenzen eingehalten. AID gibt 4 Bytes ab Adresse V'3' aus.

7. `%D V'100' .(%1 + %2)`
   Die Adresse V'100' wird um die Summe aus dem Inhalt von Register 1 und Register 2 erhöht.

7.2.4.2 Indirekte Adressierung "->" / "*"

Bei einer indirekten Adressierung benutzt AID eine Adresskonstante oder einen Speicherinhalt als Adresse für eine andere Speicherstelle. Wird der Pointer-Operator als unärer Operator eingesetzt, dann ist das Ergebnis eine virtuelle Adresse. Die Pointer-Operation bewirkt also einen Übergang auf die Maschinencode-Ebene. Wird die Indirektion mit dem Pointer-Operator als binärem Operator durchgeführt oder verwenden Sie den Inhaltsoperator, dann bleiben Sie auch nach der indirekten Adressierung auf der symbolischen Ebene, und das Ergebnis wird gemäß der entsprechenden Datendefinition aus dem Quellprogramm aufbereitet.

Bei jeder 4 Bytes langen Adresse, die zur indirekten Adressierung verwendet wird, berücksichtigt AID den aktuellen Adressierungsmodus des Testobjekts. Er ist mit %DISPLAY %AMODE abfragbar. Mit %AINT kann für die Pointer-Operation eine andere Adress-Interpretation vereinbart werden.

Pointer-Operator

```
indirekte-adressierung mit Pointer-Operator - - - - - - - - - - - - - - - - -

⎧adresskonstante                        ⎫     ⎡⎧strukturkomponente⎫⎤
⎨                                       ⎬ -> ⎢⎨                  ⎬⎥
⎩speicherreferenz [%A[Ln] | %S | %SX]   ⎭     ⎣⎩BASED-variable    ⎭⎦

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

-> Pointer-Operator

adresskonstante
: Anweisungsnamen, Source-Referenzen und das Ergebnis einer Adressselektion sind Adresskonstanten. Namen von Marken müssen vor "->" in L'...' gesetzt werden.

speicherreferenz
: kann jede Speicherstelle sein, die eine Adresse enthält. Daten vom Typ Adresse können Sie ohne Typmodifikation verwenden.

[%A[Ln] | %S | %SX]
: Typmodifikation, mit der eine Speicherstelle als Adresse interpretiert werden kann. Mit %S und %SX können Sie die Adressbildung von Maschinenbefehlen nachvollziehen. Aus Basisregister und Distanz (%S) oder Indexregister, Basisregister und Distanz (%SX) errechnet AID die Adresse wie die Hardware (siehe Abschnitt „Speichertypen zur Interpretation von Maschinenbefehlen“ auf Seite 112).

```
⎧strukturkomponente⎫
⎩BASED-variable    ⎭
```

: In diesen beiden Fällen wird der Pointer-Operator verwendet, um eine indirekte Adressierung nachzuvollziehen, die zum Sprachumfang der Programmiersprache gehört, nämlich um in C++/C eine Strukturkomponente über einen Zeiger zu referenzieren oder um in PL/I eine BASED-Variable über den zugehörigen Pointer anzusprechen. Für das Ergebnis der Indirektion gelten die im Quellprogramm definierten Attribute der Strukturkomponenten bzw. BASED-Variablen.

Inhaltsoperator

In C++/C können Sie zur Dereferenzierung statt des Pointer-Operators auch den Inhaltsoperator verwenden.

Die mit dem Inhaltsoperator referenzierte Adresse wird entsprechend ihrem im Programm vereinbarten Datentyp interpretiert. Sie wechseln also nicht auf Maschinencode-Ebene wie dies bei unärer Dereferenzierung durch den Pointer-Operator geschieht.
Im Gegensatz zu C++/C, wo Sie den Inhaltsoperator auch auf Vektoren anwenden können, ist der Inhaltsoperator in AID nur für Pointer zugelassen.

```
indirekte-adressierung mit Inhaltsoperator - - - - - - - - - - - - - - - - - -

[(]* {...} pointer-variable[)]

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

* Inhaltsoperator

pointer-variable
typbezogener Zeiger eines C++/C-Programms

Der Inhaltsoperator kann mehrfach wiederholt werden. Gegebenenfalls müssen Sie die Reihenfolge der Verarbeitung durch Klammerung festlegen. Der Inhaltsoperator wird nachrangig nach Pointer-Operator, Adressversatz und Indizierung ausgewertet wird.

Auf indirekte Adressierung kann folgen:
– Adressversatz "•"
– indirekte Adressierung (->)
– Längenmodifikation
– Typmodifikation

Beispiele

```
1. %DISPLAY V'10A', V'10A'->
```

AID-Ausgabe

```
V'0000010A' = ABSOLUT  + #'0000010A'
0000010A (0000010A) 00000478                                    ....

V'00000478' = ABSOLUT  + #'00000478'
00000478 (00000478) E3C5E7E3                                    TEXT
```

AID gibt vier Bytes ab Adresse V’10A’ im Format DUMP aus. Zur Ausgabe des zweiten Operanden verwendet AID diesen Speicherinhalt (X’00000478’) als Adresse in einer Pointer-Operation und gibt vier Bytes ab Adresse V’478’ im Format DUMP aus.

2. `%FIND C'***'`
   `%DISPLAY %1G->`
   Mit %FIND suchen Sie die Zeichenfolge '***' im Speicher. Wenn AID die Zeichenfolge findet, wird im AID-Register %1G die Fortsetzungsadresse abgelegt, das ist die Adresse des ersten Bytes nach der gefundenen Zeichenfolge. Mit dem %DISPLAY-Kommando lassen Sie sich den auf den Suchbegriff folgenden Speicherinhalt ausgeben.

3.

```
%SET %7 INTO V'14C0'%SX->
Inhalt von Mehrzweckregister 4:        X'00000100'
Inhalt von Mehrzweckregister 6:        X'00004000'
Speicherinhalt ab Adresse V'14C0':     X'50746B00' ≙ ST R7,X'B00'(R4,R6)
```

   AID simuliert eine Übertragung mit dem Befehl Speichern (ST, Befehlscode X'50') und überträgt den Inhalt von Mehrzweckregister 7 ab Adresse V'4C00'. Die Adresse errechnet sich AID aus dem Speicherinhalt X'50746B00' wie folgt:

```
X'507' wird ignoriert
X'4' nimm Inhalt von Register 4:           '00000100'
X'6' addiere Inhalt von Register 6:        '00004000'
X'B00' addiere Distanz:                         'B00'
-------------------------------------------------------
ergibt die Adresse                             '4C00'
```

4. `%SET X'C1C2C3C4' INTO V'14C2'%S->`
   Register- und Speicherinhalte sind dieselben wie in Beispiel 3.
   Mit diesem %SET übertragt AID das Sedezimal-Literal X'C1C2C3C4' auf die Speicherstelle mit der Adresse V'4B00'. Die Adresse errechnet sich AID aus dem Speicherinhalt X'6B00' wie folgt:

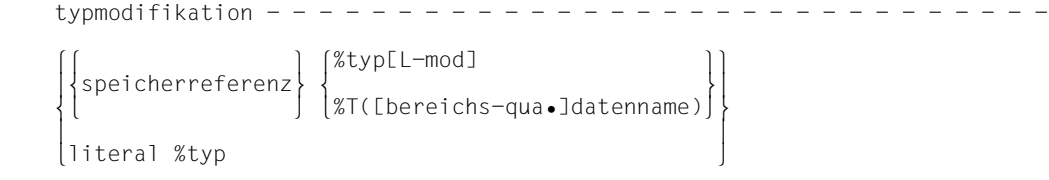
```
X'6' nimm Inhalt von Register 6:         '00004000'
X'B00' addiere Distanz:                       'B00'
-----------------------------------------------------
ergibt die Adresse                           '4B00'
```

7.2.4.3 Typmodifikation

Die Typmodifikation verwenden Sie, um einen Speicherinhalt anders interpretieren zu lassen als sein Speichertyp-Attribut vorgibt. Das kann in den folgenden Fällen notwendig sein:

- Anpassung der Typen beim %SET
- anderes Ausgabeformat bei %DISPLAY
- Konvertierung eines Literals (nur bei %DISPLAY zugelassen)
- Interpretation als Adresse vor einem Pointer-Operator
- Interpretation bzw. Aufbereitung in einer anderen Struktur (Redefinition einer Speicherstelle)
- Interpretation als Ganzzahl in einem Ausdruck.

Außer vor einem Pointer-Operator ist die Typmodifikation nur am Ende einer komplexen Speicherreferenz sinnvoll, um den Speicherinhalt ab der errechneten Adresse oder das Literals im gewünschten Speichertyp zu interpretieren.



speicherreferenz
kann jede, wie auch immer angesprochene Speicherstelle sein: virtuelle Adresse, Datenname, Schlüsselwort, C-Qualifikation

literal
Die AID-Literale sind ab Seite 103 beschrieben.

%typ[L-mod]
Schlüsselwort für Speichertypen mit wahlweiser Längenangabe:
%X, %C, %P, %D, %F, %A und %UTF16 (siehe Kapitel „Schlüsselwörter“ auf Seite 111).
Die Länge können Sie mit allen Möglichkeiten der Längenmodifikation angeben. Ohne Längenangabe bleibt das Längenattribut des modifizierten Speicherobjekts erhalten.
Die Speichertypen %H, %Y, %S, %SX haben eine feste Länge. Sie können deshalb nur ohne Längenangabe verwendet werden.
Beim Speichertyp %UTF16 muss die Länge ein Vielfaches von 2 sein.
Die Längenmodifikation ist bei Literalen nicht zugelassen.

%T([bereichs-qua•]datenname)
: Mit dem Typselektor verwenden Sie Speichertyp und Länge von anderen Datendefinitionen zur Interpretation einer Speicherstelle. Sie müssen deshalb auch die Regeln der Längenmodifikation beachten, z.B Überschreiten der Bereichsgrenzen.
datenname kann qualifiziert sein, d.h. aus einem anderen Programmteil stammen. Eine Basisqualifikation können Sie jedoch nicht angeben.

Bei der Typmodifikation überprüft AID, ob der Speicherinhalt zu dem gewählten Speichertyp passt. Andernfalls gibt AID eine Fehlermeldung aus.

Jedem Speichertyp ist ein Ausgabetyp zugeordnet (siehe Kapitel „Schlüsselwörter“ auf Seite 111). Sie können deshalb mit der Typmodifikation auch den Ausgabetyp verändern.

Die Speichertypen %D, %P, %F und %A haben nur bestimmte zugelassene Längen (siehe Kapitel „Schlüsselwörter“ auf Seite 111). Wenn sie ohne Längenangabe eingesetzt werden, muss die Länge des modifizierten Speicherobjekts mit einer der zugelassenen Längen des Speichertyps übereinstimmen. Andernfalls weist AID eine Typmodifikation mit einem Längenfehler ab.
Die Speichertypen %S, %H und %Y haben die feste Länge 2 Bytes, %SX hat die feste Länge 4 Bytes. Mit ihnen wird zugleich eine Längenmodifikation ausgeführt. Die Längenmodifikation muss innerhalb der Bereichsgrenzen stattfinden können.

Die Typmodifikation mit dem Speichertyp %UTF16 ist erlaubt, wenn die (implizite) Länge der Speicherstelle ein Vielfaches von 2 ist.
Die Typmodifikation %UTF16 ist auf X-Literalen erlaubt, aber auf C- und U-Literalen verboten.

Da das %SET-Kommando bei der Übertragung den Typ und die Länge berücksichtigt und gegebenenfalls vor einer numerischen Übertragung den Speicherinhalt des Sendefeldes in den des Empfangsfeldes konvertiert, müssen die Datentypen von Sende- und Empfangsfeld verträglich sein. Dazu finden Sie eine Tabelle in der %SET-Beschreibung der sprachspezifischen Handbücher [2]-[6]. Sind die Datentypen nicht vereinbar, so können Sie einen zum Speicherinhalt passenden und mit dem Empfangsfeld verträglichen Speichertyp angeben.
Für Programmiersprachen, die die Definition von Strukturen vorsehen, gilt bezüglich des %SET die folgende Einschränkung: Strukturen können Sie mit einem %SET-Kommando nur modifizieren, wenn Sende- und Empfangsfeld dieselbe Struktur haben. Wurde eine der Adressen beim Schreiben des Programms nicht als Struktur beschrieben, so können Sie ihr mit der Typselektion die erforderliche Struktur zuweisen. Dann muss allerdings der aktuelle Speicherinhalt der Definition dieser Struktur entsprechen.

Beispiele

1. `%DISPLAY V'10A'%F`
   Der Speicherinhalt ab Adresse V'10A' wird als Binärwert mit Vorzeichen interpretiert und als Ganzzahl mit Vorzeichen ausgegeben. Ohne die Typmodifikation hätte die virtuelle Adresse den Speichertyp Sedezimal (%X) und die Länge 4 Bytes und damit verbunden den Ausgabetyp DUMP.

2. `%INSERT V'4710'%SX->`
   Der Speicherinhalt der Adresse V'4710' wird zur Berechnung des Testpunktes entsprechend dem %SX-Format ausgewertet.

3. `%SET RECORD.10%PL5 INTO AMOUNT`
   In einem COBOL-Programm gibt es ein Datenfeld RECORD mit einer Länge von 45 Bytes, das eine Folge von gepackten Zahlen mit je 5 Bytes Länge enthält. AMOUNT ist ein numerisches, entpacktes Datenelement. Die ersten beiden Zahlen werden mit dem Adressversatz übersprungen. Durch die Typ- und Längenmodifikation wird die dritte gepackte Zahl aus RECORD entpackt und rechtsbündig nach AMOUNT übertragen.

4. `%D V'134'.(INDEX * 4)%T(LINE)`
   Zur virtuellen Adresse V'134' werden durch Adressversatz soviel Bytes dazugerechnet, wie sich aus dem Inhalt von INDEX multipliziert mit 4 ergeben. Der Speicherinhalt an der errechneten Adresse wird entsprechend dem Typ und der Länge der Datendefinition zu LINE aufbereitet und ausgegeben.

5. `%DISPLAY %1%F`
   Ohne Typmodifikation würde AID den Inhalt des Registers 1 als Sedezimalzahl ausgeben. Mit der Typmodifikation %F bereitet AID den Registerinhalt vor der Ausgabe als Ganzzahl mit Vorzeichen auf.

6. `%DISPLAY X'20AC'%UTF16`
   Durch die Typmodifikation erfolgt die Ausgabe im Dump-Format, d.h. neben dem Sedezimal-Code 20AC wird die Interpretation als UTF16-Code ausgegeben, in diesem Fall das Eurozeichen.

7.2.4.4 Längenmodifikation

Mit der Längenmodifikation weichen Sie von der Länge einer Speicherreferenz ab. AID greift dann nicht auf die im Längenattribut hinterlegte Länge zu, sondern übernimmt die von Ihnen angegebene. Der Wert einer Längenmodifikation muss zwischen 1 und 65 535 liegen.

Ohne Angabe von *typ* beinhaltet die Längenmodifikation eine Typmodifikation in den Speichertyp %X.

Mit der Längenmodifikation dürfen die Bereichsgrenzen des modifizierten Speicherobjekts nicht überschritten werden, d.h. mit der neugebildeten Länge dürfen Sie die Endadresse nicht überschreiten.

```
längenmodifikation - - - - - - - - - - - - - - - - - - - - - - - - - - - -

                          ⎧Ln                    ⎫
speicherreferenz %[typ]   ⎨L(speicherreferenz)   ⎬
                          ⎩L=(ausdruck)          ⎭

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

speicherreferenz
: kann jede, wie auch immer angesprochene Speicherstelle sein: virtuelle Adresse, Datenname, Schlüsselwort, C-Qualifikation oder komplexe Speicherreferenz

typ
: Wenn Sie eine Typ- und Längenmodifikation durchführen wollen, geben Sie ein Schlüsselwort für Speichertypen an (%X, %C, %P, %D, %F, %A, %UTF16) und danach das L ohne weiteres Prozentzeichen.
Beispiel:
VAR1%XL5 oder VAR1%CL5

%Ln
: Eine Längenmodifikation, die mit %L beginnt, beinhaltet zugleich eine Typmodifikation in den Standard-Speichertyp %X.

  n ist eine positive Ganzzahl oder Sedezimalzahl mit $0 \leq n \leq 65\,535$, entsprechend dem zulässigen Wert einer Längenmodifikation.

%L(speicherreferenz)
: Mit dem Längenselektor verwenden Sie das Längenattribut einer anderen Speicherreferenz für die Längenmodifikation. Der Längenselektor ist sinnvoll bei Datennamen, C- und COM-Qualifikationen.

%L=(ausdruck)

Mit der Längenfunktion lassen Sie sich die Länge von AID errechnen. *(ausdruck)* ist in Abschnitt „Adress-, Typ- und Längenselektor“ auf Seite 95 beschrieben. Er wird gebildet aus Ganzzahlen, Inhalten von Speicherreferenzen, die vom Typ Integer (%F oder %A) und einer Länge ≤ 8 sein müssen, dem Ergebnis von Adressselektor, Längen-selektor und Längenfunktion und den arithmetischen Operatoren (+ – * /).
Die beteiligten Operanden und das resultierende Ergebnis müssen im Wertebereich eines %FL8 Feldes liegen.
Steht im *ausdruck* einer Längenfunktion nur eine Speicherreferenz, nimmt AID den Inhalt, nicht die Länge, als Wert für die Längenmodifikation.

Es wird der Wertebereich $-2^{63} \leq n < +2^{64}$ unterstützt. Damit können die Datentypen %FL8 mit den Werten $-2^{63} \leq n < +2^{63}$ und %AL8 mit den Werten $0 \leq n < +2^{64}$ korrekt dargestellt werden. Wenn das Ergebnis nicht in den Wertebereich passt, kommt es zu einer Fehlermeldung AID0470.

Verwenden Sie einen Längenselektor auf einen Vektor, ohne einen Index dazu anzugeben, wird die Länge des gesamten Vektors selektiert. Nur mit Angabe des Index kann AID auf die Länge eines Elements des Vektors zugreifen.

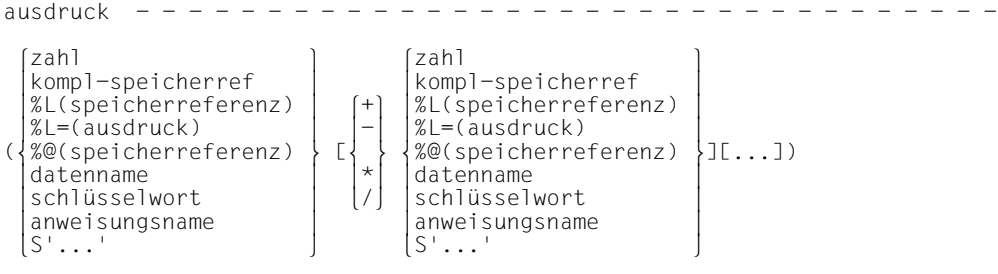
Beispiele

1. `%DISPLAY V'10A'%L=(VAR1)`
   VAR1 ist vom Typ Integer und enthält den Wert 23. Ab der Adresse V’10A’ werden 23 Bytes im Ausgabeformat DUMP ausgegeben.
2. `%SET CVAR1%CL(CVAR) INTO CVAR`
   CVAR1 und CVAR seien zwei Character-Variablen in einem Fortran-Programm. und CVAR1 sei länger als CVAR. Die Längenmodifikation ermöglicht es Ihnen, CVAR1 linksbündig in der Länge von CVAR zu übertragen.
3. `%SET %L(CVAR) INTO %2G`
   Die Länge der Variablen CVAR wird in AID-Register %2G übertragen.
4. `%DISPLAY V'10A'%AL3->`
   Der Inhalt der drei Bytes ab Adresse V’10A’ wird als Adresse interpretiert. AID gibt ab der damit angesprochenen Speicherstelle 4 Bytes im Dump-Format (%XL4) aus.
5. `%D V'10A'%L=(INDEX*12-%L(NAME))`
   Hier ergibt sich die Länge aus der Multiplikation des Inhalts von INDEX mit 12. Davon wird Länge von NAME abgezogen.
6. `%D V'4700'%L=(%L(C=CS1)-%L(INDAT))`
   Die Länge wird berechnet aus der Länge der CSECT CS1 abzüglich der Länge von INDAT.

7.2.4.5 Arithmetischer Ausdruck

Beim Adressversatz, in einer Längenfunktion und im Index können Sie einen arithmetischen Ausdruck schreiben, um von AID den benötigten Wert errechnen zu lassen. Einen Ausdruck können Sie also überall dort einsetzen, wo ein ganzzahliger Wert erforderlich ist. Die Rechenoperatoren werden von AID entsprechend den mathematischen Regeln bei der Auflösung eines arithmetischen Ausdrucks abgearbeitet. Durch Setzen von runden Klammern können Sie eine andere Reihenfolge der Abarbeitung festlegen. Vor und nach einem Minuszeichen "–" empfiehlt es sich, ein Leerzeichen zu schreiben, so dass auch bei der Einstellung %AID SYMCHARS[=STD] keine Fehlinterpretationen vorkommen können.

Für jeden Schritt der Abarbeitung von *ausdruck* gilt: $-2^{63} \leq$ Zwischenergebnis $\leq 2^{63}-1$
Für den Adressversatz gilt: $-2^{31} \leq$ Endergebnis $\leq 2^{31}-1$
Für die Längenfunktion gilt: $0 \leq$ Endergebnis $\leq 65\,535$
Für den Index gelten die im Quellprogramm festgelegten Grenzwerte. Außerdem können nur die Operanden *zahl* und *datenname* verwendet werden (siehe Abschnitt „Datennamen“ auf Seite 75).



zahl
: ist eine Ganzzahl oder eine Sedezimalzahl mit $-2^{63} \leq zahl \leq 2^{63}-1$.

kompl-speicherref
: kann jede, wie auch immer angesprochene Speicherstelle sein. Ihr Inhalt muss ganzzahlig sein, also vom Typ %F oder %A mit einer Länge ≤8.

  Auf diese Art können Sie den Inhalt einer Speicherreferenz für einen Adressversatz, zur Längenmodifikation oder als Index/Subskript einsetzen.

%L(speicherreferenz)
: Mit dem Längenselektor greifen Sie auf das Längenattribut einer Speicherreferenz zu. Das Ergebnis ist ein ganzzahliger Wert. Der Längenselektor ist sinnvoll bei Datennamen, C- und COM-Qualifikationen. Die Länge von anderen Speicherreferenzen, z.B. Schlüsselwörtern, ist ja bekannt.

%L=(ausdruck)
: Mit der Längenfunktion lassen Sie sich von AID einen ganzzahligen Wert errechnen. Ausdruck entspricht den hier beschriebenen Regeln.

%@(speicherreferenz)
Mit dem Adressselektor greifen Sie auf das Adressattribut einer Speicherreferenz zu. Das Ergebnis ist eine Adresskonstante (%AL4).

datenname
muss im Quellprogramm vom Typ Integer oder Adresse mit einer Länge ≤ 8 definiert sein. Der Inhalt von *datenname* wird zur Berechnung des arithmetischen Ausdrucks herangezogen.

schlüsselwort
Der Inhalt von *schlüsselwort* wird zur Berechnung des arithmetischen Ausdrucks verwendet. Sie können die folgenden Schlüsselwörter (siehe Kapitel „Schlüsselwörter“ auf Seite 111) angeben:

```
%n                  Mehrzweckregister, 0 £ n £ 15
%nG                 AID-Mehrzweckregister, 0 £ n £ 15
%PC                 Befehlszähler (Program Counter)
%•[subkdoname]      Durchlaufzähler. Mit der Kurzform %• bezeichnen Sie
                    den Durchlaufzähler des gerade aktiven Subkommandos.
```

anweisungsname
Da Anweisungsnamen Adresskonstanten sind, können sie in einem Ausdruck eingesetzt werden.

S'...'
Source-Referenzen sind ebenfalls Adresskonstanten. Auch sie können in einem Ausdruck eingesetzt werden.

Beispiele

1. `%D %L=(%1+5)`
   Die Länge, die sich aus dem Inhalt von Register %1 erhöht um 5 ergibt, wird ausgegeben.

2. `%D V'0'.(V'100'%AL2 + %L(C=CSECT))`
   Ab Adresse V'0' wird ein Adressversatz in der Länge des in Klammern angegebenen Ausdrucks durchgeführt. Zunächst wird der Inhalt der beiden Bytes mit den Adressen V'100' und V'101' als positive Ganzzahl in der Länge 2 interpretiert. Anschließend wird die Länge von CSECT dazuaddiert. Ab der so errechneten Speicherstelle gibt AID 4 Bytes im Dump-Format aus.

3. `%S %L=((V'0'%AL4 + V'4'%AL1) * NUM1) INTO %2G`
   Der Wert, den die Längenfunktion errechnet, wird in AID-Register %2G übertragen. V'0' enthält X'00000005', V'4' enthält X'FFF5003A', und NUM1 enthält den Wert 3. Das ergibt nach den Typ- und Längenmodifikationen: (5 + 255) * 3 = 780. Der Wert 780 wird in %2G übertragen.

7.2.4.6 Adress-, Typ- und Längenselektor

Mit den Selektoren greifen Sie auf die Attribute einer Speicherreferenz zu.

```
selektoren - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

%@(speicherreferenz)
%T([bereichs-qua•]datenname)
%L(speicherreferenz)

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

%@(speicherreferenz)
: Mit dem Adressselektor greifen Sie auf das Adressattribut einer Speicherreferenz zu. Das Ergebnis ist eine Adresskonstante (%AL4). Sie kann vor einem Pointer-Operator zum Zugriff auf eine Speicherstelle oder als Binärzahl ohne Vorzeichen in einem Ausdruck eingesetzt werden. Mit %DISPLAY können Sie das Ergebnis der Adressselektion ausgeben lassen.
Der Adressselektor ist sinnvoll bei Datennamen, C- und COM-Qualifikationen. Die Adressen von Schlüsselwörtern, die außerhalb des Benutzerbereichs liegen, können Sie sich zwar mit %DISPLAY ausgeben lassen, jedoch nicht mit einem nachfolgenden Pointer-Operator als Speicherreferenz verwenden.

%T([bereichs-qua•]datenname)
: Mit dem Typselektor greifen Sie auf das Typ- und Längenattribut einer Speicherreferenz zu. Der selektierte Datentyp kann nur zur Typmodifikation eingesetzt werden. *datenname* kann qualifiziert sein.

%L(speicherreferenz)
: Mit dem Längenselektor greifen Sie auf das Längenattribut einer Speicherreferenz zu. Das Ergebnis ist eine positive Ganzzahl. Sie kann zur Längenmodifikation oder in einem Ausdruck eingesetzt werden. Mit %DISPLAY können Sie das Ergebnis der Längenselektion ausgeben lassen.
Der Längenselektor ist sinnvoll bei Datennamen, C- und COM-Qualifikationen. Die Länge von anderen Speicherreferenzen, z.B. Schlüsselwörtern und virtuellen Adressen, ist ja bekannt.

Beispiele

1.
```
%D %@(VAR)
%D %@(VAR)->.8
%S V'2E'.(%@(VAR))%CL2 INTO X
%D V'A1A'%XL=(2+@(VAR))
```
Einsatzmöglichkeiten des Adressselektors:
Im ersten %DISPLAY wird eine Adresse ausgegeben.
Im zweiten %DISPLAY wechseln Sie mit Adressselektion und Pointer-Operator auf Maschinencode-Ebene, um beim Adressversatz nicht durch die Bereichsgrenzen von VAR behindert zu werden.
Im %SET benutzen Sie die Adresse von VAR als Wert für einen Adressversatz.
Im letzten %DISPLAY benutzen Sie den Wert der Adresse von VAR, um die Länge zu errechnen.

2.
```
%D V'1OO'%T(VAR)
%S V'1OO'%T(INT) INTO NUM1
```
Einsatzmöglichkeiten des Typselektors:
Im %DISPLAY wird der Inhalt einer virtuellen Adresse in Typ und Länge von VAR ausgegeben (Redefinition).
Im %SET wird der Inhalt einer virtuellen Adresse im Typ und der Länge der Integervariablen INT interpretiert, damit er werterhaltend in die numerische Variable NUM1 übertragen werden kann.

3.
```
%D %L(VAR)
%S %L(VAR) INTO NUM1
%D V'A1A'.(2+%L(VAR))
%D V'A1A'%L=(%L(VAR)*5)
```
Einsatzmöglichkeiten des Längenselektors:
Im ersten %DISPLAY wird die Länge von VAR ausgegeben.
Im %SET wird der Wert der Länge von VAR in die numerische Variable NUM1 übertragen.
Im zweiten %DISPLAY benutzen Sie die Länge von VAR als Wert in einem Adressversatz-Ausdruck.
Im letzten %DISPLAY benutzen Sie den Wert der Länge von VAR in einer Längenmodifikation.

7.2.4.7 Besonderheiten beim Zusammenwirken der verschiedenen Bestandteile

Beginnt eine komplexe Speicherreferenz mit einer Adresskonstanten (z.B. mit einer Source-Referenz oder einer Marke), muss anschließend der Pointer-Operator geschrieben werden. Namen von Marken müssen dabei stets in L'...' gesetzt werden.
Ohne den Pointer-Operator können Adresskonstanten innerhalb der *kompl-speicherref* überall da stehen, wo auch Sedezimalzahlen geschrieben werden können.
Nach Adressversatz oder Pointer-Operation (Ausnahmen bei C++/C und PL/I, siehe Abschnitt „Adressversatz "."“ auf Seite 82 und Abschnitt „Indirekte Adressierung "->" / "*"“ auf Seite 84) gehen impliziter Speichertyp und implizite Länge der Ausgangsadresse verloren. Falls Sie nicht explizit einen anderen Speichertyp und eine andere Länge vereinbaren, gilt an der errechneten Stelle der Speichertyp %X in der Länge 4, außer Sie setzen die komplexe Speicherreferenz als Empfänger im Kommando %MOVE ein. In diesem Fall geht der Bereich, der mit %MOVE überschrieben werden darf, von der Anfangsadresse von *kompl-speicherref* bis zum Ende des von Ihrem Programm belegten Speichers.

Für keinen Operanden in einer komplexen Speicherreferenz darf der zugeordnete Speicherbereich durch einen Adressversatz oder eine Längenmodifikation überschritten werden, sonst schreibt AID eine Fehlermeldung. Wenn Sie jedoch die Anfangsadresse eines Speicherobjekts nutzen wollen, ohne auf die Bereichsgrenzen achten zu müssen, verwenden Sie die Adressselektion in Verbindung mit dem Pointer-Operator (%@(...)->). Sie verlassen damit die symbolische Ebene, was gleichzeitig beinhaltet, dass Sie auf das Typ- und Längenattribut des angesprochenen Objekts dann nur noch über die entsprechenden Selektoren zugreifen können.

Manche Compiler, wie C++/C und PL/I, erzeugen bei der Übersetzung zu jedem Programm oder Unterprogramm einen Prolog. Mit *funktion* (C++/C) bzw. *entry* (PL/I) ohne anschließenden Pointer-Operator bezeichnen Sie die erste ausführbare Anweisung der entsprechenden Funktion oder Prozedur. Wenn Sie jedoch an *funktion* oder *entry* den Pointer-Operator anschließen, um vom Funktions- oder Prozeduranfang ausgehend weiterzupositionieren, müssen Sie beachten, dass die Adressrechnung dann mit der Anfangsadresse des Prologs beginnt.

Zeichencodierung einer Zeichenfolge

Mit den Konvertierungsfunktkionen %C() und %UTF16() kann die Interpretation der Zeichenfolgen-Codierung verändert werden. Die Speicherreferenz muss vom Typ „Zeichenfolge“ sein, also z.B. vom Typ %C oder %UTF16.

Die Konvertierungsfunktkionen haben nur dann eine Wirkung, wenn %C() auf den Typ %UTF16 oder %UTF16() auf den Typ %C angewandt wird. Hierbei bleiben die Speicherstellen unverändert. AID arbeitet implizit mit der konvertierten Speicherstelle weiter.

Der CCSN für den Typ %C wird aus der %AID EBCDIC Einstellung verwendet. Die aktuell gültigen Einstellungen können mit dem Kommando %SHOW %AID angezeigt werden.

Beipiel:

An der Speicherstelle V'00' steht das Byte X'BB' .

1. `%AID EBCDIC=EDF03IRV`
   Das Byte X'BB' beschreibt demzufolge das Zeichen '[' im CCSN EDF03IRV.
   %UTF16(V'00' %CL1) ergibt den Hexadezimalwert X'005B'.
2. `%AID EBCDIC=EDF03DRV`
   Das Byte X'BB' beschreibt das Zeichen C'Ä' im CCSN EDF03DRV
   %UTF16(V'00' %CL1) ergibt den Hexwert X'00C4'.

8 Operand Medium-und-Menge

Der Operand *medium-u-menge* legt fest, mit welchem Ausgabemedium AID arbeiten soll und ob es außer dem Inhalt des bezeichneten Speicherbereichs (Daten) Zusatzinformationen ausgeben soll.
Dieser Operand kann, durch Komma getrennt, mehrfach angegeben werden, so dass z.B. mit T=MIN, P=MAX minimaler Informationsumfang am Terminal und maximaler Informationsumfang auf SYSLST ausgegeben wird.

Den Operanden *medium-u-menge* können Sie in den folgenden Kommandos angeben:
%DISPLAY
%HELP
%SDUMP
%OUT

Der *medium-u-menge*-Operand des %OUT-Kommandos wirkt außerdem auf:
%DISASSEMBLE
%TRACE

Mit dem Kommando %OUT kann *medium-u-menge* für die Kommandos %DISASSEMBLE, %DISPLAY, %HELP, %SDUMP und %TRACE festgelegt werden. Die Vereinbarung gilt für die gesamte Testsitzung bis zu einer neuen Vereinbarung oder der Beendigung mit /EXIT-JOB.

In %DISPLAY, %SDUMP und %HELP kann *medium-u-menge* direkt angegeben werden; die Vereinbarung gilt nur für das jeweilige Kommando, danach tritt wieder *medium-u-menge* des Kommandos %OUT oder der Standardwert T=MAX in Kraft. Wird in %DISPLAY, %SDUMP oder %HELP kein *medium-u-menge* angegeben, so gilt der *medium-u-menge* des %OUT-Kommandos. Wurde auch in %OUT *medium-u-menge* nicht angegeben, so gilt der Standardwert T=MAX.

Für alle Ausgabemedien berücksichtig AID den zugeordneten Coded-Character-Set Namen (kurz CCSN) immer dann, wenn UTF16-/ UTFE-Zeichen auf das Ausgabemedium ausgegeben werden sollen. AID unterstützt jedoch nur UTFE bzw. 1-Byte-EBCDIC-Codes.

```
medium-u-menge-OPERAND - - - - - - - - - - - - - - - - - - - - - - - - -

{ T  }   { MAX }
{ H  } = {     }
{ Fn }   { MIN }
{ P  }

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

T Terminal-Ausgabe über SYSOUT.

H Hardcopy-Ausgabe (schließt die Terminal-Ausgabe mit ein und kann nicht gemeinsam mit *T* angegeben werden)

Fn Datei-Ausgabe. Fn bezeichnet den Linknamen für die Ausgabedatei.
nist eine Zahl mit einem Wert $0 \leq n \leq 7$.
Es gibt drei Wege, die zugehörige Datei anzulegen:

1. %OUTFILE-Kommando mit dem Link- und Dateinamen.
2. /FILE-Kommando für Fn.
3. Für einen Linknamen, dem kein Dateiname zugewiesen ist, setzt AID einen FILE-Makro mit dem Dateinamen AID.OUTFILE.Fn ab, entsprechend dem Linknamen Fn. Die Datei wird mit `FCBTYPE=SAM, OPEN=EXTEND und RECFORM=V` erstellt.

Beachten Sie bei der Verwendung des Linknamens F6, dass F6 der Standard-Linkname für REP-Dateien ist.

P Ausgabe auf SYSLST.

MAX

Die maximalen Zusatzinformationen werden mit den Daten ausgegeben.
Das sind zum einen Informationen über den AID-Arbeitsbereich und über die Unterbrechungsstelle, zum anderen Informationen über die auszugebenden Daten.

Bei %HELP hat die Angabe {MIN | MAX} keine Auswirkungen, eine der beiden Angaben ist aber syntaktisch erforderlich.
Wenn für ein Ausgabemedium zum ersten Mal eine Ausgabe mit dem Operandenwert MAX durchgeführt wird bzw. wenn sich der Zeileninhalt gegenüber einer vorhergegangenen Ausgabe geändert hat, werden der Ausgabe bis zu drei Zeilen vorangestellt:

- Taskzeile, sie informiert über den aktuellen AID-Arbeitsbereich und enthält den Task Identifier (TID) sowie die Prozessfolge-Nummer (TSN) oder den Linknamen der Dump-Datei.
- Kopfzeile, sie enthält Informationen über die Unterbrechungsstelle, das ist die Adresse, an der das Programm zum Zeitpunkt der Ausgabe steht.

– Zielzeile, sie enthält Informationen über die auszugebende Adresse, das ist die CSECT/der COMMON, in der sich die Adresse befindet, und die Distanz zum CSECT-/COMMON-Anfang.

Für jeden Datenbereich werden ausgegeben:
– Datenname und bei Vektoren auch die Indexgrenzliste
– Inhalt und bei Vektoren auch der zugehörige Index; bei gleichen Zeilen wird der Text "repeated lines: n" ausgegeben.
– virtuelle Adresse des ersten Bytes einer jeden Datenzeile bei Ausgaben auf Maschinencode-Ebene
– Distanz des ersten Bytes einer jeden Datenzeile zum Beginn einer CSECT, falls die Adresse einer CSECT zugeordnet werden kann; ansonsten die Adresse des ersten Bytes einer jeden Datenzeile relativ zum Anfang des Datenbereiches.

MIN

Daten werden ohne Zusatzinformationen ausgegeben.

Beispiele

```
%OUT %DA T=MIN
%C1 %IO <%DA FROM %PC->;%STOP>
%R
00000BB6  L     R1,A8(R0,R11)
00000BBA  L     R15,98(R0,R11)
00000BBE  BALR  R14,R15
00000BC0  DC    X'0001'  INVALID OPCODE
00000BC2  CLI   396(R12),X'F0'
00000BC6  L     R13,B4(R0,R2)
00000BCA  BC    B'1100',E0(R0,R13)
00000BCE  L     R15,A4(R0,R11)
00000BD2  BAL   R14,4(R0,R15)
00000BD6  DC    X'0000'  INVALID OPCODE

STOPPED AT SRC_REF: 540PE, SOURCE: TEST, PROC: TEST
```

Ohne Zusatzinformationen sollen die Daten, die das %DISASSEMBLE-Kommando ermittelt, ausgegeben werden. Die virtuelle Adresse des jeweiligen Befehls wird als 8-stellige Sedezimal-Zahl vorangestellt.

```
%OUT %DA T=MAX
%C1 %IO <%DA FROM %PC->;%STOP>
%R
TEST+C22          L     R1,A8(R0,R11)          58 10 B0A8
TEST+C26          L     R15,9C(R0,R11)         58 F0 B09C
TEST+C2A          BALR  R14,R15                05 EF
TEST+C2C          CLI   396(R12),X'F2'         95 F2 C396
TEST+C30          L     R13,B4(R0,R2)          58 D0 20B4
TEST+C34          BC    B'1100',14A(R0,R13)    47 C0 D14A
TEST+C38          L     R15,A4(R0,R11)         58 F0 B0A4
TEST+C3C          BAL   R14,4(R0,R15)          45 E0 F004
TEST+C40          DC    X'0000'  INVALID OPCODE 00 00
TEST+C42          CLI   396(R12),X'F0'         95 F0 C396

STOPPED AT SRC_REF: 58REA, SOURCE: TEST, PROC: TEST
```

Mit Zusatzinformationen sollen die Daten, die das %DISASSEMBLE-Kommando ermittelt, ausgegeben werden. Die Adresse des Befehls wird als relative Adresse angegeben, d.h. als Programmname plus Abstand zum Programmanfang. Hinter den disassemblierten Befehlen steht der Speicherinhalt in sedezimaler Form.

```
%OUT %D T=MIN
%D ABC-TAB

01          ABC-TAB
 02         ZEICHEN( 1: 26)
            |A|  |B|  |C|  |D|  |E|  |F|  |G|  |H|  |I|  |J|  |K|  |L|  |M|
            |N|  |O|  |P|  |Q|  |R|  |S|  |T|  |U|  |V|  |W|  |X|  |Y|  |Z|
```

Ohne Zusatzinformationen soll die Tabelle ABC-TAB aus einem COBOL-Programm mit %DISPLAY ausgegeben werden. Die Stufennummern und der Inhalt der Tabellenfelder werden ausgegeben.

```
%D ABC-TAB T=MAX

*** TID: 000000D1 *** TSN: 8438 ************************************************
SRC_REF:     58ADD   SOURCE: MOBS      PROC: MOBS  ******************************
01          ABC-TAB
 02         ZEICHEN( 1: 26)
            (  1) |A|  (  2) |B|  (  3) |C|  (  4) |D|  (  5) |E|  (  6) |F|
            (  7) |G|  (  8) |H|  (  9) |I|  ( 10) |J|  ( 11) |K|  ( 12) |L|
            ( 13) |M|  ( 14) |N|  ( 15) |O|  ( 16) |P|  ( 17) |Q|  ( 18) |R|
            ( 19) |S|  ( 20) |T|  ( 21) |U|  ( 22) |V|  ( 23) |W|  ( 24) |X|
            ( 25) |Y|  ( 26) |Z|
```

Mit Zusatzinformationen soll die Tabelle ABC-TAB aus einem COBOL-Programm mit %DISPLAY ausgegeben werden. Eine Task- und eine Kopfzeile wird der eigentlichen Ausgabe vorangestellt. Die Stufennummern, der Inhalt der Tabellenfelder und der zugehörige Index werden ausgegeben.

9 AID-Literale

In den AID-Kommandos %DISPLAY, %FIND, %MOVE und %SET ist als Operand die Angabe eines AID-Literals möglich.

9.1 Zeichen-Literale

9.1.1 Character-Literal

9.1.1.1 Eingabeformate

{C'x...x' | 'x...x'C | 'x...x'| U'x...x'}

maximale Länge: 80 Zeichen.
Zeichenvorrat für *x*: jedes Zeichen, das Sie am Terminal eingeben können.

Wenn der Coded-Character-Set für das Eingabemedium nicht UTFE ist, kann mit Hilfe des U-Literals ein UTFE-Character-String spezifiziert werden.

Kleinbuchstaben können nur eingegeben werden, wenn %AID LOW[=ON] eingestellt ist. Voreinstellung ist, dass Kleinbuchstaben in Großbuchstaben umgesetzt werden (siehe %AID). Dann können Sie Kleinbuchstaben nur als Sedezimal-Literale eingeben. Apostrophen, die im Literal enthalten sein sollen, müssen verdoppelt werden (").

%DISPLAY
Das Literal wird ausgegeben. Es kann u.U. über eine Typmodifikation konvertiert werden.

%FIND
x kann auch das Wildcard-Symbol '%' sein. Es steht jeweils für ein beliebiges Zeichen und wird von %FIND immer als Treffer gemeldet.
Bei Verwendung von %C() und %UTF16() auf das Suchliteral wird '%' nicht mehr als Wildcard-Symbol unterstützt.

%MOVE
Das Literal wird linksbündig in der Länge des Literals in das Empfangsfeld übertragen. Ist das Literal länger als das Empfangsfeld, wird die Übertragung mit einer Meldung abgewiesen.

%SET
Besteht das Literal nur aus Ziffern, ist seine Länge ≤ 18 und soll es in ein numerisches Feld übertragen werden, so wird es wie ein numerisches Literal konvertiert und werterhaltend übertragen. Ist der Inhalt nicht rein numerisch oder seine Länge > 18, so kann das Literal alphanumerisch in ein Character-Feld (%C) oder binär in ein Feld mit Typmodifikation %X übertragen werden. Dort wird es linksbündig abgelegt. Ist das Empfangsfeld länger als das Literal, wird rechts aufgefüllt, bei alphanumerischer Übertragung mit Leerzeichen (C'␣' ≙ X'40'), bei binärer Übertragung mit X'00'. Ist das Literal länger als das Empfangsfeld, wird rechts abgeschnitten und eine Warnung ausgegeben.

9.1.1.2 Zeichencodierung

Mit der Einführung von Unicode unterstützt AID für Eingabe- und Ausgabemedien den diesen Medien zugeordneten **C**ode-**C**haracter-**S**et-**N**ame (CCSN).
Einer Datei kann mit dem Kommando MODIFY-FILE-ATTRIBUTES über den Operanden CODED-CHARACTER-SET ein CCSN zugewiesen werden.

Beim Eingabemedium TERMINAL benutzt AID den CCSN, der über das BS2000 Kommando MODIFY-TERMINAL-OPTIONS eingestellt wurde. Dabei sind AID alle Einstellungen unbekannt, die direkt an der Terminal-Emulation vorgenommen, aber nicht über das Kommando MODIFY-TERMINAL-OPTIONS bekannt gemacht wurden.

Wurden für Eingabe- oder Ausgabemedien keine CCSNs vereinbart, verwendet AID als Standard-CCSN den CCSN des Join-Eintrages der Benutzerkennung. Diese Einstellung kann über das Kommando %AID EBCDIC= … geändert werden.

AID unterstützt für Eingabe- und Ausgabemedien nur diejenigen CCSNs, die auch von XHCS unterstützt werden und außer UTFE einen 1-Byte-EBCDIC-Code repräsentieren. Voraussetzung hierfür ist das Subsystem XHCS-SYS in der Version ≥V02.0.

Mit dem AID-Kommando %SHOW %CCSN können die aktuell von XHCS-SYS unterstützten CCSNs ausgegeben werden.

9.1.1.3 Konvertierungsfunktion %C() und %UTF16()

Mit diesen Funktionen können sie die Art der Zeichencodierung eines Character-Literals verändern.

%UTF16() konvertiert das Literal in eine UTF16-Zeichenfolge.
%C() konvertiert ein UTF16-Literal in eine 1-Byte-EBCDIC-Codierung, die durch das Kommando %AID EBCDIC festgelegt wurde.

Liegt das Literal in einer 1-Byte-EBCDIC-Codierung vor, so hat %C() keine Wirkung.

Bei der Konversion wird ein Zeichen durch das Ersatzzeichen '.' ersetzt, wenn es im UTF16 oder 1-Byte-EBCDIC-Zeichensatz nicht vorhanden ist. In diesem Fall gibt AID eine Meldung aus.

9.1.2 Sedezimal-Literal

{X'f...f' | 'f...f'X}

maximale Länge: 80 Sedezimal-Stellen (entspricht 40 Bytes).
Ein Literal mit ungerader Stellenzahl wird rechts mit X'0' ergänzt.
Zeichenvorrat für *f*: jedes Zeichen von 0 - 9 und A - F.

Die Typmodifikation %UTF16 ist für ein Sedezimal-Literal zulässig. Durch diese Typmodifikation wird das Literal wie ein Character-Literal behandelt (siehe Abschnitt „Character-Literal“ auf Seite 103).

%DISPLAY
Das Literal wird ausgegeben. Es kann über eine Typmodifikation konvertiert werden.

%FIND
f kann auch das Wildcard-Symbol '%' sein. Es steht jeweils für ein beliebiges Zeichen und wird von %FIND immer als Treffer gemeldet.

%MOVE
Das Literal wird linksbündig in der Länge des Literals in das Empfangsfeld übertragen. Ist das Literal länger als das Empfangsfeld, wird die Übertragung mit einer Meldung abgewiesen.

%SET
Das Literal wird linksbündig übertragen. Ist das Empfangsfeld länger als das Literal, wird rechts mit X'00' aufgefüllt. Ist das Literal länger als das Empfangsfeld, wird rechts abgeschnitten.
Dieses Literal kann für die Übertragung in ein Empfangsfeld mit beliebiger Datentyp-Definition verwendet werden.

9.1.3 Binär-Literal

{B'b...b' | 'b...b'B}

maximale Länge: 80 Binärstellen (entspricht 10 Bytes).
Es wird rechts auf Bytelänge (8 Binärstellen) mit binär Null B'0' ergänzt.
Zeichenvorrat für b: die Zeichen 0 und 1

%DISPLAY
Das Literal wird ausgegeben. Es kann über eine Typmodifikation konvertiert werden.

%FIND
B'b...b' kann nicht angegeben werden.

%MOVE
Das Literal wird linksbündig in der Länge des Literals in das Empfangsfeld übertragen. Ist das Literal länger als das Empfangsfeld, wird die Übertragung mit einer Meldung abgewiesen.

%SET
Das Literal wird linksbündig übertragen. Ist das Empfangsfeld länger als das Literal, wird rechts mit binär Null aufgefüllt. Ist das Literal länger als das Empfangsfeld, wird rechts abgeschnitten.
Dieses Literal kann in ein Empfangsfeld mit beliebiger Datentyp-Definition übertragen werden.

9.2 Numerische Literale

9.2.1 Ganzzahl

[{±}]n

maximale Länge: 20 Ziffern
Wertebereich: $-10^{21} \leq n \leq +10^{20}-1$

Die interne Darstellung einer Ganzzahl ist für den Anwender undefiniert, das heißt, referenziert der Anwender in einem AID-Kommando die interne Darstellung, ist das Ergebnis undefiniert.
Beispiel: `%D 12345 %X / %M 123456789 INTO V'xxxx'`

Aus Kompatibilitätsgründen ist die interne Darstellung im Bereich von $2^{31} \leq n \leq +2^{31}-1$ wie %FL4.

%DISPLAY
Das Literal wird ausgegeben. Es kann über eine Typmodifikation konvertiert werden.

%FIND
Eine Ganzzahl kann nicht angegeben werden.

%MOVE
Die Ganzzahl wird als sedezimaler Wert in einem Wort (4 Bytes) aufbereitet und linksbündig im Empfangsfeld abgelegt. Ist das Empfangsfeld zu kurz, wird die Übertragung mit einer Fehlermeldung abgewiesen.

%SET
Die Ganzzahl kann in jedes numerische Empfangsfeld übertragen werden; sie wird gegebenenfalls dem Typ des Empfangsfeldes angepasst und werterhaltend übertragen.

9.2.2 Sedezimalzahl

#'x...x'

maximale Länge: 16 Sedezimal-Stellen (entspricht dem Speichertyp %FL8, Ganzzahl mit Vorzeichen in Integer-Darstellung)

Wertebereich:
Bei einer max. Länge von 8 Sedezimalstellen: $-2^{31} \leq$ #'x...x' $\leq +2^{31}-1$ (%FL4)
Bei einer Länge von mindestens 9 Sedezimalstellen: $-2^{63} \leq$ #'x...x' $\leq +2^{63}-1$ (%FL8)

Zeichenvorrat für x:

Negative Sedezimalzahl:
32-Bit-Zahlen:
haben genau 8 Sedezimalziffern, wobei das erste Bit in der linkesten Ziffer das Vorzeichen beschreibt. Um einen negativen Wert zu beschreiben, muss diese Ziffer aus dem Bereich X'8' , X'9', ..., X'F' kommen.

64-Bit-Zahlen:
haben genau 16 Sedezimalziffern; wie bei einer 32-Bit Sedezimalzahl muss die linkeste Ziffer aus dem Bereich X'8' , X'9', ..., X'F' kommen.

Beispiel:
#'FFFFFFFF' beschreibt eine 32-Bit-Sedezimalzahl mit Wert -1;
#'0FFFFFFFF' beschreibt eine 64-Bit-Sedezimalzahl mit dem Wert +2**32-1;

Damit bleibt das bisherige Verhalten bei 32-Bit-Sedezimalzahlen von älteren AID Versionen erhalten.

%DISPLAY
Das Literal wird ausgegeben. Es kann über eine Typmodifikation konvertiert werden.

%FIND
Eine Sedezimalzahl kann nicht angegeben werden.

%MOVE
Die Sedezimalzahl wird in Wortlänge (4 Bytes) aufbereitet und linksbündig im Empfangsfeld abgelegt. Ist das Empfangsfeld zu kurz, wird die Übertragung mit einer Fehlermeldung abgewiesen.

%SET
Die Sedezimalzahl kann in jedes numerische Empfangsfeld übertragen werden; sie wird gegebenenfalls dem Typ des Empfangsfeldes angepasst und werterhaltend übertragen.

9.2.3 Dezimalpunktzahl

[{±}]n.m

maximale Länge: 18 Ziffern, ein Dezimalpunkt und ein Vorzeichen.

Vor der höchstwertigen Stelle kann ein Vorzeichen stehen. Ein Dezimalpunkt kann an jeder Position innerhalb der Ziffernfolge angegeben werden. Soll er an höchster Stelle stehen, muss eine Null davor gesetzt werden.

%DISPLAY
Das Literal wird ausgegeben.

%FIND/%MOVE
Eine Dezimalpunktzahl kann nicht angegeben werden.

%SET
Die Dezimalpunktzahl kann in jedes numerische Empfangsfeld übertragen werden; sie wird gegebenenfalls dem Typ des Empfangsfeldes angepasst und werterhaltend übertragen.

9.2.4 Gleitpunktzahl

[{±}]mantisseE[{±}]exponent

Die Gleitpunktzahl wird intern mit doppelter Genauigkeit (8 Bytes) aufgebaut. Haben *mantisse* oder *exponent* kein Vorzeichen, werden sie positiv interpretiert. Innerhalb der Gleitpunktzahl dürfen keine Leerzeichen geschrieben werden.

mantisse
maximale Länge: 16 signifikante Ziffern, ein Dezimalpunkt und ein Vorzeichen
mantisse muss einen Dezimalpunkt enthalten, der an jeder beliebigen Stelle innerhalb stehen kann. Soll er an der höchsten Stelle stehen, muss eine Null vorangehen.

exponent
maximale Länge: 2 Ziffern und ein Vorzeichen
Wertebereich: $-75 \leq exponent \leq 76$.

%DISPLAY
Das Literal wird ausgegeben.

%FIND/%MOVE
Eine Gleitpunktzahl kann nicht angegeben werden.

%SET
Die Gleitpunktzahl kann in jedes numerische Empfangsfeld übertragen werden; es wird entsprechend dem Typ des Empfangsfelds angepasst und werterhaltend übertragen.

10 Schlüsselwörter

Schlüsselwörter sind für AID festgelegte Vereinbarungen, die mit einem Prozent-Zeichen beginnen. Sie stehen für Speichertypen, Register, Befehlszähler, Speicherklassen, Systeminformationen, Durchlaufzähler, logische Werte, Vorschubsteuerung, Adressumschaltung, Ausgabe der aktuellen Aufrufhierarchie, Befehlstypen und Ereignisse.
In einer komplexen Speicherreferenz können Sie die Schlüsselwörter für Speicherklassen, Register, Befehlszähler und Durchlaufzähler verwenden. Die impliziten Speichertypen und Längen sind im jeweiligen Abschnitt angegeben.

10.1 Allgemeine Speichertypen

Mit den Schlüsselwörtern für Speichertypen können Sie eine Speicherstelle oder ein Literal anders als definiert interpretieren lassen (siehe Abschnitt „Typmodifikation“ auf Seite 88).
Das kann in den folgenden Fällen notwendig oder sinnvoll sein:
Beim %SET sind die Speichertypen von *sender* und *empfänger* nicht miteinander verträglich.

Beim %DISPLAY wollen Sie sich eine Speicherstelle oder ein Literal konvertiert ausgeben lassen. Jedem Speichertyp ist implizit ein Ausgabetyp zugeordnet, der festlegt, wie der Speicherinhalt ausgegeben wird.
Bei der Adressierung wollen Sie eine Speicherstelle in die Rechnung mit einbeziehen.

Mit der wahlweisen Längenangabe *L-mod* können Sie zugleich die Länge modifizieren (siehe Abschnitt „Längenmodifikation“ auf Seite 91). Dies ist bei Literalen nicht zugelassen.
Zwischen Typ und Längenangabe darf kein Leerzeichen geschrieben werden. Die Längenangabe kann jede Form der Längenmodifikation annehmen.

```
%X[L-mod]   Sedezimal, Länge 1 ≤ n ≤ 65 535
            Der Standard-Speichertyp für eine virtuelle Adresse ist %XL4.
            Ausgabetyp: Dump (Sedezimal und Character)

%C[L-mod]   Character, Länge 1 ≤ n ≤ 65 535
            Ausgabetyp: Character

%UTF16[L-mod]
            Unicode-Character, Länge 2 ≤ n ≤ 65 534,
            Die Länge muss ein Vielfaches von 2 haben.
            Ausgabetyp: Dump (Hexadezimal und Character)
```

```
%P[L-mod]   Gepackt, Länge 1 ≤ n ≤ 9, darf außer dem Vorzeichen
            (letztes Halbbyte) nur Ziffern enthalten.
            Ausgabetyp: numerisch (Ganzzahl mit Vorzeichen)
            Für Literale nicht zugelassen.

%D[L-mod]   Gleitpunkt, Länge n = 4, 8 oder 16 Bytes
            Ausgabetyp: numerisch (Gleitpunktdarstellung)
            Für Literale nicht zugelassen.

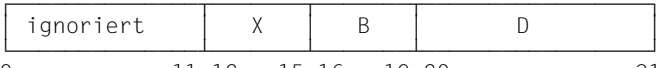
%F[L-mod]   Binär mit Vorzeichen (Integer), Länge n = 1..8 Bytes
%H          entspricht %FL2
            Ausgabetyp: numerisch (Ganzzahl mit Vorzeichen)
            %H ist für Literale nicht zugelassen.

%A[L-mod]   Adresse, Länge n = 1..8 Bytes
%Y          entspricht %AL2
            Ausgabetyp: numerisch (Ganzzahl ohne Vorzeichen)
            Für Literale nicht zugelassen.
```

10.2 Speichertypen zur Interpretation von Maschinenbefehlen

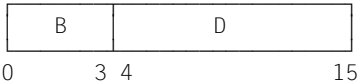
Diese Speichertypen werden benutzt, um eine Adresse zu errechnen, die in den Hauptspeicher-Operanden von Maschinenbefehlen durch Basisregister und Distanz oder Index-, Basisregister und Distanz hinterlegt ist. Erst der nachfolgende Pointer-Operator "->" veranlasst die Errechnung der Adresse. Ohne nachfolgenden Pointer-Operator wirkt %SX wie %XL4 und %S wie %XL2. Beispiele dazu finden Sie im Abschnitt „Indirekte Adressierung "->" / "*"“ auf Seite 84.

```
%SX     SX-Adresse, Länge 4 Bytes, Index-Basis-Distanz (X-B-D),
        entspricht einem Maschinenbefehl im SX-Format
        Die Indexregister-Nummer X wird nur ausgewertet, wenn sie ≠ 0 ist.
        Ausgabetyp: Sedezimal
```




```
        Bit 0          11 12   15 16   19 20          31
```

```
%S      S-Adresse, Länge 2 Bytes Basis-Distanz (B-D)
        Die Basisregister-Nummer B wird nur ausgewertet, wenn sie ≠ 0 ist.
        Ausgabetyp: Sedezimal
```




```
                                Bit 0      3 4             15
```

10.3 Programmregister und Befehlszähler

Die Programmregister (Mehrzweckregister und Gleitpunktregister) und der Befehlszähler (Program Counter) werden von AID über Schlüsselwörter angesprochen. Sie können den Inhalt anzeigen lassen (%DISPLAY), verändern (%MOVE, %SET) oder zur Adressierung benutzen. Die Programmregister liegen im privilegierten Bereich und können über ihre virtuellen Adressen nicht angesprochen werden. Die Gleitpunktregister belegen einen gemeinsamen Speicherplatz:
%0Q überdeckt %0D und %2D; %4Q überdeckt %4D und %6D.

Der Inhalt von Mehrzweckregistern wird in Subkommandobedingungen und arithmetischen Ausdrücken von AID als numerischer Wert mit Vorzeichen behandelt, wie es seinem Typ %FL4 entspricht.
Vor einem Pointer-Operator hingegen nimmt AID den Typ %AL4 an und der Registerinhalt kann ohne Typmodifikation als Adresse verwendet werden. Ausgegeben wird ein Register als Sedezimalzahl. Um den numerischen Wert mit Vorzeichen ausgeben zu lassen, müssen Sie für die Ausgabe eine Typmodifikation mit %F vornehmen.

In ASSEMBH haben die Programmregister auch symbolische Namen _Rn, die jedoch nur beim symbolischen Testen von Assemblerprogrammen verwendet werden können.

```
%PC   Befehlszähler (Program Counter), Typ %AL4
%n    Mehrzweckregister 0 ≤ n ≤ 15, Speichertyp %FL4,
      Ausgabe als Sedezimalzahl
%nE   Gleitpunktregister mit einfacher Genauigkeit n = {0,2,4,6}, Typ %DL4
%nD   Gleitpunktregister mit doppelter Genauigkeit n = {0,2,4,6}, Typ %DL8
%nQ   Gleitpunktregister mit vierfacher Genauigkeit n = {0,4}, Typ %DL16

%MR   alle 16 Mehrzweckregister in Tabellenform aufbereitet
%FR   alle 4 Gleitpunktregister mit doppelter Genauigkeit
      in Tabellenform aufbereitet

%nAR  Zugriffsregister 0 ≤ n ≤ 15, Speichertyp %FL4, Ausgabe als
      Sedezimalzahl (ESA-Unterstützung)
%AR   alle 16 Zugriffsregister in Tabellenform aufbereitet
      (ESA-Unterstützung)
```

Das Schlüsselwort %PC und die Register können Sie bei Bedarf mit einem Index versehen. Diesen Index brauchen Sie nur, wenn Sie ein Programm bearbeiten, das Contingency-Prozesse definiert hat bzw. in der STXIT-Verarbeitung von AID unterbrochen wurde. Wenn Sie nicht die Informationen zum unterbrochenen Contingency- bzw. STXIT-Prozess, sondern zu einer anderen Prozess-Ebene, z.B. dem Basis-Prozess, wollen, dann müssen Sie den entsprechenden Index angeben. Diesen Index können Sie mit %DISPLAY %PCBLST feststellen (siehe „Makroaufrufe an den Ablaufteil“ [11]).
Der Index wird folgendermaßen angegeben: schlüsselwort(index).

10.4 AID-Register

Die AID-Register liegen in dem Speicherbereich, der für AID reserviert ist, damit können die AID-Register von jedem AID-Arbeitsbereich aus angesprochen werden, ohne die Arbeitsbereich-Vereinbarung zu verändern. Die AID-Register entsprechen in Typ und Länge den Programmregistern.
Schlüsselwörter zum Ansprechen aller AID-Register gibt es nicht.

```
%nG     AID-Mehrzweckregister 0 ≤ n ≤ 15, Speichertyp %FL4,
        Ausgabe als Sedezimalzahl
%nGD    AID-Gleitpunktregister mit doppelter Genauigkeit n = {0,2,4,6},
        Typ %DL8
```

10.5 Speicherklassen

Mit den folgenden Schlüsselwörtern können Sie Speicherbereiche ansprechen. Sie können als Bereich bei %CONTROL*n*, %DISASSEMBLE, %DISPLAY, %FIND und %TRACE angegeben werden. In einer komplexen Speicherreferenz können Sie die Schlüsselwörter für Speicherklassen mit allen Attributen verwenden:
Name, Adresse, Inhalt, Länge, Typ.

Im Klasse-5-Speicher belegt Ihr Programm privilegierte und nicht-privilegierte Bereiche. Auf beide Bereiche können Sie mit dem Schlüsselwort %CLASS5 zugreifen, auf den privilegierten Bereich jedoch nur mit höherer Test-Privilegierung.

Auf XS-Anlagen bzw. in XS-Dumps wird mit den Schlüsselwörtern %CLASS5 und %CLASS5BELOW derselbe Adressraum bezeichnet. Entsprechendes gilt für %CLASS6 und %CLASS6BELOW.

```
%CLASS5             Klasse-5-Speicher, Typ %X
%CLASS6             Klasse-6-Speicher, Typ %X
```

Zusätzlich gibt es beim Testen von Programmen in BS2000-Versionen ≥V9 oder beim Bearbeiten von Speicherabzügen, die in solch einer BS2000-Version erzeugt wurden, noch die folgenden Schlüsselwörter, ebenfalls alle vom Typ %X:

```
%CLASS5BELOW        Klasse-5-Speicher, unterhalb der 16MB-Grenze (≙%CLASS5)
%CLASS5ABOVE        Klasse-5-Speicher, oberhalb der 16MB-Grenze
%CLASS6BELOW        Klasse-6-Speicher, unterhalb der 16MB-Grenze (≙%CLASS6)
%CLASS6ABOVE        Klasse-6-Speicher, oberhalb der 16MB-Grenze
```

10.6 Systeminformationen

Über die Schlüsselwörter für Systeminformationen erhalten Sie mit %DISPLAY die entsprechenden Informationen über eine Task. Wird mehr als nur ein Wert für das entsprechende Schlüsselwort zurückgeliefert, führt AID eine Aufbereitung durch und gibt eine Tabelle aus.

```
%CC                     Condition Code
%PCB                    Process Control Block (kann wie %PC indiziert werden)
%PCBLST                 Liste aller Process Control Blocks
%LINK                   Name des zuletzt nachgeladenen Segments, das mit
                        %ON %LPOV festgestellt wurde
%PM                     Program Mask
%AUD1                   Hardware-Audit-Tabelle beginnend mit dem jüngsten
                        Eintrag; nur wenn bei Systemgenerierung angelegt
%AMODE                  Systeminformationsfeld zum Adressierungsmodus.
                        Kann nur mit %MODE24 oder %MODE31 verändert werden
%ASC                    ASC-Modus auf ESA-Anlagen
                        (bzgl. AR-Modus bedeutet: X'00' = aus; X'01' = ein
%DS[(ALET/SPID-qua)]    Information über SPIDs und/oder ALETs der aktiven
                        Datenräume auf ESA-Anlagen

%LOC(speicherref)       maschinennahe Lokalisierungsinformation zu einer
                        Adresse im ausführbaren Teil
%HLLOC(speicherref)     symbolische Lokalisierungsinformation zu einer
                        Adresse im ausführbaren Teil

%SORTEDMAP              Liste aller CSECTs und COMMONs des Benutzerprogramms
                        (namen- und adresssortiert)
```

```
                ⎧ CTX=kontext [•L=ladeeinheit] ⎫
                ⎪ L=ladeeinheit                ⎪
%MAP [(         ⎨              ⎧USER⎫          ⎬)]
                ⎪ SCOPE =      ⎨    ⎬          ⎪
                ⎩              ⎩ALL ⎭          ⎭
```

{CTX=kontext | L=ladeeinheit}
: Bei Angabe eines Pfads werden alle CSECTs/COMMONs des angegebenen Kontextes oder der angegebenen Ladeeinheit aufgelistet.

SCOPE=USER
: CSECTs/COMMONs der Standard-Kontexte CTXPHASE bzw. LOCAL#DEFAULT und der durch den BIND-Makro mit Operand LNKCTX[@] gebildeten Kontexte werden aufgelistet.

SCOPE=ALL
: Zusätzlich zu den CSECTs/COMMONs der benutzerdefinierten Kontexte wird die MAP aller Kontexte ausgegeben, an die sich das Programm konnektiert hat wie z.B. DSSM-Subsysteme oder Userpool-Kontexte.

Alle BLS-Namen (Kontext, Ladeeinheit, CSECT und COMMON) werden ungekürzt ausgegeben. Innerhalb der Kontexte und Ladeeinheiten wird die ausgegebene Liste nach CSECT-Namen sortiert.

Beispiele

1. `/%D %HLLOC(PROG=UPRONUM.S'22DIS'->)`

   AID-Ausgabe

```
V'000083EC' = CONTEXT  : LOCAL#DEFAULT
              SMOD     :  UPRONUM
              PROC     :  UPRONUM
              PARAGRAPH:  ENTPA
              SRC-REF  :     22DIS
              LABEL    :  ENTPA
```

2. `/%D %LOC(PROG=UPRONUM.S'22DIS'->)`

   AID-Ausgabe

```
V'000083EC' = CONTEXT  :  LOCAL#DEFAULT
              LMOD     :  %UNIT
              SMOD     :  UPRONUM
              OMOD     :  UPRONUM
              CSECT    :  UPRONUM (00008018) + 000003D4
```

3. `/%D %MAP`

   AID-Ausgabe

```
*** TID: 00230056 *** TSN: OFZB ***********************************************
CURRENT PC: 00002000    CSECT: LLMTEST2  ************************************
**CSECT-LISTING(MAP) OF CONTEXT : LOCAL#DEFAULT
**MAP OF LOAD UNIT             : %UNIT
  CSECT-NAME                      START     SIZE    VER/DATE_OF_MOD
  ASSTEST                         00000150  000830   ............
  ASS2                            00000980  000008   ............
  COMM1                           00001000  00012C   %COMMON.....
  LLMTEST1                        00000000  000150   ............
**CSECT-LISTING(MAP) OF CONTEXT : CTX2
**MAP OF LOAD UNIT             : LLMTEST2
  CSECT-NAME                      START     SIZE    VER/DATE_OF_MOD
  ASSTEST                         00002078  000650   ............
  ASS2                            000026C8  000008   ............
  COMM1                           FFFFFFFF  000000   %COMMON.....
  LLMTEST2                        00002000  000078   ............
```

10.7 Durchlaufzähler

Der Durchlaufzähler wird zu jedem Subkommando angelegt. In ihm wird gezählt, wie oft das Subkommando durchlaufen wurde. Innerhalb des Subkommandos kann der eigene Zähler immer mit %• angesprochen werden. Erhält das Subkommando einen Namen, so bekommt auch der Zähler einen Namen. In diesem Fall kann der Zähler außerhalb des Subkommandos mit %•subkdoname angesprochen werden. Die Ausführung eines Subkommandos können Sie vom Stand des Durchlaufzählers %•subkdoname abhängig machen, indem Sie ihn in einer Bedingung abfragen (siehe Kapitel „Subkommando“ auf Seite 47).

Die Zuweisung eines numerischen Wertes kann mit %SET erfolgen. Der Inhalt kann mit %DISPLAY gelesen werden. Der Zähler wird erhöht, wenn das zugehörige Subkommando durchlaufen wird. Den Durchlaufzähler können Sie überall dort verwenden, wo ein numerischer Wert zulässig ist.

```
%•[subkdoname]  Variable vom Typ %FL4
                subkdoname ist der Name des zugehörigen Subkommandos.
                Mit der Kurzform %• bezeichnen Sie den Durchlaufzähler des
                gerade aktiven Subkommandos.
```

10.8 Logische Werte

Diese beiden Schlüsselwörter können zur Wertzuweisung (%SET) in logische Variablen aus Fortran-Programmen verwendet werden.

```
%TRUE
%FALSE
```

10.9 Vorschubsteuerung

Die beiden Schlüsselwörter zur Vorschubsteuerung wirken nur auf das Ausgabemedium SYSLST. Sie können nur bei %DISPLAY angegeben werden.

```
%NP             Beginn einer neuen Seite
%NL[(n)]        Ausgabe von n Leerzeilen, 1 ≤ n ≤ 255
                Standardwert für n ist 1.
```

10.10 Adressumschaltung

Bei der XS-Programmierung werden 31-Bit-Adressen verwendet statt der sonst üblichen 24-Bit-Adressen. Mit diesen beiden Schlüsselwörtern können Sie den Adressierungsmodus des Testobjekts oder die Adressinterpretation bei indirekter Adressierung verändern.
`%MOVE %MODE31 INTO %AMODE` verändert den Adressierungsmodus
`%AINT %MODE24` verändert die Adressinterpretation in AID bei indirekter Adressierung

```
%MODE24    24-Bit-Adressierung
%MODE31    31-Bit-Adressierung
```

10.11 Aktuelle Aufrufhierarchie

Bei %SDUMP veranlasst das Schlüsselwort %NEST die Ausgabe der aktuellen Aufrufhierarchie.

```
%NEST      Ausgabe der aktuellen Aufrufhierarchie
```

10.12 Kriterium bei %CONTROL*n* und %TRACE

Mit diesen Schlüsselwörtern werden Befehle oder Anweisungen aus den Programmiersprachen in Typ-Gruppen eingeteilt. Die Schlüsselwörter können in den AID-Kommandos %CONTROL*n* und %TRACE als Überwachungskriterium (Operand *kriterium*) angegeben werden.
Bei %CONTROL*n* wird das zugehörige Subkommando bearbeitet, wenn ein Befehl oder eine Anweisung der zu überwachenden Gruppe zur Ausführung ansteht.
Bei %TRACE wird eine Protokollzeile ausgegeben, wenn ein Befehl oder eine Anweisung der zu überwachenden Befehlsgruppe ausgeführt wird. Beim symbolischen Test findet die Ausgabe statt, bevor die entsprechende Anweisung ausgeführt wird, beim maschinennahen Testen wird protokolliert, nachdem der Befehl ausgeführt wurde.

Standardwert ist das symbolische *kriterium* %STMT. Das hat zur Folge, dass bei einem %CONTROL*n* oder %TRACE mit einer Bereichsangabe auf Maschinencode-Ebene immer ein Schlüsselwort für *kriterium* angegeben werden muss, falls nicht noch aus einem vorhergehenden %CONTROL*n* bzw. %TRACE ein Überwachungskriterium gültig ist.

kriterium	**Befehls- oder Anweisungsgruppe**
Beim Testen auf Maschinencode-Ebene:	
%INSTR	alle Maschinenbefehle, die durchlaufen werden
%B	Verzweigungsbefehle (das sind die Maschinenbefehle BAL, BALR, BAS, BASSM, BASR, BC, BCR, BCT, BCTR, BSM, BXH und BXLE)
%BAL	Unterprogrammaufrufe (durch die Maschinenbefehle BAL, BALR, BAS, BASSM und BASR).
Beim Testen auf symbolischer Ebene:	
%STMT	alle Anweisungen, die durchlaufen werden
%ASSGN	Zuweisungs-Anweisungen
%CALL	SUBROUTINE-Aufrufe (CALL-Anweisungen)
%COND	IF(...) THEN-,ELSE IF(...) THEN-,ELSE- und IF(...)-Anweisungen
%DB	Anweisungen zum Aufruf einer Datenbank
%EXCEPTION	bedingte Anweisungszweige
%GOTO	GOTO-Anweisungen
%IO	Ein-/Ausgabe-Anweisungen
%LAB	Anweisungen nach Label
%PROC	STOP-, END-, RETURN-, SUBROUTINE- und FUNCTION-Anweisungen
%SORT	MERGE- und SORT-Anweisungen

10.13 Ereignis bei %ON

Diese Schlüsselwörter stehen für die Schreibüberwachung, Programmfehler, Programmbeendigungen, Supervisor-Calls und andere Ereignisse während des Programmablaufs. Sie können in dem Kommando %ON (Operand *ereignis*) angegeben werden. Der Operand *ereignis* legt fest, bei welchem Vorfall das Programm unterbrochen werden soll, um das zugehörige Subkommando zu bearbeiten.
Wenn mehrere %ON-Kommandos mit unterschiedlichen *ereignis*-Vereinbarungen gleichzeitig aktiv sind und auch zutreffen, führt AID die zugehörigen Subkommandos in der Reihenfolge aus, in der die Schlüsselwörter in der folgenden Tabelle aufgeführt sind. Treffen verschiedene %TERM-Ereignisse zu, werden die zugehörigen Subkommandos nach dem FIFO-Prinzip abgearbeitet.

Die Schreibüberwachung kann nicht gleichzeitig für verschiedene Bereiche angemeldet sein. Ein neuer %ON %WRITE(...) überschreibt einen früher eingegebenen (siehe auch Abschnitt „%ON %WRITE mit %INSERT, %CONTROLn und %TRACE“ auf Seite 125).

Zur Auswahl des geeigneten %TERM finden Sie weitere Informationen im Handbuch „Makroaufrufe an den Ablaufteil“ [11].

Ereignis	**Subkommando wird bearbeitet:**
%WRITE(speicherref)	nach Überschreiben des mit speicherref bezeichneten Speicherbereichs (ab BS2000/OSD V1.0).
%ERRFLG(z)	nach Auftreten eines Fehlers mit dem angegebenen Ereigniscode und vor Abbruch des Programms.
%INSTCHK	nach Auftreten eines Adressierungsfehlers, eines ungültigen Systemaufrufs (SVC), nicht decodierbaren Operations-Codes, Seitenwechsel-Fehlers oder einer privilegierten Operation und vor Abbruch des Programms.
%ARTHCHK	nach Auftreten eines Datenfehlers, Divisionsfehlers, Exponenten-Überlaufs oder einer Mantisse Null und vor dem Abbruch des Programms.
%ABNORM	nach Auftreten eines der Fehler, die mit den vorher beschriebenen Ereignissen erfasst werden, sowie eines DMS-Errors (ab BS2000 V10) oder eines %ILLSTX.
%ERRFLG	nach Auftreten eines Fehlers mit beliebigem Fehlergewicht.
%SVC(z) %SVC	vor Ausführung des Systemaufrufs (SVC) mit der angegebenen Nummer. vor Ausführung eines beliebigen Systemaufrufs (SVC).
%LPOV(name) %LPOV	nach dem Laden des Segmentes mit dem angegebenen Namen. nach dem Laden eines beliebigen Segmentes.

Ereignis	Subkommando wird bearbeitet:
%TERM(N[ORMAL])	vor Programmbeendigung mit TERM MODE = NORMAL, TERM oder TERMD.
%TERM(A[BNORMAL])	vor Programmbeendigung mit TERM MODE = ABNORMAL, TERMJ oder TRMJD.
%TERM(D[UMP])	vor Programmbeendigung mit TERM DUMP = Y, TERMD oder TRMJD.
%TERM(ND|NODUMP)	vor Programmbeendigung mit TERM DUMP = NO, TERM oder TERMJ.
%TERM(P[RGR])	vor Programmbeendigung mit TERM UNIT = PRGR, TERM oder TERMD.
%TERM(S[TEP])	vor Programmbeendigung mit TERM UNIT = STEP, TERMJ oder TRMJD.
%TERM	vor Programmbeendigung mit TERM, TERMD, TERMJ oder TRMJD.
%ANY	vor der Beendigung des Programms auf Grund eines Programmfehlers bzw. durch einen TERM mit beliebigen Operandenwerten oder TERMJ, TERMD oder TRMJD, oder auf Grund eines DMS-Errors (ab BS2000 V10) oder eines %ILLSTX.
%ILLSTX	vor Auftreten eines STXIT-Aufrufs während der Abarbeitung eines vorangegangenen STXIT-Aufrufs (STXIT im STXIT)

z ist eine Ganzzahl mit: $1 \leq z \leq 255$. z kann als maximal dreistellige vorzeichenlose Dezimalzahl oder als zweistellige Sedezimalzahl (#'ff') angegeben werden.

Es wird nicht überprüft, ob der angegebene Ereigniscode oder die SVC-Nummer sinnvoll oder zulässig ist.

11 Spezielle Anwendungen

11.1 %ON und STXIT

Es gibt verschiedene Möglichkeiten, auf Ereignisse zu reagieren, die während des Programmablaufs auftreten:

– Einzelnen Ereignissen können im Programm STXIT-Routinen zugeordnet werden, die beim Eintreten solcher Ereignisse zu deren Bearbeitung durchlaufen werden (siehe „Makroaufrufe an den Ablaufteil“ [11]).
– Beim Testen mit AID können Ereignisse über das Kommando %ON angemeldet werden. Tritt ein solches Ereignis ein, wird das im %ON-Kommando angegebene Subkommando bearbeitet.

Ereignisse, für die im Programm STXIT-Routinen angemeldet wurden, können von AID nicht bearbeitet werden: AID bekommt vom Eintreten solcher Ereignisse keine Kenntnis. Für solche Ereignisse werden im %ON-Kommando angegebene Subkommandos deshalb nicht durchlaufen.

STXIT-Routinen werden u.a. von den Laufzeitsystemen der Compiler, von ILCS, von openUTM und den Datenbanksystemen angemeldet, z.B. für die Ereignisse "Programmfehler" oder "nicht behebbarer Programmfehler". Diese STXIT-Ereignisse entsprechen den Ereignissen %ERRFLG(zzz), %ERRFLG, %INSTCK, %ARTHCHK und %ABNORM im %ON-Kommando. Diese Ereignisse können mit AID nur dann bearbeitet werden, wenn das Anmelden der STXIT-Routinen unterdrückt wurde.

Für FOR1-Programme ohne Standard-Linkage wird das Anmelden von STXIT-Routinen durch Angabe der Option RUNOPT STXIT=NO unterdrückt. Für Fortran90 ist dies nicht möglich, jedoch kann die Behandlung von EXPONENT-UNDERFLOW und INTEGER-OVERFLOW gesondert durch die Laufzeitoption EXCEPTION gesteuert werden.
Bei COBOL-Programmen und Programmen, die mit Standard-Linkage arbeiten (C ab V2.0A, C++ ab V2.1A, COBOL85 ab V1.1A, COBOL2000 ab V1.0A, FOR1 ab V2.2A, Fortran90 ab V1.0A und PLI1 ab V4.1A) kann das Anmelden von STXIT-Ereignissen nicht verhindert werden.
Jedoch bietet ILCS bei Fehlern, die den Speicher nicht verändern wie Adressenfehler oder unzulässiger Operationscode das folgende Verfahren an: nach der Bearbeitung der STXIT-Routinen stellt ILCS den alten Befehlszählerstand wieder her, reproduziert den Fehler und übergibt die Kontrolle an das System, so dass es anschließend möglich ist, den Fehler mit

dem %ON-Kommando zu bearbeiten. Bei allen anderen Fehlern bricht ILCS das Programm ab. Über %ON %ANY oder %ON %TERM können Sie jedoch den Programmablauf vor dem Entladen stoppen und die Fehlerursache mit AID-Kommandos untersuchen.
Bei openUTM-Anwendungen können die UTM-STXIT-Routinen ab V3.2A (für UTM-T und UTM-P im Dialog) durch Angabe der Option STXIT=OFF im START-Parameter ausgeschaltet werden. Falls openUTM unter ILCS läuft, was ab V3.2A möglich ist (Operand PROGRAM COMP=ILCS in der KDCDEF-Anweisung), dann bleiben jedoch die ILCS-STXIT-Routinen auch nach dem Abschalten der UTM-STXIT wirksam.

In Assembler- und C++/C - Programmen können eigene Routinen (bei C++/C: Signalbehandlung über Bibliotheksfunktion signal(), bei Assembler: Makro STXIT) zur Fehlerbehandlung geschrieben werden. Auch in diesem Fall ist es für AID nicht möglich, über das %ON-Kommando auf einen durch selbst programmierte Routinen abgefangenen Fehler zu reagieren. Allerdings können Sie mit %INSERT einen Testpunkt in die Fehlerbehandlungsroutinen setzen, dessen zugehöriges Subkommando im Fehlerfall ausgeführt wird.

11.2 Programme mit Überlagerungsstruktur

Standardmäßig geht AID davon aus, dass ein Programm ohne Überlagerungsstruktur gebunden ist. Es benutzt die einmal geladenen LSD-Sätze, ohne jedes Mal zu prüfen, ob die angesprochene CSECT in einem mittlerweile nachgeladenen Segment liegt. Falls Sie jedoch ein Programm testen, das statisch als Overlay gebunden wurde oder das Segmente dynamisch mit den Makroaufrufen BIND/UNBIND nach- oder entlädt, müssen Sie im Kommando %AID den Operanden OV=YES angeben, um sicherzustellen, dass AID bei jedem Zugriff auf die LSD überprüft, ob zwischenzeitlich nachgeladen wurde.
Bei Programmen mit Überlagerungsstruktur können Sie einen Testpunkt nur in ein Segment setzen, das zum Zeitpunkt der Kommandoeingabe geladen ist. Ebenso können Sie einen Testpunkt nur löschen, wenn das zugehörige Segment geladen ist. Wird das Segment ent- oder überladen, bleibt der Testpunkt dennoch erhalten, falls er nicht zuvor explizit mit %REMOVE gelöscht wurde. Wird das Segment erneut geladen, so ist auch der Testpunkt wieder gesetzt.

12 Einschränkungen und Wechselwirkungen

12.1 %ON %WRITE mit %INSERT, %CONTROL*n* und %TRACE

Zwischen %ON *write-ereignis* und den AID-Kommandos %CONTROL*n*, %INSERT und %TRACE sind die folgenden Wechselwirkungen zu beachten:

– Wenn ein %CONTROL*n* oder ein %TRACE mit maschinennahem *kriterium* angemeldet ist, wird die Eingabe von %ON *write-ereignis* mit einer Fehlermeldung abgewiesen und umgekehrt.

– Wenn ein Maschinenbefehl durch einen %CONTROL*n* oder %TRACE mit symbolischem *kriterium* mit der AID-internen Markierung (X'0A81') überschrieben wurde, bemerkt AID den schreibenden Zugriff dieses Befehls nicht.

– Wenn ein Maschinenbefehl durch den mit %INSERT vereinbarten Testpunkt mit der AID-internen Markierung überschrieben wurde, erkennt AID auch hier den schreibenden Zugriff dieses Befehls nicht.

Für eine lückenlose Schreibüberwachung empfiehlt es sich, alle %CONTROL*n*- und %INSERT-Kommandos mit %REMOVE zu löschen und einen eventuell noch eingetragenen %TRACE zu löschen, indem Sie nach dem %ON mit %RESUME fortfahren.

12.2 Wechselwirkungen zwischen Ablaufüberwachung und Ausgabe oder Modifikation von Speicherinhalten

Wenn Sie mit %INSERT einen Testpunkt setzen, überschreibt AID den Befehlscode an der Adresse des Testpunkts mit einem SVC X'81'. Ebenso markiert AID den ersten Befehl der ausführbaren Anweisungen in *control-* und *trace-bereich* bei symbolischem *kriterium* mit X'0A81'. Diese Markierungen können Sie sich ansehen, wenn Sie mit %DISPLAY den betreffenden Befehlscode ausgeben lassen. Wenn Sie hingegen den Code mit %DISASSEMBLE rückübersetzen, tauscht AID den eingetragenen SVC gegen den Originalbefehl aus und Sie sehen den Befehlscode, so wie ihn der Compiler erzeugt hat. Ebenfalls greift AID auf den Originalcode zurück, wenn aus dem entsprechenden Befehl über die Typmodifikation %S oder %SX eine Adresse berechnet werden soll, die dann über Pointer (->) angesprochen wird.
Mit dem %FIND-Kommando können Sie die gesetzten Markierungen im Befehlscode suchen, indem Sie X'0A81' als Suchbegriff angeben.
Bei den Kommandos %MOVE und %SET werden eventuell eingetragene Markierungen nicht ersetzt. Das kann dazu führen, dass beim Übertragen von Befehlscode Markierungen verschwinden bzw. neue gesetzt werden, die AID mit seiner internen Kommandoverwaltung nicht mehr in Verbindung bringen kann. Sie müssen also selbst sicherstellen, dass vor dem Übertragen bzw.Überschreiben im betroffenen Befehlscode keine von AID eingetragenen SVCs enthalten sind. Testpunkte und Markierungen des %CONTROL*n* löschen Sie mit dem Kommando %REMOVE. Ein eventuell noch eingetragener %TRACE wird gelöscht, indem Sie das Programm mit %RESUME starten oder %TRACE 1 %INSTR eingeben.

12.3 Testpunkte in Common Memory Pools

Wird ein Testpunkt in einen Common Memory Pool gesetzt, so ist dieser Testpunkt nur der Task bekannt, die den Testpunkt gesetzt hat. Alle weiteren Tasks, die ebenfalls an den Common Memory Pool angeschlossen sind, bleiben an diesem Testpunkt hängen.

Ein weiterer Benutzer des Common Memory Pools müsste von Hand den Originalcode wieder einsetzen. Andernfalls muss die hängengebliebene Task warten, bis der Testpunkt durch die Task, die ihn eingetragen hat, explizit mit %REMOVE zurückgesetzt wird oder bis die Task sich beendet. Dadurch werden die von dieser Task gesetzten Testpunkte implizit entfernt.

Ein weiteres Problem entsteht, wenn die Verbindung von der Task, die den Testpunkt gesetzt hat, zum Common Memory Pool getrennt wird, ohne dass der Testpunkt mit %REMOVE entfernt wurde. Wenn die Task so beendet wird, unterbleibt die implizite Rücksetzung des Testpunkts, da zwischen der Task und dem Common Memory Pool keine Verbindung mehr besteht. Der Originalcode kann dann nur noch von Hand wiederhergestellt werden.

Um die geschilderten Probleme zu vermeiden, sollte der Code zum Testen lokal ablaufen, also nicht in einem Common Memory Pool. Falls aber ein Testpunkt in einem Common Memory Pool benötigt wird, sollte dieser unmittelbar nach Erreichen mit %REMOVE entfernt werden. Zumindest sollte während des Testens der Verbindungsabbau zum Common Memory unterdrückt werden, damit die Testpunkte implizit entfernt werden können.

12.4 Low-Level-Trace und -Control in Verbindung mit Contingencies

12.4.1 %TRACE

Bei Tasks mit ereignisgesteuerter Verarbeitung kann es beim Ablauf von %TRACE auf Maschinencode-Ebene zur fehlerhaften Protokollierung einzelner Maschinenbefehle kommen. Wenn aufgrund eines asynchrones Ereignisses eine Contingency-Routine aufgerufen wird und gleichzeitig ein Low-Level-Trace abläuft, können beim Eintritt in die Contingency-Routine Fehler auftreten. Dies gilt auch bei der Rückkehr in die von dem Ereignis unterbrochene Task.

Abhängig von der jeweiligen Programm- und Test-Konstellation kann an der Schnittstelle Basisprozess/Contingency folgendes Fehlverhalten auftreten:

– ein Befehl wird falsch protokolliert, z.B. mit falschen Registerinhalten
– ein Befehl geht verloren
– ein Befehl wird doppelt angezeigt

12.4.2 %CONTROL

Analog kann es bei der Befehlsüberwachung mit %CONTROL vorkommen, dass ein Subkommando ausgelassen oder zu häufig ausgeführt wird.

i STXIT-Routinen, die bei Programmfehlern gestartet werden, sind von dem oben geschilderten Fehlverhalten nicht betroffen. Hier werden bei %TRACE und %CONTROL alle Befehle richtig und vollständig abgearbeitet.

13 Meldungen

AID0250 Internal AID error in module (&00) . Please contact maintenance (CMD: (&01))
AID0250 Interner AID-Fehler in Modul (&00) . Wartung verstaendigen (KDO: (&01))

Bedeutung
Fehler in AID.
(&00) Modulnummer des Moduls, das den Fehler entdeckt hat.

Maßnahme
Systemverwalter verstaendigen.

AID0251 No memory available
AID0251 Speicherengpass

Bedeutung
AID hat keinen Speicher zur Verfuegung.

Maßnahme
Test beenden und Systemverwalter verstaendigen.

AID0252 AID error in module (&00) : RTC (&01) (CMD: (&02))
AID0252 AID-Fehler in Modul (&00) : RTC (&01) (KDO: (&02))

Bedeutung
Fehler in AID.
(&00) Modulnummer des Moduls, das den Fehler entdeckt hat
(&01) internes Fehlerkennzeichen von AID.

Maßnahme
Systemverwalter verstaendigen.

AID0253 Function not yet implemented (CMD: (&00))
AID0253 Funktion noch nicht implementiert (KDO: (&00))

Bedeutung
Die Funktion wird von der aktuellen AID- oder BS2000-Version nicht unterstuetzt.

AID0254 System error (&00) (CMD: (&01))
AID0254 System-Fehler (&00) (KDO: (&01))

Bedeutung
Systemfehler.
(&00) Modulnummer des Moduls, das den Fehler entdeckt hat.

Maßnahme
Systemverwalter verstaendigen.

```
AID0255    System error in module (&00) : RTC (&01) (CMD: (&02))
AID0255    System-Fehler in Modul (&00) : RTC (&01) (KDO: (&02))
```

Bedeutung
Systemfehler.
(&00) Modulnummer des Moduls, das den Fehler entdeckt hat
(&01) internes Fehlerkennzeichen des Systems

Maßnahme
Systemverwalter verstaendigen.

```
AID0256    AID message file not available or inconsistent.
AID0256    AID-Meldungsdatei nicht verfuegbar oder inkonsistent.
```

Bedeutung
Die AID-Textdatei, welche die AID-spezifischen Meldungen enthaelt, ist nicht vorhanden oder inkonsistent.

Maßnahme
Systemverwaltung verstaendigen.

```
AID0257    Text for message I (&00) cannot be issued.
AID0257    Text fuer Meldung I (&00) kann nicht ausgegeben werden.
```

Bedeutung
Die AID-Textdatei, welche die AID-spezifischen Meldungen enthaelt, ist nicht vorhanden oder inkonsistent. Der Text fuer die Meldung mit der Nummer (&00) kann nicht ausgegeben werden.

Maßnahme
Systemverwalter verstaendigen.

```
AID0258    Test point already set by another task (CMD: (&00))
AID0258    Testpunkt bereits von einer anderen Task gesetzt (KDO: (&00))
```

Bedeutung
Der Testpunkt wurde bereits von einer anderen Task gesetzt und wird daher fuer diese Task zurueckgewiesen.

```
AID0259    Exponent overflow in floating point literal
AID0259    Exponentenueberlauf in einem Gleitpunkt-Literal
```

Bedeutung
Exponentenueberlauf; es wird nur ein Bereich von -76 <= exp <= +75 unterstuetzt.

```
AID0260    File error on (&00) ; DMS error code (&01) (CMD: (&02))
AID0260    Dateizugriffsfehler auf (&00) ; DMS-Fehler-Code (&01) (KDO: (&02))
```

Bedeutung
Fehler beim Zugriff auf eine Datei.
(&00) Linkname oder Dateiname
(&01) DMS-Fehlercode oder Blank

Maßnahme
Datei gemaess DMS-Fehlercode (siehe BS2000-Meldungen) bereinigen und Kommando erneut eingeben.

```
AID0261   File contains no recognizable dump (CMD: (&00))
AID0261   Datei enthaelt unbekanntes Dump-Format (KDO: (&00))
```

Bedeutung
Der Inhalt der Datei kann nicht als Dump erkannt werden.

```
AID0262   Insufficient memory to process command (CMD: (&00))
AID0262   Wegen AID-internem Speichermangel keine Kommandoausfuehrung (KDO: (&00))
```

Bedeutung
Speichermangel; die zu verarbeitende Datenmenge ist zu gross.

Maßnahme
Daten ggf. in Teilmengen verarbeiten.

```
AID0263   Symbolic linkage information (ESD/ESV) missing (CMD: (&00))
AID0263   Symbolische Binde-Information (ESD/ESV) fehlt (KDO: (&00))
```

Bedeutung
Das Kommando konnte nicht ausgefuehrt werden, weil keine symbolische Binde-Information vorhanden ist.

Maßnahme
Das zu testende Programm mit der Angabe SYMTEST= ALL oder SYMTEST=MAP bzw. TEST-SUPPORT=YES neu binden.

```
AID0264   Name of (&00) qualification not found (CMD: (&01))
AID0264   Name der (&00) Qualifikation nicht gefunden (KDO: (&01))
```

Bedeutung
Der Name der angegebenen Qualifikation wurde in der gueltigen Umgebung nicht gefunden.
(&00) Typ der nicht durchfuehrbaren Qualifikation: CTX, T, L, O, C, E, D

Maßnahme
Namen oder gueltige Umgebung aendern und Kommando erneut eingeben.

```
AID0265   Invalid qualification:(&00)
AID0265   Ungueltige Qualifikation:(&00)
```

Bedeutung
Fehlerhafte Qualifikation.

Maßnahme
Richtige Qualifikation eingeben.

```
AID0266    O qualification not supported by old linkage format (CMD: (&00))
AID0266    O-Qualifikation wird von altem Bindeformat nicht unterstuetzt (KDO: (&00))
```

Bedeutung
Objektmodul-Qualifikation wird nicht unterstuetzt fuer Programme, die mit einer TSOSLNK-Version < V15 gebunden wurden.

Maßnahme
Programm mit neuerer TSOSLNK-Version binden und Kommando erneut eingeben.

```
AID0267    Index not allowed for keyword (CMD: (&00))
AID0267    Schluesselwort kann nicht indiziert werden (KDO: (&00))
```

Bedeutung
Zum angegebenen Schluesselwort ist kein Index erlaubt.

Maßnahme
Schluesselwort ohne Index eingeben.

```
AID0268    No additional parameter allowed for event (CMD: (&00))
AID0268    Fuer dieses Ereignis sind keine weiteren Parameter zugelassen (KDO: (&00))
```

Bedeutung
Zum angegebenen Ereignis ist kein Zusatz erlaubt.

Maßnahme
Ereignis ohne Zusatz eingeben.

```
AID0269    AID allows only LSDCHECK=YES
AID0269    AID erlaubt nur LSDCHECK=YES
```

Bedeutung
LSDCHECK=NO ist nicht erlaubt (opt. REP einsetzen)

```
AID0270    Invalid keyword / event (CMD: (&00))
AID0270    Unzulaessiges Schluesselwort / Ereignis (KDO: (&00))
```

Bedeutung
Ungueltiges Schluesselwort oder Ereignis.

```
AID0271    Syntax error (CMD: (&00))
AID0271    Syntaxfehler (KDO: (&00))
```

Bedeutung
? Syntax-Fehler/ Regelverstoss. Moegliche Ursache:
- Syntax entspricht nicht der Beschreibung.
- Syntax ist nicht eindeutig.
! Syntax korrigieren/ Eindeutigkeit z.B. durch Klammerung erreichen.

```
AID0272    Command too long (CMD: (&00))
AID0272    Kommando ist zu lang (KDO: (&00))
```

Bedeutung
Eingegebenes Kommando ist zu lang.

Maßnahme
Kommando in mehrere Teilkommandos aufteilen und erneut eingeben.

```
AID0273    Address overflow (CMD: (&00))
AID0273    Adress-Ueberlauf (KDO: (&00))
```

Bedeutung
Adresse liegt ausserhalb des gueltigen Adressbereichs.

Maßnahme
Gueltige Adresse eingeben.

```
AID0274    Change desired? Reply (Y=Yes; N=No)
AID0274    Aenderung durchfuehren? Antwort (J=Ja; N=Nein)
```

Bedeutung
Soll der alte Inhalt durch den neuen Inhalt ersetzt werden ?

```
AID0275    CSECT qualification required
AID0275    CSECT-Qualifikation erforderlich
```

Bedeutung
Fuer das eingegebene Kommando ist eine CSECT-Qualifikation unbedingt erforderlich.

Maßnahme
CSECT-Qualifikation ergaenzen.

```
AID0276    %INSERT or %ON too deeply nested (CMD: (&00))
AID0276    %INSERT oder %ON zu tief verschachtelt (KDO: (&00))
```

Bedeutung
Schachtelungstiefe des eingegebenen %INSERT- oder %ON-Kommandos zu gross.

Maßnahme
Subkommandofolge eines inneren %INSERT- bzw. %ON- Kommandos aendern in '%STOP'. Beim Erreichen dieses Testpunktes/Ereignisses haelt AID an und der Benutzer kann erneut %INSERT- bzw. %ON- Kommandos verschachteln .

```
AID0277    No qualification defined (CMD: (&00))
AID0277    Keine Vorqualifikation definiert (KDO: (&00))
```

Bedeutung
Zur Aufloesung der angegebenen Vorqualifikation gibt es kein aktuelles %QUALIFY-Kommando.

Maßnahme
%QUALIFY-Kommando eingeben und Kommando wiederholen.

```
AID0278   Value outside supported range (CMD: (&00))
AID0278   Wert ausserhalb des unterstuetzten Werte-Bereichs (KDO: (&00))
```

Bedeutung
Wert liegt ausserhalb des von AID unterstuetzten Wertebereichs.

```
AID0279   Type not supported (CMD: (&00))
AID0279   Typ wird nicht unterstuetzt (KDO: (&00))
```

Bedeutung
Typ wird nicht von AID unterstuetzt.

```
AID0280   Task / file qualification not allowed (CMD: (&00))
AID0280   Task-/ Datei-Qualifikation nicht zulaessig (KDO: (&00))
```

Bedeutung
T-Qualifikation oder E=Dn-Qualifikation im eingegebenen Kommando nicht erlaubt.

```
AID0281   Explicit qualification ignored
AID0281   Explizite Qualifikation wird ignoriert
```

Bedeutung
Warnung: explizite Qualifizierung des Operanden wird ignoriert, weil unnoetig.

```
AID0282   TSN not in use
AID0282   TSN unbekannt
```

Bedeutung
TSN nicht vorhanden.

```
AID0283   No information given for dump file (CMD: (&00))
AID0283   Fuer Dump-Datei existiert keine Information (KDO: (&00))
```

Bedeutung
Zu diesem Kommando sind keine Informationen bei Dump-Datei-Qualifikation verfuegbar.

```
AID0284   Address inaccessible
AID0284   Adresse nicht zugreifbar
```

Bedeutung
Zugriff auf Adresse nicht moeglich.

```
AID0285   Warning: (&00) parameter too long; shortened to "(&01)"
AID0285   Warnung: (&00)-Parameter ist zu lang; wird auf "(&01)" gekuerzt
```

Bedeutung
Der in dem Kommando zum Aufbau einer Benutzeroberflaechenverbindung angegebene Parameter ist zu lang und wird gekuerzt.
(&00) Parametertyp
(&01) gekuerzter Parameter

```
AID0286   Variable type is not a pointer type (CMD: (&00))
AID0286   Variable ist nicht vom Typ Zeiger (KDO: (&00))
```

Bedeutung
Indirekte Adressierung kann nur mit Variablen vom Typ %A durchgefuehrt werden.

Maßnahme
Inhalt der Variablen, die zur indirekten Adressierung verwendet werden soll, mit Typmodifikation als %A interpretieren.

```
AID0287   Address not within code or linkage info (CSECT list) missing (CMD: (&00))
AID0287   Adr. ausserhalb des Codes o. fehlende Bindeinf. (CSECT-Liste) (KDO: (&00))
```

Bedeutung
Adresse liegt ausserhalb des Codes; oder es gibt keine CSECT-Liste fuer das geladene Programm.

Maßnahme
CSECT-Liste durch Neu-Binden des Programms mit Parameter SYMTEST=ALL oder SYMTEST=MAP bzw. TEST-SUPPORT=YES erzeugen

```
AID0288   No message with this number
AID0288   Meldungsnummer nicht belegt
```

Bedeutung
Diese Meldungsnummer ist nicht belegt

```
AID0289   No message with this number
AID0289   Meldungsnummer nicht belegt
```

Bedeutung
Diese Meldungsnummer ist nicht belegt

```
AID0290   Test point does not exist (CMD: (&00))
AID0290   Testpunkt nicht gesetzt (KDO: (&00))
```

Bedeutung
Der zu loeschende Testpunkt ist nicht oder nicht mehr gesetzt.

```
AID0291   Event does not exist (CMD: (&00))
AID0291   Ereignis nicht angemeldet (KDO: (&00))
```

Bedeutung
Das zu loeschende Ereignis ist nicht oder nicht mehr gesetzt.

```
AID0292   %INSERT / %ON rejected (CMD: (&00))
AID0292   %INSERT / %ON zurueckgewiesen (KDO: (&00))
```

Bedeutung
Testpunkt oder Ereignis konnte nicht gesetzt werden.

```
AID0293   Event not allowed in this OS version
AID0293   Ereignis in dieser Betriebssystem-Version nicht erlaubt
```

Bedeutung
Ereignis ist in dieser Betriebssystem Version noch nicht erlaubt.

```
AID0294   Page(s) from address (&00) to (&01) not dumped
AID0294   Seite(n) von Adresse (&00) bis (&01) nicht abgezogen
```

Bedeutung
Die Seite(n) mit den Adressen (&00) bis (&01) liegen nicht in der angegebenen Dump-Datei.

```
AID0295   Page(s) from address (&00) to (&01) not allocated
AID0295   Seite(n) von Adresse (&00) bis (&01) nicht zugewiesen
```

Bedeutung
Die Seite(n) mit den Adressen (&00) bis (&01) sind dieser Task nicht zugewiesen.

```
AID0296   Invalid memory class (CMD: (&00))
AID0296   Unzulaessige Speicherklasse (KDO: (&00))
```

Bedeutung
Das angesprochene Schluesselwort fuer Speicherklassen ist unzulaessig.

```
AID0297   TSN qualification required for dump file (&00) (CMD: (&01))
AID0297   TSN-Qualifikation fuer Dump-Datei (&00) erforderlich (KDO: (&01))
```

Bedeutung
Fuer den Zugriff auf die angegebene Dump-Datei ist die Angabe einer T-Qualifikation noetig
(&00) Linkname

Maßnahme
T-Qualifikation angeben.

```
AID0298   Testpoint removed; code not restored because of user modification
AID0298   Testpunkt entfernt; Code wegen Benutzer-Modif. nicht wiederhergestellt
```

Bedeutung
Am Testpunkt wurde mit einem %MOVE oder %SET der alte Code ueberschrieben; daher wird der alte Code nicht wiederhergestellt; der Testpunkt ist geloescht

```
AID0299   RECFORM parameter error
AID0299   Fehler im RECFORM-Parameter
```

Bedeutung
Satzformat ist ungleich 'V'.

Maßnahme
Satzformat auf 'V' aendern.

```
AID0300   Error on medium (CMD: (&00))
AID0300   Ausgabe-Medium macht Fehler (KDO: (&00))
```

Bedeutung
Ausgabemedium meldet Fehler.

Maßnahme
Anderes Ausgabemedium angeben.

```
AID0301    Segment of task (&00) inaccessible (CMD: (&01))
AID0301    Segment der Task (&00) nicht zugreifbar (KDO: (&01))
```

Bedeutung
Zugriff auf die angegebene Adresse einer Fremdtask ist nicht moeglich.
(&00) TSN

```
AID0302    System table does not exist or inaccessible (CMD: (&00))
AID0302    Systemtabelle existiert nicht oder ist nicht zugreifbar (KDO: (&00))
```

Bedeutung
Systemtabelle existiert nicht oder der Zugriff ist nicht moeglich. (Moegliche Ursachen: kein Programm geladen, Tabelle nicht in Dump-Datei enthalten, etc.).

```
AID0303    Memory allocation error (CMD: (&00))
AID0303    Speicherzugriffsfehler (KDO: (&00))
```

Bedeutung
Fehler beim Zugriff auf Speicher.

```
AID0304    Keyword / event parameter out of range (CMD: (&00))
AID0304    Bereichsverletzung fuer Schluesselwort- / Ereignis-Parameter (KDO: (&00))
```

Bedeutung
Parameter des angegebenen Schluesselwortes bzw. Ereignisses liegt ausserhalb des zulaessigen Wertebereichs.

```
AID0305    Illegal keyword / event parameter (CMD: (&00))
AID0305    Unzulaessiger Parameter fuer Schluesselwort / Ereignis (KDO: (&00))
```

Bedeutung
Parameter des angegebenen Schluesselwortes bzw. Ereignisses ist nicht zulaessig.

Maßnahme
Kommando korrekt eingeben.

```
AID0306    Invalid range specification
AID0306    Ungueltige Bereichsangabe
```

Bedeutung
Die Bereichsangabe ist falsch.

Maßnahme
Korrekte Bereichsangabe eingeben.

```
AID0307    Output medium (&00) not available
AID0307    Ausgabe-Medium (&00) nicht verfuegbar
```

Bedeutung
Ausgabemedium (&00) ist nicht verfuegbar.

```
AID0308    Exception handling: testpoint could not be set
AID0308    Exception Handling: Testpunkt konnte nicht gesetzt werden
```

Bedeutung
Der Code fuer das Exception Handling ist noch nicht geladen; Programm in einer C/C++ Funktion anhalten und dann den Testpunkt setzen

```
AID0309    PCB index out of range (CMD: (&00))
AID0309    PCB-Index nicht im zulaessigen Bereich (KDO: (&00))
```

Bedeutung
PCB mit angegebenem Index existiert nicht.

```
AID0310    Terminal request not honored (CMD: (&00))
AID0310    CONSOL-Umschaltung nicht moeglich (KDO: (&00))
```

Bedeutung
Umschalten der Ein-/Ausgabe auf Operateurkonsole bzw. Ruecksetzen auf SYSOUT nur unter TSOS moeglich.

```
AID0311    Error during execution of a system command (CMD: (&00))
AID0311    Fehler waehrend der Ausfuehrung eines System-Kommandos (KDO: (&00))
```

Bedeutung
Bei der Ausfuehrung eines Systemkommandos in einer AID-Kommandofolge ist ein Fehler aufgetreten.

Maßnahme
Systemkommando und Rest der Kommandofolge korrekt eingeben.

```
AID0312    Variable / literal not convertible
AID0312    Variable / Literal nicht konvertierbar
```

Bedeutung
Die Variable bzw. das Literal ist nicht konvertierbar.

```
AID0313    No program loaded (CMD: (&00))
AID0313    Kein Programm geladen (KDO: (&00))
```

Bedeutung
Das angegebene Kommando ist nur durchfuehrbar, wenn ein Programm geladen ist.

```
AID0314    TID not in use
AID0314    TID unbekannt
```

Bedeutung
Diese TID ist unbekannt.

```
AID0315    Dump file not open (CMD: (&00))
AID0315    Dump-Datei ist nicht geoeffnet (KDO: (&00))
```

Bedeutung
Dump-Datei wurde nicht geoeffnet.

Maßnahme
Dump-Datei mit dem %DUMPFILE-Kommando geoeffnen und Kommando erneut eingeben.

```
AID0316    %TITLE string too long (CMD: (&00))
AID0316    String im '%TITLE'-Kommando ist zu lang (KDO: (&00))
```

Bedeutung
Der im %TITLE-Kommando angegebene String ist zu lang; maximale Laenge: 80.

Maßnahme
%TITLE-String abkuerzen und Kommando erneut eingeben.

```
AID0317    Program enters STXIT / CONTINGENCY routine (PC: (&00))
AID0317    Benutzer-Programm laeuft in eine STXIT-/CONTINGENCY-Routine (PC: (&00))
```

Bedeutung
Das Benutzerprogramm betritt eine STXIT bzw. eine CONTINGENCY Routine.
(&00) Inhalt des Befehlszaehlers

```
AID0318    Further (&00) byte(s) not displayed
AID0318    Weitere (&00) Byte(s) werden nicht angezeigt
```

Bedeutung
Von den zu uebertragenden Bytes werden (&00) Bytes nicht mehr angezeigt.

```
AID0319    PARTNER/ROUTE/IP address unknown or inactive
AID0319    Partner/Route/IP-Adresse unbekannt oder nicht aktiv
```

Bedeutung
Eine Verbindung zur Benutzeroberflaeche konnte nicht hergestellt werden, weil Partner/Route/ IP-Adresse unbekannt oder nicht aktiv sind.

```
AID0320    User interface or version of user interface not supported
AID0320    Benutzeroberflaeche oder Vers. dieser Benutzeroberfl. nicht unterstuetzt
```

Bedeutung
Die Benutzeroberflaeche oder diese Version einer Benutzeroberflaeche wird nicht unterstuetzt.

```
AID0321    TID / TSN not in use
AID0321    TID / TSN unbekannt
```

Bedeutung
Diese TID/TSN ist unbekannt.

```
AID0322    ONLY parameter invalid (CMD: (&00))
AID0322    Ungueltiger ONLY-Parameter (KDO: (&00))
```

Bedeutung
Der Parameter ONLY des %INSERT- / %ON-Kommandos ist fehlerhaft.

Maßnahme
ONLY-Parameter korrigieren und Kommando erneut eingeben.

```
AID0323   No match found for virtual address (CMD: (&00))
AID0323   Virtuelle Adresse nicht gefunden (KDO: (&00))
```

Bedeutung
Die eingegebene virtuelle Adresse ist nicht im Programm enthalten, d.h. es ist nicht moeglich Programm-, Lademodul-, Objektmodul- und CSECT- Name anzugeben, in denen die Adresse enthalten ist (z.B. weil vom Binder keine CSECT-Liste erzeugt wurde).

Maßnahme
Wenn keine CSECT-Liste existiert, Programm neu binden mit dem Parameter SYMTEST=ALL oder SYMTEST=MAP bzw. TEST-SUPPORT=YES und Kommando erneut eingeben.

```
AID0324   Privilege too low but can be raised
AID0324   Testprivilegierung zu niedrig, kann jedoch erhoeht werden
```

Bedeutung
Testprivilegierung ist fuer die gewuenschte Funktion zu niedrig, koennte aber mit dem SDF-Kommando /MODIFY-TEST-OPTIONS noch erhoeht werden.

Maßnahme
Testprivilegierung mit dem SDF-Kommando /MODIFY-TEST-OPTIONS erhoehen.

```
AID0325   Privilege too low and cannot be raised
AID0325   Testprivilegierung zu niedrig und kann nicht erhoeht werden
```

Bedeutung
Testprivilegierung ist fuer die gewuenschte Funktion zu niedrig, und kann auch mit dem SDF-Kommando /MODIFY-TEST-OPTIONS nicht genuegend erhoeht werden.

Maßnahme
Der Benutzer kann die geforderte Funktion unter seiner Kennung nicht erhalten. Hoehere Testprivilegierung beim Systemverwalter beantragen.

```
AID0326   IEEE floatingpoint value not a number
AID0326   IEEE Gleitpunktzahlenwert ist keine Zahl
```

Bedeutung
Der Gleitpunktzahlenwert ist keine Zahl.

```
AID0327   %LOC parameter invalid (CMD: (&00))
AID0327   Ungueltiger %LOC-Parameter (KDO: (&00))
```

Bedeutung
Falscher Parameter bei %LOC.

Maßnahme
Parameter korrigieren und Kommando erneut eingeben.

```
AID0328    Invalid test point location (CMD: (&00))
AID0328    Ungueltige Testpunktadresse (KDO: (&00))
```

Bedeutung
Ungueltige Testpunkt-Adresse.

Maßnahme
Testpunkt-Adresse korrigieren und Kommando erneut eingeben.

```
AID0329    Execution of %HELP command impossible (CMD: (&00))
AID0329    %HELP-Kommando kann nicht ausgefuehrt werden (KDO: (&00))
```

Bedeutung
%HELP-Kommando kann wegen fehlender oder inkonsistenter AID-Textdatei nicht ausgefuehrt werden.

Maßnahme
Systemverwalter verstaendigen.

```
AID0330    System table empty (CMD: (&00))
AID0330    System-Tabelle ist leer (KDO: (&00))
```

Bedeutung
Die angesprochene Systemtabelle ist leer.

```
AID0331    Invalid link name (CMD: (&00))
AID0331    Ungueltiger Link-Name (KDO: (&00))
```

Bedeutung
Linkname ungueltig.

Maßnahme
Linkname korrigieren und Kommando erneut eingeben.

```
AID0332    Access to dump file on tape not supported (CMD: (&00))
AID0332    Zugriff auf Dump-Datei auf Band wird nicht unterstuetzt (KDO: (&00))
```

Bedeutung
Zugriff auf Dump-Dateien, die auf Band stehen, wird nicht unterstuetzt.

Maßnahme
Dump-Datei auf Platte bringen und Kommando erneut eingeben.

```
AID0333    Inconsistency detected by AID in module (&00) (CMD: (&01))
AID0333    AID hat in Modul (&00) eine Inkonsistenz festgestellt (KDO: (&01))
```

Bedeutung
Inkonsistenz wurde von AID entdeckt.
(&00) Modulnummer des Moduls, das den Fehler entdeckt hat

Maßnahme
Systemverwalter informieren.

```
AID0334   Invalid length (CMD: (&00))
AID0334   Ungueltige Laenge (KDO: (&00))
```

Bedeutung
Laenge ungueltig.

Maßnahme
Laenge korrigieren und Kommando erneut eingeben.

```
AID0335   No message with this number
AID0335   Meldungsnummer nicht belegt
```

Bedeutung
Diese Meldungsnummer ist nicht belegt

```
AID0336   File not open (CMD: (&00))
AID0336   Datei ist nicht geoeffnet (KDO: (&00))
```

Bedeutung
Datei ist nicht geoeffnet.

```
AID0337   Invalid command name or invalid msg nr. in %HELP command (CMD: (&00))
AID0337   Fehlerhafte(r) Kommandoname/Meldungsnummer im %HELP-Kommando (KDO: (&00))
```

Bedeutung
Falscher Kommandoname im %HELP-Kommando oder angegebene Meldungsnummer ist nicht vorhanden.

```
AID0338   %INSERT / %ON not allowed for foreign task or dump file (CMD: (&00))
AID0338   %INSERT / %ON fuer Fremdtask oder Dump-Datei unzulaessig (KDO: (&00))
```

Bedeutung
Mit %INSERT- oder %ON-Kommando kann nicht auf eine fremde Task oder Dump-Datei zugegriffen werden.

```
AID0339   Interrupt in connection to user interface. Connection closed
AID0339   Unterbrechung in Verbindung zur Benutzeroberflaeche. Verbindung abgebaut
```

Bedeutung
Die Verbindung zum Partner-Rechner weist eine Unterbrechung auf und wurde deshalb abgebaut

```
AID0340   Terminal output terminated by user interrupt
AID0340   Terminalausgabe von Benutzer unterbrochen
```

Bedeutung
Die Ausgabe auf SYSOUT wurde beendet aufgrund einer vom Benutzer erzeugten Unterbrechung.

```
AID0341   %MOVE/%SET rejected because of intermediate modification of old content
AID0341   %MOVE/%SET wegen zwischenzeitl. Aenderung des alten Inhalts abgelehnt
```

Bedeutung
Bei einem %MOVE/%SET-Kommando hat sich zwischen dem Zeitpunkt der Ausgabe des alten Inhalts und der Beantwortung der Frage 'Aenderung ... (J=Ja; N=Nein)' mit 'J' der alte Inhalt veraendert. Die Aenderung wird deshalb nicht durchgefuehrt.

Maßnahme
Kommando erneut eingeben. Eventuell den CHECK-Parameter im %AID-Kommando ausschalten.

```
AID0342   Nothing changed
AID0342   Keine Aenderung
```

Bedeutung
Es wurde keine Aenderung durchgefuehrt. Der Benutzer hat die Frage 'Aenderung ... (J=Ja; N=Nein)' mit 'N' beantwortet.

```
AID0343   Change of dump file not supported (CMD: (&00))
AID0343   Aenderung in einer Dump-Datei wird nicht unterstuetzt (KDO: (&00))
```

Bedeutung
Eine Aenderung in einer Dump-Datei wird von AID nicht unterstuetzt.

```
AID0344   LSD version for SOURCE module (&00) not supported
AID0344   LSD-Version fuer Source-Modul (&00) wird nicht unterstuetzt
```

Bedeutung
Der Compiler hat fuer die Programmeinheit (&00) die LSD in einer Version erzeugt, die von AID nicht bearbeitet werden kann.

Maßnahme
Die Programmeinheit (evtl. auch noch weitere) muss mit Hilfe eines anderen Compilers, der LSD in einer von AID unterstuetzten LSD-Version erzeugen kann, neu uebersetzt werden oder man muss eine andere AID-Version benutzen, die auf der vom Compiler erzeugten LSD-Version arbeiten kann.

```
AID0345   %MOVE/%SET exceeds segment boundaries; only (&00) bytes moved
AID0345   %MOVE/%SET ueberschreitet Segmentgr. ; nur (&00) Bytes uebertragen
```

Bedeutung
Aenderung geht ueber Segmentgrenzen hinaus; es wurden nur (&00) Bytes geaendert.

Maßnahme
Restliche zu aendernde Bytes in eigenem %MOVE- Kommando uebertragen.

```
AID0346   Location to be changed not allocated (CMD: (&00))
AID0346   Zu aendernde Speicherstelle nicht zugewiesen (KDO: (&00))
```

Bedeutung
Zu aendernde Speicherstelle ist nicht zugewiesen.

```
AID0347    Array (&00) must be subscripted (CMD: (&01))
AID0347    Das Feld (&00) muss indiziert werden (KDO: (&01))
```

Bedeutung
Hier ist es zwingend erforderlich, dass das Feld (&00) indiziert wird.

```
AID0348    Program stopped due to (&00) event (&01)
AID0348    Programm angehalten wegen (&00)-Ereignis (&01)
```

Bedeutung
Ein Ereignis (&00), dessen Ueberwachungdurch einen AID-Schalter (FORK oder EXEC) aktiviert wurde, ist eingetreten, oder es wurde ein %STOP von einer anderen Task der Familie abgesetzt. In einer Fork-Task wird mit (&01) die PID ausgegeben.

```
AID0349    No message with this number
AID0349    Meldungsnummer nicht belegt
```

Bedeutung
Diese Meldungsnummer ist nicht belegt

```
AID0350    %FIND without parameters not allowed in this context (CMD: (&00))
AID0350    %FIND ohne Parameter in diesem Zusammenhang nicht erlaubt (KDO: (&00))
```

Bedeutung
%FIND-Kommando ohne Parameter in diesem Zusammenhang nicht zulaessig.

Maßnahme
%FIND-Kommando mit Parametern (zumindest mit Suchbegriff) eingeben.

```
AID0351    No match in range
AID0351    Kein Treffer im Suchbereich
```

Bedeutung
Kein Treffer im %FIND-Bereich.

```
AID0352    No additional match in range
AID0352    Kein weiterer Treffer im Suchbereich
```

Bedeutung
Kein weiterer Treffer im %FIND-Bereich.

```
AID0353    Length exceeds boundaries; only (&00) bytes moved.
AID0353    Laenge ueberschreitet Grenzen ; nur (&00) Bytes uebertragen.
```

Bedeutung
Angegebene Laenge ueberschreitet Grenzen; nur (&00) Bytes wurden uebertragen.

```
AID0354    Given length exceeds boundaries; nothing moved (CMD: (&00))
AID0354    Angegebene Laenge ueberschreitet Grenzen; nichts uebertragen (KDO: (&00))
```

Bedeutung
Angegebene Laenge ueberschreitet Schluesselwort- oder Variablen-Grenzen; es wurde nichts uebertragen.

```
AID0355    %MOVE / %SET rejected
AID0355    %MOVE / %SET zurueckgewiesen
```

Bedeutung
Das %MOVE / %SET -Kommando wurde zurueckgewiesen.

```
AID0356    Keyword not allowed in command (CMD: (&00))
AID0356    Schluesselwort im Kommando nicht erlaubt (KDO: (&00))
```

Bedeutung
Dieses Schluesselwort ist im eingegebenen Kommando nicht zugelassen.

```
AID0357    Modification not allowed for keyword (CMD: (&00))
AID0357    Aenderung fuer dieses Schluesselwort nicht erlaubt (KDO: (&00))
```

Bedeutung
Bei diesem Schluesselwort ist die eingegebene Modifikation nicht zugelassen.

```
AID0358    No information given for foreign task (CMD: (&00))
AID0358    Keine Information ueber Fremdtask vorhanden (KDO: (&00))
```

Bedeutung
Zu diesem Kommando sind keine Informationen bei Fremdtaskqualifikation verfuegbar.

```
AID0359    (&00) is neither a class nor a namespace specification
AID0359    (&00) stellt weder einen Klassen- noch einen Namespace-Namen dar
```

Bedeutung
Der vor dem '::'-Operator angegebene Name ist weder ein Klassen-Name noch ein Namespace- Name .
(&00) Klassen- bzw. Namespace-Name

```
AID0360     "::" must not succeed a BLK/PROC qualification
AID0360     "::" darf nicht hinter einer BLK-/PROC-Qualifikation stehen
```

Bedeutung
Der globale Daten-Scope-Operator '::' darf nach einer Block- oder Prozedur-Qualifikation nicht stehen .

```
AID0361    The command is not allowed in dump file environment
AID0361    Das Kommando ist bei Basisqualifikation auf Dumpfile nicht erlaubt
```

Bedeutung
Das Kommando ist nur zulaessig bei Basisqualifikation auf den virtuellen Speicher.

```
AID0362    Function is locked (CMD: (&00))
AID0362    Funktion ist gesperrt (KDO: (&00))
```

Bedeutung
Funktion kann z.Zt. nicht aufgerufen werden.

```
AID0363    %MOVE exceeds CSECT boundaries; REP(s) suppressed (CMD: (&00))
AID0363    %MOVE ueberschreitet CSECT-Grenzen; REP(s) unterdrueckt (KDO: (&00))
```

Bedeutung
Es konnte kein REP erstellt werden, weil das Empfangsfeld im %MOVE-Kommando CSECT-Grenze(n) ueberschreitet.

Maßnahme
Uebertragung mit mehreren %MOVE-Kommandos durchfuehren, bei denen keine CSECT-Grenze ueberschritten wird.

```
AID0364   No match found for target address; REP(s) suppressed (CMD: (&00))
AID0364   Zieladresse nicht gefunden; REP(s) unterdrueckt (KDO: (&00))
```

Bedeutung
Empfaenger konnte aufgrund fehlender CSECT- Information nicht lokalisiert werden; es wurde(n) kein(e) REP(s) erstellt.

Maßnahme
Evtl. Programm mit SYMTEST=ALL oder SYMTEST=MAP bzw. TEST-SUPPORT=YES neu binden.

```
AID0365   REP generation error; no REP issued (CMD: (&00))
AID0365   Fehler bei der REP-Erzeugung; kein REP erzeugt (KDO: (&00))
```

Bedeutung
Fehler bei der Ausgabe von REP-Information auf Datei, z.B. formaler Fehler, kein Medium bzw. keine Datei vorhanden.

```
AID0366   File error : (&00) ; no REP issued (CMD: (&01))
AID0366   Fehler beim Zugriff auf Datei (&00) ; kein REP erzeugt (KDO: (&01))
```

Bedeutung
Bei der Ausgabe von REP-Information in die Datei (&00) wurde ein Fehler festgestellt. Der REP wurde nicht erstellt.

```
AID0367   File (&00) cannot be opened; DMS error code (&01); no REP issued (CMD: (&02))
AID0367   Datei (&00) kann nicht geoeff. werden; DMS (&01); kein REP erz. (KDO: (&02))
```

Bedeutung
Bei der Ausgabe von REP-Information in die Datei (&00) wurde vom DMS der Fehler (&01) gemeldet. der REP wurde nicht erstellt.

```
AID0368   FCB type not supported for file (&00) ; no REP issued (CMD: (&01))
AID0368   Fehlerhafter FCB-Typ der Datei (&00) ; kein REP erzeugt (KDO: (&01))
```

Bedeutung
Es wurde versucht, einen REP auf eine Datei auszugeben, die vom Benutzer erstellt wurde; Zuordnung erfolgt ueber den Linknamen (&00). der FCB-Typ der Datei wird von AIDSYS nicht unterstuetzt. Es wurde kein REP ausgegeben.

Maßnahme
FCB-Typ aendern, Kommando wiederholen.

```
AID0369    No linkage information available; REP(s) suppressed (CMD: (&00))
AID0369    Keine Binde-Information verfuegbar; REP(s) unterdrueckt (KDO: (&00))
```

Bedeutung
Das zu testende Programm wurde entweder mit einer Binder-Version < V16 gebunden, oder die Erstellung von CSECT-Listen wurde unterdrueckt. Es ist keine Lokalisierung auf Objektmodule moeglich. Eine LMS-Korrektur-Anweisung kann nicht ausgegeben werden und wird deshalb unterdrueckt.

Maßnahme
Programm mit TSOSLNK >= V16 und Parameter SYMTEST=ALL oder SYMTEST=MAP bzw. TEST-SUPPORT=YES neu binden. Kommando wiederholen.

```
AID0370    File error (&00) or SLED without virtual address space (CMD: (&01))
AID0370    Dateifehler (Linkname (&00)) oder SLED-Datei ohne virt. Adressraum (KDO: '
```

Bedeutung
Fehler beim Zugriff auf die Datei mit dem Linknamen (&00) oder SLED-Datei ohne virtuellen Adressraum.

```
AID0371    Task/file qualification not allowed for event (CMD: (&00))
AID0371    Task-/Datei-Qualifikation ist fuer ein Ereignis nicht erlaubt (KDO: (&00))
```

Bedeutung
Bei Angabe von Ereignissen ist eine Fremdtask- oder E=Dn-Qualifizierung nicht erlaubt.

```
AID0372    Task/file qualification not allowed for test point (CMD: (&00))
AID0372    Task-/Datei-Qualifikation ist fuer einen Testpunkt nicht erlaubt (KDO: '
```

Bedeutung
Bei Angabe von Testpunkten ist eine Fremdtask- oder E=Dn-Qualifizierung nicht erlaubt.

```
AID0373    Stop not possible for given (&00). (Reason (&01))
AID0373    Stop Kommando fuer angegebene (&00) nicht moeglich. (Grund (&01))
```

Bedeutung
Die Anweisung wird nicht ausgefuehrt, wenn (&00)
Grund -10 : die eigene Task ist.
-14 : keine Fork Task / nicht vorhanden ist.
-18 : kein Mitglied der Taskfamilie ist.

```
AID0374    Requested information not in dump file
AID0374    Angeforderte Information wurde in der Dump-Datei nicht gefunden
```

Bedeutung
Zur Erfuellung der gewuenschten Funktion sind nicht alle benoetigten Seiten in Dump-Datei enthalten.

```
AID0375     (&00) (&01) not found
AID0375     (&00) (&01) nicht gefunden
```

Bedeutung
Sourcemodul,Prozedur/Block, Symbol oder Source- Referenz in der momentan gueltigen bzw. explizit angegebenen Umgebung nicht gefunden oder Source- modul im geladenen Programm nicht gefunden.
(&00) SOURCE_MODUL, PROC/BLK, SYMBOL, SRC_REF_#
(&01) Name

```
AID0376  Ambiguous or incomplete qualification for (&00) (&01)
AID0376  Mehrdeutige oder unvollstndige Qualifikation fuer (&00) (&01)
```

Bedeutung
Mehrdeutige oder unvollstaendige Qualifikation, falls Vollqualifikation erforderlich, fuer Prozedur oder Symbol.
(&00) PROC oder SYMBOL
(&01) Name der Prozedur (PROC) oder desSymbols (SYMBOL)

Maßnahme
Eindeutige bzw., falls erforderlich, vollstaendige Qualifikation waehlen.

```
AID0377  Symbolic information inconsistent for (&00)
AID0377  Inkonsistente Symbolinformation fuer (&00)
```

Bedeutung
Die symbolische Information wurde nicht beim gleichen Uebersetzungsvorgang erzeugt wie das geladene Objekt.
(&00) Sourcemodul

Maßnahme
Richtige Bibliothek angeben bzw. neu uebersetzen.

```
AID0378  Symbolic information missing
AID0378  Symbolinformation fehlt
```

Bedeutung
Es wurde ein Kommando eingegeben, das LSD-Infor-mation voraussetzt. Diese ist falsch oder nicht vorhanden.

Maßnahme
Bibliothek fuer LSD-Information angeben oder mit mit Symbolinformations-Option uebersetzen, binden und laden oder nur Kommandos fuer das Testen auf Maschinencode-Ebene verwenden (z.B. %TRACE %INSTR)

```
AID0379  S and PROC qualification required or LSD information missing
AID0379  S- und PROC-Qualifikation ist notwendig oder die LSD-Information fehlt
```

Bedeutung
Die Unterbrechungsstelle befindet sich ausserhalb einer eindeutigen Sourcemodul-/Prozedurumgebung oder es ist keine LSD-Information (Symbolinformation) vorhanden.

Maßnahme
S- und PROC-Qualifikation angeben oder Bibliothek fuer LSD-Information zuweisen.

```
AID0380   Invalid explicit basing (&00) (&01) (CMD: (&02))
AID0380   Fehlerhafte explizite Basis (&00) (&01) (KDO: (&02))
```

Bedeutung
Basis nicht vom Typ 'pointer' oder Typ der Variablen nicht als 'based' deklariert.
(&00) Name
(&01) ->

```
AID0381    (&00) (&01) not of type INTEGER (CMD: (&02))
AID0381    (&00) (&01) ist nicht vom Typ INTEGER (KDO: (&02))
```

Bedeutung
Index entspricht nicht dem Typ 'integer' oder Multiplier ist keine Ganzzahl.
(&00) INDEX oder MULTIPLIZIERER
(&01) Name des Indexes (INDEX) oder des Multipliers (MULTIPLIZIERER)

```
AID0382   Invalid dimension of array (&00) (CMD: (&01))
AID0382   Fehlerhafte Dimensionsangabe des Feldes (&00) (KDO: (&01))
```

Bedeutung
Tabelle (&00) hat falsche Dimension.

```
AID0383   No subscript allowed for symbol (&00)
AID0383   Das Symbol (&00) darf nicht indiziert werden
```

Bedeutung
Symbol (&00) darf nicht subskribiert werden.

```
AID0384   Component list conflicts with structure type/DSECT (&00) (CMD: (&01))
AID0384   Komponentenl. stimmt nicht mit Strukturtyp/DSECT (&00) ueberein (KDO: (&01))
```

Bedeutung
Das Layout fuer die Mode-Konstante stimmt nicht mit dem Base-Model (&00) ueberein.

```
AID0385    (&00) (&01) not within nest (CMD: (&02))
AID0385    (&00) (&01) liegt nicht in der Aufrufhierarchie (KDO: (&02))
```

Bedeutung
Sourcemodul oder Prozedur ist nicht in der momentanen Aufrufhierarchie enthalten.
(&00)SOURCE-MODUL oder PROC (&01)Name des Souercemoduls bzw. der Prozedur

```
AID0386   No message with this number
AID0386   Meldungsnummer nicht belegt
```

Bedeutung
Diese Meldungsnummer ist nicht belegt

```
AID0387   Too many PROC/BLK qualifications (CMD: (&00))
AID0387   Zu viele PROC/BLK - Qualifikationen (KDO: (&00))
```

Bedeutung
PROC- bzw. BLK-Qualifikation hat eine zu tiefe Schachtelung.
(&00) PROC oder BLK

Maßnahme
Kuerzere, noch eindeutige Qualifikation waehlen.

```
AID0388   Types are not convertible; nothing changed/compared (CMD: (&00))
AID0388   Typen lassen sich nicht konvert.; nichts geaendert/verglichen (KDO: (&00))
```

Bedeutung
Typen sind nicht konvertierbar. Es wurde entweder der Wert des Empfaengers nicht veraendert (%SET) oder der Vergleich in einer Subkommando-Bedingung (CONDITION) nicht durchgefuehrt und auf FALSE gesetzt.

```
AID0389   %SET array INTO array not yet implemented (CMD: (&00))
AID0389   %SET <feld> INTO <feld> wird noch nicht unterstuetzt (KDO: (&00))
```

Bedeutung
Funktion ist nicht in der aktuellen AID-Version enthalten.

```
AID0390   Warning: source truncated
AID0390   Warnung: Quelldatum wurde abgeschnitten
```

Bedeutung
Warnung: der konvertierte Wert entspricht nicht dem urspruenglichen.

Maßnahme
Falls gewuenscht keine, andernfalls Kommando mit korrigierter Eingabe wiederholen.

```
AID0391   Task with (&00) is unknown.
AID0391   Task mit (&00) ist unbekannt.
```

Bedeutung
Die Task mit der angegebenen TSN/PID (&00) wurde nicht gefunden.

```
AID0392   Value(s) of >(&00)< does(do) not match to type declaration
AID0392   Wert(e) von >(&00)< passt (passen) nicht zur Typdeklaration
```

Bedeutung
Der aktuelle Inhalt des vorliegenden oder nach folgenden Datums oder Datenelements (&00) entspricht nicht dem deklarierten Typ. Bei Feldern koennen dies fuer mehrere Elemente zutreffen.

```
AID0393   Symbol too complex or a too deeply nested struct component (CMD: (&00))
AID0393   Symbol ist zu komplex oder zu tief geschachtelte Komponente (CMD: (&00))
```

Bedeutung
Komplexe Speicherreferenz hat zuviele Komponenten. oder auf Grund von AID Restriktionen ist sie nicht testbar.

Maßnahme
Schrittweise mit Teilkomponenten arbeiten.

```
AID0394   Structure type/DSECT without comp. list/explicit basing not supported
AID0394   Strukturtyp/DSECT ohne Komponentenl./explizite Basis wird nicht unterst.
```

Bedeutung
Model-Namen duerfen nur als Layout-Bezeichner von Mode-Konstanten verwendet werden.

```
AID0395   Division by zero
AID0395   Division durch Null
```

Bedeutung
Division durch Null ist nicht erlaubt.

```
AID0396   Invalid address for (&00)
AID0396   (&00) hat eine ungueltige Adresse
```

Bedeutung
Das Speicherobjekt (&00) ist aktuell nicht greifbar.

```
AID0397   (Next component of) (&00) has length <= 0 or is an empty string
AID0397   (Naechste Komponente von) (&00) hat die Laenge <= 0 oder ist Leerstring
```

Bedeutung
Die Komponente von (&00) hat die Laenge <= 0 oder ist ein Leerstring (C/C++)

```
AID0398   Symbol (&00) represents no instruction address
AID0398   Das Symbol (&00) stellt keine Befehlsadresse dar
```

Bedeutung
Symbol (&00) stellt keine Befehlsadresse dar.

```
AID0399   Odd or unallocated address
AID0399   Ungerade oder nicht allokierte Speicheradresse
```

Bedeutung
Die Adresse ist entweder ungerade oder nicht belegt.

```
AID0400   Dimension (&00) of array (&01) out of range or array has no element
AID0400   Dimension (&00) des Feldes (&01) gibt es nicht oder das Feld hat kein Elem.
```

Bedeutung
Dimensionswert (&00) der Tabelle (&01) liegt ausserhalb des zulaessigen Wertebereichs.

```
AID0401   %MOVE / %SET into constant not allowed
AID0401   Konstante als Ziel einer %MOVE / %SET - Anweisung nicht zugelassen
```

Bedeutung
Konstanten koennen mit %MOVE / %SET nicht ueberschrieben werden.

```
AID0402   Warning: absolute value moved!
AID0402   Warnung: der absolute Wert wurde uebertragen!
```

Bedeutung
Warnung: Das Empfangsfeld ist ein vorzeichenloses numerisches Datum. Es wurde der Absolutbetrag des Sendefeldes uebertragen.

```
AID0403   (&00) must not be indexed by (&01)
AID0403   (&00) darf durch (&01) nicht indiziert werden
```

Bedeutung
(&00) ist Spezialindexdatenfeld oder Spezialindex mit anderem Basisfeld als (&01).

```
AID0404   AID cannot reference statement (&00) due to compiler optimization
AID0404   Die Anweisung (&00) kann von AID wegen Optimierung nicht referenz. werden
```

Bedeutung
Wegen Compiler-Optimierung ist das Statement mit der Source-Referenz (&00) fuer AID nicht vorhanden

```
AID0405   Too many libraries
AID0405   Zu viele Bibliotheken
```

Bedeutung
Ueber das %SYMLIB-Kommando wurden mehr als 15 Bibliotheken angemeldet.

Maßnahme
Mit dem leeren %SYMLIB-Kommando alle Bibliotheken abmelden und dann weniger als 15 Bibliotheken anmelden.

```
AID0406   AID cannot access library (&00); DMS error code (&01)
AID0406   Zugriff auf Bibliothek (&00) nicht moeglich; DMS-Fehler-Code (&01)
```

Bedeutung
Die Bibliothek (&00) ist fuer AID momentan nicht zugreifbar; (&01) DMS-Fehler-Code. Moegliche Gruende -geschuetzt/gesperrt/Passwort fehlt.

Maßnahme
Problem moeglicherweise beseitigen und %SYMLIB- Kommando erneut eingeben.

```
AID0407   File (&00) is not a PLAM library
AID0407   Die Datei (&00) ist keine PLAM-Bibliothek
```

Bedeutung
Die Datei (&00) ist keine PLAM-Bibliothek. Die restlichen Bibliotheken des %SYMLIB-Kommandos sind angemeldet worden.

```
AID0408   Member (&00) of library (&01) is LOCKED
AID0408   Das Element (&00) der Bibliothek (&01) ist gesperrt
```

Bedeutung
Auf das Element (&00) der Bibliothek (&01) kann AID momentan nicht zugreifen.

Maßnahme
Nach Freigabe des Elements AID-Kommando erneut eingeben.

```
AID0409   Only positive INTEGER values allowed for special index
AID0409   Fuer einen Spezialindex sind nur positive Zahlen erlaubt
```

Bedeutung
Empfaenger ist vom Typ 'index', dessen Inhalt stets positiv und ganzzahlig sein muss.

```
AID0410   %TRACE / %CONTROL not supported in foreign task or dump file
AID0410   Fuer Fremdtask/Dump-Datei wird %TRACE/%CONTROL nicht unterstuetzt
```

Bedeutung
Das %TRACE- und %CONTROL-Kommando ist nur in der eigenen Task zugelassen.

Maßnahme
Kommando mit korrigierter Eingabe wiederholen - unter Beruecksichtigung eines noch geltenden %BASE-Kommandos.

```
AID0411   File with specified link (&00) is already open
AID0411   Die Datei mit dem Linknamen (&00) ist bereits offen
```

Bedeutung
Datei mit dem Linknamen (&00) ist bereits geoeffnet.

Maßnahme
a) Dump-Datei mit %DUMPFILE Dn schliessen und erneut mit %DUMPFILE Dn=dateiname oeffnen oder
b) anderen Linknamen verwenden.

```
AID0412   Symbol (&00) not within nest and not of storage class STATIC/CONSTANT
AID0412   Symbol (&00) nicht in der Aufrufhierarchie u. nicht vom Typ stat./konstant
```

Bedeutung
Das Symbol (&00) kann aktuell nicht referenziert werden.

Maßnahme
Kommando zu einem Zeitpunkt wiederholen, zu dem sich das Symbol innerhalb der Aufrufhierarchie befindet.

```
AID0413   Range exceeds segment / CSECT boundaries
AID0413   Segment-/CSECT-Grenzen werden ueberschritten
```

Bedeutung
Bereich ueberschreitet die Segment- oder CSECT- Grenzen.

Maßnahme
Bereich neu angeben.

```
AID0414   Range specification incorrect
AID0414   Die Bereichsangabe ist falsch
```

Bedeutung
Falsche Bereichsangabe.

```
AID0415   Combination of HIGH LEVEL range and LOW LEVEL mode illegal
AID0415   Kombination von High-Level-Bereich und Low-Level-Kriterium unzulaessig
```

Bedeutung
Mischen von symbolischen und maschinennahen Operanden innerhalb des Kommandos nicht zulaessig.

```
AID0416    (&00) was not set
AID0416    (&00) wurde nicht gesetzt
```

Bedeutung
Das zu loeschende %CONTROL-Kommando (%C1 -%C7) war nicht gesetzt.

```
AID0417   Program counter not within code; use expl. qualification (PROG= / S=)
AID0417   Befehlszaehler nicht im Code; expl. Qualifikation benutzen (PROG= / S=)
```

Bedeutung
Befehlszaehler ist ausserhalb des Benutzerprogramms; S- bzw. PROG-Qualifikation verwenden.

```
AID0418   Invalid parameter combination in this context (CMD: (&00))
AID0418   In diesem Zusammenhang ungueltige Parameterkombination (KDO: (&00))
```

Bedeutung
Es wurden ein oder mehrere in diesem Zusammenhang ungueltige Parameter bei dem Kommando angegeben.

```
AID0419   Compiler register optimization prohibits modification of variable (&00)
AID0419   Compiler-Register-Optimierung verbietet Aenderung der Variablen (&00)
```

Bedeutung
Wegen potentieller Inkonsistenz im weiteren Programmablauf kann der Wert der Variablen (&00) derzeit nicht veraendert werden.

Maßnahme
An anderer Stelle im Programm wiederholen.

```
AID0420   AID cannot reference variable (&00) due to compiler optimization
AID0420   Wegen Compiler-Optimierung kann AID auf die Variable (&00) nicht zugreifen
```

Bedeutung
Die Variable (&00) wurde zwar deklariert, wird aber im Programm nicht referenziert. Wegen Compiler-Optimierung ist sie fuer AID nicht ansprechbar.

```
AID0421   Nested %INSERT on label or entry variable not allowed (CMD: (&00))
AID0421   Geschachtelte %INSERTs auf LABEL- oder ENTRY-Var. unzulaessig (KDO: (&00))
```

Bedeutung
Wenn ein Testpunkt erreicht wird oder ein Ereignis eintritt, darf das Subkommando kein %INSERT auf eine Label- bzw. Entry-Variable sein.

```
AID0422   Program stack corrupted ( invalid back link or stack address )
AID0422   Fehler im Programm-Stack ( falsche Ruecksprung- oder Stack-Adresse )
```

Bedeutung
AID kann einen Programm-Stack nicht interpretieren. Dies kommt vor, wenn die Stack-Verkettung zerstoert wurde (z.B. durch einen Programm-Fehler)

Maßnahme
Programm-Fehler mit maschinennahen AID-Funktionen eingrenzen.

```
AID0423   No (&00) set
AID0423   Kein (&00) gesetzt
```

Bedeutung
Das eingegebene %REMOVE/%SHOW-Kommando blieb ohne Wirkung, da kein %CONTROLn, %INSERT oder %ON angemeldet war.

```
AID0424   File (&00) could not be opened; DMS error code (&01)
AID0424   Die Datei (&00) kann nicht geoeffnet werden; DMS-Fehler-Code (&01)
```

Bedeutung
Bei der Ausgabe von Informationen in die Datei (&00) wurde von DMS der Fehler (&01) (siehe BS2000 Meldungen) gemeldet.

Maßnahme
Fehler beheben, Kommando wiederholen.

```
AID0425   FCB type not supported for file (&00)
AID0425   FCB-Typ fuer Datei (&00) wird nicht unterstuetzt
```

Bedeutung
Es wurde versucht einen REP auf eine Datei auszugeben, die vom Benutzer erstellt wurde; Zuordnung erfolgt ueber den Linknamen (&00). Der FCB-Typ der Datei wird von AIDSYS nicht unterstuetzt.

Maßnahme
FCB-Typ aendern, Kommando wiederholen.

```
AID0426   Warning: ambiguity of (&00) (&01) not completely checked (OVERLAY loading)
AID0426   Warnung: Mehrdeutigkeit von (&00) (&01) nicht vollst. ueberpr. ( OVERLAY )
```

Bedeutung
Warnung: Prozedur oder Symbol wurde im geladenen Teil des mit Overlay-Technik gebundenen Programms als eindeutig erkannt. Ob der Name der Prozedur oder des Symbols ueber den gesamten Sourcemodul eindeutig ist, konnte nicht ueberprueft werden.
(&00 PROZEDUR oder SYMBOL
(&01) Name der Prozedur oder des Symbols

```
AID0427    (&00) (&01) not found in loaded part of SOURCE module ( OVERLAY loading )
AID0427    (&00) (&01) wurde im gelad. Teil des Source-Mod. nicht gefunden ( OVERLAY )
```

Bedeutung
Name der Prozedur oder des Symbols konnte im geladenen Teil des mit Overlay-Technik gebundenen Programms nicht gefunden werden. Der nicht geladene Teil konnte nicht ueberprueft werden.
(&00) PROZEDUR oder SYMBOL
(&01) Name der Prozedur oder des Symbols

```
AID0428   Warning: LSD information corrupted; failure possible
AID0428   Warnung: Fehler in LSD-Information; fehlerhaftes Verhalten moeglich
```

Bedeutung
Die LSD-Information (Symbolinformation) ist fehlerhaft, AID arbeitet eventuell nicht richtig, es kann aber symbolisch getestet werden

```
AID0429   No message with this number
AID0429   Meldungsnummer nicht belegt
```

Bedeutung
Diese Meldungsnummer ist nicht belegt

```
AID0430   Name of dump file too long
AID0430   Name der Dump-Datei zu lang
```

Bedeutung
Die angegebene Dump-Datei konnte nicht geoeffnet werden, da AID nur Dateinamen verarbeiten kann, die nicht laenger als 54 Zeichen sind.

Maßnahme
Dump-Datei mit /MODIFY-FILE-ATTRIBUTES umbenennen und AID-Kommando entsprechend aendern.

```
AID0431   Requested information not within dump file
AID0431   Angeforderte Information befindet sich nicht in der Dump-Datei
```

Bedeutung
Gewuenschte Information kann nicht beschafft werden.

```
AID0432   LSD-extra-information for source-modul (&00) is invalid
AID0432   LSD-Zusatz-Information fuer Source-Modul (&00) ist fehlerhaft
```

Bedeutung
Die zusaetzliche LSD-Information fuer das Testen mit graphischer Oberflaeche auf Quellprogrammbasis ist falsch erzeugt oder ueberschrieben.

```
AID0433   LSD information corrupted; symbolic test not possible
AID0433   Fehler in der LSD-Information; symbolischer Test nicht moeglich
```

Bedeutung
Die LSD-Information (Symbolinformation) wurde falsch erzeugt oder ueberschrieben. Ein symbolischer Test ist nicht moeglich.

Maßnahme
Programm neu uebersetzen bzw. Fehlermeldung an die Systemdiagnose schicken.

```
AID0434   Offset operation only admitted for an operand yielding an address
AID0434   Offset-Operation ist nur bei Operanden, die eine Adr. liefern, erlaubt
```

Bedeutung
Eine Adressversatz kann nur auf solche Operanden angewandt werden, die eine Adresse liefern.

Maßnahme
Speicherreferenz mit Adressattribut vor Adressversatz einfuegen.

```
AID0435   Warning: no output given for specified operands (CMD: (&00))
AID0435   Warnung: Keine Ausgabe fuer angegebene Operanden (KDO: (&00))
```

Bedeutung
Warnung: das angegebene Kommando erzeugte keine Ausgabe, da z.B. Name nicht gefunden wurde oder nicht eindeutig war.

```
AID0436   AID cannot reference symbol (&00) due to incomplete LSD information
AID0436   Wegen unvollst. LSD-Info. kann AID das Symbol (&00) nicht referenzieren
```

Bedeutung
Das Symbol (&00) wurde zwar deklariert, kann aber von AID nicht angesprochen werden, da vom Uebersetzer keine vollstaendige LSD-Information (Symbolinformation) erzeugt wurde.

```
AID0437   CSECT/ENTRY has length 0
AID0437   CSECT/ENTRY hat die Laenge 0
```

Bedeutung
CSECT/ENTRY hat die Laenge Null; sie kann nicht ausgegeben werden.

```
AID0438   String too long
AID0438   String zu lang
```

Bedeutung
Eingegebener String ueberschreitet die maximale Laenge.

Maßnahme
Gekuerzten String eingeben.

```
AID0439   Name too long
AID0439   Name zu lang
```

Bedeutung
Eingegebener Name ueberschreitet die maximale Laenge.

Maßnahme
Korrekten Namen eingeben.

```
AID0440   Source info file for %TRACE doesn't match with loaded object (CMD: (&00))
AID0440   Source-Info-Datei passt nicht zum geladenen Objekt (KDO: (&00))
```

Bedeutung
Die aufbereitete Textdatei (SIF) fuer %TRACE wurde nicht beim gleichen Uebersetzungsvorgang erzeugt wie das geladene Objekt.

Maßnahme
Source neu uebersetzen, binden und Textdatei fuer %TRACE neu erzeugen. Sonst erfolgen die %TRACE- Ausgaben im LSD-definierten Format.

```
AID0441   Wrong continuation in %TRACE command
AID0441   Falscher Fortsetzungsparameter in %TRACE - Kommando
```

Bedeutung
Als Fortsetzungsparameter fuer das %TRACE - Kommando sind nur 'R' und 'S' erlaubt

```
AID0442   Keyword does not yet exist in this OS version
AID0442   Das Schluesselwort gibt es in dieser Betriebssystemversion noch nicht
```

Bedeutung
Das angegebene Schluesselwort wurde erst in einer neueren Version des Betriebssystems eingefuehrt; in der BS-Version des Testobjekts ist es undefiniert.

```
AID0443   Keyword no longer exists in this OS version
AID0443   Das Schluesselwort gibt es in dieser Betriebssystemversion nicht mehr
```

Bedeutung
Das angegebene Schluesselwort existiert zwar in einer frueheren Betriebssystem-Version, in der BS-Version des Testobjekts ist es jedoch undefiniert.

```
AID0444   Outfile could not be opened
AID0444   Ausgabe-Datei kann nicht geoeffnet werden
```

Bedeutung
Die angegebene Datei konnte nicht als Ausgabe- Datei geoeffnet werden.

Maßnahme
Pruefen ob Datei eventuell schon offen ist.

```
AID0445   Source modul (&00) has no LSD for source based debugging.
AID0445   Source-Modul (&00) hat kein LSD fuer den Test auf Quellprogrammbasis.
```

Bedeutung
Der Test mit graphischer Benutzeroberflaeche auf Quellprogrammbasis ist nicht moeglich, da das saetzliche LSD entweder fehlt oder als fehlerhaft erkannt wurde.

```
AID0446   Variable (&00) has neither implicit nor explicit base
AID0446   Variable (&00) besitzt weder eine implizite noch explizite Basis
```

Bedeutung
Die Variable (&00) wurde ohne jeglichen Based-Pointer deklariert. Ein Bezug auf diese Variable ist mit AID nur dann moeglich, wenn ein expliziter Based-Pointer gegeben ist.

Maßnahme
Bitte expliziten Based-Pointer angeben (z.B. expl_ptr -> Variable).

```
AID0447    No libraries existing to be released
AID0447    Es gibt keine zugewiesenen Bibliotheken
```

Bedeutung
Den spezifizierten Linknamen (%SYMLIB (E/D)=...) oder allen Linknamen (%SYMLIB) ist zur Zeit keine vom Benutzer angeforderte Bibliothek zugewiesen.

```
AID0448    Pointer value exceeds BIT pointer boundary
AID0448    Wert des Zeigers ist unzulaessig fuer BIT-Zeiger
```

Bedeutung
Sendefeld vom Typ 'pointer', Empfangsfeld vom Typ 'bit-pointer'; ausserdem High-Value-Byte des Sendefeldes ungleich X'00'.

Maßnahme
Empfangsfeld explizit mit %SET X'00' into ... %XL1 besetzen.

```
AID0449    OFFSET ptr value exceeds BIT OFFSET ptr boundary; nothing changed
AID0449    Wert des OFFSET-Zeigers ist unzulaessig fuer BIT-OFFSET-Zeiger
```

Bedeutung
Sendefeld vom Typ 'offset-pointer', Empfangsfeld vom Typ 'bit-offset-pointer'; ausserdem High-Value-Byte des Sendefeldes ungleich X'00'.

Maßnahme
Empfangsfeld explizit mit %SET X'00' into ... %XL1 besetzen.

```
AID0450    Warning: BIT offset ignored
AID0450    Warnung: BIT-Versatz wird ignoriert
```

Bedeutung
Das High-Value-Byte des Sendefeldes ist ungleich X'00'
Sendefeld:bit-pointer
Zielfeld:pointer
oder
Sendefeld:bit-offset-pointer
Zielfeld:offset-pointer
und High-Value-Byte (entspricht Bit-Offset) des Sendefeldes ungleich X'00'.

```
AID0451    Length of (&00) could not be determined by AID
AID0451    AID kann die Laenge von (&00) nicht bestimmen
```

Bedeutung
Die Laenge des variablen Strings (&00) kann von AID nicht bestimmt werden, weil entsprechende Information vom Uebersetzer in der LSD-Information nicht zu erzeugen ist (z.B. weil abhaengig von einem Ausdruck).

```
AID0452  Boundary of dimension (&00) of array (&01) couldn't be determined by AID
AID0452  Grenze der Dimension (&00) des Feldes (&01) kann von AID nicht best. werden
```

Bedeutung
Die Grenzen des variablen Feldes koennen von AID nicht bestimmt werden, weil entsprechende Infor mation vom Uebersetzer in der LSD-Information nicht zu erzeugen ist (z.B. weil abhaengig von einem Ausdruck) bzw. weil die Feldgrenzen von nicht initialisierbaren Variablen abhaengen.
(&00) Dimension
(&01) Name des Feldes

Maßnahme
Wertzuweisung an nicht initialisierte Variable.

```
AID0453  Type of symbol (&00) not described in LSD information
AID0453  Typ des Symbols (&00) ist in der LSD-Information nicht beschrieben
```

Bedeutung
Das Symbol (&00) kann von AID nicht ausgegeben werden, weil die LSD-Information (Symbolinformation) keine Angabe ueber den Typ des Symbols enthaelt.

```
AID0454  Range parameter error: upper bound lower than lower bound
AID0454  Unzulaessige Bereichsangabe: Obergrenze kleiner als Untergrenze
```

Bedeutung
Beim %CONTROL- und %TRACE-Kommando muessen die Bereichsgrenzen in aufsteigender Ordnung angegeben werden.

Maßnahme
Bitte Bereich mit richtiger Ordnungsrelation angeben.

```
AID0455  Unknown CPU-Type or operating system version
AID0455  CPU-Typ oder Betriebssystem-Version unbekannt
```

Bedeutung
AIDSYS liefert unbekannte oder falsche Informationen ueber Hardwaretyp oder Betriebssystemversion. Dadurch koennen bei Low-Level-%TRACE Befehle oder SVC's falsch protokolliert werden.

Maßnahme
Systemverwalter verstaendigen.

```
AID0456  No message with this number
AID0456  Meldungsnummer nicht belegt
```

Bedeutung
Diese Meldungsnummer ist nicht belegt

```
AID0457   Surplus bits of source have been ignored
AID0457   Ueberfluessige Bits des Quelldatums ignoriert
```

Bedeutung
Sendefeld groesser als Empfangsfeld. Ueberzaehlige Bits im Empfangsfeld ignoriert.

```
AID0458   Test point not set; entry couldn't be confirmed within LSD Information
AID0458   Testpunkt nicht gesetzt, da Entry-Name nicht in LSD beschrieben
```

Bedeutung
Bevor AID einen Testpunkt auf einen Entrynamen setzen kann, muss die Adresse der Einsprungstelle mit Hilfe der LSD-Information als Entry verifiziert werden. Moegliche Fehlerursachen:

1) Entryvariable enthaelt eine ungueltige Adresse (nicht initialisiert ueberschrieben)
2) fuer die Einsprungstelle existiert keine LSD- Information
3) Einsprungstelle kann in der LSD-Information nicht als Entry verifiziert werden

Maßnahme

1) Variable initialisieren bzw. erst nach Zuweisung ansprechen
2) LSD-Information nachladen

```
AID0459   %SET may only be applied to components of complex number variables
AID0459   Zuweisung von komplexen Zahlen nur komponentenweise moeglich
```

Bedeutung
Variablen vom Typ COMPLEX koennen nu komponentenweise (Realteil oder Imaginaerteil) veraendert werden.

Maßnahme
Bitte fuehren Sie die Veraenderung komponentenweise durch.

```
AID0460   Actual value of variable (&00) doesn't match with predefined TRUE / FALSE
AID0460   Aktueller Wert der Variablen (&00) passt nicht zum vordef. TRUE / FALSE
```

Bedeutung
Warnung: Die logische Variable (&00) hat aktuell einen Wert, der nicht mit den vom Uebersetzer festgelegten Standardwerten fuer TRUE/FALSE uebereinstimmt. Gemaess Uebersetzerkonventionen wird immer ein Wert TRUE/FALSE bestimmt.

```
AID0461   Too many statement types
AID0461   Zu viele Anweisungstypen angegeben
```

Bedeutung
Der Programmadresse zu dieser %TRACE-Ausgabezeile sind mehr Anweisungs- bzw. Befehlstypen zugeordnet als ausgegeben werden koennen. Es werden so viele Anweisungs- bzw. Befehlstypen wie moeglich ausgegeben.

```
AID0462   %JUMP to given target (&00) not allowed
AID0462   %JUMP auf (&00) nicht erlaubt
```

Bedeutung
Die Adresse (&00) ist als Operand 'fortsetzung' des %JUMP-Kommandos nicht zugelassen.

Maßnahme
Andere Adresse angeben.

```
AID0463   Given target (&00) is not within the actual valid program/procedure
AID0463   Fortsetzungsadr. (&00) liegt nicht innerhalb d. guelt. Programms/Prozedur
```

Bedeutung
Mit dem %JUMP-Kommando kann nur innerhalb der gerade aktiven Prozedur bzw. des gerade aktiven Programms verzweigt werden.
(&00) als Operand 'fortsetzung' angegebene Adresse

Maßnahme
Andere Adresse angeben.

```
AID0464   Given target (&00) of %JUMP command doesn't represent a program label
AID0464   Fortsetzung (&00) im %JUMP-Kommando ist kein Anweisungsname
```

Bedeutung
Im %JUMP-Kommando koennen als Operand 'fortsetzung' nur Symbole angegeben werden, die eine Programmadresse darstellen; Datenadressen sind nicht zugelassen.
(&00) als Operand 'fortsetzung' angegebene Adresse

Maßnahme
Andere Adresse angeben.

```
AID0465   Offset / length exceeds (&00) boundaries
AID0465   Versatz / Laenge ueberschreitet (&00) - Grenzen
```

Bedeutung
Der angegebene Wert des Adressversatzes oder die explizite/implizite Laenge oder die Kombination von beiden ueberschreitet die Bereichsgrenzen der Speicherreferenz (&00).

Maßnahme
Entweder Wert des Adressversatzes oder Laenge entsprechend korrigieren, oder mit %@(...)-> auf Maschinencode-Ebene wechseln.

```
AID0466   Back tracking information not found or inconsistent
AID0466   Rueckverfolgungs-Information nicht gefunden oder fehlerhaft
```

Bedeutung
Der Modul AIDIT0, der die Rueckverfolgungs- Information enthaelt,
- ist im gebundenen Programm nicht vorhanden,
- ist nur zum Teil vorhanden,
- enthaelt eine alte, inkonsistente Version oder

- ist unmittelbar nach dem Laden des Programms noch nicht vorhanden.
- enthaelt keinen Eintrag fuer einen innerhalb einer Csect der aktiven Aufrufhierarchie spezifizierten Sprach-Indikator.

```
AID0467   Start/end address exceeds CSECT/keyword/symbol boundaries
AID0467   Start-/End-Adresse ueberschreitet CSECT-/Schluesselwort/Symbol-Grenzen
```

Bedeutung
Start- oder Endadresse liegt ausserhalb des durch CSECT/Schluesselwort/Symbol beschriebenen Bereichs.

Maßnahme
Kommando mit korrigierten Eingabewerten wiederholen.

```
AID0468   Label not allowed for range description
AID0468   Anweisungsname als Bereichsangabe nicht zulaessig
```

Bedeutung
Anweisungsnamen sind nicht fuer die Beschreibung eines %TRACE- oder %CONTROL-Bereiches zugelassen.

```
AID0469   Illegal jumping off place
AID0469   %JUMP-Kdo an dieser Unterbrechungsstelle nicht zugelassen
```

Bedeutung
An der aktuellen Unterbrechungsstelle ist eine Verzweigung mit Hilfe des %JUMP-Kommandos nicht moeglich, weil entweder eine notwendige Initialisierung des Programmes noch nicht durchlaufen wurde oder das Programm aktuell im Laufzeitsystem unterbrochen ist oder man sich nicht in einer Prozedur befindet, fuer die LSD vorhanden ist.

Maßnahme
Programm bis auf gueltige Absprungstelle weiter- laufen lassen.

```
AID0470   Arithmetic overflow when (&00) is calculated
AID0470   Arithmetischer Ueberlauf bei der Berechnung von (&00)
```

Bedeutung
Benutzerfehler: arithmetischer Ueberlauf bei Berechnung von (&00) aufgetreten.
(&00) Adresse, Laenge bzw. Subscript

Maßnahme
Kommando mit korrigierten Eingabewerten wiederholen.

```
AID0471   HIGH LEVEL trace / control not allowed in program (&00)
AID0471   High-Level-Trace/-Control in Programmeinheit (&00) nicht erlaubt
```

Bedeutung
In dieser Programmeinheit werden %TRACE- bzw. %CONTROL-Kommandos mit symbolischem ’kriterium’ nicht unterstuetzt.

Maßnahme
Kommando mit maschinennahem 'kriterium' verwenden.

```
AID0472   LSD information is incomplete or wrong for CSECT "(&00)"
AID0472   LSD-Information fuer die CSECT "(&00)" ist unvollstaendig oder fehlerhaft
```

Bedeutung
Die LSD-Information (Symbolinformation) ist un-vollstaendig oder fehlerhaft fuer die CSECT (&00).

Maßnahme
Programm neu binden bzw. neu uebersetzen.

```
AID0473   Condition not supported (CMD: (&00))
AID0473   Bedingung wird nicht unterstuetzt (KDO: (&00))
```

Bedeutung
Bedingung (&00) wird nicht von AID unterstuetzt. Die logischen Operanden muessen Vergleiche sein.

```
AID0474   No message with this number
AID0474   Meldungsnummer nicht belegt
```

Bedeutung
Diese Meldungsnummer ist nicht belegt

```
AID0475   Subcommand label already exists
AID0475   Subkommando-Name existiert bereits
```

Bedeutung
Der eingegebene Subkommando-Name existiert bereits.

Maßnahme
Anderen Namen fuer Subkommando verwenden.

```
AID0476   Too many subcommand labels defined
AID0476   Zu viele Subkommando-Namen definiert
```

Bedeutung
Es wurden bereits zu viele Namen fuer Subkommandos definiert.

Maßnahme
Nicht mehr benoetigte Subkommandos loeschen.

```
AID0477   Subcommand label does not exist: (&00)
AID0477   Der Subkommando-Name existiert nicht: (&00)
```

Bedeutung
Der Subkommando-Name (&00) wurde nicht definiert oder bereits geloescht.

Maßnahme
Richtigen Subkommando-Namen eingeben.

```
AID0478   No message with this number
AID0478   Meldungsnummer nicht belegt
```

Bedeutung
Diese Meldungsnummer ist nicht belegt

```
AID0479   Labels, source-references, template_instances must not be subscripted
AID0479   Anweisungsnamen, Source-Referenzen, Template-Instances ohne Subskript
```

Bedeutung
Syntax-Fehler

Maßnahme
Syntax korrigieren

```
AID0480   Address selection, offset, type modification not allowed for constants
AID0480   Adress-Selektion,Versatz oder Typ-Modifikation auf Konst. nicht anwendb.
```

Bedeutung
Der Adress-Selektor, der Offset-Operator und die Typ-Modifikation duerfen auf Konstanten nicht angewandt werden

```
AID0481   No LSD information provided for address to be located
AID0481   Zu der zu lokalisierenden Adresse liegt keine LSD-Information vor
```

Bedeutung
Die zu lokalisierende Adresse ist nicht im LSD (Symbolinformation) beschrieben.

Maßnahme
LSD aus Bibliothek nachladen oder Modul neu mit LSD-Generierung uebersetzen.

```
AID0482   Address cannot be located in LSD information
AID0482   Adresse kann nicht in der LSD-Information lokalisiert werden
```

Bedeutung
Adresse ist keiner Source-Referenz zuzuordnen (z.B. Adressen des Prozedur-Prologs).

```
AID0483   Reference to undefined subcommand label
AID0483   Bezug auf einen undefinierten Subkommando-Namen
```

Bedeutung
Eine Bezugnahme auf den Namen der aktuellen Subkommandofolge durch %. ist nur innerhalb des gerade aktiven Subkommandos erlaubt.

Maßnahme
Angabe des Namens der Subkommandofolge durch %.subkdoname.

```
AID0484   Subcommand label too long
AID0484   Subkommando-Name zu lang
```

Bedeutung
Die Laenge eines Subkommando-Namens darf nicht groesser als 30 sein.

Maßnahme
Laenge des Subkommando-Namens auf 30 begrenzen.

```
AID0485   "*" operator is only allowed for type pointers
AID0485   "*"-Operator ist nur auf Typ-Zeiger anwendbar
```

Bedeutung
Dem '*'-Operator muss eine 'type-pointer' Variable folgen.

```
AID0486   Type modification of symbol (&00) not allowed
AID0486   Der Typ des Symbols (&00) kann nicht modifiziert werden
```

Bedeutung
Die Adresse des Symbols (&00) ist nicht Byte- ausgerichtet bzw. das Symbol hat eine Bitlaenge.

```
AID0487   Test point is not a CLASS6 address
AID0487   Testpunkt liegt nicht im Klasse-6-Speicher
```

Bedeutung
Testpunkt liegt nicht im Klasse-6-Speicher

```
AID0488   OVERLAY not loaded
AID0488   OVERLAY-Segment nicht geladen
```

Bedeutung
Segment, in das ein Testpunkt gesetzt bzw. aus dem ein Testpunkt geloescht werden soll, ist nicht geladen.

Maßnahme
Testpunkt zu einem Zeitpunkt setzen oder loeschen, zu dem das Segment geladen ist.

```
AID0489   Address in %ON %WRITE command invalid
AID0489   Die Adresse im %ON %WRITE-Kommando ist ungueltig
```

Bedeutung
Die im %ON %WRITE-Kommando angegebene Adresse ist ungueltig.

Maßnahme
Gueltige Adresse angeben.

```
AID0490   %ON %WRITE is not allowed when LOW LEVEL %CONTROL/%TRACE is active
AID0490   %ON %WRITE nicht erlaubt, falls ein LOW-LEVEL-%TRACE/%CONTROL aktiv ist
```

Bedeutung
Es ist nicht erlaubt, ein %ON %WRITE-Kommando einzugeben, solange ein %TRACE- oder ein %CONTROLn-Kommando mit maschinennahem 'kriterium' (%INSTR, %B, %BAL) aktiv ist.

Maßnahme
Entfernen des %TRACE durch Eingabe des Kommandos %TRACE 1 %INSTR; entfernen des %CONTROLn durch Eingabe des Kommandos %REMOVE %Cn.

```
AID0491   LOW LEVEL %CONTROL/%TRACE is not allowed when %ON %WRITE is active
AID0491   LOW-LEVEL-%CONTROL/%TRACE nicht erlaubt, falls %ON %WRITE aktiv ist
```

Bedeutung
Es ist nicht erlaubt, ein %TRACE- oder %CONTROLn- Kommando mit maschinennahem 'kriterium' (%INSTR, %B, %BAL) einzugeben, solange ein %ON %WRITE- Kommando aktiv ist.

Maßnahme
Entfernen des %ON %WRITE-Kommandos durch %REMOVE %WRITE.

```
AID0492   %STOP was sent to fork task ((&00)).
AID0492   %STOP wurde an die Fork-Task mit (&00) geschickt.
```

Bedeutung
Es wurde eine Contingency-Routine fuer die angegeben Task eingehaengt, die bei der naechsten Aktivierung der Task den Debug-Modus veranlasst.

```
AID0493   %ON %WRITE command is only allowed in status TU
AID0493   Das %ON %WRITE-Kommando ist nur im Status TU zulaessig
```

Bedeutung
Das Kommando %ON %WRITE ist nur im nicht-privilegierten Zustand erlaubt.

Maßnahme
Uebergang in den nicht-privilegierten Zustand.

```
AID0494   Test point must not be set into CLASS6 memory pools
AID0494   In Klasse-6-Memorypools darf kein Testpunkt gesetzt werden
```

Bedeutung
Das Setzen von Testpunkten in Klasse-6-Memory- Pools ist nicht erlaubt.

```
AID0495   No message with this number
AID0495   Meldungsnummer nicht belegt
```

Bedeutung
Diese Meldungsnummer ist nicht belegt

```
AID0496   Warning: previously defined event %WRITE is replaced
AID0496   Warnung: vorher definiertes Ereignis %WRITE wurde ueberschrieben
```

Bedeutung
Warnung: Da jeweils nur ein %WRITE-Ereignis angemeldet sein kann, wird bei der Eingabe eines weiteren %ON %WRITE-Kommandos das vorher eingegebene ueberschrieben.

```
AID0497   Length of area in %ON %WRITE command exceeds 65535 bytes
AID0497   Der Bereichs im %ON %WRITE-Kommando ist laenger als 65535 Bytes
```

Bedeutung
Mit dem %ON %WRITE-Kommando kann maximal ein Bereich von 65535 Bytes ueberwacht werden.

Maßnahme
Kleinere Laenge angeben.

```
AID0498   Hardware does not support ALET/SPID qualification
AID0498   Die Hardware unterstuetzt keine ALET/SPID-Qualifikation
```

Bedeutung
ALET/SPID-Qualifikationen bei Operanden werden nur auf ESA-Anlagen unterstuetzt.

Maßnahme
Eingabe ohne ALET/SPID-Qualifikation

```
AID0499   Value of ALET is unknown
AID0499   Unbekannter ALET-Wert
```

Bedeutung
Der Wert des ALET ist unbekannt.

Maßnahme
Richtigen Wert angeben.

```
AID0500   Number of DATA SPACES changed during execution of command
AID0500   Die Anzahl der DATA SPACES hat sich waehrend der Kdo-Ausfuehrung geaend.
```

Bedeutung
Die Anzahl der Datenraeume hat sich waehrend der Ausfuehrung des Kommandos geaendert.

Maßnahme
Kommando spaeter noch einmal eingeben.

```
AID0501   CTX qualification not allowed in this OS version
AID0501   CTX-Qualifikation in dieser Betriebssystem-Version nicht zugelassen
```

Bedeutung
AID erlaubt keine CTX-Qualifikation in dieser BS2000 Version

```
AID0502   No DATA SPACES dumped
AID0502   Dump-Datei enthaelt keine DATA SPACES
```

Bedeutung
In der angegebenen Dump-Datei gibt es keine Dump-Bereiche aus Datenraeumen.

```
AID0503   DATA SPACE with specified SPID not found in dump file
AID0503   DATA SPACE mit angegebener SPID befindet sich nicht in der Dump-Datei
```

Bedeutung
Der Datenraum mit der angegebenen SPID befindet sich nicht in der Dump-Datei.

```
AID0504   DATA SPACE with specified ALET not found in dump file
AID0504   DATA SPACE mit angegebener ALET befindet sich nicht in der Dumpdatei
```

Bedeutung
Der Datenraum mit dem angegebenen ALET befindet sich nicht in der Dump-Datei.

```
AID0505    Hardware does not support ESA
AID0505    Hardware unterstuetzt ESA nicht
```

Bedeutung
Auf dieser Anlage wird ESA nicht unterstuetzt.

```
AID0506    No DATA SPACES defined
AID0506    Keine DATA-SPACEs definiert
```

Bedeutung
Zu diesem Zeitpunkt sind keine Datenraeume definiert.

```
AID0507    DATA SPACE not allowed for event
AID0507    Bei Ereignissen sind DATA SPACES unzulaessig
```

Bedeutung
Die Angabe von Data Spaces fuer Events ist nicht erlaubt.

```
AID0508    No symbolic library opened
AID0508    Keine Symbolbibliothek geoeffnet
```

Bedeutung
Es gibt keine offene Symbolbibliothek

```
AID0509    No qualification defined
AID0509    Keine Vorqualifikation definiert
```

Bedeutung
Es existiert keine Vorqualifikation

```
AID0510    No %FIND command entered
AID0510    Es wurde kein %FIND-Kommando eingegeben
```

Bedeutung
Es wurde noch kein %FIND-Kommando eingegeben

```
AID0511    No outfile assigned or opened
AID0511    Keine zugewiesene oder geoeffnete Ausgabe-Datei vorhanden
```

Bedeutung
Es gibt keine mit dem %OUTFILE-Kommando zugezugewiesene oder keine geoeffnete Ausgabedatei.

```
AID0512    No AID command entered in this task
AID0512    Es wurde noch kein AID-Kommando in diesem Task eingegeben
```

Bedeutung
Es wurde noch kein echtes %AID-Kommando eingegeben.

```
AID0513    No subcommand label defined
AID0513    kein Subkommando-Name definiert
```

Bedeutung
Es wurde noch kein Subkommando-Name definiert

```
AID0514    Connection name (OWN) already busy.
AID0514    Verbindungsname (OWN) belegt.
```

Bedeutung
Der angegebene Name des eigenen Verbindungsendpunktes ist bereits belegt.

```
AID0515    Connection name (APPL) or route unknown.
AID0515    Verbindungsname (APPL) oder Route zum Partner unbekannt.
```

Bedeutung
Der angegebene Name des fernen Verbindungsendpunktes oder der Weg dorthin ist unbekannt.

```
AID0516    Connection name (APPL) busy or inactive.
AID0516    Verbindungsname (APPL) belegt oder inaktiv.
```

Bedeutung
Der angegebene Name des fernen Verbindungsend-punktes ist belegt oder nicht aktiv.

```
AID0517    User interface internal error (&00) in module (&01)
AID0517    Fehler in der Benutzeroberflaeche (&00) im Modul (&01)
```

Bedeutung
Es ist ein Fehler in der Benutzeroberflaeche oder in der Kommunikation zwischen AID und der Benutzeroberflaeche aufgetreten.
(&00) Internes Fehlerkennzeichen von AID.
(&01) Modulnummer des Moduls, das denFehler entdeckt hat.

Maßnahme
Systemverwalter verstaendigen.

```
AID0518    Connection ID wrong.
AID0518    Verbindungs-ID falsch.
```

Bedeutung
Die Verbindungs-ID ist falsch.

Maßnahme
Richtige Verbindungs-ID eingeben.

```
AID0519    Warning: no output given
AID0519    Warnung: keine Ausgabe
```

Bedeutung
Es gibt keine Informationen fuer eine Ausgabe.

```
AID0520    User interface not connected.
AID0520    Verbindung zur Benutzeroberflaeche nicht aufgebaut.
```

Bedeutung
Wegen eines aufgetretenen Fehlers wurde keine Verbindung zur Benutzeroberflaeche aufgebaut.

```
AID0521   User interface already connected; input ignored.
AID0521   Benutzeroberflaeche ist bereits angeschlossen; Eingabe wird ignoriert.
```

Bedeutung
Es ist jeweils nur eine Verbindung zu einer Benutzeroberflaeche zugelassen. Ein erneuter Ver-such eines Verbindungsaufbaus wird abgelehnt.

```
AID0522   Check dialog is not allowed, when user interface is connected.
AID0522   Bei angeschlossener Benutzeroberflaeche ist ein Check Dialog verboten.
```

Bedeutung
Ist eine Benutzeroberflaeche angeschlossen, so ist ein Check Dialog nicht zugelassen.

```
AID0523   Warning: Check dialog aborted.
AID0523   Warnung: Check Dialog abgebrochen.
```

Bedeutung
Wird dei aktivem Check Dialog eine Benutzeroberflaeche angeschlossen, fuehrt dies zu einer automatischen Beendigung des Check Dialogs.

```
AID0524   Partial array operand not allowed for function/selector
AID0524   Partielles Feld als Selektor-/Funktions-Operand verboten.
```

Bedeutung
Ein Teilfeld darf nicht als Operand einer Funktion oder Selektion wie %@ oder %L auftreten.

```
AID0525   Register specification is not allowed on /390 objects.
AID0525   Registerangabe ist auf /390-Objekten nicht erlaubt.
```

Bedeutung
Bei /390-Objekten ist diese Registerangabe unzulaessig.

```
AID0526   Hardware does not support %(&00).
AID0526   Die Hardware unterstuetzt %(&00) nicht.
```

Bedeutung
Die Angabe des Schluesselwortes (&00) ist bei dieser Hardware unzulaessig.

```
AID0527   Program mode RM does not support keyword.
AID0527   Der Programmodus RM unterstuetzt Schluesselwort nicht.
```

Bedeutung
Waehrend des Programmodus RM wird das Schluesselwort nicht unterstuetzt.

```
AID0528   Warning: /390-trapcode for absolute-address
AID0528   Warnung: auf eine Absolut-Adresse wurde ein /390-Trapcode gesetzt
```

Bedeutung
auf eine nicht lokalisierbare Adresse (Absolut) wurde ein /390-Trap gesetzt;aufRisc-Code ist der Testpunkt wieder zu entfernen!

```
AID0529    Symbol (&00) ambiguous because of using directive .
AID0529    Symbol (&00) ist aufgrund einer USING Direktive mehrdeutig .
```

Bedeutung
Aufgrund einer USING Direktive ist das Symbol (&00) nicht eindeutig zu bestimmen .

```
AID0530    Alias name (&00) is undeclared.
AID0530    Alias-Name (&00) ist nicht deklariert.
```

Bedeutung
Der angegebenen Alias-Name ist nicht deklariert oder bereits deaktiviert worden

```
AID0531    Alias name (&00) is ambiguous.
AID0531    Der Alias-Name (&00) ist mehrdeutig.
```

Bedeutung
Der angegebene Alias-Name ist bereits vergeben.

```
AID0532    No message with this number
AID0532    Meldungsnummer nicht belegt
```

Bedeutung
Diese Meldungsnummer ist nicht belegt

```
AID0533    Too many active alias declarations.
AID0533    Zu viele Alias-Deklarationen.
```

Bedeutung
Es gibt zu viele Alias-Deklarationen. Unnoetige Deklarationen bitte entfernen

```
AID0534    Pool overflow for alias declarations.
AID0534    Pool-Ueberlauf fuer Alias-Deklaration.
```

Bedeutung
Der Speicherbereich fuer die Aufnahme von Alias-Deklarationen ist voll. Bitte unnoetige De-
klarationen deaktivieren.

```
AID0535    Alias declaration contains an alias name (&00) specified previously.
AID0535    Die Alias-Deklaration enthaelt einen spezifizierten Alias-Namen (&00).
```

Bedeutung
Alias-Deklarationen duerfen keine bereits vorher deklarierten Alias-Namen enthalten. Ersetzen Sie den vorher deklarierten Alias-Namen durch seinen Wert oder verwenden Sie einen anderen Namen.

```
AID0536    Warning! Alias name (&00) is an non-percent aid-keyword.
AID0536    Warnung! Der Alias-Name (&00) ist auch ein %-freies AID-Keyword.
```

Bedeutung
Der Alias-Name ist auch ein %-freies AID- Schluesselwort. In bestimmten Kommandos kann die Ersetzung des Alias-Namens zu einem Syntax- Fehler fuehren.

```
AID0537   Alias name is too long.
AID0537   Alias-Name ist zu lang.
```

Bedeutung
Die Laenge von Alias-Namen ist auf 32 Bytes beschraenkt.

```
AID0538   There exist no alias declarations.
AID0538   Es gibt keine Alias-Deklarationen.
```

Bedeutung
Keine aktive Alias-Deklaration vorhanden.

```
AID0539   SPL4-Stack-globals not supported.
AID0539   SPL4-Stack-Globals werden nicht unterstuetzt.
```

Bedeutung
SPL4-Stack-Globals werden nicht unterstuetzt. SPL4-Stack-Globals werden wegen einer Laufzeitsystem abhaengigen Adressierung nicht unterstuetzt

```
AID0540   Constant expressions in template arguments not supported.
AID0540   Konstante Ausdruecke in Template-Argumenten nicht unterstuetzt.
```

Bedeutung
Momentan wird die Auswertung von konstanten Ausdruecken in Template-Argumenten nicht unterstuetzt. Der Ausdruck ist durch das Ergebnis zu ersetzen.

```
AID0541   Simple type specification too long
AID0541   Specification eines einfachen Typen zu lang.
```

Bedeutung
Specification eines einfachen Typen zu lang. Spezifikation eines einfachen Typs ist zu lang. Ueberfluessige Typ-Angaben sind zu beseitigen.

```
AID0542   Alias names not allowed in %QUALIFY command
AID0542   Aliasnamen im %QUALIFY - Kommando nicht erlaubt
```

Bedeutung
Die Angabe eines Aliasnamens in einem %QUALIFY -Kommando ist nicht erlaubt. Mit %SHOW %ALIAS erfahren Sie, welche Aliasnamen im Moment vereinbart sind.

```
AID0543   A proc-qualification may not contain a structured name.
AID0543   Eine Proc-Qualifikation darf kein strukturierter Name sein.
```

Bedeutung
Struktierte Namen sind fuer eine Procedur-Qualifikation nicht zugelassen, wenn die generierte LSD-Version kleiner gleich 6 ist

```
AID0544    No message with this number
AID0544    Meldungsnummer nicht belegt
```

Bedeutung
Diese Meldungsnummer ist nicht belegt

```
AID0545    Pointer to member value invalid
AID0545    Ungueltiger Pointer-to-Member-Wert
```

Bedeutung
Der interne Wert des Pointer-to-Member laesst sich nicht in einen Pointernamen transformieren

```
AID0546    Right operand is no 'pointer to member' type
AID0546    Rechter Operand ist nicht vom Typ 'Pointer-to-Member'
```

Bedeutung
Rechter Operand ist nicht vom Typ 'Pointer-to-Member'.

```
AID0547    Left operand does not refer to a class object
AID0547    Linker Operand bezieht sich nicht auf ein Klassenobjekt.
```

Bedeutung
Linker Operand bezieht sich nicht auf ein Klassenobjekt.

```
AID0548    Operand refers to class types which are not compatible
AID0548    Operand bezieht sich auf unvertraegliche Klassentypen
```

Bedeutung
Linker und rechter Operand der Operatoren ->* oder
.* beziehen sich auf unvertraegliche Klassentypen.

```
AID0549    Class of pointer to member has ambiguous subobjects
AID0549    Pointer-to-Member-Klasse enthaelt mehrdeutige Subobjekte
```

Bedeutung
Pointer-to-Member-Klasse enthaelt mehrdeutige Subobjekte in der Klasse des linken Operanden.

```
AID0550    sizeof-/&-selector may not be applied to HLL/LL-operands.
AID0550    sizeof-/&-selektor nicht auf HLL/LL-Operanden ansetzen.
```

Bedeutung
sizeof-/-& Selektoren duerfen nur rein symbolische Ausdruecke als Argument enthalten, aber keine HLL/LL-Uebergaenge.

```
AID0551    sizeof-Operator may not be applied to functions.
AID0551    sizeof-Operator auf Funktion nicht anwendbar
```

Bedeutung
Gemass C/C++ darf sizeof auf Funktionen nicht angesetzt werden.

```
AID0552    syntax-error by alias-name/prequalifiction (CMD: (&00))
AID0552    Syntax-Fehler durch Alias-Name,Vorqualifikation(KDO: (&00))
```

Bedeutung
Die Auswertung des Alias-Namen bzw. der Vorqualifikation fuehrt zu einem syntaktisch fehlerhaftenKommando.

```
AID0553    Partial page(s) from address (&00) to (&01) not accessible
AID0553    Teilseite(n) von Adresse (&00) bis (&01) nicht zugreifbar
```

Bedeutung
Programmzugriff auf nicht angeforderte Teilseite einer 8K SPARC-Seite fuehrt auf SPARC HSI nicht immer zu einer Programmunterbrechung

```
AID0554    Change on register %g0 not allowed.
AID0554    Aenderungen im Register %g0 nicht erlaubt.
```

Bedeutung
%g0 darf nicht geaendert werden.

```
AID0555    Specified CCS-name not supported
AID0555    Der spezifizierte CCS-Name wird nicht unterstuetzt
```

Bedeutung
XHCS unterstuetzt den CCS-Namen nicht oder OSD-Version unterstuetzt UNICODE nicht

```
AID0556    Specified CCS-name not supported by AID
AID0556    Spezifizierter CCS-Name wird von AID nicht unterstuetzt
```

Bedeutung
AID unterstuetzt nur Einbyte EBCDIC-Codes

```
AID0557    No even number for UTF16 typ length specification
AID0557    Laengenangaben fuer typ UTF16 muessen geradzahlig sein
```

Bedeutung
Typ UTF16 erfordert geradzahlig Laenge

```
AID0558    UTFE type HEX value doesn't end on character boundary
AID0558    HEX wert eines UTFE Typs endet nicht an Zeichen Grenze
```

Bedeutung
Ein Operand mit Typ UTFE hat einen hexadezimalen Wert, der nicht an Zeichengrenze endet.

```
AID0559    value is not an UTFE-string
AID0559    Wert ist keine zulaessige UTFE-Zeichenkette
```

Bedeutung
XHCS hat eine unzulaessige UTFE-Codierung festgestellt Zeichenkette wird als HEX-String intepretiert

```
AID0560   CCS-name of output media unknown to actual XHCS subsystem
AID0560   CCS-Name eines Ausgabe Mediums ist XHCS unbekannt
```

Bedeutung
Es kann keine Konversion nach CCS der Ausgabe durchgefuehrt werden.

```
AID0561   CCS-name of output media not supported by AID
AID0561   Der CCS-Name eines Ausgabe Mediums wird von AID nicht unterstuetzt
```

Bedeutung
AID unterstuetzt als CCSN nur Einbyte EBCDIC
Codes bzw. UTFE

```
AID0562   Subsystem XHCS not available, no unicode support
AID0562   Subsystem XHCS nicht verfuegbar, keine UNICODE Konvertierungen
```

Bedeutung
Das Subsystem XHCS ist nicht im System. AID kann keine UNICODE Typen bearbeiten.

```
AID0563   Version of subsystem XHCS does not support UNICODE
AID0563   Keine UNICODE Unterstuetzung durch aktuelle XHCS Version
```

Bedeutung
Mit der im System befindlichen XHCS Version sind UNICODE Typen durch AID nicht bearbeitbar.

```
AID0564   Version of Subsystem SYSFILE cannot provide CCS-name
AID0564   Aktuelle SYSFILE Subsystem Version liefert keinen CCS-Namen
```

Bedeutung
Die im System befindliche Version des Subsystem SYSFILE ist zu niedrig, um den EBCDIC von SYSOUT/ SYSLST zu bestimmen. Die default EBCDIC Einstellung wird angenommen.

```
AID0565   LSD information for UNICODE types requires at least LSD Version 10
AID0565   LSD-Information mit UNICODE Datentypen erfordern mindestens Version 10
```

Bedeutung
Die LSD-Information (Symbolinformation) enthaelt Zeichen-Datentypen mit UNICODE Codierung. Hierfuer ist mindestens LSD Version 10 notwendig.

Maßnahme
Compiler Fehler, ungueltige LSD Version

```
AID0566   Variable boundary of (component of) (&00) is out of valid range.
AID0566   Variable Grenze von (Komponente. von) (&00) verletzt gueltigen Bereich.
```

Bedeutung
Die Variable Grenze ein Feldes liegt nicht im spezifizierten Gueltigkeitsbereich. (Bsp. OCCURS DEPENDING Feld in COBOL).

Maßnahme
AID nimmt Feld mit 0 Elementen an.

```
AID0567   No redefinition of a char literal by a different char type.
AID0567   Keine Redefinition eines Character Literals durch anderen CHAR-Typ.
```

Bedeutung
Die Ausgabetyp eines Character-Literals darf nicht durch eine Redefinition mit Hilfe eines anderen Charactertyps geaendert werden. Beispielsweise macht es wenig Sinn, fuer das Bitmuster von C'A' eine UTF16 Konvertierung zu erzwingen.

```
AID0568   Invalid type of argument of string conversion.
AID0568   Der Argumenttyp ist fuer eine Zeichenkonversion nicht erlaubt.
```

Bedeutung
Der Typ des Argument einer Zeichenkonversion muss ein Character Typ sein. Andernfalls vorher explizite Typmodifikation mit %C bzw. %UTF16 durchfuehren.

```
AID0569   Size of argement of string conversion exceeds 80 characters.
AID0569   Laenge des Arguments der Zeichenkonversion ueberschreitet 80 Zeichen.
```

Bedeutung
Die Laenge des Arguments der Zeichenkonversion ist auf 80 Zeichen beschraenkt. Andernfalls vorher mit Laengenmodifikation entsprechend kuerzen.

```
AID0570   String conversion with substitution by default characters performed.
AID0570   Zeicherkonversion wurde mit Ersatzzeichen durchgefuehrt.
```

Bedeutung
Bei der Zeicherkonversion wurden Zeichen der Quelle durch ein Ersatzzeichen ersetzt, da sie im Ziel CCS nicht darstellbar waren.

```
AID0571   Search critera in %FIND command too long.
AID0571   Zu langes Suchkriterium im %FIND.
```

Bedeutung
Das Suchkriterium im %FIND ist auf 80 Bytes bzw. 40 Bytes bei X-Literalen beschraenkt.

14 Anhang

14.1 In Kommandofolgen und Subkommandos unzulässige SDF- und ISP-Kommandos

Liste der BS2000-Kommandos, die nicht in Kommandofolgen und Subkommandos stehen dürfen.

Kommando	Kurzbeschreibung	Handbuch
ABORT	Prozedur abbrechen	[17]
ADD-SHARED-PROGRAMM	erklärt Bindemodule als gemeinsam benutzbar	[10]
BEGIN-PROCEDURE	Prozedurdateimerkmale festlegen	[8]
BREAK	Kommando-Modus anfordern	[17]
CATEGORY	Lastverteilung einer Anlage steuern	[9]
CANCEL-PROCEDURE	Prozedur(ablauf) abbrechen	[8]
CHANGE-ACCOUNTING-FILE	aktuelle Abrechnungsdatei schließen, neue Datei anlegen	[9]
CHANGE-CONSLOG	aktuelle Protokolldatei schließen, neue Datei anlegen	[18]
CHANGE-SERSLOG	aktuelle SERSLOG-Datei schließen, neue Datei anlegen	[10],[9]
DELON	ON-Kommando löschen	[17]
END	SPOOLIN-Datei schließen	[17]
ENDON	beendet ON-Anweisungsfolge	[17]
ENDP	beendet Prozedurdatei	[17]
END-PROCEDURE	beendet Prozedurdatei	[8]
EOF	Dateiende für SYSDTA kennzeichnen	[17]
ESCAPE	Prozedurablauf unterbrechen	[17]
EXIT-PROCEDURE	Prozedurablauf beenden und Rückkehr zur zuletzt verlassenen Prozedurdatei	[8]
HOLD-JOB	Auftrag (Job) anhalten	[9],[10]
HOLD-PROCEDURE	unterbricht Prozedurablauf und ermöglicht die Kommandoeingabe vom Datensichtgerät	[8]

Kommando	Kurzbeschreibung	Handbuch
HOLD-RSO	Wartezustand für Subsystem RSO	[10]
HOLD-SPOOL	Wartezustand für Subsystem SPOOL	[10]
HOLD-SPOOLOUT	Spooloutauftrag anhalten	[10]
INTR	"INTR-Ereignis" an Programm senden	[17]
LOAD-EXECUTABLE-PROGRAM	Programm laden	[8]
LOAD-PROGRAM	Programm laden	[8]
LOADAID	AID laden	[9]
LOGON	Auftrag (Job) einleiten	[17]
MARGIN	Zeilenlänge des Datensichtgerätes ändern	[17]
MODIFY-ACCOUNTING-PARAMETERS	legt Abrechnungssätze und Satzerweiterungen für die Accountingdatei fest	[9],[10]
MODIFY-CHANNEL-OPTIONS		
MODIFY-SYMBOLIC-PARAMETER		
MODIFY-TASK-CATEGORIES	begrenzt die Anzahl aktiver Tasks	[10]
ON	bedingte Ausführung einer Kommandofolge	[17]
PROCEDURE	Prozedurdateimerkmale festlegen	[17]
READ-CHANNEL		
RESTART	Programm bei Fixpunkt starten	[17]
RESTORE		
RESUME-JOB	Wartezustand für Anwenderauftrag aufheben	[10]
RESUME-PROCEDURE	unterbrochenen Prozedurablauf fortsetzen	[10]
RESUME-RSO	Wartezustand für Subsystem RSO aufheben	[10]
RESUME-SPOOL	Wartezustand für Subsystem SPOOL aufheben	[10]
RFD	Diskettengerät für wartende SPOOLIN-Aufträge zuweisen	[9]
RTI	Rückkehr zur unterbrochenen Prozedur	[17]
SAVEFILE		
SET-JOB-STEP	beendet SPIN-OFF	[8]
SET-SPACE-SATURATION-LEVEL	Festlegen der Speicher-Sättigungsstufen auf einen Pubset	[10]
SHARE	erklärt Bindemodule als gemeinsam benutzbar	[9]
SHOW-ACCOUNTING-STATUS	informiert über das Abrechnungssystem	[9],[10]
SHOW-FILE	Datei auf Bildschirm ausgeben	[17]

Kommando	Kurzbeschreibung	Handbuch
SHOW-RSO-STATUS	informiert über Zustand des Subsystems RSO	[10]
SHOW-SERSLOG	gibt Informationen über SERSLOG aus	[10]
SHOW-SPOOL-STATUS	informiert über Zustand des Subsystem SPOOL	[10]
SKIP	bedingt (Auftragsschalter) springen	[17]
SKIPJV	bedingt (Jobvariable) springen	[17]
SKIPUS	bedingt (Benutzerschalter) springen	[17]
SPMGT	Speicherplatz verwalten	[9]
START-ACCOUNTING	Abrechnungssystem einschalten	[9],[10]
START-DISKETTE-INPUT	Diskettenlaufwerk für SPOOLIN zuweisen	[10]
START-EXECUTABLE-PROGRAM	Programm laden und starten	[8]
START-PROGRAM	Programm laden und starten	[8]
START-RSO	Subsystem RSO laden und starten	[9]
START-SERSLOG	aktiviert SERSLOG	[9],[10]
START-SPOOL	SPOOL laden und initialisieren	[18]
STOP-ACCOUNTING	Abrechnungssystem ausschalten	[9],[10]
STOP-RSO	beendet Subsystem RSO	[10]
STOP-SERSLOG	SERSLOG beenden	[10]
STOP-SPOOL	beendet Subsystem SPOOL	[10]
WAIT	bedingte Wartezeit (Stapelauftrag) angeben	[17]
WHEN	Stapelauftrag bedingt (Benutzerschalter) anhalten	[17]
WRITE-CHANNEL		

Liste der ISP-Kommandos, die in AID-Kommandofolgen und in Subkommandos das geladene Programm abbrechen:

Kommando	Funktion	Handbuch
CALL	Call-Prozedur aufrufen	[17]
DO	DO-Prozedur aufrufen	[17]
EXECUTE	Modul laden und starten	[17]
LOAD	Modul laden	[17]
LOGOFF	Auftrag (Job) beenden	[17]

14.2 Letztmalig beschriebene Operanden

In dieser AID-Version werden die Operanden *AS ausgabetyp* des Kommandos %DISPLAY (nur maschinennahes Testen) und *steuerung* des Kommandos %ON (maschinennahes und symbolisches Testen) letztmalig beschrieben.

14.2.1 Operand "AS ausgabetyp"

```
%DISPLAY    {daten [AS ausgabetyp]}{,...}    [medium-u-menge][,...]
```

Im %DISPLAY-Kommando kann der Operand *AS ausgabetyp* angegeben werden. Er folgt auf den zugehörigen *daten*-Operanden, dessen Ausgabetyp modifiziert werden soll.
Ohne diesen Operanden können Sie den Ausgabetyp mit einer Typmodifikation ändern (siehe Abschnitt „Typmodifikation“ auf Seite 88).

ausgabetyp

Jedem Adressoperanden ist in AID ein Typ zugeordnet, der angibt, wie eine bestimmte (vom Typ abhängige) Anzahl von Bytes im Speicher zu interpretieren ist (Speichertyp) und wie ihr Wert ausgegeben werden soll (Ausgabetyp). Jedem Speichertyp ist ein Ausgabetyp zugeordnet (siehe Abschnitt „Allgemeine Speichertypen“ auf Seite 111 und Abschnitt „Speichertypen zur Interpretation von Maschinenbefehlen“ auf Seite 112). Durch explizite Angabe von *ausgabetyp* kann diese Zuordnung geändert werden.

ausgabetyp kann die folgenden Werte annehmen:

HEX	sedezimal
DEC	Dezimalzahl
CHAR	character
BIN	binär
DUMP	Dump (sedezimal und character)

14.2.2 Operand "steuerung" bei %ON

```
%ON        ereignis        [<subkdo>]        [steuerung]
```

steuerung wird als letzter Operand im %ON angegeben, also nach *ereignis* und *subkdo*.

steuerung

gibt an, ob *ereignis* nach dem n-ten Auftreten gelöscht werden soll und ob AID zu diesem Zeitpunkt die Eingabe neuer Kommandos erwartet. Wird der Operand *steuerung* nicht angegeben, setzt AID die Standardwerte 65 535 (für *n*) und K (*KEEP*) ein.

```
steuerung-OPERAND - - - - - - - - - - - - - - - - - - - - - - - - - - - -

                  ⎧ K ⎫
ONLY  n [         ⎨ C ⎬ ]
                  ⎩ S ⎭

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

n ist eine Zahl mit dem Wert $1 \leq n \leq 65\,535$.
Sie gibt an, beim wievielten Auftreten von *ereignis* die weiteren Vereinbarungen dieses *steuerung*-Operanden ausgeführt werden sollen.

K *ereignis* wird nicht gelöscht (KEEP).
Der Programmablauf wird unterbrochen, und AID erwartet die Eingabe von Kommandos.

S *ereignis* wird gelöscht (STOP).
Der Programmablauf wird unterbrochen, und AID erwartet die Eingabe von Kommandos.

C *ereignis* wird gelöscht (CONTINUE).
Keine Unterbrechung des Programms.

14.2.3 Binden mit TSOSLNK

Beim statischen Binden mit TSOSLNK müssen Sie stets darauf achten, die ESD-Sätze mitzuführen (Anweisung LINK-SYMBOLS).

Mit dem Operanden SYMTEST in der Anweisung PROGRAM können Sie steuern, wie die LSD-Sätze aus Bindemodulen behandelt werden sollen (siehe „TSOSLNK“ [13]). Hier werden nur die zwei Operandenwerte beschrieben, die die Verwendung der Namen von Daten und Anweisungen beim Testen unterstützen.

```
PROGRAM...,SYMTEST = { ALL }
                     { MAP }
```

ALL Die LSD-Sätze werden aus dem Bindemodul in die Ladeeinheit übernommen. Der Binder überprüft jedoch nicht, ob der verarbeitete Bindemodul wirklich LSD-Sätze enthält.

Damit sind beim Laden des Programms zwei Wege möglich:

1. Das Programm wird mit LSD-Sätzen geladen.
2. Das Programm wird ohne LSD-Sätze geladen. Steht der Bindemodul in einer PLAM-Bibliothek, kann diese mit %SYMLIB für AID angemeldet werden. Dann kann AID bei Bedarf die LSD-Sätze nachladen.

MAP Aus den ESD-Sätzen erzeugt der Binder eine Objekt-Strukturliste, die mit in die Ladeeinheit geschrieben wird. Mit dieser Information können Aufrufhierarchien rückverfolgt werden. LSD-Sätze werden nicht mitgebunden, auch wenn sie im Bindemodul vorhanden sind. AID kann dann aber, falls gewünscht, LSD-Sätze zum symbolischen Testen nachladen.

i Beim Binden von Großmodulen (TSOSLNK-Anweisung MODULE) gibt es keine Möglichkeit, LSD-Sätze in die Ladeeinheit miteinzubinden. Außerdem geht wichtige Information im ESD verloren. Wenn die Bindemodule mit den LSD-Sätzen in einer PLAM-Bibliothek stehen, die Sie mit %SYMLIB anmelden, so kann AID die LSD-Sätze nachladen, und symbolisches Testen ist mit der folgenden Einschränkung möglich: Liegt die Unterbrechungsstelle in einem Modul, zu dem es auch in der PLAM-Bibliothek keine LSD gibt, dann können Sie auch die Daten und Anweisungen aus Modulen mit LSD nicht symbolisch ansprechen.

Beispiel

```
/START-PROGRAM FROM-FILE=$TSOSLNK
 PROG BEISPIEL, FILENAM=BEISPIEL.FOR1, SYMTEST=ALL
 INCLUDE *
 RESOLVE, FOR1MODLIB
 END
```

Aus der temporären Bindemoduldatei wird das Programm BEISPIEL gebunden und mit den LSD-Sätzen in der Datei BEISPIEL.FOR1 abgelegt. Alle unbefriedigten Externverweise sollen per Autolink aus der Bibliothek FOR1MODLIB befriedigt werden.

14.3 Ereigniscodes

Folgende Tabelle ordnet den STXIT-Ereignisklassen die zugehörigen Unterbrechungsereignisse und die Ereigniscodes zu:

STXIT-Ereignisklasse	Unterbrechungsereignis	Ereigniscode
Programmfehler	unzulässiger SVC unzulässiger Operationscode Datenfehler Exponentenüberlauf Divisionsfehler Mantisse = 0 Exponentenunterlauf Dezimalüberlauf Festpunktüberlauf	X' 04' X' 58' X' 60' X' 64' X' 68' X' 6C' X' 70' X' 74' X' 78'
Intervallzeitgeber für CPU-Zeit	"SETIC-Intervall" abgelaufen für CPU-Zeit	X' 20'
Intervallzeitgeber für Realzeit	"SETIC-Intervall" abgelaufen für Realzeit	X' A0'
Ende Programmlaufzeit	Ende der Programmlaufzeit	X' 80'
nicht behebbarer Programmfehler	priviligierter SVC Zugriff auf eine nicht vorhandene Speicherseite priviligierte Operation Adressenfehler XA-Fehler (falscher Adressierungsmodus) Realtimer (Condition Error) Ausrichtungsfehler Validierungsfehler nicht behebbarer Vektorprozessor Fehler	X' 08' X' 48' X' 54' X' 5C' X' 9C' X' A4' X' AC' X' B0' X' B4'
Mitteilung an das Programm	Kommando	X' 44'
ESCPBRK	BREAK/ESCAPE (über Tasten)	X' 84'
ABEND	Systemfehler, Leitungsverlust START-EXECUTABLE-PROGRAM, LOAD-EXECUTABLE-PROGRAM, ABEND, LOGOFF, CANCEL-JOB Adress-Übersetzungsfehler wegen Hardwarefehler Hardwarefehler (CPU) erzwungenes Entladen eines Subsystems (Systemverwaltung)	X' 88' X' 8C' X' 94' X' A8' X' B8'
Programmbeendigung	TERM CMD	X' 90' X' 98'
SVC-Unterbrechung	SVC-Aufruf eines angegebenen SVCs	X' 50'
Hardwarefehler	Ein-/Ausgabefehler bei Data-In-Virtual-Technik	X' 28'

Fachwörter

Ablaufüberwachung

%CONTROLn, %INSERT und %ON sind Kommandos zur Ablaufüberwachung. Kommt der Programmablauf an eine Anweisung oder einen Befehl der gewählten Gruppe (%CONTROLn) oder an die vereinbarte Programmadresse (%INSERT) oder tritt das ausgewählte Ereignis ein (%ON), wird der Programmablauf unterbrochen, und AID bearbeitet das vereinbarte Subkommando.

Ablaufverfolgung

%TRACE ist das Kommando zur Ablaufverfolgung. Mit ihm vereinbaren Sie, welche und wieviele Befehle oder Anweisungen protokolliert werden sollen. Im Standardfall können Sie den Programmablauf am Bildschirm mitverfolgen.

Adressoperand

ist ein Operand, mit dem Sie eine Speicherstelle oder einen Speicherbereich adressieren. Sie können virtuelle Adressen, Datennamen, Anweisungsnamen, Source-Referenzen, Schlüsselwörter, komplexe Speicherreferenzen, eine C-Qualifikation (maschinennahes Testen) oder eine PROG-Qualifikation (symbolisches Testen) angeben. Die Speicherstelle bzw. der Speicherbereich liegen entweder im geladenen Programm oder in einem Speicherabzug in einer Dump-Datei.
Wenn ein Name in Ihrem Programm mehrfach vergeben wurde und somit kein eindeutiger Bezug auf eine Adresse gewährleistet ist, können Sie ihn mit Bereichsqualifikationen oder über eine *kennzeichnung* (COBOL) eindeutig der gewünschten Adresse zuordnen.

Adressierungsmodus

AID übernimmt den Adressierungsmodus des Testobjekts (24-Bit-Adressen oder 31-Bit-Adressen). Sie können mit AID auch Programme testen, die aus Moduln mit unterschiedlichem Adressierungsmodus gebunden wurden. Im Systeminformationsfeld %AMODE ist stets der aktuelle Adressierungsmodus eingetragen. Ändern können Sie den Adressierungsmodus mit `%MOVE %MODE{24|31} INTO %AMODE`, anschauen mit `%DISPLAY %AMODE`.

Änderungsdialog

Mit dem Kommando %AID CHECK=ALL schalten Sie den Änderungsdialog ein. Er wird bei der Ausführung von %MOVE oder %SET wirksam. AID fragt im Dialog nach, ob die Änderung des Speicherinhalts wirklich durchgeführt werden soll. Wird als Antwort ein N eingegeben, unterbleibt die Änderung; wird ein Y eingegeben, führt AID die Übertragung aus.

AID-Arbeitsbereich

ist der Adressraum, in dem Sie Speicherreferenzen ohne Angabe einer Basisqualifikation ansprechen können.
Er umfasst den nicht-privilegierten Teil des virtuellen Speichers in Ihrer Task, der vom Programm samt allen konnektierten Subsystemen belegt ist, oder den entsprechenden Bereich in einem Speicherabzug.
Mit dem Kommando %BASE können Sie den AID-Arbeitsbereich vom geladenen Programm in einen Speicherabzug verlegen oder umgekehrt. In einem Kommando können Sie vom AID-Arbeitsbereich abweichen, indem Sie im Adressoperanden eine Basisqualifikation angeben.

AID-Ausgabedateien

sind die Dateien, in die Sie sich die Ausgaben der Kommandos %DISASSEMBLE, %DISPLAY, %HELP, %SDUMP und %TRACE schreiben lassen können. Die Dateien werden in den Ausgabekommandos über ihre Linknamen F0 bis F7 angesprochen (siehe %OUT und %OUTFILE).
In die Datei, die dem Linknamen F6 zugewiesen wurde, werden die REP-Sätze geschrieben (siehe %AID REP=YES und %MOVE).
Es gibt drei Wege, eine Ausgabedatei anzulegen:

1. /%OUTFILE-Kommando mit dem Link- und Dateinamen
2. /FILE-Kommando mit dem Link- und Dateinamen
3. Für einen Linknamen, dem noch kein Dateiname zugewiesen ist, setzt AID einen FILE-Makro mit dem Dateinamen AID.OUTFILE.Fn ab.

Eine AID-Ausgabedatei hat stets das Format FCBTYPE=SAM, RECFORM=V und OPEN=EXTEND.

AID-Eingabedateien

sind Dateien, die AID zur Ausführung von AID-Funktionen benötigt, im Unterschied zu Eingabedateien, die das Programm benutzt. AID verarbeitet nur Platten-Dateien. AID-Eingabedateien sind:

1. Dump-Dateien, in denen sich Speicherabzüge befinden (%DUMPFILE)
2. PLAM-Bibliotheken, in denen sich Bindemodule befinden. Wird die Bibliothek mit %SYMLIB zugewiesen, kann AID die LSD-Sätze nachladen.

AID-Literale

AID stellt Ihnen Zeichen-Literale und numerische Literale zur Verfügung (siehe Kapitel „AID-Literale“ auf Seite 103):

{C'x...x' | 'x...x'C | 'x...x'| U'x...x'}	Character-Literal
{X'f...f' | 'f...f'X}	sedezimal-Literal
{B'b...b' | 'b...b'B}	Binär-Literal
[{±}]n	Ganzzahl
#'f...f'	Sedezimalzahl
[{±}]n.m	Dezimalpunktzahl
[{±}]mantisseE[{±}]exponent	Gleitpunktzahl

AID-Standard-Adressinterpretation

Indirekte Adressen, d.h. Adressen vor einem Pointer-Operator, werden im Standardfall entsprechend dem gerade gültigen Adressierungsmodus interpretiert. Mit %AINT können Sie von der Standard-Adressinterpretation abweichen und festlegen, ob AID bei indirekter Adressierung mit 24-Bit-Adressen oder mit 31-Bit-Adressen arbeiten soll.

AID-Standard-Arbeitsbereich

Das ist der nicht-privilegierte Teil des virtuellen Speichers in Ihrer Task, der vom Programm samt allen konnektierten Subsystemen belegt ist.
Ohne Vereinbarung mit %BASE und ohne Angabe einer Basisqualifikation gilt der AID-Standard-Arbeitsbereich.

Aktuelle Aufrufhierarchie

ist der Stand der Unterprogrammverschachtelung an der Unterbrechungsstelle. Sie reicht von der Unterprogrammebene, auf der das Programm unterbrochen wurde über die durch CALL-Anweisungen verlassenen Unterprogramme mittlerer Hierarchiestufen bis zum Hauptprogramm.
Sie wird mit %SDUMP %NEST ausgegeben.

Aktuelle CSECT

ist die CSECT, in der das Programm unterbrochen wurde. Die STOP-Meldung gibt ihren Namen aus.

Aktuelles Programm

ist das Programm, das in der Task geladen ist, in der Sie AID-Kommandos eingeben.

Aktueller Programmteil

ist der Programmteil, in dem das Programm unterbrochen wurde. Die STOP-Meldung gibt seinen Namen aus.

Anweisungsname

bezeichnet einen im Quellprogramm vergebenen Namen, mit dem in AID eine ausführbare Anweisung angesprochen werden kann. Das sind Marken oder Namen von Haupt- oder Unterprogrammen. Hierzu ist in den LSD-Sätzen eine Adresskonstante hinterlegt, die die Adresse der ersten Anweisung nach der Marke bzw. im Haupt- oder Unterprogramm enthält. Genauer ist es die Adresse des ersten Befehls, der zu der ersten ausführbaren Anweisung nach einer Marke bzw. im Haupt- oder Unterprogramm erzeugt wurde.

Attribute

Jedes Speicherobjekt hat bis zu sechs Attribute:
Adresse, Name (opt), Inhalt, Länge, Speichertyp, Ausgabetyp.
Mit Selektoren können Sie auf Adresse, Länge und Speichertyp zugreifen. Über den Namen findet AID in den LSD-Sätzen alle zugehörigen Attribute, um das zugehörige Speicherobjekt richtig zu interpretieren.
Adresskonstanten und Konstanten aus dem Quellprogramm haben nur bis zu fünf Attribute:
Name (opt), Wert, Länge, Speichertyp, Ausgabetyp.
Sie haben keine Adresse. Beim Ansprechen einer Konstanten greift AID nicht auf ein Speicherobjekt zu, sondern setzt nur den Wert der Konstanten ein.

Ausgabetyp

Attribut eines Speicherobjekts, das bestimmt, wie der Speicherinhalt von AID ausgegeben wird. Jedem Speichertyp ist ein Ausgabetyp zugeordnet. In Kapitel „Schlüsselwörter“ auf Seite 111 sind die AID-spezifischen Speichertypen samt zugehörigen Ausgabetypen aufgelistet. Für die Datentypen in der jeweiligen Programmiersprache gilt eine entsprechende Zuordnung. Eine Typmodifikation bei %DISPLAY und %SDUMP bewirkt eine Änderung des Ausgabetyps.

Basisqualifikation

ist die Qualifikation, mit der Sie das geladene Programm oder einen Speicherabzug in einer Dump-Datei bezeichnen. Sie wird mit E={VM | Dn} angegeben. Die Basisqualifikation können Sie global mit %BASE vereinbaren oder im Adressoperanden für eine einzelne Speicherreferenz angeben.

Bereichsgrenzen

Jedem Speicherobjekt ist ein bestimmter Bereich zugeordnet, der bei Datennamen und Schlüsselwörtern durch die Attribute Adresse und Länge festgelegt ist. Bei virtuellen Adressen liegen die Bereichsgrenzen zwischen V’0’ und der letzten Adresse des virtuellen Speichers (V’7FFFFFFF’). Bei Bereichsqualifikationen ergeben sich die Bereichsgrenzen aus Anfangs- und Endadresse des damit bezeichneten Programmteils (siehe Kapitel „Adressierung in AID“ auf Seite 65).

Bereichsqualifikationen

Mit diesen Qualifikationen bezeichnen Sie einen Teil des Arbeitsbereiches. Endet ein Adressoperand mit einer dieser Qualifikationen, so wirkt das Kommando nur in dem Teil, der mit der letzten Qualifikation bezeichnet wurde. Mit einer Bereichsqualifikation begrenzen Sie den Wirkungsbereich eines Kommandos oder machen damit einen Daten- oder Anweisungsnamen im Arbeitsbereich eindeutig oder Sie erreichen damit einen Namen, der an der aktuellen Unterbrechungsstelle sonst nicht ansprechbar ist.

Bereichsüberprüfung

AID überprüft bei Adressversatz, Längenmodifikation und bei *empfänger* in einem %MOVE, ob die Bereichsgrenzen der angesprochenen Speicherobjekte überschritten werden und gibt im Fehlerfall eine entsprechende Meldung aus.

Benutzerbereich

ist der Bereich des virtuellen Speichers, der vom geladenen Programm samt allen seinen konnektierten Subsystemen belegt ist. Er entspricht dem Bereich, der durch das Schlüsselwort %CLASS6 bzw. %CLASS6ABOVE und %CLASS6BELOW repräsentiert wird.

CSECT-Informationen

stehen in der Objekt-Strukturliste.

Datenname

steht für alle Namen, die im Quellprogramm für Daten vergeben wurden. Mit einem Datennamen sprechen Sie Variablen, Konstanten, Strukturen, Tabellen und Struktur- oder Tabellenelementen beim symbolischen Testen an. Auf Elemente von Tabellen können Sie sich wie in der jeweiligen Programmiersprache über einen Index beziehen.

Datentyp

Gemäß dem im Quellprogramm deklarierten Datentyp ordnet AID allen Datenelementen einen AID-Speichertyp zu:

- Binärstring (≙ %X)
- Character (≙ %C oder %UTF16)
- numerisch, wobei nicht alle Datentypen, die in der jeweiligen Programmiersprache numerisch behandelt werden, auch für AID vom Speichertyp numerisch sind (siehe hierzu die einzelnen sprachspezifischen AID-Handbücher [2]-[6]).

Der zugeordnete Speichertyp bestimmt, wie das Datenelement von %DISPLAY ausgegeben, von %MOVE oder %SET übertragen bzw. überschrieben und wie es in der Bedingung eines Subkommandos verglichen wird.

Eingabepuffer

AID hat einen internen Eingabepuffer. Reicht er für die Aufnahme der Eingabe eines Kommandos nicht aus, wird das Kommando mit einer Fehlermeldung als zu lang abgewiesen. Die gewünschte Operation müssen Sie auf zwei Kommandos aufteilen, damit AID sie durchführen kann.

ESD/ESV

External Symbol Dictionary bei OMs / External Symbols Vector bei LLMs ist das Verzeichnis der Externbezüge eines Moduls. Es wird vom Compiler erstellt. Hierin sind unter anderem Informationen über CSECTs, DSECTs und COMMONs enthalten. Der Binder greift auf dieses Verzeichnis zu, wenn er die Objekt-Strukturliste erzeugt.

globale Einstellungen

AID stellt Ihnen Kommandos zur Verfügung, mit denen Sie das Verhalten von AID Ihren Testerfordernissen anpassen können, die Ihnen die Adressierung erleichtern oder Schreibarbeit ersparen. Die Voreinstellungen gelten während der gesamten Testsitzung (siehe %AID, %AINT, %BASE und %QUALIFY).

Hauptprogramm

wird in diesem Handbuch als Sammelbegriff für das Programm (COBOL), die Funktion (main bei C++/C) oder die externe Prozedur (PL/I) verwendet, die beim Anlaufen des Programms vom System gestartet wird.

Index

ist ein Teil eines Adressoperanden. Mit einem Index wird die Position eines Vektorelements bestimmt. Er kann wie in der Programmiersprache angegeben werden oder durch einen arithmetischen Ausdruck, aus dem AID den Wert des Index errechnet.

Kommandofolge

Mehrere Kommandos werden mit Semikolon (;) zu einer Folge verbunden, die von links nach rechts abgearbeitet wird. Wie im Subkommando darf eine Kommandofolge AID- und BS2000-Kommandos enthalten. In Kommandofolgen nicht zugelassen sind die AID-Kommandos %AID, %BASE, %DUMPFILE, %HELP, %OUT, %QUALIFY und die im Anhang aufgelisteten BS2000-Kommandos.

Enthält eine Kommandofolge eines der Kommandos zur Ablaufsteuerung, wird die Kommandofolge an der Stelle abgebrochen und das Programm gestartet (%CONTINUE, %RESUME, %TRACE) oder angehalten (%STOP). Nachfolgende Kommandos aus der Kommandofolge werden nicht mehr ausgeführt.

Kommandomodus

Mit Kommandomodus wird in den AID-Handbüchern der EXPERT-Modus der SDF-Kommandosprache bezeichnet. Falls Sie gerade in einem anderen Modus (GUIDANCE={MAXIMUM | MEDIUM | MINIMUM | NO}) arbeiten, sollten Sie mit Kommando MODIFY-SDF-OPTIONS GUIDANCE=EXPERT in den EXPERT-Modus umschalten, wenn Sie AID-Kommandos eingeben wollen. AID-Kommandos verfügen nicht über eine SDF-Syntax:

– Operanden werden nicht über Menüs abgefragt.
– Im Fehlerfall gibt AID eine Fehlermeldung aus, führt aber keinen Korrekturdialog.

Im EXPERT-Modus fordert Sie das System mit "/" zur Kommandoeingabe auf.

Konstante

Eine Konstante repräsentiert einen Wert, der nicht über eine Adresse im Programmspeicher hinterlegt ist.
Zu den Konstanten gehören die im Quellprogramm definierten Konstanten, die Ergebnisse von Längenselektion, Längenfunktion und Adressselektion sowie die Anweisungsnamen und die Source-Referenzen.

Eine Adresskonstante repräsentiert eine Adresse. Adresskonstanten sind Anweisungsnamen, Source-Referenzen und das Ergebnis einer Adressselektion.
Eine Adresskonstante kann in einer komplexen Speicherreferenz nur vor einem Pointer-Operator (->) eingesetzt werden.

LIFO

Last In First Out; treffen an einem Testpunkt (%INSERT) oder bei Auftreten eines Ereignisses (%ON) Anweisungen aus verschiedenen Eingaben zusammen, so werden die zuletzt eingegebenen zuerst abgearbeitet (siehe Abschnitt „Ketten“ auf Seite 58).

Lokalisierungsinformation

Mit %DISPLAY %HLLOC(speicherref) für die symbolische Ebene und mit %DISPLAY %LOC(speicherref) für die Maschinencode-Ebene gibt Ihnen AID die statische Programmverschachtelung zu der angegebenen Speicherstelle aus.
Im Gegensatz dazu erhalten Sie mit %SDUMP %NEST die dynamische Programmverschachtelung, das ist die Aufrufhierarchie zur aktuellen Programmunterbrechungsstelle.

LSD

List for Symbolic Debugging ist ein Verzeichnis der im Modul definierten Daten- und Anweisungsnamen. Ebenso sind dort die vom Compiler erzeugten Source-Referenzen hinterlegt. Die LSD-Sätze werden vom Compiler erzeugt. AID holt sich hieraus die Informationen zur symbolischen Adressierung.

Namensraum

umfasst alle zu einem Programmteil in den LSD-Sätzen verzeichneten Datennamen.

Objekt-Strukturliste

Auf Basis des ESD (External Symbol Dictionary) erstellt der Binder die Objekt-Strukturliste, wenn die Standardeinstellung SYMTEST=MAP gilt bzw. wenn Sie SYMTEST=ALL angegeben haben.

Programmeinheit

Begriff, der in Fortran für das gebraucht wird, was in anderen Programmiersprachen mit Übersetzungseinheit bezeichnet wird. Die Programmeinheit können Sie mit der S-Qualifikation ansprechen.

Programmteil

ist ein Sammelbegriff für den Teil eines Programms, der mit einer Bereichsqualifikation angesprochen werden kann. Jede Programmiersprache hat dafür eigene Bezeichnungen, die in den sprachspezifischen AID-Handbüchern beschrieben sind.

Programmzustand

AID unterscheidet drei Programmzustände, in denen sich das zu testende Programm befinden kann:

1. Das Programm steht.
%STOP oder K2-Taste unterbrechen ein laufendes Programm. Außerdem wird das Programm unterbrochen, wenn ein %TRACE abgearbeitet ist. Die Task befindet sich im Kommandomodus. Sie können Kommandos eingeben.

2. Das Programm läuft ohne Ablaufverfolgung.
%RESUME startet ein Programm oder setzt es fort. %CONTINUE bewirkt dasselbe; ist allerdings ein %TRACE noch nicht ganz abgearbeitet, so setzt %CONTINUE das Programm mit Ablaufverfolgung fort.

3. Das Programm läuft mit Ablaufverfolgung.
%TRACE startet ein Programm oder setzt es fort. Der Programmablauf wird entsprechend der Vereinbarungen im %TRACE protokolliert. %CONTINUE bewirkt dasselbe, wenn noch ein %TRACE aktiv ist.

Qualifikation

Mit einer Qualifikation können Sie eine Speicherreferenz ansprechen, die nicht im AID-Arbeitsbereich liegt oder außerhalb des aktuellen Haupt- oder Unterprogramms oder darin nicht eindeutig ist. Die Basisqualifikation gibt an, ob die Speicherreferenz im geladenen Programm oder in einem Speicherabzug liegt. Eine Bereichsqualifikation gibt an, in welchem Programmteil eine Speicherreferenz liegt. Wenn ein Operand durch eine Qualifikation überbestimmt ist (d.h. die Qualifikation ist überflüssig oder widersprüchlich), wird die Qualifikation ignoriert. Das ist z.B. der Fall, wenn zu einer virtuellen Adresse eine Bereichsqualifikation angegeben wird.

Source-Referenz

bezeichnet eine ausführbare Anweisung. Sie wird mit S'nummer/name' angegeben. *nummer/name* wird vom Compiler erzeugt und in den LSD-Sätzen hinterlegt.

Speicherobjekt

ist eine bestimmte Anzahl von zusammenhängenden Bytes im Speicher. Auf Programmebene sind das die Daten des Programms, sofern ihnen ein Speicherbereich zugewiesen ist, und der Befehlscode. Außerdem gehören alle Register, der Befehlszähler sowie alle anderen Bereiche, die nur über Schlüsselwörter angesprochen werden können, ebenfalls zu den Speicherobjekten. Keine Speicherobjekte hingegen sind alle im Programm definierten Konstanten, die Anweisungsnamen, Source-Referenzen, die Ergebnisse von Adressselektion, Längenselektion und Längenfunktion und die AID-Literale. Sie repräsentieren einen Wert, der nicht verändert werden kann.

Speicherreferenz

Mit einer Speicherreferenz sprechen Sie ein Speicherobjekt an. Es gibt einfache und komplexe Speicherreferenzen. Einfache Speicherreferenzen sind virtuelle Adressen, Namen, zu denen AID sich die Adresse aus den LSD-Informationen holen kann, und Schlüsselwörter. Anweisungsnamen und Source-Referenzen sind in den AID-Kommandos %CONTROLn, %DISASSEMBLE, %INSERT, %JUMP, %REMOVE und %TRACE als Speicherreferenz erlaubt, obwohl sie nur Adresskonstanten sind.
Mit den komplexen Speicherreferenzen geben Sie AID eine Vorschrift an, wie die gewünschte Adresse errechnet werden soll und welcher Typ und welche Länge gelten sollen. Folgende Operationen können in einer komplexen Speicherreferenz vorkommen: Adressversatz, indirekte Adressierung, Typmodifikation, Längenmodifikation und Adressselektion.

Speichertyp

ist entweder der Datentyp, der im Quellprogramm festgelegt wurde oder der durch Typmodifikation gewählte. AID kennt die Speichertypen %X, %C, %P, %D, %F, %A (siehe %SET und Kapitel „Adressierung in AID“ auf Seite 65 und Kapitel „Schlüsselwörter“ auf Seite 111).

Subkommando

ist ein Operand der Überwachungskommandos %CONTROLn, %INSERT oder %ON. Ein Subkommando besteht aus einem Kommandoteil, dem wahlweise ein Name und eine Bedingung vorangestellt sein kann. Der Kommandoteil kann aus einem einzelnen Kommando oder aus einer Kommandofolge bestehen. Er kann AID- und BS2000-Kommandos enthalten. Jedes Subkommando hat einen Durchlaufzähler. Wie eine Ausführungsbedingung formuliert wird, wie Name und Durchlaufzähler vergeben und angesprochen werden, und welche Kommandos innerhalb von Subkommandos nicht erlaubt sind, ist in Kapitel „Subkommando“ auf Seite 47 beschrieben.
Der Kommandoteil des Subkommandos wird dann ausgeführt, wenn die Überwachungsbedingung des entsprechenden Kommandos (*kriterium*, *testpunkt*, *ereignis*) zutrifft und die eventuell definierte Ausführungsbedingung erfüllt ist.

Übersetzungseinheit

Der Teil eines Programms, der als eine Einheit übersetzt wurde. Bei Fortran wird hierfür der Begriff Programmeinheit verwendet. Die Übersetzungseinheit können Sie mit der S-Qualifikation ansprechen.

Unterbrechungsstelle

Die Adresse, an der ein Programm unterbrochen wurde, wird Unterbrechungsstelle genannt. Aus der STOP-Meldung können Sie entnehmen, an welcher Adresse und in welchem Programmteil die Unterbrechungsstelle liegt. Dort wird das Programm fortgesetzt. Für COBOL- und FOR1-Programme können Sie mit %JUMP eine andere Fortsetzungsadresse vereinbaren.

Unterprogramm

wird in diesem Handbuch als Sammelbegriff für Funktionen (C++/C, Fortran, COBOL), Prozeduren (PL/I) oder Programme (COBOL) verwendet, die in der Aufrufhierarchie dem Hauptprogramm untergeordnet sind.

Zeichenkonvertierungsfunktionen

AID stellt zwei Funkionen zur Zeichenkonvertierung %C() und %UTF16() zur Verfügung.
Die Funktion %UTF16() wandelt Strings von einer 1-Byte-EBCDIC-Codierung in die UTF16-Codierung um, die Funktion %C realisiert die Konvertierung umgekehrt.

Literatur

Die Handbücher sind online unter *http://manuals.fujitsu-siemens.com* zu finden oder in gedruckter Form gegen gesondertes Entgelt unter *http://FSC-manualshop.com* zu bestellen.

[1] **AID** (BS2000/OSD)
Testen auf Maschinencode-Ebene
Benutzerhandbuch

Zielgruppe
Programmierer und Tester
Inhalt
– Beschreibung der AID-Kommandos für das Testen auf Maschinencode-Ebene
– Anwendungsbeispiel
Aufgenommen wurden die Kommandos %SHOW und %SDUMP %NEST, Kontext-COMMON-Qualifikation sowie auf ESA-Anlagen für Daten-Räume die ALET/SPID-Qualifikationen. Es gibt zusätzliche Schlüsselwörter.

[2] **AID** (BS2000)
Advanced Interactive Debugger
Testen von COBOL-Programmen
Benutzerhandbuch

Zielgruppe
COBOL-Programmierer
Inhalt
– Beschreibung der AID-Kommandos für das symbolische Testen von COBOL-Programmen
– Anwendungsbeispiel
Einsatz
Testen von COBOL-Programmen im Dialog- und Stapelbetrieb

[3] **AID** (BS2000)
Advanced Interactive Debugger
Testen von FORTRAN-Programmen
Benutzerhandbuch

Zielgruppe
FORTRAN-Programmierer
Inhalt
– Beschreibung der AID-Kommandos für das symbolische Testen von FORTRAN-Programmen
– Anwendungsbeispiel
Einsatz
Testen von FORTRAN-Programmen im Dialog- und Stapelbetrieb

[4] **AID** (BS2000)
Advanced Interactive Debugger
Testen von PL/I-Programmen
Benutzerhandbuch

Zielgruppe
PL/I-Programmierer
Inhalt
– Beschreibung der AID-Kommandos für das symbolische Testen von PL/I-Programmen
– Anwendungsbeispiel

[5] **AID** (BS2000/OSD)
Advanced Interactive Debugger
Testen von ASSEMBH-Programmen
Benutzerhandbuch

Zielgruppe
Assembler-Programmierer
Inhalt
– Beschreibung der AID-Kommandos für das symbolische Testen von ASSEMBH-Programmen
– Anwendungsbeispiel
Einsatz
Testen von ASSEMBH-Programmen im Dialog- und Stapelbetrieb

[6] **AID** (BS2000/OSD)
Testen von C/C++-Programmen
Benutzerhandbuch

Zielgruppe
Das Handbuch richtet sich an C/C++-Programmierer.
Inhalt
Das Handbuch enthält die Beschreibung der AID-Kommandos und der C/C++-spezifischen Adressoperanden zum symbolischen Testen von C/C++-Programmen. Es enthält Informationen zum Testen unter POSIX sowie zum Testen auf RISC-Anlagen und ausführliche Anwendungsbeispiele.
Einsatz
Testen von C/C++-Programmen im Dialog- und Stapelbetrieb

[7] **AID** (BS2000)
Advanced Interactive Debugger
Tabellenheft
Benutzerhandbuch

Zielgruppe
Programmierer im BS2000
Inhalt
– Testen von Programmen der Programmiersprachen ASSEMBH, C/C++, COBOL, FORTRAN, PL/I und auf Maschinencode-Ebene
– Kurzfassung der AID-Kommandos und Operanden
– %SET-Tabellen
Einsatz
Testen von Programmen im Dialog- und Stapelbetrieb

[8] **BS2000/OSD-BC**
Kommandos Band 1 - 5
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich sowohl an den nichtprivilegierten Anwender als auch an die Systembetreuung.
Inhalt
Die Bände 1 - 5 enthalten die Kommandos ADD-... bis WRITE-... (BS2000/OSD-Grundausbau und ausgewählte Produkte) mit der Funktionalität für alle Privilegien. Die Kommando- und Operandenfunktionen werden ausführlich beschrieben; viele Beispiele unterstützen das Verständnis. Am Anfang jedes Bandes informiert eine Übersicht über alle in den Bänden 1 - 5 beschriebenen Kommandos.
Der Anhang von Band 1 enthält u.a. Informationen zur Kommandoeingabe, zu bedingten Jobvariablenausdrücken, Systemdateien, Auftragsschaltern, Geräte- und Volumetypen. Der Anhang der Bände 4 und 5 enthält jeweils eine Übersicht zu den Ausgabespalten der SHOW-Kommandos der Komponente NDM. Der Anhang von Band 5 enthält zusätzlich eine Übersicht aller START-Kommandos.
In jedem Band ist ein umfangreiches Stichwortverzeichnis mit allen Stichwörtern der Bände 1 - 5 enthalten.

[9] **BS2000/OSD-BC**
Einführung in die Systembetreuung
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich an die Systembetreuung und das Operating des Betriebssystems BS2000/OSD.
Inhalt
Es sind u.a. folgende Themen zur Verwaltung und Überwachung des BS2000/OSD-Grundausbaus enthalten: Systemeinleitung, Parameterservice, Job- und Tasksteuerung, Speicher-, Geräte-, Benutzer-, Datei-, Pubset- und Systemzeit-Verwaltung, Privilegienvergabe, Accounting und Operatorfunktionen.

[10] BS2000
Systemverwalter-Kommandos (SDF-Format)
Beschreibung

Zielgruppe
BS2000-Systemverwalter
Inhalt
SDF-Kommandos für den Systemverwalter
Einsatz
Systemverwaltung

[11] **BS2000/OSD**
Makroaufrufe an den Ablaufteil
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich an alle BS2000/OSD-Assembler-Programmierer.
Inhalt
Das Handbuch enthält eine Zusammenstellung der Makroaufrufe an den Ablaufteil, die ausführliche Beschreibung jedes Makroaufrufs mit Hinweisen und Beispielen sowie einen ausführlichen allgemeinen Lernteil.

[12] **LMS** (BS2000/OSD)
SDF-Format
Benutzerhandbuch

Zielgruppe
BS2000-Anwender
Inhalt
Beschreibung der Anweisungen zum Erstellen und Verwalten von PLAM-Bibliotheken und darin enthaltenen Elementen.
Häufige Anwendungsfälle werden an Hand von Beispielen erklärt.

[13] BS2000
TSOSLNK
Benutzerhandbuch

Zielgruppe
Software-Entwickler
Inhalt
– Anweisungen und Makroaufrufe des Binders TSOSLNK zum Binden von Lade- und Großmodulen
– Kommandos des statischen Laders ELDE

[14] **BINDER**
Binder in BS2000/OSD
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich an Software-Entwickler
Inhalt
Es beschreibt die BINDER-Funktionen und enthält Beispiele dazu. Im Nachschlageteil sind die BINDER-Anweisungen und der Makroaufruf BINDER beschrieben.

[15] **BLSSERV**
Bindelader-Starter in BS2000/OSD
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich an Software-Entwickler und geübte BS2000/OSD-Benutzer.
Inhalt
Es beschreibt die Funktionen, die Unterprogrammschnittstelle, die XS-Unterstützung und den Aufruf des Bindeladers DBL als Bestandteil des Subsystems BLSSERV.

[16] **SDF** (BS2000/OSD)
Einführung in die Dialogschnittstelle SDF
Benutzerhandbuch

Zielgruppe
BS2000/OSD-Anwender
Inhalt
Das Handbuch beschreibt die Dialog-Eingabe von Kommandos und Anweisungen im SDF-Format. Ein Schnelleinstieg mit leicht nachvollziehbaren Beispielen und weitere umfangreiche Beispiele erleichtern die Anwendung. SDF-Syntaxdateien werden erklärt.

[17] BS2000
Benutzerkommandos (ISP-Format)
Benutzerhandbuch

Zielgruppe
BS2000-Anwender (nicht privilegiert)
Inhalt
Alle BS2000-Systemkommandos in lexikalischer Reihenfolge mit Hinweisen und Beispielen.
Folgende Liefereinheiten sind berücksichtigt:
BS2000-GA, MSCF, JV, FT, TIAM
Einsatz
BS2000-Dialogbetrieb, -Prozeduren, -Stapelbetrieb

[18] **BS2000/OSD-BC**
Systembedienung
Benutzerhandbuch

Zielgruppe
Das Handbuch "Systembedienung" ist eine Beschreibung für den Operator an Anlagen des Betriebssystems BS2000/OSD.
Inhalt
Es beschreibt die Aufgabengebiete und Funktionsbereiche des Operators sowie die Kommandos, die ihm an der Bedienstation zur Wahrnehmung seiner Aufgaben zur Verfügung stehen. Es enthält folgende Kapitel:
– Systemeinleitung und -beendigung (Arten des Systemstarts, SHUTDOWN)
– Kommandos in alphabetischer Reihenfolge
– Geräteverwaltung (Rekonfiguration, Betriebsmittelbelegung, Datenträgerüberwachung, Umgang mit NDM, Duplex-Rekonfiguration)
– Hilfsmittel und Methoden zur Vereinfachung der Systembedienung
– Speicherauszüge (SLED)
– Meldungen und Maßnahmen bei Sättigungszuständen

[19] **Fortran90** (BS2000/OSD)
Fortran90-Compiler
Benutzerhandbuch

Zielgruppe
Fortran90-Anwender im BS2000.
Inhalt
Das Handbuch beschreibt alle Tätigkeiten zum Erzeugen eines ablauffähigen Fortran90-Programms: Übersetzen, Binden, Laden, Testen. Es beinhaltet Programmierhinweise und weitergehende Informationen zu Dateiveratbeitung und Sprachverknüpfung.

[20] BS2000
Programmiersystem
Technische Beschreibung

Zielgruppe
– BS2000-Anwender und -Betreiber, die sich für den technischen Hintergrund ihres Systems interessieren (Softwareentwickler, Systemanalytiker, RZ-Leiter, Systemverwalter)
– Informatiker, die ein konkretes "General-Purpose"-Betriebssystem studieren wollen
Inhalt
Funktionen und Realisierungsprinzipien
– des Binders
– des Laders
– des Binde-Laders
– der Test- und Diagnosehilfen
– des Programmbibliothekssystems

Stichwörter

T

Fujitsu Siemens Computers GmbH
Handbuchredaktion
81730 München

Kritik
Anregungen
Korrekturen

Fax: 0 700 / 372 00001

e-mail: manuals@fujitsu-siemens.com
http://manuals.fujitsu-siemens.com

Absender

Kommentar zu AID V3.2A
Basishandbuch

Information on this document

On April 1, 2009, Fujitsu became the sole owner of Fujitsu Siemens Computers. This new subsidiary of Fujitsu has been renamed Fujitsu Technology Solutions.

This document from the document archive refers to a product version which was released a considerable time ago or which is no longer marketed.

Please note that all company references and copyrights in this document have been legally transferred to Fujitsu Technology Solutions.

Contact and support addresses will now be offered by Fujitsu Technology Solutions and have the format *…@ts.fujitsu.com*.

The Internet pages of Fujitsu Technology Solutions are available at *http://ts.fujitsu.com/*...
and the user documentation at *http://manuals.ts.fujitsu.com*.

Hinweise zum vorliegenden Dokument

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 aus dem Dokumentenarchiv bezieht sich auf eine bereits vor längerer Zeit freigegebene oder nicht mehr im Vertrieb befindliche 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/*..., und unter *http://manuals.ts.fujitsu.com* finden Sie die Benutzerdokumentation.