
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 neu übersetzt werden, sondern kann ohne LSD oder evtl. auch mit AID-Korrekturen, in den produktiven Einsatz gehen. Der Funktionsumfang von AID und seine Testsprache, die AID-Kommandos, sind primär auf die Dialoganwendung zugeschnitten. AID kann aber ebensogut 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" können Sie Daten, Anweisungen und Programmteile mit den im Quellcode vereinbarten Namen ansprechen, und Anweisungen ohne Namen mit der vom Compiler erzeugten Source-Referenz. Die BS2000-Kommandos, die in der AID-Dokumentation vorkommen, sind im SDF-Format (System Dialog Facility), EXPERT-Form beschrieben. SDF ist die Dialogschnittstelle zum BS2000. Die SDF-Kommandosprache löst die bisherige Kommandosprache im ISP-Format ab.

AID V2.1A können Sie ab BS2000 V10 einsetzen; die ESA-Unterstützung und die Schreibüberwachung ab BS2000/OSD V1.0.

1.1 Zielgruppe

AID wendet sich an alle Software-Entwickler, die im BS2000 mit den Programmiersprachen ASSEMBLER, COBOL, Fortran, C, C++, PL/I arbeiten und Programme testen und auch korrigieren wollen. Dieses Handbuch wendet sich an Tester, die auf Maschinencode-Ebene mit AID arbeiten wollen.

1.2 Konzept der AID-Handbücher

Die Dokumentation für 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 ein Tabellenheft [11] vor, in dem die Syntax der Kommandos und der Operanden mit kurzen Erläuterungen aufgeführt sind. Außerdem gibt es darin die %SET-Tabellen und eine Gegenüberstellung AID - IDA. 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 [1]

Im Basishandbuch finden Sie einen Überblick über AID und die Beschreibung der Sachverhalte und Operanden, die in allen 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, die Operanden Subkommando, komplexe Speicherreferenz und Medium-und-Menge, die AID-Literale und die Schlüsselwörter. Das Handbuch enthält außerdem die Meldungen, die in Kommandofolgen unzulässigen BS2000-Kommandos, die nur noch in dieser Version unterstützten Operanden und eine Gegenüberstellung von AID und IDA.

AID Benutzerhandbücher

In den Benutzerhandbüchern finden Sie die Kommandos in alphabetischer Reihenfolge. Alle einfachen Speicherreferenzen sind in diesen Handbüchern beschrieben. Neben dem vorliegenden Handbuch

AID - Testen auf Maschinencode-Ebene

gibt es noch die Benutzerhandbücher

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 [12]

In den sprachspezifischen Handbüchern und dem Handbuch für das Testen auf Maschinencode-Ebene finden Sie die Kommandos in alphabetischer Reihenfolge. Alle einfachen Speicherreferenzen sind in diesen Handbüchern beschrieben.

In den sprachspezifischen Handbüchern ist die Beschreibung der Operanden auf die jeweilige Programmiersprache zugeschnitten. Es wird vorausgesetzt, daß Ihnen der Sprachumfang und die Handhabung des jeweiligen Compilers geläufig sind.

Das vorliegende Handbuch 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 Änderungen gegenüber AID V2.0A

Zur ESA-Unterstützung wurden folgende Erweiterungen eingeführt:

- Die ALET/SPID-Qualifikation ermöglicht den Zugriff auf virtuelle Adressen in einem Datenraum.
- Über die Schlüsselwörter %0AR bis %15AR können Sie die Zugriffsregister lesen und verändern.
- Das Schlüsselwort %DS[(ALET/SPID-qua)] liefert Informationen über Datenräume und über die zugehörigen SPIDs und ALETs.
- Das Schlüsselwort %ASC informiert über den ASC-Modus, der bestimmt, ob der AR-Modus gesetzt ist und Datenräume adressiert werden oder nicht.
- Das %TRACE-Protokoll enthält zusätzlich zum bisherigen Umfang nun auch den Inhalt der Zugriffsregister, die neuen ESA-Maschinenbefehle und einen Stern (*) vor Adressen aus Datenräumen.
- Adressen aus Datenräumen sind bei der Ausgabe mit %DISPLAY mit einem Stern (*) gekennzeichnet.

Die neuen CTX- und COM-Qualifikationen wurden in das Kapitel 4 Maschinencode-spezifische Adressierung und die Beschreibung der Adreßoperanden aufgenommen. Ergänzende Informationen finden Sie im AID-Basishandbuch, Kapitel 7.1.

Im Kapitel 5 „AID-Kommandos“ wurden die in jedem Adreßoperanden auftretenden Texte z.B. zu den Qualifikationen, Speicherreferenzen u.s.w., soweit als möglich beschränkt. Zu Beginn wird es dadurch sicher öfter erforderlich, im Kapitel 4 „Maschinencode-spezifische Adressierung“ nachzuschlagen. Nach etwas Übung mit dem Produkt ist es dann angenehmer, wenn die Wiederholungen entfallen.

Das Kommando %SHOW wurde beschrieben. Damit können Sie Informationen anfordern über aktuelle Kommandos, Testpunkte und Ereignisse mit den zugehörigen Subkommandos sowie Operandenwerte und Subkommandos.

Die Schlüsselwörter %AMODE, %LINK und %MAP wurden aufgenommen. Mit %DISPLAY %LINK wird der Name des Segments ausgegeben, das beim letzten erfolgreichen %LPOV-Ereignis nachgeladen wurde (siehe %ON).

Mit %DISPLAY %MAP können Sie sich eine Übersicht über Kontexte, mit den Commons und CSECTs ausgeben lassen. Im Gegensatz zum Schlüsselwort %SORTEDMAP werden hier lange Namen ungekürzt ausgegeben.

Die Schlüsselwörter %IFR, %IMR, %ISR werden **nicht** mehr beschrieben.

In AID-Ausgaben werden jetzt auch der Kontextname und vorhandene COMMON-Bereiche ausgegeben.

Mit %ON kann der schreibende Zugriff auf eine Speicherstelle überwacht werden. Hierzu wurde mit AID V2.0B das Ereignis %WRITE (speicherref) eingeführt. Gegenüber der ersten Version kann jetzt ein Bereich von 64KB überwacht werden.

Für das %ON-Kommando hat sich die Reihenfolge der Abarbeitung geändert, wenn mehrere Ereignisse gleichzeitig auftreten.

LMS-Korrektur-Anweisungen werden nur noch im SDF-Format erzeugt. Sie können nur mit LMS \geq V 2.0A verarbeitet werden.

Der Meldungsschlüssel der AID-Meldungen wurde von *In* in *AIDOn* geändert. Außerdem gibt es jetzt auch deutsche AID-Meldungstexte. Die Meldungen können jetzt mit /HELP-MES-SAGE-INFORMATION abgefragt werden. Mit dem AID-Kommando %HELP können die Meldungen in dieser Version noch mit dem alten Meldungsschlüssel *In* angesprochen werden.

1.4 Verwendete Darstellungsmittel

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

halbfett Hervorhebungen werden **halbfett** gekennzeichnet.



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

2 Metasyntax

Für die Darstellung der Kommandos 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 }
```

```
{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, steht für eine *vorqualifikation* (siehe %QUALIFY), ist der Operator für einen Adreßversatz oder 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 Voraussetzungen zum Testen

Als Arbeitsunterlage benötigen Sie vom zu testenden Programm das Assembler-Listing oder vergleichbare Informationen wie z.B. Locatormap und/oder Objectlisting des Compilers. Stehen Ihnen keine Struktur-Informationen zur Verfügung, können Sie das Binder-Listing hinzuziehen.

3.1 Übersetzen, Binden, Laden

Für das Testen auf Maschinencode-Ebene brauchen Sie beim Übersetzen, Binden und Laden keine speziellen Optionen bzw. Operanden anzugeben, um den in diesem Handbuch beschriebenen Funktionsumfang von AID voll ausnutzen zu können. Dann wird standardmäßig aus der vom Compiler erzeugten ESD (External Symbol Dictionary) bzw. ESV (External Symbol Vector) vom verwendeten Binder eine Objektstrukturliste/ein Externadreibuch erzeugt, die bei Verwendung der Standard-Optionen auch mitgeladen werden (siehe AID - Basishandbuch, Kapitel 4.1 [1]). Daraus kann AID Informationen über CSECTs und COMMONs entnehmen.

Wurden sie beim Binden mit TSOSLNK (SYMTEST=NO) oder mit BINDER (Anweisung SAVE-LLM, Operand SYMBOL-DICTIONARY=NO) nicht erzeugt oder wurden sie nicht mitgeladen, können Sie die folgenden Funktionen nicht ausführen:

- Ausgeben der Liste aller CSECTs des Benutzerprogramms (%D %SORTEDMAP oder %D %MAP)
- Ausgeben der maschinennahen Lokalisierungs-Information (%D %LOC (speicherref))
- Angeben einer CSECT/COM-Qualifikation in einer Speicherreferenz
- Ablaufüberwachung durch die Kommandos %CONTROLn und %TRACE, wenn sie implizit oder explizit auf eine CSECT/COMMON beschränkt sein sollen
- CSECT-relative Angaben bei %DISASSEMBLE, %FIND, %TRACE und in der STOP-Meldung
- Erzeugen von REPs für Änderungen

CSECTs, die umbenannt wurden, sprechen Sie in AID-Kommandos mit ihren neuen Namen an (siehe AID-Basishandbuch, Kapitel 4).



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

Beim Laden Ihres Programms mit ELDE oder Binden und Laden mit DBL (DLL bis BS2000 V9.5) brauchen Sie keine Operanden anzugeben.

3.2 Kommandos zu Beginn einer Testsitzung

AID-Kommandos verfügen **nicht** über eine SDF-Syntax. Das bedeutet:

- Operanden werden nicht über Menüs abgefragt.
- Im Fehlerfall gibt AID eine Fehlermeldung aus, führt aber keinen Korrekturdialog.

Falls Ihre Task gerade nicht im SDF-EXPERT-Modus ist, sollten Sie mit Kommando `MODIFY-SDF-OPTIONS GUIDANCE=EXPERT` in den EXPERT-Modus umschalten, wenn Sie AID-Kommandos eingeben wollen.

Um eine längere AID-Ausgabe mit der K2-Taste unterbrechen zu können, muß mit dem Kommando `/MODIFY-TERMINAL-OPTIONS OVERFLOW-CONTROL=USER-ACKNOWLEDGE` folgende Option eingestellt sein:

4 Maschinencode-spezifische Adressierung

In diesem Kapitel werden die Qualifikationen und Speicherreferenzen beschrieben, die Sie für das Testen auf Maschinencode-Ebene verwenden können. Eine allgemeine Beschreibung der Adressierung finden Sie im AID-Basishandbuch, Kapitel 7 [1]. Wenn Sie ein Speicherobjekt angeben, das nicht im aktuellen AID-Arbeitsbereich liegt oder dort nicht eindeutig ansprechbar ist, schreiben Sie vor die Speicherreferenz die Qualifikationen, die AID die Zuordnung ermöglichen. Auf alle Speicherreferenzen können Sie die im AID-Basishandbuch, Kapitel 7 [1] beschriebenen Operationen anwenden.

Wenn Sie Programme mit ESA-Befehlen auf ESA-Anlagen mit einer BS2000/OSD-BC-Version $\geq V1.0$ testen, können Sie die ALET- und SPID-Qualifikation verwenden. Mit den beiden Qualifikationen können Sie virtuelle Adressen in einem Datenraum ansprechen. Zum Speicherkonzept auf ESA-Anlagen siehe Handbuch „Makroaufrufe an den Ablaufteil“ [7].

4.1 Qualifikationen

Beim Testen auf Maschinencode-Ebene können Sie die Basisqualifikation, und als Bereichsqualifikationen die CTX-, L-, O-, C-, COM-, ALET- und SPID-Qualifikationen verwenden.

Qualifikationen müssen Sie immer in der Reihenfolge angeben, in der sie hier beschrieben sind. Sie werden durch Punkte getrennt. Zwischen der letzten Qualifikation und der anschließenden Adresse muß ebenfalls ein Punkt stehen. In der folgenden Übersicht ist dargestellt, wie Sie Qualifikationen einsetzen können:

$$\left[E = \left\{ \begin{array}{l} VM \\ Dn \end{array} \right\} \cdot \right] \left\{ \begin{array}{l} \left\{ \begin{array}{l} ALET = \left\{ \begin{array}{l} X'f \dots f' \\ \%nAR \\ \%nG \end{array} \right\} \\ SPID = X'f \dots f' \end{array} \right\} \\ \\ \left[CTX = \text{kontext} \cdot \right] \left[L = \text{ladeinheit} \cdot \right] \left[O = \text{objektmodul} \cdot \right] \left\{ \begin{array}{l} C = \text{csect} \\ COM = \text{common} \end{array} \right\} \end{array} \right\}$$

Basisqualifikation

Mit der Basisqualifikation vereinbaren Sie den AID-Arbeitsbereich. Sie legen fest, ob eine anschließende Adresse im virtuellen Speicher oder in einer Dump-Datei liegen soll.

$E=\{VM \mid Dn\}$

legt fest, ob AID auf den virtuellen Speicher (VM) oder auf eine Dump-Datei (Dn) zugreifen soll. Die E-Qualifikation ist im AID-Basishandbuch, Kapitel 7 [1] und bei %BASE beschrieben. Auf sie können unmittelbar Bereichsqualifikationen, eine virtuelle Adresse, ein Schlüsselwort oder eine komplexe Speicherreferenz folgen.

Bereichsqualifikationen

Mit einer Bereichsqualifikation bezeichnen Sie einen zusammenhängenden Teilbereich eines Programms. Endet ein Adreßoperand mit einer dieser Qualifikationen, begrenzen Sie die Wirkung eines Kommandos auf diesen Bereich.

CTX=kontext

bezeichnet einen Kontext (siehe AID Basishandbuch, Kapitel 7.1 [1]). Diese Qualifikation geht einer L-, O-, C- oder COM-Qualifikation voran. Nur im Kommando %QUALIFY kann ein Adreßoperand mit der CTX-Qualifikation enden. Sie brauchen diese Qualifikation, wenn Sie eine Übersetzungseinheit, eine CSECT oder einen COMMON ansprechen wollen, der oder die in mehreren Kontexten enthalten ist, und worin nicht die aktuelle Unterbrechungsstelle liegt.

kontext ist der im BIND-Makro explizit vergebene Name des Kontextes oder der implizit vergebene Name LOCAL#DEFAULT oder CTXPHASE. Auch mit dem DBL geladene Programme erhalten den standardmäßig vergebenen Kontextnamen LOCAL#DEFAULT. Wenn sie mit TSOSLNK gebunden wurden, heißt der Kontext CTXPHASE. Weitere Kontexte eines Programms können durch Anschließen an ein Shared-Code-Programm entstehen.

kontext ≤ 32stelliger Name eines Kontextes.

[L=ladeeinheit.] [O=objektmodul.] {C=csect | COM=common}

Die L- und/oder O-Qualifikation brauchen Sie nur, wenn Sie eine andere als die aktuelle CSECT oder den aktuellen COMMON ansprechen wollen und es im aktuellen Kontext mehrere CSECTs/COMMONs mit demselben Namen gibt.

Sie müssen nur die L- und/oder O-Qualifikationen angeben, die zur eindeutigen Ansprache erforderlich sind. Es muß eine C/COM-Qualifikation folgen.

Informationen über L-, O- und C/COM-Qualifikationen zu einer virtuellen Adresse erhalten Sie über

%DISPLAY %LOC

L=ladeeinheit

bezeichnet alle Bindelademodule bzw. einen Lademodul, der sich zum Zeitpunkt, an dem er angesprochen wird, im virtuellen Speicher oder entsprechend im Speicherabzug befinden muß.

ladeeinheit ≤ 8stelliger Name eines Bindemoduls (OM)

≤ 32stelliger Name eines Bindelademoduls (LLM).

O=objektmodul

bezeichnet einen Bindemodul (OM), wie er beim Übersetzen eingetragen wurde. Die *ladeeinheit*, in der sich der Bindemodul befindet, muß sich zum Zeitpunkt, an dem der Bindemodul angesprochen wird, im virtuellen Speicher oder entsprechend im Speicherabzug befinden.

objektmodul ≤ 8stelliger Name eines Bindemoduls.

C=csect

bezeichnet eine CSECT. Der Name der CSECT wurde im Quellprogramm vergeben oder mit LMS, TSOSLNK oder BINDER geändert (siehe AID Basishandbuch, Kapitel 4 [1]). *csect* ist immer der zuletzt vereinbarte Name. Sie können auch eine maskierte CSECT angeben. **Nicht** verwenden können Sie CSECT-Namen zur Adressierung, wenn vom Binder keine Objekt-Strukturliste erstellt wurde.

csect ≤ 32stelliger Name einer CSECT.

Unmittelbar im Anschluß an *C=* schreiben Sie *csect*, wenn der Name keine Sonderzeichen enthält. *C=N'...'* verwenden Sie, wenn er Sonderzeichen enthält.

Die Namen der CSECTs Ihres Testobjekts gibt AID Ihnen mit %DISPLAY %SORTEDMAP oder %DISPLAY %MAP aus.

Mit einer C-Qualifikation kann ein Adreßoperand enden. Die gesamte CSECT bezeichnen Sie damit im %DISPLAY, %CONTROLn, %FIND, %MOVE, %SET und %TRACE.

Die Anfangsadresse der CSECT bezeichnen Sie damit im %DISASSEMBLE und %INSERT.

Die C-Qualifikation kann auch als Speicherreferenz eingesetzt werden, da sie ein Adreß- und ein Längen-Attribut hat (siehe Abschnitt Speicherreferenzen).



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

COM=common

bezeichnet einen COMMON. Die COM-Qualifikation können Sie genau wie die C-Qualifikation verwenden. Bitte lesen Sie das dort nach.

ALET={X'f...f' | %nAR | %nG}

bezeichnet einen Datenraum über einen seiner ALETs. Die ALET-Qualifikation wird nur beim Testen von ESA-Programmen benötigt. Der ALET wird vom Makro ALESRV zurückgeliefert. Er kann als Sedezimal-Literal angegeben werden oder einem Zugriffsregister bzw. einem AID-Register entnommen werden.

Auf eine ALET-Qualifikation kann eine virtuelle Adresse oder eine komplexe Speicherreferenz ohne symbolische Komponenten folgen.

X'f...f' ist ein 8-stelliges Sedezimal-Literal

%nAR ist ein Zugriffsregister; $0 \leq n \leq 15$

%nG ist ein AID-Register, in das der ALET zwischengespeichert wurde;
 $0 \leq n \leq 15$

SPID=X'f...f'

bezeichnet einen Datenraum über seine SPID, ein 8-byte-langes Kennzeichen. Die SPID-Qualifikation wird nur beim Testen von ESA-Programmen benötigt. Die SPID wird vom Makro DSPSRV zurückgeliefert. Sie kann nur als Sedezimal-Literal angegeben werden.

Auf eine SPID-Qualifikation kann nur eine virtuelle Adresse oder eine komplexe Speicherreferenz ohne symbolische Komponenten folgen.

X'f...f' ist ein 16-stelliges Sedezimal-Literal.

Informationen über die ALET/SPID-Qualifikationen erhalten Sie, wenn Sie eingeben:
 %DISPLAY %DS[(ALET/SPID-qua)]

4.2 Speicherreferenzen

Beim Testen auf Maschinencode-Ebene bezeichnen Sie eine Speicherstelle mit einer virtuellen Adresse, durch eine komplexe Speicherreferenz, mit einer C/COM-Qualifikation oder einem Schlüsselwort.

Testen Sie ein Programm oder einen Programmteil, zu dem LSD-Sätze erzeugt wurden, können Sie in allen Operanden, in denen *kompl-speicherref* möglich ist, auch die im Quellprogramm definierten Daten- und Anweisungsnamen verwenden.

V'f...f' bezeichnet eine virtuelle Adresse. Sie hat implizit die Länge 4, den Speichertyp %X und damit den Ausgabety *Dump*.

f..f ist eine maximal 8stellige Sedezimal-Adresse, wobei *f* einen Wert von 0 - 9 und A - F annehmen kann.

Wenn Sie beim Testen von ESA-Programmen eine Adresse in einem Datenraum ansprechen wollen, müssen Sie vor der virtuellen Adresse eine ALET- oder SPID-Qualifikation angeben.

Ansonsten ist eine virtuelle Adresse im jeweiligen AID-Arbeitsbereich eindeutig und nur die Angabe einer Basisqualifikation sinnvoll. Die Angabe von Bereichs-Qualifikationen ist somit überflüssig und wird von AID ignoriert.

{C=csect | COM=common}

bezeichnet eine CSECT oder einen COMMON. Sie hat alle Attribute einer virtuellen Adresse und zusätzlich das Attribut Name. Deshalb kann die C/COM-Qualifikation auch als Anfangsadresse einer *kompl-speicherref* verwendet werden. Die Bereichsgrenzen der CSECT/COMMON können Sie nur durch indirekte Adressierung (->) überschreiten. Weitere Informationen finden Sie unter „Bereichsqualifikationen“ auf Seite 11.

schlüsselwort

bezeichnet je nach Operandentyp eine Speicherklasse, ein Programmregister, AID-Register, Systeminformationen, Durchlaufzähler oder den Adressierungsmodus (siehe AID-Basishandbuch, Kapitel 10 [1]). Jedes Schlüsselwort hat alle Attribute eines Speicherobjektes (AID-Basishandbuch, Kapitel 7.2.3 [1]). Die Bereichsgrenzen eines Schlüsselwortes können Sie nur durch indirekte Adressierung (->) überschreiten.

kompl-speicherref

Eine komplexe Speicherreferenz bezeichnet eine errechnete Adresse. Ohne entsprechende Typ- oder Längen-Modifikation hat sie die Länge 4, den Speichertyp %X und den Ausgabetypp Dump.

Beim Testen von Programmen, die **nicht** auf ESA-Anlagen laufen, können folgende Operationen in einer *kompl-speicherref* vorkommen (siehe AID-Basishandbuch, Kapitel 7 [1]):

- Adreßversatz (•)
- indirekte Adressierung (->)
- Typmodifikation (%T(datenname), %X, %C, %P, %D, %F, %A, %S, %SX)
- Längenmodifikation (%L(...), %L=(ausdruck), %Ln)
- Adreßselektion (%@(...))

Nach Adreßversatz oder indirekter Adressierung gehen impliziter Speichertyp und implizite Länge der Ausgangsadresse verloren und es gilt %XL4. Trotzdem bleiben die Bereichsgrenzen der Ausgangsadresse z.B. CSECT oder Schlüsselwort wirksam. Sie dürfen mit keinem Operanden in einer *kompl-speicherref* (Adreßversatz, Längen- oder Typmodifikation) überschritten werden, sonst gibt AID eine Fehlermeldung aus. Erst durch die Verbindung von Adreßselektor mit Pointeroperator/indirekter Adressierung wechseln Sie auf die Maschinencode-Ebene, auf der der Bereich den gesamten vom geladenen Programm belegten virtuellen Speicher umfaßt.

Beim Testen von Programmen auf ESA-Anlagen gibt es Unterschiede bezüglich der Operationen, die in *kompl-speicherref* vorkommen können:

- für Speicherobjekte im **Programmraum** können Sie symbolische und maschinennahe Adressierung mischen; die Möglichkeiten entsprechen denen für Programme auf Nicht-ESA-Anlagen.
- für Speicherobjekte im **Datenraum** können Sie die **symbolische** Adressierung **nicht** verwenden. Sie können **nur** maschinennahe Komponenten verwenden. Die ALET- oder SPID-Qualifikation wirkt sich **auf alle** Komponenten der komplexen Speicherreferenz aus. Nur in einer abschließenden Typmodifikation können Sie einen Namen verwenden. Ohne vorhergehende Bereichsqualifikation ergänzt AID nur die aktuelle Basisqualifikation.

```
%DISPLAY ALET=%2G.V'34'%T(NAME) entspricht
```

```
%DISPLAY E=VM.ALET=%2G.V'34'%T(E=VM.NAME)
```

Beispiele

Adressierung bzw. spezielle Operanden beim Testen von ESA-Programmen

```
1. %DISPLAY %ASC
   %ASC           = 01
```

Über das Schlüsselwort %ASC erfahren Sie, ob in die Adreßumsetzung Zugriffsregister einbezogen und damit Datenräume adressiert werden. Der Wert 01 bedeutet, daß Zugriffsregister einbezogen werden.

```
2. %DISPLAY %DS
   SPID           ALET           SIZE
0001003300000031 00000000    0
0000000080000800 0001001B   409600
0000000080000800 0001001C   409600
0000000080000900 00010011   131072
0000000080000900 00010012   131072
```

Alle Angaben zu den aktiven Datenräumen werden ausgegeben.

3. %DISPLAY %AR
%AR (0 : 15)
(0) 00000000 (1) 000000FF (2) 00000000 (3) 00000090 (4) 00000094
(5) 000000B4 (6) 92020711 (7) 09252742 (8) 00000000 (9) 00000000
(10) 000000AC (11) 000000B0 (12) 000000B4 (13) 92020711 (14) 09252742
(15) 00000000

Alle Zugriffsregister werden ausgegeben. Die Zugriffsregister mit einem Wert < 0 enthalten einen ALET.

4. %FIND C'01' IN ALET=X'00010003'.V'0'%L100
ABSOLUT +0000000F=0000000F : F0F1405E2 C5C9E3C5 40F0F0F0 01 SEITE 000
%FIND C'01' IN ALET=%8AR.V'0'%L100
ABSOLUT +0000000F=0000000F : F0F1405E2 C5C9E3C5 40F0F0F0 01 SEITE 000
%FIND C'01' IN SPID=X'0000000080000200'.V'0'%L100
ABSOLUT +0000000F=0000000F : F0F1405E2 C5C9E3C5 40F0F0F0 01 SEITE 000

Drei Möglichkeiten dieselbe Stelle im Datenraum zu adressieren.

5 AID-Kommandos

CHECK

ALL Vor der Ausführung eines %MOVE oder %SET führt AID folgenden Änderungsdialog:

```

OLD CONTENT:
AAAAAAA
NEW CONTENT:
BBBBBBB
% AID0274 CHANGE DESIRED? REPLY (Y = YES; N = NO) ?

```

N

AID0342 NOTHING CHANGED

Nach der Eingabe **Y** wird der alte Speicherinhalt ohne weitere Meldung überschrieben. In Prozeduren im Stapelbetrieb kann AID keinen Dialog führen und nimmt immer **Y** an.

Der alte bzw. neue Inhalt wird auf SYSOUT ausgegeben. Wird SYSOUT umgewiesen, dann sind diese Ausgaben am Terminal nicht zu sehen. Das gilt auch, wenn das %MOVE- bzw. %SET-Kommando mit dem CMD-Makro gegeben wurde und eine Ausgabe auf SYSOUT vereinbart ist. Dagegen geht die Meldung AID0274 und gegebenenfalls AID0342 immer auf das Medium Terminal.

NO %MOVE und %SET werden ohne Änderungsdialog ausgeführt.

Wird der *CHECK*-Operand ohne Wertangabe eingegeben, setzt AID den Standardwert NO ein.

REP

YES Zu Änderungen im Speicher mit %MOVE werden LMS-Korrekturanweisungen im SDF-Format (REPs) erstellt. Wenn die Objekt-Strukturliste nicht zur Verfügung steht, erstellt AID keine Korrektursätze und gibt eine Fehlermeldung aus.

AID hinterlegt die Korrekturen in einer Datei mit dem Linknamen F6. Für den LMS-Lauf muß dann noch die MODIFY-ELEMENT-Anweisung eingefügt werden. Achten Sie deshalb darauf, daß Sie in die Datei mit dem Linknamen F6 keine anderen Ausgaben schreiben lassen.

Ist keine Datei mit dem Linknamen F6 angemeldet (siehe %OUTFILE), legt AID die Datei AID.OUTFILE.F6 an, in die es dann den REP schreibt. Benutzerspezifische REP-Dateien müssen mit Zugriffsmethode SAM angelegt sein. Von AID angelegte REP-Dateien werden ebenfalls mit Zugriffsmethode SAM, Satzformat V und Eröffnungsart EXTEND angelegt. Die Datei bleibt geöffnet, bis sie mit %OUTFILE geschlossen wird oder bis /LOGOFF bzw. /EXIT-JOB.

NO Es werden keine REPs erstellt.

Wird der *REP*-Operand ohne Wertangabe eingegeben, setzt AID den Standardwert NO ein. Der *REP*-Operand des %MOVE-Kommandos kann die mit %AID getroffene Vereinbarung für dieses eine %MOVE-Kommando ersetzen. Für nachfolgende %MOVE-Kommandos ohne *REP*-Operand gilt dann wieder die Vereinbarung mit %AID.

SYMCHARS

STD Der Bindestrich (-) wird als alphanumerisches Zeichen interpretiert und kann somit in Programm-, Daten- und Anweisungsnamen verwendet werden. Er wird nur dann als Minus-Zeichen interpretiert, wenn vor dem Bindestrich ein Leerzeichen steht.

NOSTD

Der Bindestrich (-) wird immer als Minus-Zeichen interpretiert und kann in Namen nicht verwendet werden.

Wird der *SYMCHARS*-Operand ohne Wertangabe eingegeben, setzt AID den Standardwert (STD) ein.

OV

YES müssen Sie angeben, wenn Sie ein Programm mit Überlagerungsstruktur (Overlay) testen. AID überprüft dann jedesmal, ob der angesprochene Programmteil eventuell aus einem nachgeladenen Segment stammt.

NO AID geht davon aus, daß das zu testende Programm ohne Überlagerungsstruktur gebunden ist. AID benutzt die einmal geladenen LSD-Sätze, ohne zu prüfen, ob der angesprochene Programmteil in einem nachgeladenen Segment liegt.

Wird der *OV*-Operand ohne Wertangabe eingegeben, setzt AID den Standardwert (NO) ein.

LOW

ON Kleinbuchstaben in Character-Literalen und in Programm-, Daten- und Anweisungsnamen werden nicht in Großbuchstaben konvertiert.

OFF Alle Kleinbuchstaben aus Benutzereingaben werden in Großbuchstaben umgesetzt.

Wenn in einer Testsitzung noch kein *LOW*-Operand eingegeben wurde, gilt die Voreinstellung OFF. Wird der *LOW*-Operand ohne Wertangabe eingegeben, setzt AID den Standardwert (ON) ein. Um wieder die Umsetzung in Großbuchstaben einzuschalten, müssen Sie LOW=OFF eingeben.

DELIM

C'x' | 'x'C | 'x'

Mit diesem Operanden legen Sie ein Zeichen als linke und rechte Begrenzung (Delimiter) für die AID-Ausgabe symbolischer Daten vom Typ Character mit %DISPLAY fest.

| Der Standard-Begrenzer ist der senkrechte Strich.

Wird der *DELIM*-Operand ohne Wertangabe eingegeben, setzt AID den Standardwert (|) ein.

LANG

D AID gibt die Informationen, die mit %HELP angefordert wurden, in Deutsch aus.

E AID gibt die Informationen, die mit %HELP angefordert wurden, in Englisch aus.

Wird der *LANG*-Operand ohne Wertangabe eingegeben, setzt AID den Standardwert (D) ein. Mit dem SDF-Kommando `MODIFY-MSG-ATTRIBUTES TASK-LANGUAGE=D` bekommen Sie auch die AID-Meldungen auf Deutsch. Der Änderungsdialog (siehe CHECK-Operand) wird dadurch nicht beeinflusst.

%AINT

Mit %AINT legen Sie fest, ob AID bei indirekter Adressierung mit 24-Bit-Adressen oder mit 31-Bit-Adressen arbeiten soll. Die Adresse vor dem Pointer-Operator (->) besteht für AID dann entsprechend aus 24 oder 31 Bits.

Der Adressierungsmodus des Testobjekts wird damit nicht beeinflusst.

- Mit *aid-mode* legen Sie die Adreßinterpretation für indirekte Adressierung innerhalb eines AID-Arbeitsbereiches fest.

Kommando	Operand
%AINT	[aid-mode] [...]

%AINT ist nur für den Test von Programmen auf XS-Rechnern sinnvoll.

Standardmäßig interpretiert AID indirekte Adreßangaben entsprechend dem aktuellen Adressierungsmodus des Testobjekts. Der aktuelle Adressierungsmodus ist der, der im Systeminformationsfeld %AMODE eingetragen ist. Solange ein %AINT gilt, unterbleibt diese Anpassung und indirekte Adressen werden gemäß der Vereinbarungen im %AINT interpretiert. Den aktuellen Adressierungsmodus (%AMODE) des Testobjektes können Sie mit %DISPLAY abfragen und mit %MOVE verändern.

Ohne die Angabe einer Qualifikation gilt %AINT für AID-Kommandos, die indirekte Adressen des aktuellen AID-Arbeitsbereiches ansprechen oder benutzen.

Ein %AINT ohne Operanden schaltet zurück auf die Standard-Adreßinterpretation. Dasselbe bewirkt ein %AINT mit Basisqualifikation und ohne Angabe zur Adreßinterpretation.

%AINT verändert den Programmzustand nicht.

aid-mode

legt für den aktuellen oder den mit der angegebenen Basisqualifikation bezeichneten AID-Arbeitsbereich fest, wie indirekte Adressen in später folgenden AID-Kommandos interpretiert werden sollen.

Geben Sie ein Schlüsselwort für die Adreßinterpretation und keine Qualifikation an, so gilt das %AINT-Kommando für die Bearbeitung des aktuellen AID-Arbeitsbereiches.

Geben Sie eine Basisqualifikation und kein Schlüsselwort für die Adreßinterpretation an, wird für den entsprechenden AID-Arbeitsbereich die AID-Standard-Adreßinterpretation wirksam.

aid-mode-OPERAND -----

$$[\cdot] [E = \left\{ \begin{array}{l} \text{VM} \\ \text{Dn} \end{array} \right\} [\cdot]] [\left\{ \begin{array}{l} \%M[ODE]31 \\ \%M[ODE]24 \end{array} \right\}]$$

- Steht der Punkt an führender Stelle, ist er das Kennzeichen für eine *vorqualifikation*. Sie muß mit einem vorhergehenden %QUALIFY-Kommando definiert worden sein. Zwischen Basisqualifikation und dem Schlüsselwort zur Adreßinterpretation muß ein Punkt gesetzt werden. Geben Sie nur eine Basisqualifikation an, so dürfen Sie keinen abschließenden Punkt eingeben.

E={VM | Dn}

geben Sie an, wenn die Umschaltung der Adreßinterpretation nicht für den aktuellen AID-Arbeitsbereich gelten soll. Geben Sie nur eine Basisqualifikation an, so gilt für den damit angesprochenen Bereich wieder die Standard-Adreßinterpretation.

{%M[ODE]31 | %M[ODE]24}

Schlüsselwort, das angibt, wie viele Bits bei indirekter Adressierung in AID-Kommandos berücksichtigt werden sollen.

%M[ODE]31	31-Bit-Adressierung
%M[ODE]24	24-Bit-Adressierung

Beispiele

Die Adresse V'100' hat den Inhalt: 1200000C

Register 5 hat den Inhalt: 010001A0

1. %AINT %MODE24

```
%DISPLAY V'100' ->
```

```
%MOVE %5-> INTO %5G
```

Mit %AINT stellen Sie auf 24-Bit-Adreßinterpretation um. Die Umstellung gilt für den aktuellen AID-Arbeitsbereich.

Der %DISPLAY gibt 4 Bytes ab Adresse V'00000C' aus.

Der %MOVE überträgt 4 Bytes ab Adresse V'0001A0' in das AID-Register 5.

2. %AINT %MODE31

```
%DISPLAY V'100' ->
```

```
%MOVE %5-> INTO %5G
```

Sie stellen die Adreßinterpretation für den aktuellen AID-Arbeitsbereich auf 31-Bit-Interpretation um.

Der %DISPLAY gibt 4 Bytes ab Adresse V'1200000C' aus.

Der %MOVE überträgt 4 Bytes ab Adresse V'010001A0' in das AID-Register 5.

Beispiele

Die Adresse V'100' hat den Inhalt: 1200000C

Register 5 hat den Inhalt: 010001A0

1. %AINT %MODE24

```
%DISPLAY V'100' ->
```

```
%MOVE %5-> INTO %5G
```

Mit %AINT stellen Sie auf 24-Bit-Adreßinterpretation um. Die Umstellung gilt für den aktuellen AID-Arbeitsbereich.

Der %DISPLAY gibt 4 Bytes ab Adresse V'00000C' aus.

Der %MOVE überträgt 4 Bytes ab Adresse V'0001A0' in das AID-Register 5.

2. %AINT %MODE31

```
%DISPLAY V'100' ->
```

```
%MOVE %5-> INTO %5G
```

Sie stellen die Adreßinterpretation für den aktuellen AID-Arbeitsbereich auf 31-Bit-Interpretation um.

Der %DISPLAY gibt 4 Bytes ab Adresse V'1200000C' aus.

Der %MOVE überträgt 4 Bytes ab Adresse V'010001A0' in das AID-Register 5.

%BASE

Mit %BASE legen Sie die Basisqualifikation fest. Alle nachfolgend eingegebenen Speicherreferenzen ohne eigene Basisqualifikation übernehmen die mit %BASE vereinbarte. Zugleich legen Sie damit fest, wo sich der AID-Arbeitsbereich befinden soll.

- Mit *basis* bezeichnen Sie den virtuellen Speicherbereich des geladenen Programms oder einen Speicherabzug in einer Dump-Datei.

Kommando	Operand
%BASE	[basis]

Beim Testen auf Maschinencode-Ebene entspricht der AID-Arbeitsbereich dem nicht privilegierten Teil im virtuellen Speicher, den das geladene Programm mit allen konnektierten Subsystemen belegt (AID-Standard-Arbeitsbereich) oder dem entsprechenden Bereich in einem Speicherabzug. Geben Sie in einer Testsitzung kein %BASE oder geben Sie ein %BASE ohne Operanden ein, gilt die Basisqualifikation E=VM (Standardwert).

Ein %BASE gilt bis zum nächsten %BASE, bis /LOGOFF bzw. /EXIT-JOB oder bis zum Schließen der Dump-Datei (siehe %DUMPFIL), die als Basisqualifikation vereinbart war.

Unmittelbar bei der Eingabe werden alle Speicherreferenzen in einem Kommando, auch in einem Subkommando mit der aktuellen Basisqualifikation ergänzt. Keine Auswirkung hat %BASE auf Subkommandos, die vorher vereinbart wurden.

%BASE darf nur als Einzelkommando eingegeben werden; es darf nicht in einer Kommando- oder einem Subkommando stehen.

%BASE verändert den Programmzustand nicht.

basis

legt die Basisqualifikation fest. Alle nachfolgend eingegebenen Speicherreferenzen ohne eigene Basisqualifikation übernehmen die mit %BASE vereinbarte.

basis-OPERAND - - - - -

$$E = \left\{ \begin{array}{l} VM \\ Dn \end{array} \right\}$$

- - - - -

E=VM Der virtuelle Speicherbereich des geladenen Programms ist als Basisqualifikation vereinbart. VM ist der Standardwert

E=Dn Ein Speicherabzug in einer Dump-Datei mit dem Linknamen Dn ist als Basisqualifikation vereinbart.

n ist eine Zahl mit einem Wert $0 \leq n \leq 7$.

Bevor Sie eine Dump-Datei als Basisqualifikation vereinbaren, müssen Sie mit %DUMPFILe die entsprechende Dump-Datei einem Linknamen zuweisen und öffnen.

%CONTINUE

Mit %CONTINUE starten Sie das geladene Programm oder setzen es an der unterbrochenen Stelle fort. Im Gegensatz zu %RESUME wird ein aktiver %TRACE durch %CONTINUE nicht beendet, sondern entsprechend den Vereinbarungen fortgesetzt.

Kommando	Operand
----------	---------

%CONT[INUE]

Ein %TRACE ist aktiv, sobald er eingegeben wurde. In folgenden Fällen ist der %TRACE nur unterbrochen und kann mit %CONTINUE fortgesetzt werden:

1. Ein Subkommando wurde ausgeführt, weil eine Überwachungsbedingung aus einem %CONTROLn, %INSERT oder %ON zutraf, und das Subkommando enthielt ein %STOP.
2. Ein %INSERT endet mit einer Programmunterbrechung, weil der *steuerung*-Operand K oder S lautet.
3. Die K2-Taste wurde gedrückt (siehe Abschnitt „Kommandos zu Beginn einer Testsitzung“ auf Seite 8).

Steht in einem Subkommando nur das Kommando %CONTINUE, wird nur der Durchlaufzähler erhöht.

Steht %CONTINUE in einer Kommandofolge oder in einem Subkommando, werden nachfolgende Kommandos nicht mehr ausgeführt.

%CONTINUE verändert den Programmzustand.

%CONTROLn

Mit %CONTROLn können Sie nacheinander bis zu sieben Ablaufüberwachungs-Funktionen vereinbaren, die dann gleichzeitig wirken. Es gibt %CONTROL1 bis %CONTROL7.

- Mit *kriterium* wählen Sie verschiedene Typen von Maschinenbefehlen aus. Steht ein Befehl des ausgewählten Typs zur Ausführung an, unterbricht AID das Programm und bearbeitet *subkdo*.
- Mit *control-bereich* legen Sie den Programmbereich fest, in dem *kriterium* überwacht werden soll.
- Mit *subkdo* definieren Sie ein Kommando oder eine Kommandofolge und eventuell eine Bedingung. Bei zutreffendem *kriterium* und erfüllter Bedingung wird *subkdo* ausgeführt. *subkdo* muß angegeben werden.

Kommando	Operand		
%[CONTROL]n	kriterium	[IN control-bereich]	<subkdo>

Ein %CONTROLn auf Maschinencode-Ebene kann **nicht gleichzeitig** mit einem *write-ereignis* des %ON angemeldet sein.

Mehrere %CONTROLn mit unterschiedlichen Nummern beeinflussen einander nicht, so daß Sie mehrere Kommandos mit demselben *kriterium* für verschiedene Bereiche oder mit unterschiedlichen *kriterien* für denselben Bereich aktivieren können. Treffen an einem Maschinenbefehl mehrere %CONTROLn zusammen, so werden die zugehörigen Subkommandos in der Reihenfolge %C1 bis %C7 bearbeitet.

%CONTROLn **wirkt** beim Testen auf Maschinencode-Ebene **anders** als beim Testen auf symbolischer Ebene:

Auch außerhalb von *control-bereich* wird hier *kriterium* **überwacht**, aber *subkdo* wird nur innerhalb von *control-bereich* ausgeführt. Der **zusätzliche** Überwachungsaufwand verlangsamt den Programmablauf. Deshalb empfiehlt es sich, für lange Befehlsstrecken das %CONTROLn-Kommando gezielt an Testpunkten als Subkommando des %INSERT einzusetzen und nach der Ausführung wieder zu löschen (siehe %INSERT und AID-Basis-handbuch, Kapitel 6 [1]).

Der einzelne Operandenwert eines %CONTROLn gilt so lange, bis Sie ihn durch neue Angaben in einem späteren %CONTROLn mit derselben Nummer überschreiben, ihn löschen oder bis zum Programmende.

Mit %REMOVE löschen Sie eine einzelne oder alle aktiven %CONTROLn-Vereinbarungen.

Für %CONTROLn muß die Basisqualifikation E=VM eingestellt sein (siehe %BASE) oder explizit angegeben werden.

%CONTROLn verändert den Programmzustand nicht.

kriterium

Schlüsselwort, das den Typ der Maschinenbefehle festlegt, vor deren Ausführung AID *subkdo* bearbeiten soll.

Da der **Standardwert** für *kriterium* das **symbolische** %STMT ist, **muß** für das maschinen-nahe Testen **immer** ein *kriterium* angegeben werden, falls nicht noch aus einem vorhergehenden %CONTROLn ein zulässiges *kriterium* gilt.

<i>kriterium</i>	Bearbeitung von <i>subkdo</i> vor
%INSTR	jedem Maschinenbefehl, der zur Ausführung ansteht
%B	jedem Verzweigungsbefehl, der zur Ausführung ansteht (das sind die Maschinenbefehle BAL, BALR, BAS, BASSM, BASR, BC, BCR, BCT, BCTR, BSM, BXH und BXLE)
%BAL	jedem Unterprogrammaufruf, der ansteht (durch die Maschinenbefehle BAL, BALR, BAS, BASSM und BASR)

control-bereich

legt den Programmbereich fest, in dem die Überwachungsfunktion wirksam wird. Beim Verlassen des festgelegten Programmbereichs läuft die Überwachung von *kriterium* weiter, aber *subkdo* wird nicht bearbeitet. Steht wieder ein Maschinenbefehl zur Ausführung an, der im zu überwachenden Programmbereich liegt, wird auch *subkdo* wieder bearbeitet.

Eine *control-bereich*-Definition gilt bis zum nächsten %CONTROLn derselben Nummer mit neuer Definition, bis zum entsprechenden %REMOVE oder bis zum Programmende. Ein %CONTROLn ohne eigenen *control-bereich*-Operanden übernimmt eine wirksame Bereichsdefinition. Ein wirksamer *control-bereich* muß in einem %CONTROLn mit derselben Nummer definiert sein, und die aktuelle Unterbrechungsstelle muß innerhalb dieses Bereichs liegen. Gibt es keine wirksame Bereichsdefinition, so umfaßt der *control-bereich* standardmäßig das gesamte Benutzerprogramm.

control-bereich-OPERAND - - - - -

IN [•][qua•] { C=csect | COM=common
(V'f...f': V'f...f')
schlüsselwort }

- - - - -

- Steht der Punkt an führender Stelle, ist er das Kennzeichen für eine *vorqualifikation*. Sie muß mit einem vorhergehenden %QUALIFY definiert worden sein. Aufeinanderfolgende Qualifikationen werden durch einen Punkt getrennt. Außerdem muß zwischen der letzten Qualifikation und dem anschließenden Operandenteil ein Punkt stehen.
- qua Eine oder mehrere Qualifikationen geben Sie an, wenn Sie ein Speicherobjekt ansprechen wollen, das nicht im aktuellen AID-Arbeitsbereich liegt oder darin nicht eindeutig ist. Sie geben nur die Qualifikationen an, die zur eindeutigen Ansprache genügen.
- E=VM
Da *control-bereich* nur im virtuellen Speicher des geladenen Programms liegen kann, geben Sie *E=VM* nur an, wenn als aktuelle Basisqualifikation eine Dump-Datei vereinbart ist (siehe %BASE).
- CTX=kontext
geben Sie nur an, um eine CSECT oder einen COMMON im Programmsystem eindeutig ansprechen zu können.
- [L=ladeeinheit.][O=objektmodul.]
geben Sie nur an, wenn Sie eine CSECT oder einen COMMON ansprechen wollen, die im aktuellen Kontext ohne diese Qualifikationen nicht eindeutig adressierbar sind. Sie geben nur die Qualifikationen an, die zur eindeutigen Ansprache nötig sind.
- {C=csect | COM=common}
Der *control-bereich* umfaßt die gesamte angegebene CSECT oder den COMMON.
- (V'f...f' : V'f...f')
Der *control-bereich* wird durch die Angabe einer virtuellen Anfangs- und Endadresse festgelegt.
Die Adressen müssen im ausführbaren Teil eines geladenen Programms liegen.
Anfangsadresse muß \leq Endadresse sein.

schlüsselwort

control-bereich umfaßt den Speicherbereich, der durch eines der folgenden Schlüsselwörter (siehe AID-Basishandbuch, Kapitel 9 [1]) angesprochen wird. Bei XS- bzw. ESA-Anlagen bezeichnen Sie mit %CLASS6 und %CLASS6BELOW denselben Adreßraum.

%CLASS6	Klasse-6-Speicher
%CLASS6BELOW	Klasse-6-Speicher, unterhalb der 16MB-Grenze
%CLASS6ABOVE	Klasse-6-Speicher, oberhalb der 16MB-Grenze

subkdo

wird immer dann bearbeitet, wenn im *control-bereich* ein Maschinenbefehl zur Ausführung ansteht, der *kriterium* entspricht. *subkdo* wird vor der Ausführung des Befehls bearbeitet. *subkdo* muß angegeben werden, da AID im %CONTROLn kein <%STOP> einsetzt.

Vollständig beschrieben finden Sie *subkdo* im AID Basishandbuch, Kapitel 6 [1].

subkdo-OPERAND

```
<[subkdoname:] [(bedingung):] [ { AID-kommando } { BS2000-kommando } { ; ... } ]>
```

Ein Subkommando kann einen Namen, eine Bedingung und einen Kommandoteil enthalten. Zu jedem Kommando gehört ein Durchlaufzähler. Der Kommandoteil kann aus einem einzelnen Kommando oder einer Kommandofolge bestehen, er kann AID- und BS2000-Kommandos und Kommentare enthalten.

Wenn das Subkommando nur aus einem Namen oder einer Bedingung besteht, aber keinen Kommandoteil hat, erhöht AID beim Erreichen einer Anweisung vom Typ *kriterium* nur den Durchlaufzähler.

Im *subkdo* eines %CONTROLn sind zusätzlich zu den Kommandos, die in allen Subkommandos nicht zugelassen sind, die AID-Kommandos %CONTROLn, %INSERT und %ON nicht erlaubt.

Die Kommandos in einem *subkdo* werden nacheinander ausgeführt. Danach wird das Programm fortgesetzt. Die Kommandos zur Ablaufsteuerung verändern auch in einem Subkommando sofort den Programmzustand. Sie brechen *subkdo* ab und starten das Programm (%CONTINUE, %RESUME, %TRACE) oder halten es an (%STOP). Sie sind nur als letztes Kommando in einem *subkdo* sinnvoll, da nachfolgende Kommandos im *subkdo* nicht mehr ausgeführt werden. Auch ein Löschen des gerade aktiven Subkommandos mit %REMOVE ist nur als letztes Kommando in *subkdo* sinnvoll.

Beispiele

1. `%C1 %INSTR <C1_ZAEHLER: %CONT>`
Vor jedem Befehl wird der Zähler des Subkommandos C1_ZAEHLER um eins erhöht und danach das Programm fortgesetzt.
2. `%C1 %B IN C=M1BS <%D %PC->%XL8>`
In der CSECT M1BS werden bei jedem Verzweigungsbefehl 8 Bytes des Programmcodes an der Unterbrechungsstelle ausgegeben (%PC->). Danach wird das Programm fortgesetzt, ein eventuell aktiver %TRACE wird weitergeführt. Der Programmablauf wird allerdings auch außerhalb von M1BS überwacht und verlangsamt sich wesentlich.
3. `%INSERT V'B40' <%REM %C5>`
`%INSERT V'A38' <%C5 %B IN (V'A38':V'B40') <%D V'798'>>`
Von Adresse V'A38' bis V'B40' sollen bei jedem Verzweigungsbefehl 4 Bytes ab der Adresse V'798' ausgegeben werden. Das Kommando wird jedoch erst aktiv, wenn der Programmablauf an die Adresse V'A38' (%INSERT) kommt, und bei Ende des zu überwachenden Bereichs wird der %CONTROL5 wieder gelöscht. Der Programmablauf wird so außerhalb des Bereichs nicht verlangsamt.
4. `%C2 %BAL <(%PC->%L2 EQ X'05EF'): %D %PC, %15>`
Alle BALR 14,15 werden protokolliert und zwar sowohl die Absprungstelle (%PC) als auch die Zieladresse (%15).

%DISASSEMBLE

Mit %DISASSEMBLE können Sie Speicherinhalte in symbolische Assembler-Notation rückübersetzen und ausgeben lassen.

- Mit *anzahl* legen Sie fest, wieviele Ausgabezeilen mit interpretiertem Speicherinhalt ausgegeben werden sollen.
- Mit *start* bestimmen Sie die Adresse, bei der AID mit der Rückübersetzung beginnen soll.

Kommando	Operand
<pre>{%DISASSEMBLE} {%DA}</pre>	<pre>[anzahl] [FROM start]</pre>

Die Rückübersetzung des Speicherinhalts beginnt mit dem ersten Byte. Für Speicherinhalt, der nicht als Befehl interpretiert werden kann, wird eine Ausgabezeile erzeugt, die die sechszahlige Darstellung des Speicherinhalts und den Hinweis INVALID OP CODE enthält. Die Suche nach gültigen Befehls-codes geht dann in 2-Byte-Schritten vorwärts.

Mit einem %DISASSEMBLE ohne *start*-Operanden können Sie ein vorher gegebenes %DISASSEMBLE-Kommando solange fortsetzen, bis Sie mit einem BS2000- oder AID-Kommando (/START-PROGRAM, /LOAD-PROGRAM, %BASE) das Testobjekt wechseln oder einen neuen Operandenwert vereinbaren. AID setzt die Rückübersetzung an der Speicheradresse fort, die an die Adresse anschließt, die mit dem vorhergehenden %DISASSEMBLE-Kommando zuletzt bearbeitet wurde. Ist auch *anzahl* nicht angegeben, so erzeugt AID dieselbe Anzahl von Ausgabezeilen wie bisher vereinbart.

Haben Sie in einer Testsitzung noch kein %DISASSEMBLE-Kommando gegeben, oder haben Sie das Testobjekt gewechselt und geben nun im %DISASSEMBLE-Kommando keine aktuellen Werte für einen oder beide Operanden an, dann arbeitet AID mit den Standardwerten 10 für *anzahl* und V'0' für *start*. Wurde das Programm nicht ab V'0' geladen, muß *start* angegeben werden.

Mit dem %OUT-Kommando können Sie steuern, wie die aufbereitete Speicherinformation dargestellt wird und ob sie auf SYSOUT, SYSLST oder in eine katalogisierte Datei ausgegeben werden soll. Der Aufbau der möglichen Ausgabezeilen ist im Anschluß an die Beschreibung des *start*-Operanden nachzulesen.

%DISASSEMBLE verändert den Programmzustand nicht.

anzahl

gibt an, wieviele Assembler-Befehle ausgegeben werden sollen. Wird für *anzahl* kein Wert angegeben und ist aus einem vorherigen %DISASSEMBLE auch kein Wert gültig, so setzt AID den Standardwert 10 ein.

anzahl ist eine Ganzzahl mit einem Wert:
 $1 \leq \textit{anzahl} \leq 2^{31}-1$

start

legt die Adresse fest, an der die Rückübersetzung von Speicherinhalt in Assembler-Befehle beginnen soll. Wird *start* nicht angegeben, so setzt AID beim ersten %DA nach dem Laden des Programms die Adresse V'0' ein. Bei jedem weiteren %DA wird hinter dem zuletzt rückübersetzten Assembler-Befehl fortgesetzt.

```

start-OPERAND - - - - -
FROM [•][qua•]{
  { C=csect | COM=common
    V'f...f'
  }
  { schlüsselwort
    kompl-speicherref
  }
}

```

- Steht der Punkt an führender Stelle, ist er das Kennzeichen für eine *vorqualifikation*. Sie muß mit einem vorhergehenden %QUALIFY definiert worden sein. Aufeinanderfolgende Qualifikationen werden durch einen Punkt getrennt. Außerdem muß zwischen der letzten *qualifikation* und dem anschließenden Operandenteil ein Punkt stehen.
- qua Eine oder mehrere Qualifikationen geben Sie an, wenn Sie ein Speicherobjekt ansprechen wollen, das nicht im aktuellen AID-Arbeitsbereich liegt oder darin nicht eindeutig ist. Sie geben nur die Qualifikationen an, die zur eindeutigen Ansprache genügen.
 - E={VM | Dn}
 - geben Sie nur an, wenn für *start* die aktuelle Basisqualifikation nicht gelten soll (siehe %BASE).
 - CTX=kontext
 - geben Sie nur an um eine CSECT oder einen COMMON im Programmsystem eindeutig ansprechen zu können.

[L=ladeeinheit.][O=objektmodul.]

geben Sie nur an, wenn Sie eine CSECT oder einen COMMON ansprechen wollen, die im aktuellen Kontext ohne diese Qualifikationen nicht eindeutig adressierbar sind. Sie geben nur die Qualifikationen an, die zur eindeutigen Ansprache nötig sind.

{C=csect | COM=common}

damit legen Sie *start* auf die Anfangsadresse der angegebenen CSECT oder des COMMON.

V'f...f' ist eine virtuelle Adresse, die im ausführbaren Teil des Programms liegen und die Anfangsadresse eines Maschinenbefehls sein muß. Sie erhalten andernfalls eine unsinnige Disassemblierung.

schlüsselwort

damit legen Sie *start* auf die Anfangsadresse einer Speicherklasse (siehe AID-Basis-Handbuch, Kapitel 10.5 [1]). Bei XS- bzw. ESA-Anlagen bezeichnen Sie mit %CLASS6 und %CLASS6BELOW denselben Adreßraum.

%CLASS6	Klasse-6-Speicher
%CLASS6BELOW	Klasse-6-Speicher, unterhalb der 16MB-Grenze
%CLASS6ABOVE	Klasse-6-Speicher, oberhalb der 16MB-Grenze

kompl-speicherref

sollte die Anfangsadresse eines Maschinenbefehls ergeben, andernfalls erhalten Sie eine unsinnige Disassemblierung. Folgende Operationen können in *kompl-speicherref* vorkommen (siehe AID-Basis-Handbuch, Kapitel 7.2.4 [1] und wegen Einschränkungen beim Testen von ESA-Programmen vorliegendes Handbuch, Kapitel 4, Maschinencode-spezifische Adressierung):

- Adreßversatz (•)
- indirekte Adressierung (->)
- Typmodifikation (%A, %S, %SX)
- Längenmodifikation (%L(...), %L=(ausdruck), %Ln)
- Adreßselektion (%@(...))

Mit der indirekten Adressierung (->) können Sie die Adresse, die im Befehlszähler oder in einem Register hinterlegt ist, als *start* verwenden.

Beispiel: %PC-> legt *start* auf die Unterbrechungsstelle.

Ausgabe des %DISASSEMBLE-Protokolls

Das %DISASSEMBLE-Protokoll wird standardmäßig mit Zusatzinformationen über SYS-OUT ausgegeben (T=MAX). Mit %OUT können Sie die Ausgabe-Medien wählen und festlegen, ob AID Zusatzinformationen ausgeben soll oder nicht.

Folgende Elemente enthält eine %DA-Ausgabezeile, wenn der Standardwert T=MAX gilt:

- CSECT-relative Speicheradresse≥
- in symbolische Assembler-Notation rückübersetzter Speicherinhalt, wobei Distanzen im Gegensatz zum Assembler-Format als Sedezimalzahlen dargestellt werden.
- für Speicherinhalt, der nicht mit einem gültigen Operationscode beginnt, wird ein Assembler-Befehl DC im Sedezimal-Format mit der Länge von 2 Bytes aufgebaut, dem der Hinweis INVALID OP CODE folgt.
- sedezimale Darstellung des Speicherinhalts (Maschinencode).

Der CSECT-Name steht in einer separaten Zeile, wenn er länger als 8 Stellen ist. Die dann folgende Zeile beginnt mit dem auf 7 Stellen gekürzten und mit Stern (*) abgeschlossenen CSECT-Namen und entspricht sonst dem vorher beschriebenen Aufbau.≥

Beispiel zum Zeilenaufbau (T=MAX) für eine CSECT mit kurzem Namen.

```
MOBS+A30      LM    R0,R1,C0(R11)          98 01 B0C0
MOBS+A34      L     R15,B4(R0,R11)        58 F0 B0B4
MOBS+A38      BALR  R14,R15                05 EF
MOBS+A3A      IDL   5F0(R9),X'00'         80 00 95F0
MOBS+A3E      LPDR  R2,R0                  20 20
```

Beispiel zum Zeilenaufbau (T=MAX) für eine CSECT mit langem Namen.

```
BCL45678901234567890&@
BCL4567**+62  CLC   105(6,R2),11E(R2)     D5 05 2105 211E
```

Mit dem %OUT-Operandenwert T=MIN baut AID verkürzte Ausgabezeilen auf, in denen die CSECT-relative Adresse durch die virtuelle Adresse ersetzt wird und die sedezimale Darstellung des Speicherinhalts entfällt.

Beispiel zum Zeilenaufbau (T=MIN)

```
00000A30 LM    R0,R1,C0(R11)
00000A34 L     R15,B4(R0,R11)
00000A38 BALR  R14,R15
00000A3A IDL   5F0(R9),X'00'
00000A3E LPDR  R2,R0
```

Beispiele

1. %DISASSEMBLE 15 FROM CTX=CTXPHASE.V'A18'
Im Kontext CTXPHASE wird von der virtuellen Adresse 'A18' an der Speicherinhalt in Assembler-Befehle rückübersetzt. Es werden 15 Ausgabezeilen mit gültigem oder ungültigem Befehlscode erzeugt. Speicherinhalt, der keinen zugelassenen Befehlscode enthält, wird als Sedezimalstring mit der Länge von 2 Bytes ausgegeben mit dem Hinweis: INVALID OPCODE.
2. %DA 1 FROM %PC->
Ab der Unterbrechungsstelle wird ein Befehl rückübersetzt.
3. %DA FROM C=DRUCK.8
Die Rückübersetzung beginnt mit der Anfangsadresse des Programmabschnitts DRUCK+8. Ohne vorhergehende Vereinbarung für *anzahl* werden 10 Ausgabezeilen erzeugt.
4. %QUALIFY L=RESI.0=OSS1.C=KOPF
%DA 5 FROM .#'A0'->
AID ergänzt das Kommando wie folgt:
%DA 5 FROM L=RESI.0=OSS1.C=KOPF.#'A0'->
Zur Anfangsadresse des Programmabschnitts KOPF im Objektmodul OSS1, das wieder im Lademodul RESI enthalten ist, wird #'A0' addiert.
Unter der errechneten Adresse steht die Adresse, mit der AID die Rückübersetzung beginnen soll.
5. %OUT %DISASSEMBLE T=MIN
%DA 50 FROM V'128'
Von der virtuellen Adresse V'128' an sollen 50 Zeilen mit rückübersetztem Speicherinhalt erzeugt werden. Mit dem vorhergehenden %OUT wurde festgelegt, daß auf die sedezimale Darstellung des Speicherinhalts und die CSECT-relative Angabe der Speicheradresse verzichtet wird.
6. %FIND X'D5' ALIGN=2
%DA 1 FROM %OG->
Sie suchen die Anfangsadresse eines CLC-Befehls. %FIND hinterlegt im AID-Register %OG die Trefferadresse. Ab dieser Adresse soll AID einen Befehl disassemblieren.
7. %DA 1 FROM C=CS1.(%L(C=CS1)-4)
Die letzten 4 Bytes der CSECT CS1 sollen disassembliert werden.

%DISPLAY

Mit %DISPLAY veranlassen Sie die Ausgabe von Speicherinhalten, Adressen, Längen, Systeminformationen und AID-Literalen, und Sie steuern damit den Vorschub nach SYSLST. Die Ausgabe geht auf SYSOUT, SYSLST oder in eine katalogisierte Datei.

- Mit *daten* bezeichnen Sie eine Speicherstelle, deren Inhalt, Adresse oder Länge Sie ausgeben lassen wollen. Oder Sie bezeichnen damit Systeminformationen, definieren AID-Literale oder steuern den Vorschub nach SYSLST.
- Mit *medium-u-menge* geben Sie an, welche Ausgabe-Medien AID verwenden soll und ob Zusatzinformationen ausgegeben werden sollen. Mit diesem Operanden setzen Sie eine mit %OUT getroffene Vereinbarung für das aktuelle %DISPLAY-Kommando außer Kraft.

Kommando	Operand
%[DISPLAY]	daten {,...} [medium-u-menge][,...]

Ohne Qualifikation zu *daten* sprechen Sie *daten* im aktuellen AID-Arbeitsbereich an.

Beim Testen von ESA-Programmen werden Adressen, die in einem Datenraum liegen, bei der Ausgabe mit einem Stern markiert.

Ohne den *medium-u-menge*-Operanden gibt AID die Daten entweder gemäß der Vereinbarung im %OUT-Kommando oder standardmäßig mit Zusatzinformationen über SYSOUT aus.

%DISPLAY verändert den Programmzustand nicht.

daten

beschreibt, welche Informationen AID ausgeben soll. Sie können sich Speicherinhalte, Adressen, Längen und Systeminformationen ausgeben lassen. Um die Protokolle Ihrer Tests übersichtlicher zu gestalten, können Sie AID-Literale definieren oder den Vorschub nach SYSLST steuern.

Geben Sie in einem %DISPLAY mehrere *daten*-Operanden an, so können Sie von Operand zu Operand wechseln zwischen den hier beschriebenen nicht-symbolischen Angaben und den symbolischen Angaben, wie sie in den sprachspezifischen Benutzerhandbüchern [2] - [5] beschrieben sind. Auch innerhalb einer komplexen Speicherreferenz können Sie symbolische und maschinennahe Angaben mischen. Für symbolische Angaben müssen allerdings auch die LSD-Sätze zur Verfügung stehen (siehe AID-Basishandbuch, Kapitel 3 [1]).

daten-OPERAND - - - - -

{	[.] [qua.]	{ C=csect COM=common V'f...f' schlüsselwort kompl-speicherref }	}
	{ %@ %L } ([.] [qua.]	{ C=csect COM=common schlüsselwort kompl-speicherref }	
%L=(ausdruck)			
AID-Literal			
vorschubsteuerung			

- - - - -

- Steht der Punkt an führender Stelle, ist er das Kennzeichen für eine *vorqualifikation*. Sie muß mit einem vorhergehenden %QUALIFY definiert worden sein. Aufeinanderfolgende Qualifikationen werden durch einen Punkt getrennt. Außerdem muß zwischen der letzten *qualifikation* und dem anschließenden Operandenteil ein Punkt stehen.
- qua Eine oder mehrere Qualifikationen geben Sie an, wenn Sie ein Speicherobjekt ansprechen wollen, das nicht im aktuellen AID-Arbeitsbereich liegt oder darin nicht eindeutig ist. Sie geben nur die Qualifikationen an, die zur eindeutigen Ansprache genügen.
- E={VM | Dn}
 geben Sie nur an, wenn für *daten* nicht die aktuelle Basisqualifikation gelten soll.
- {ALET={X'f...f' | %nAR | %nG} | SPID=X'f...f'}
 geben Sie nur an, wenn Sie beim Testen von ESA-Programmen eine Adresse in einem Datenraum ansprechen wollen. Diesen Qualifikationen kann nur eine V-Adresse bzw. eine *kompl-speicherref* folgen.
- CTX=kontext
 geben Sie nur an, um eine CSECT oder einen COMMON im Programmsystem eindeutig ansprechen zu können.
- [L=ladeeinheit.][O=objektmodul.]
 geben Sie nur an, wenn Sie eine CSECT oder einen COMMON ansprechen wollen, die im aktuellen Kontext ohne diese Qualifikationen nicht eindeutig adressierbar sind. Sie geben nur die Qualifikationen an, die zur eindeutigen Ansprache nötig sind.

{C=csect | COM=common}

Endet die Adressierung mit einer C/COM-Qualifikation, so wird der ganze ausgewählte Programm-Abschnitt ausgegeben.

V'f...f' ist eine virtuelle Adresse. Standardmäßig wird der Inhalt an einer virtuellen Adresse sedezimal in der Länge 4 Bytes (%XL4) ausgegeben.

schlüsselwort

Sie können hier alle Schlüsselwörter für Speicherbereiche, Programmregister, AID-Register, Systeminformationen und das Schlüsselwort für den Durchlaufzähler angeben (siehe AID-Basishandbuch, Kapitel 9 [1]). Schlüsselwörter, die mehr als einen Wert liefern, werden von AID in Tabellenform aufbereitet.

%CLASS5	Klasse-5-Speicher
%CLASS5BELOW	Klasse-5-Speicher, unterhalb der 15MB-Grenze
%CLASS5ABOVE	Klasse-5-Speicher, oberhalb der 16MB-Grenze
%CLASS6	Klasse-6-Speicher
%CLASS6BELOW	Klasse-6-Speicher, unterhalb der 16MB-Grenze
%CLASS6ABOVE	Klasse-6-Speicher, oberhalb der 16MB-Grenze
%n	Mehrzweckregister, 0 ≤ n ≤ 15
%nD E	Gleitpunktregister, einfache/doppelte Genauigkeit, n = 0,2,4,6
%nQ	Gleitpunktregister, vierfache Genauigkeit, n = 0,4
%nG	AID-Mehrzweckregister, 0 ≤ n ≤ 15
%nGD	AID-Gleitpunktregister, n = 0,2,4,6
%nAR	Zugriffsregister, 0 ≤ n ≤ 15
%MR	alle 16 Mehrzweckregister in Tabellenform
%FR	alle 4 Gleitpunktregister mit doppelter Genauigkeit in Tabellenform aufbereitet
%AR	alle 16 Zugriffsregister in Tabellenform
%PC	Befehlszähler (Program Counter)
%PM	Program Mask
%CC	Condition Code
%PCB	Process Control Block
%PCBLST	Liste aller Process Control Blocks
%SORTEDMAP	Liste aller CSECTs und COMMONs des Benutzerprogramms und konnektierter Subsysteme werden ausgegeben (namens- und adreßsortiert). Lange Namen werden abgeschnitten.

%MAP [({ CTX=kontext [•L=ladeeinheit] })]

L=ladeeinheit

SCOPE = { USER }

 { ALL }

{CTX=kontext | L=ladeeinheit}

Bei Angabe eines Pfads werden alle CSECTs/Commons des angegebenen Kontextes oder der angegebenen Ladeeinheit aufgelistet.

<u>SCOPE=USER</u>	CSECTs/COMMONs der Standard-Kontexte CTXPHASE bzw. LOCAL#DEFAULT und der durch den BIND-Makro mit Operand LNKCTX[<u>@</u>] 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.
%AMODE	Adressierungsmodus (24- oder 31-Bit-Adressen)
%ASC	ASC-Modus auf ESA-Anlagen (bzgl. AR-Modus bedeutet: X'00' = aus; X'01' = ein)
%AUD1	P1-Audit-Tabelle, wenn vorhanden auch SAVE-Tabelle
%LINK	Name des zuletzt nachgeladenen Moduls/Segments (siehe %ON, Ereignis %LPOV)
%LOC(speicherref)	Lokalisierungs-Information für eine Speicherreferenz im ausführbaren Teil eines Programms
%DSE(ALET/SPID-qua)]	Information über SPIDs und/oder ALETs der aktiven Datenräume auf ESA-Anlagen.
%•[subkdoname]	Durchlaufzähler. Mit der Kurzform %• bezeichnen Sie den Durchlaufzähler des gerade aktiven Subkommandos

kompl-speicherref

bezeichnet eine errechnete Adresse. Folgende Operationen können in *kompl-speicherref* vorkommen (siehe AID-Basishandbuch, Kapitel 7.2.4 [1] und wegen Einschränkungen beim Testen von ESA-Programmen vorliegendes Handbuch, Kapitel 4):

- Adreßversatz(•)
- indirekte Adressierung (->)
- Typmodifikation (%T(datenname), %X, %C, %P, %D, %F, %A, %S, %SX)
- Längenmodifikation (%L(...), %L=(ausdruck), %Ln)
- Adreßselektion (%@(...))

Nach Adreßversatz oder indirekter Adressierung gibt AID den Speicherinhalt an der errechneten Adresse standardmäßig im Dump-Format (%XL4) aus. Mit der Typmodifikation können Sie sich *daten* in einer anderen Aufbereitung ausgegeben lassen, da sich mit dem Speichertyp auch der Ausgabetyt ändert.

Mit der Längenmodifikation können Sie die Ausgabelänge selbst bestimmen, wenn Sie z.B. nur Teile einer CSECT oder eines Speicherbereichs oder wenn Sie mehr als die standardmäßigen 4 Bytes ausgegeben lassen wollen.

Sind LSD-Sätze vorhanden, können Sie innerhalb von *kompl-speicherref* auch die im Quellprogramm definierten Daten- und Anweisungsamen verwenden.

%@(...)

Mit dem Adreßselektor können Sie sich die Anfangsadresse einer CSECT oder eines Speicherbereichs (%CLASS6, %CLASS6BELOW, %CLASS6ABOVE) oder von *kompl-speicherref* ausgeben lassen.

Der Adreßselektor liefert eine Adreßkonstante, die Sie in einer komplexen Speicherreferenz vor einem Pointer-Operator (->) verwenden können (siehe AID-Basishandbuch, Kapitel 7.2.4.4 [1]).

%L(...)

Mit dem Längenselektor können Sie sich die Länge einer CSECT oder eines Speicherbereichs (%CLASS6, %CLASS6BELOW, %CLASS6ABOVE) ausgeben lassen.

Der Längenselektor liefert eine Ganzzahl, die Sie für die Berechnung eines Adreßversatzes oder in einer Längenfunktion verwenden können (siehe AID-Basishandbuch, Kapitel 7.2.4.4 [1]).

Beispiel: %D %L(C=CS1) gibt die Länge der CSECT CS1 aus.

%L=(ausdruck)

Mit der Längenfunktion können Sie sich einen Wert errechnen lassen.

ausdruck wird gebildet aus dem Inhalt von Speicherreferenzen, Konstanten, Ganzzahlen und den arithmetischen Operatoren (+, -, *, /). Nur Speicherreferenz-Inhalte vom Typ %F oder %A mit einer Länge ≤ 4 sind zugelassen.

Die Längenfunktion liefert als Ergebnis eine Ganzzahl, die Sie für die Berechnung eines Adreßversatzes, in einer weiteren Längenfunktion oder zur Längenmodifikation verwenden können (siehe AID-Basishandbuch, Kapitel 7.2.4.4 [1]).

Beispiel: %D %L=(%1) gibt den Inhalt von Register 1 als Dezimalzahl aus.

AID-Literal

Alle im AID-Basishandbuch, Kapitel 9 [1] beschriebenen Literale können Sie angeben. Sie dienen der Gestaltung und Kommentierung des Ausgabe-Protokolls.

{C'x...x' 'x...x'C '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

vorschubsteuerung

Für das Ausgabemedium SYSLST kann die Druckaufbereitung durch die folgenden beiden Schlüsselwörter gesteuert werden:

- %NP bewirkt einen Seitenvorschub.
- %NL[n] bewirkt einen Zeilenvorschub um n Leerzeilen.
 $1 \leq n \leq 255$. Standardwert für n ist 1.

medium-u-menge

legt fest, über welches oder über welche Medien die Ausgabe erfolgen soll und ob zusätzlich Informationen neben dem Inhalt der bezeichneten Speicherbereiche ausgegeben werden sollen.

Ohne diesen Operanden und ohne eine Vereinbarung mit dem %OUT-Kommando arbeitet AID mit der Voreinstellung T=MAX.

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

$$\left\{ \begin{array}{l} \underline{T} \\ \underline{H} \\ \underline{F}_n \\ \underline{P} \end{array} \right\} = \left\{ \begin{array}{l} \underline{MAX} \\ \underline{MIN} \end{array} \right\}$$

medium-u-menge ist ausführlich im AID-Basishandbuch, Kapitel 8 [1] beschrieben.

- T Terminal-Ausgabe
- H Hardcopy-Ausgabe (schließt die Terminal-Ausgabe ein und kann nicht gemeinsam mit T angegeben werden)
- F_n Datei-Ausgabe
- P Ausgabe nach SYSLST
- MAX Ausgabe mit Zusatzinformationen.
- MIN Ausgabe ohne Zusatzinformationen.

Beispiele

1. Ausgabe des Inhalts ab der virtuellen Adresse V'8200' bis V'8203' im DUMP-Format (ohne Speichertyp-Angabe wird %XL4 verwendet).

```

/%DISPLAY V'8200'
*** TID: 001901D2 *** TSN: 1544 *****
CURRENT PC: 0000000C CSECT: M1BS *****
V'00008200' = UPRONUM + #'000001E8'
00008200 (000001E8) 002C00C1          ...A

```

2. Ausgabe des Inhalts ab der virtuellen Adresse V'8200' bis V'8203' im Character-Format (%C) und als ganze Zahl, die gepackt in 2 Bytes vorliegt (%P).

```

/%DISPLAY V'8200'%C
V'00008200' = UPRONUM + #'000001E8'
00008200 (000001E8) ...A

/%DISPLAY V'8200'%PL2
V'00008200' = UPRONUM + #'000001E8'
00008200 (000001E8) +2

```

3. An der Adresse V'100' steht der Befehl LA. Die Adresse wird im 1. %DISPLAY nur mit dem Speichertyp %SX modifiziert wodurch 4 Byte ab Adresse V'100' sedezimal ausgegeben werden. Soll AID die Adresse genauso berechnen, wie die Hardware den LA-Befehl ausführen würde, dann muß die Adresse im %DISPLAY mit nachfolgendem Pointer-Operator (->) geschrieben werden. Es wird eine Adresse errechnet: zu der Adresse im Indexregister %1 wird der Wert des Basisregisters %3 und die Distanz X'01B' hinzuaddiert. Das ergibt die Adresse V'725'. Die damit angesprochene Speicherstelle wird in der Länge 4 Byte im Dump-Format ausgegeben.

```

/%DISPLAY V'100'%SX
V'0000100' = M1BS + #'00000100'
00000100 (00000100) 4101301B

/%DISPLAY %1, %3
%1          = 00000700
%3          = 0000000A

/%DISPLAY V'100'%SX->
V'00000725' = M1BS + #'00000725'
00000725 (00000725) 58F0F008          .00.

```

4. Mit dem Schlüsselwort %PCB greift AID auf den Process Control Block zu und gibt ihn in Tabellenform aufbereitet aus. Mit dem Schlüsselwort %LOC gibt AID Informationen über die statische Programmverschachtelung der angegebenen Adresse aus.

```

/%DISPLAY %PCB
P1 LEVEL STACK AT LOCATION 720EAAB0          PRESENT LOCATION: ABSOLUT +00000060
                                                (LAST CALLED SVC: 27=SYSF-MGT)
  LNK 73AC86D0  CWRD 86010200  (PC) 00000060  (0) 00000000  (1) 9F00003C
  (2) 00000002  (3) 00000000  (4) 00000000  (5) 00000003  (6) 00000000
  (7) 00000000  (8) 00000000  (9) 00000000  (A) 00000000  (B) 00000000
  (C) 00000000  (D) 00000000  (E) 00000000  (F) 00000000  FLP0 00000000
FLP0 00000000  FLP2 00000000  FLP2 00000000  FLP4 00000000  FLP4 00000000
FLP6 00000000  FLP6 00000000  CLK 00000000  CCLK 00000000  MIX0 00000000
POST 00000000  MIX1 81000000  PTR 00000000  AIDA 70F16000  MIX2 80500000
EXRT 00000000  AUDM 00000000  MIX3 00000000  TLMW 00000000  -----

/%D %LOC(C=M1BS)
V'00000000' = CONTEXT:LOCAL#DEFAULT
             LMOD : %ROOT
             SMOD : M1BS
             OMOD : M1BS
             CSECT : M1BS      (00000000) + 00000000

```

5. Die ersten 3 Beispiele zeigen die Verwendung einer einfachen Speicherreferenz, einer einstufigen und zuletzt einer 2-stufigen indirekten Adresse. Zuletzt folgt auf die 2-stufige indirekte Adressierung noch der Adreßversatz, mit dem hier dieselbe Adresse wieder erreicht wird wie im ersten %DISPLAY-Kommando. Für weitere Informationen zur mehrfachen Verwendung von Adreßversatz und indirekter Adressierung siehe AID-Basishandbuch, Kapitel 7.2.4 [1].

```

/%D C=M1BS.#'A0'
V'000000A0' = M1BS      + #'000000A0'
000000A0 (000000A0) 000007FD      ...}

/%D C=M1BS.#'A0' ->
V'000007FD' = M1BS      + #'000007FD'
000007FD (000007FD) 00000000      ....

/%D C=M1BS.#'A0' ->->
V'00000000' = M1BS      + #'00000000'
00000000 (00000000) 58F0F008      ..0.

/%D C=M1BS.#'A0' ->->.#'A0'
V'000000A0' = M1BS      + #'000000A0'
000000A0 (000000A0) 000007FD      ...}

```

- 6. Mit %DUMPFILe wird dem AID-Linknamen D1 die Dump-Datei n M.DUMP, die zugewiesen. Daraus sollen ab der Adresse V'798' 20 Bytes im Dump-Format ausgegeben werden. Da die aktuelle Basisqualifikation E=VM gilt, müssen Sie explizit die Basisqualifikation E=D1 angeben.

```

/%DUMPFILe D1=M.DUMP
/%D E=D1.V'798'%L20
** D1: M.DUMP *****
V'00000798' = MOBS + #'00000798'
00000798 (00000798) 003DF000 00000000 C1C2C3C4 C5C6C7C8 ..0....ABCDEF GH
000007A8 (000007A8) C9D1D2D3 IJKL

```

- 7. Auf SYSLST sollen ausgegeben werden:
 - eine Kopfzeile mit dem aktuellen Stand des Befehlszählers und dem Namen der CSECT, in der das Programm gerade unterbrochen wurde (%D P=MAX).
 - der Befehlszähler (%PC)
 - eine Leerzeile (%NL(1))
 - 4 Bytes ab Adresse V'798' im Dump-Format
 - eine Leerzeile (%NL Standardwert = 1)
 - alle Mehrzweckregister (%MR)

Danach wird auf den Anfang der nächsten Seite positioniert (%NP).

```

/%OUT %D P=MAX
/%D %PC, %NL(1), V'798', %NL, %MR, %NP

```

- 8. Die letzten 4 Bytes des Klasse-6-Speichers sollte AID ausgeben. Da diese Speicherstelle nicht belegt sind, gibt AID die entsprechende Fehlermeldung aus.

```

/%DISPLAY %CLASS6.(%L(%CLASS6)-4)
V'003F8FFC' = %CLASS6 + #'003F8FFC'
AID0295 PAGE(S) FROM ADDRESS 003F8000 TO 003F8FFF NOT ALLOCATED

```

- 9. Folgenden Wert errechnet AID:
 Zu den letzten 2 Bytes in Register %1 wird der Wert addiert, der sich aus dem Inhalt des AID-Registers %6G multipliziert mit 2 ergibt. Dazu wird dann die Länge der CSECT CS1 addiert.
 Die Längenmodifikation %FL2 ist nötig, weil sonst mit dem Adreßversatz (•) die Bereichsgrenzen von Register %1 überschritten würden.

```

/%D %L=(%1.2%FL2 + (%6G * 2) + %L(C=CS1))

```

10. ESA-Unterstützung

Das geladene Programm (E=VM) arbeitet im AR-Modus, d.h., über die Zugriffsregister wird in Datenräumen adressiert.

In der Dump-Datei ESA.AR.DUMP hingegen wurde zum Zeitpunkt der Dump-Erstellung ohne AR-Modus, also nur im Programmraum, gearbeitet.

```
%DISPLAY %ASC
%ASC          = 01

%DF D3=ESA.AR.DUMP
%BASE E=D3
%DISPLAY %ASC
** D3: ESA.AR.DUMP *****
%ASC          = 00
```

11. ESA-Unterstützung

Im ersten %DISPLAY wird die Speicherstelle mit der Adresse V'34' im Programmraum ausgegeben im Typ der Datendefinition C#DS11.

Alle weiteren %DISPLAY-Kommandos geben die Speicherstelle mit der Adresse V'34' im Datenraum mit der SPID X'000000080000200' im Typ der Datendefinition C#DS11 aus. Der ALET X'0001001B' zeigt auf denselben Datenraum. Er ist in %2G zwischen gespeichert und in %7AR enthalten. Somit bewirken alle vier %DISPLAY-Kommandos dasselbe.

```
/%DISPLAY V'34' %T(C#DS11)
SRC_REF: 288 SOURCE: DS2S4A PROC: DS2S4A *****
C#DS11   = |PS1-|

/%DISPLAY ALET=%2G.V'34' %T(C#DS11)
C#DS11   = |DS11|

/%DISPLAY ALET=X'0001001B'.V'34' %T(C#DS11)
C#DS11   = |DS11|

/%DISPLAY ALET=%7AR.V'34' %T(C#DS11)
C#DS11   = |DS11|

/%DISPLAY SPID=X'000000080000200'.V'34' %T(C#DS11)
C#DS11   = |DS11|
```

12. ESA-Unterstützung

Über den ALET in Zugriffsregister 4 wird ein Datenraum angesprochen. In der nachfolgenden komplexen Speicherreferenz werden alle virtuellen Adressen (V'200', V'0', V'4') aus diesem Datenraum entnommen. AID rechnet V'1B30' + #'08' = Adresse V'1B38' und gibt ab dieser Adresse 17 Bytes (X'11') im Dump-Format aus.

```

%DISPLAY %AR, ALET=%4AR.V'200', ALET=%4AR.V'0', ALET=%4AR.V'10'
%AR ( 0: 15)
( 0) 00000000 ( 1) 00010003 ( 2) 00010004 ( 3) 00010005 ( 4) 00010006
( 5) 00000000 ( 6) 00000000 ( 7) 00000000 ( 8) 00000000 ( 9) 00000000
(10) 00000000 (11) 00000000 (12) 00000000 (13) 00000000 (14) 00000000
(15) 00000000

V'00000200' = *ABSOLUT + #'00000200'
00000200 (00000000) 00001B30          ....

V'00000000' = *ABSOLUT + #'00000000'
00000000 (00000000) 0800C301          ..C.

V'00000010' = *ABSOLUT + #'00000010'
00000010 (00000010) 110300D1          ...J

%D ALET=%4AR.V'200'->.(V'0' %FL1) %L(V'10' %XL1)
V'00001B38' = *ABSOLUT + #00001B38''
00001B38 (000004D0) D4E4C5D3 D3C5D940 C1D5D5C5 D3C9C5E2   MUELLER ANNELIES
00001B48 (000004E0) C5                                     E

```

13. ESA-Unterstützung

Bei einem %DISPLAY mit %DS werden für alle aktiven Datenräume ihre SPID, allen ihre ALETs und ihre Größe (SIZE) in Bytes ausgegeben. Geben Sie hinter dem Schlüsselwort eine ALET- oder SPID-Qualifikation an, gibt AID die zugehörige SPID bzw. die zugehörigen ALETs aus.

```

/%DISPLAY %DS
*** TID: 00020243 *** TSN: 13KG *****
CURRENT PC: 000000E0 CSECT: DS2S4A *****

SPID          ALET          SIZE
0001003300000031 00000000 0
00000000080000800 0001001B 409600
00000000080000800 0001001C 409600
00000000080000900 00010011 131072
00000000080000900 00010012 131072

/%DISPLAY %DS(SPID=X'000000080000800')

SPID = 000000080000800
ALET = 0001001B
ALET = 0001001C

/%DISPLAY %DS(ALET=X'00010011')

ALET = 00010011
SPID = 000000080000900

```

%DUMPFIL

Mit %DUMPFIL weisen Sie einem der Linknamen eine Dump-Datei zu und veranlassen AID, diese Datei zu öffnen oder zu schließen.

- Mit *link* wählen Sie den Linknamen für die Dump-Datei aus, die geöffnet oder geschlossen werden soll.
- Mit *datei* bezeichnen Sie die Dump-Datei, die geöffnet werden soll.

Kommando	Operand
$\left\{ \begin{array}{l} \%DUMPFIL \\ \%DF \end{array} \right\}$	[link [=datei]]

Ohne den *datei*-Operanden veranlassen Sie AID, die Datei zu schließen, die dem angegebenen Linknamen zugewiesen ist.

Mit einem %DUMPFIL ohne Operanden veranlassen Sie AID, alle offenen Dump-Dateien zu schließen. Lag bis dahin der AID-Arbeitsbereich in der nun geschlossenen Dump-Datei, gilt anschließend wieder der AID-Standard-Arbeitsbereich (siehe %BASE).

%DUMPFIL darf nur als Einzelkommando eingegeben werden, es darf nicht in einer Kommandofolge oder einem Subkommando stehen.

%DUMPFIL verändert den Programmzustand nicht.

link

bezeichnet einen AID-Linknamen für Dump-Dateien und hat das Format D_n , wobei n eine Zahl ist mit einem Wert $0 \leq n \leq 7$.

datei

gibt den vollqualifizierten Dateinamen an, unter dem die Dump-Datei katalogisiert ist, die AID öffnen soll.

Ohne diesen Operanden wird die Dump-Datei mit dem Linknamen *link* geschlossen.

Eine offene Dump-Datei muß erst mit einem eigenem %DUMPFIL geschlossen worden sein, bevor eine andere demselben Linknamen zugewiesen werden kann.

Beispiele

1. %DUMPFIL D3=DUMP.1234.00001
Die Datei mit dem BS2000-Katalognamen DUMP.1234.00001 wird dem Linknamen D3 zugewiesen und geöffnet.
2. %DF D3
Die Datei, die dem Linknamen D3 zugewiesen ist, wird geschlossen.
3. %DF
Alle offenen Dump-Dateien werden geschlossen.

%FIND

Mit %FIND können Sie ein Literal im Benutzerbereich des geladenen Programms oder in einem Speicherabzug in einer Dump-Datei suchen und Treffer auf Terminal (SYSOUT) ausgeben lassen. Außerdem werden in den AID-Registern %0G und %1G Trefferadresse und Fortsetzungsadresse abgelegt.

- *suchbegriff* ist ein Character- oder Sedezimal-Literal, das gesucht werden soll. Mit *ALL* geben Sie an, daß die Suche nicht nach der Ausgabe des ersten Treffers abgebrochen werden soll, sondern daß der gesamte *find-bereich* durchsucht und alle Treffer ausgegeben werden sollen. Die Suche kann dann nur mit der K2-Taste abgebrochen werden.
- Mit *find-bereich* geben Sie an, in welchem Speicherbereich AID *suchbegriff* suchen soll.
- Mit *alignment* geben Sie an, ob *suchbegriff* an Doppelwort-, Wort-, Halbwort- oder Byte-Grenze gesucht werden soll. Ohne Angabe von *alignment* wird an Byte-Grenze gesucht.

Kommando	Operanden
%F[IND]	[[ALL] suchbegriff [IN find-bereich] [alignment]]

Ein %FIND ohne den *ALL*-Operanden können Sie hinter der Adresse des letzten Treffers fortsetzen, bis das Ende von *find-bereich* erreicht ist, indem Sie ein neues %FIND-Kommando ohne eigene Operandenwerte eingeben.

In einem %FIND mit eigenem *suchbegriff* und ohne weitere Operanden ergänzt AID einen Operanden ohne aktuellen Wert mit dem entsprechenden Standardwert. Hier werden also keine Operanden aus einem vorhergehenden %FIND übernommen.

Die Ausgabe von Treffern erfolgt immer in sedezimaler Zeichendarstellung in der Länge von maximal 12 Byte auf das Medium Terminal (SYSOUT). Zusätzlich zum Treffer wird seine Adresse und (soweit möglich) der Name der CSECT/COMMON, in der der Treffer gefunden wurde, und die CSECT/COMMON-relative Adresse des Treffers ausgegeben.

Im Trefferfall wird die Trefferadresse im AID-Register %0G und die Fortsetzungsadresse (Trefferadresse + Suchstringlänge) im AID-Register %1G abgespeichert. Bei der Angabe von *ALL* wird die Adresse des letzten Treffers in %0G und die Fortsetzungsadresse des letzten Treffers in %1G abgespeichert. Falls *suchbegriff* nicht gefunden wurde, bleiben die AID-Register %0 und %1G unverändert. Die beiden Registerinhalte ermöglichen es Ihnen, das %FIND-Kommando auch in Prozeduren oder Subkommandos einzusetzen und mit den Ergebnissen weiterzuarbeiten.

%FIND verändert den Programmzustand nicht.

suchbegriff

ist ein Character- oder Sedezimal-Literal, das Sie im virtuellen Speicher oder in einer Dump-Datei suchen können.

Der *suchbegriff* kann Wildcard-Symbole enthalten. Diese Symbole sind immer Treffer. Sie werden durch '%' dargestellt.

suchbegriff-OPERAND - - - - -

{C'x...x' | 'x...x'C | 'x...x'}
{X'f...f' | 'f...f'X}

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

Character-Literal mit einer maximalen Länge von 80 Zeichen. Wenn Sie Kleinbuchstaben suchen wollen, müssen Sie mit %AID LOW[=ON] die Umsetzung in Großbuchstaben ausschalten.

x kann jedes darstellbare Zeichen annehmen, insbesondere das Wildcard-Symbol '%', welches immer einen Treffer darstellt. Das Zeichen '%' selbst kann in dieser Form nicht gesucht werden, da es als C'%' in einem Character-Literal stets zu einem Treffer führt. Es muß deshalb als Sedezimal-Literal X'6C' gesucht werden.

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

Sedezimal-Literal mit einer maximalen Länge von 80 Sedezimal-Stellen bzw. 40 Zeichen. Ein Literal mit ungerader Stellenzahl wird rechts mit X'0' ergänzt.

f kann jeden Wert zwischen 0 und F sowie das Wildcard-Symbol X'%' annehmen. Das Wildcard-Symbol stellt für jede Sedezimal-Stelle zwischen 0 und F einen Treffer dar.

find-bereich

legt einen Speicherbereich fest, in dem *suchbegriff* gesucht werden soll. *find-bereich* kann ein Teilbereich des virtuellen Speichers oder ein Teilbereich einer Dump-Datei sein.

Ist kein *find-bereich* angegeben, so setzt AID als Standardwert %CLASS6 ein und durchsucht den gesamten Klasse-6-Speicher im aktuellen AID-Arbeitsbereich, bei XS-Programmen nur den unteren Adreßraum.

find-bereich-OPERAND -----

IN [•][qua•] {
 {C=csect | COM=common
 (V'f...f': V'f...f')
 schlüsselwort
 kompl-speicherref }

- Steht der Punkt an führender Stelle, ist er das Kennzeichen für eine *vorqualifikation*. Sie muß mit einem vorhergehenden %QUALIFY definiert worden sein. Aufeinanderfolgende Qualifikationen werden durch einen Punkt getrennt. Außerdem muß zwischen der letzten *qualifikation* und dem anschließenden Operandenteil ein Punkt stehen.
- qua Eine oder mehrere Qualifikationen geben Sie an, wenn Sie ein Speicherobjekt ansprechen wollen, das nicht im aktuellen AID-Arbeitsbereich liegt oder darin nicht eindeutig ist. Sie geben nur die Qualifikationen an, die zur eindeutigen Ansprache genügen.
- E={VM | Dn}
 geben Sie nur an, wenn für *find-bereich* die aktuelle Basisqualifikation nicht gelten soll (siehe %BASE).
- {ALET={X'f...f' | %nAR | %nG} | SPID=X'f...f'}
 geben Sie nur an, wenn Sie beim Testen von ESA-Programmen eine Adresse in einem Datenraum ansprechen wollen. Diesen Qualifikationen muß eine V-Adresse bzw. eine *kompl-speicherref* folgen.
- CTX=kontext
 geben Sie nur an um eine CSECT oder einen COMMON im Programmsystem eindeutig ansprechen zu können.
- [L=ladeeinheit•][O=objektmodul•]
 geben Sie nur an, wenn Sie eine CSECT oder einen COMMON ansprechen wollen, die im aktuellen Kontext ohne diese Qualifikationen nicht eindeutig adressierbar sind. Sie geben nur die Qualifikationen an, die zur eindeutigen Ansprache nötig sind.
- {C=csect | COM=common}
 Endet die Adressierung mit einer C/COM-Qualifikation, so wird die angegebene CSECT oder der COMMON als *find-bereich* festgelegt.

schlüsselwort

legt einen Speicherbereich durch Angabe eines der folgenden Schlüsselwörter (siehe AID-Basishandbuch, Kapitel 10 [1]) fest. Bei XS- bzw. ESA-Anlagen bezeichnen Sie mit %CLASS6 und %CLASS6BELOW denselben Adreßraum.

%CLASS6	Klasse-6-Speicher
%CLASS6BELOW	Klasse-6-Speicher, unterhalb der 16MB-Grenze
%CLASS6ABOVE	Klasse-6-Speicher, oberhalb der 16MB-Grenze

(V'f...f' : V'f...f')

legt einen Speicherbereich durch die Angabe einer virtuellen Anfangs- und Endadresse fest.

Die resultierende Adreßdifferenz darf einen Wert von 65 535 nicht überschreiten und Anfangsadresse muß ≤ Endadresse sein.

f..f ist eine maximal 8stellige Sedezimal-Adresse.

kompl-speicherref

bezeichnet einen Bereich von 4 Bytes ab der errechneten Adresse. Soll eine andere Anzahl von Bytes durchsucht werden, muß *kompl-speicherref* mit der entsprechenden Längenmodifikation (≤ 64 KB) enden. Folgende Operationen können darin vorkommen (siehe AID-Basishandbuch, Kapitel 7 [1] und wegen Einschränkungen beim Testen von ESA-Programmen vorliegendes Handbuch, Kapitel 4, Maschinencode-spezifische Adressierung):

- Adreßversatz (•)
- indirekte Adressierung (->)
- Typmodifikation (%A, %S, %SX)
- Längenmodifikation (%L(...), %L=(ausdruck), %Ln)
- Adreßselektion (%@(...))

alignment

legt fest, daß *suchbegriff* nur an bestimmten ausgerichteten Adressen gesucht werden soll.



suchbegriff wird gesucht an:

- 1 Byte-Grenze (Standardwert)
- 2 Halbwortgrenze
- 4 Wortgrenze
- 8 Doppelwortgrenze

Beispiele

1. `%FIND C'***' IN %CLASS6`
Das Zeichen-Literal C'***' wird im Klasse-6-Speicher (unterhalb der 16 MB-Grenze) der aktuell geltenden Basisqualifikation gesucht. Ein Treffer wird auf SYSOUT ausgegeben.
2. `%FIND`
Die Suche wird mit den Parametern des letzten %FIND-Kommandos hinter dem letzten Treffer fortgesetzt.
3. `%F ALL X'00F%00' ALIGN 2`
An jeder Halbwortgrenze wird ein Sedezimal-Literal gesucht, das in den ersten drei Halbbytes X'00F' und im vierten Halbbyte einen beliebigen Sedezimalwert enthält (repräsentiert durch %) und mit X'00' endet.
4. `%AID LOW = ON`
`%IN V'8A0' <(%OG NE -1):%MOVE 'Hansi' INTO %OG-->`
`%IN V'8A0' <%FIND 'Berta'>`
Die Unterscheidung von Groß- und Kleinbuchstaben wird eingeschaltet. Durch Verkettung entsteht am Testpunkt V'8A0' die Kommandofolge:
`%FIND 'Berta'; (%OG NE -1):%MOVE 'Hansi' INTO %OG-->`
Die Bedingung kann nur am Anfang eines Subkommandos geschrieben werden. Deshalb sind zwei %INSERTs zu demselben Testpunkt nötig, wenn nur ein Teil der Kommandofolge, die am Testpunkt wirken soll, von der Bedingung abhängig sein soll.
Am Testpunkt mit der virtuellen Adresse V'8A0' soll der String 'Berta' gesucht werden. Im Trefferfall (AID-Register %OG ≠ -1) wird der entsprechende Speicherbereich mit 'Hansi' überschrieben.
5. `%IN V'1648' <(%OG NE -1): %SET %L=(%1G-%OG) INTO %2G>`
`%IN V'1648' <%FIND 'x...x'>`
Diese Kommandofolge können Sie in einer Prozedur einsetzen, um ein beliebiges Character-Literal zu suchen. Im Trefferfall steht die gefundene Adresse im AID-Register %OG und die Länge des gesuchten Strings in %2G.
6. `%F X'ABCD' IN V'A1A'%L=(%1)`
In einem variabel langen Speicherbereich ab Adresse V'A1A' wird ein beliebiges Character-Literal gesucht. Die Länge des Speicherbereichs steht jeweils in Register 1.


```

info-ziel-OPERAND -----
{
%[AID | %[AINT | %[BASE | %[CONT[INUE] | %[C[ONTR]OL]
%[DISASSEMBLE | %[DA | %[D[ISPLAY] | %[DUM]PFILE | %[DF
%[F[IND] | %[H[ELP] | %[IN[SE]RT] | %[JUM]P | %[M[OVE]
%[ON | %[OUT | %[OUTFILE | %[Q[UALIFY]
%[RE[M]OVE] | %[R[ESUM]E] | %[SD[U]MP] | %[S[ET]
%[SH[OW] | %[STO]P | %[SYMLIB | %[TITL]E | %[T[RACE]
}
In

```

Die AID-Kommandonamen können auch in der zulässigen Abkürzung angegeben werden.

In bezeichnet den alten Meldungsschlüssel einer Meldung, zu der Bedeutung und mögliche Maßnahmen ausgegeben werden sollen.

n ist die dreistellige Meldungsnummer.

Die Abfrage von Meldungen mit dem AID-%HELP-Kommando wird in zukünftigen AID-Versionen nicht mehr unterstützt, da dieselbe Information mit dem SDF-Kommando /HELP=MESSAGE=INFORMATION angefordert werden kann.

medium-u-menge

legt fest, über welche Medien die Informationen zu *info-ziel* ausgegeben werden sollen.

Ohne diesen Operanden und ohne Vereinbarung mit dem %OUT-Kommando arbeitet AID mit dem Standardwert T=MAX.

```

medium-u-menge-OPERAND -----
{
T
H
} = {
MAX
}
{
Fn
P
} = {
MIN
}

```

medium-u-menge ist ausführlich im AID-Basishandbuch, Kapitel 8 [1] beschrieben.

T Terminal-Ausgabe

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

Fn Datei-Ausgabe

P Ausgabe nach SYSLST

Die Angabe {MAX | MIN} hat bei %HELP keine Auswirkungen, eine der beiden Angaben ist aber syntaktisch erforderlich.

%INSERT

Mit %INSERT legen Sie einen Testpunkt fest und vereinbaren ein Subkommando. Wenn der Programmablauf den Testpunkt erreicht, bearbeitet AID das zugehörige Subkommando. Zusätzlich können Sie angeben, ob AID nach einer angegebenen Anzahl von Durchläufen den Testpunkt löschen soll und ob das Programm danach angehalten werden soll.

- Mit *testpunkt* bezeichnen Sie den Befehl im Programm, vor dessen Ausführung AID den Programmablauf unterbrechen soll, um *subkdo* zu bearbeiten.
- Mit *subkdo* vereinbaren Sie ein Kommando oder eine Kommandofolge und eventuell eine Bedingung. Wird *testpunkt* erreicht und ist die Bedingung erfüllt, wird *subkdo* ausgeführt.
- Mit *steuerung* legen Sie fest, ob *testpunkt* nach einer vorgegebenen Anzahl von Durchläufen gelöscht und ob danach das Programm angehalten werden soll.

Kommando	Operand
%IN[<i>SERT</i>]	<i>testpunkt</i> [<i><subkdo></i>] [<i>steuerung</i>]

Ein *testpunkt* wird in folgenden Fällen gelöscht:

1. Das Programmende wird erreicht.
2. Die mit *steuerung* vorgegebene Anzahl von Durchläufen wird erreicht, und das Löschen von *testpunkt* ist vereinbart.
3. Der *testpunkt* wird mit %REMOVE gelöscht.

Ohne *subkdo*-Operanden setzt AID das *subkdo* *<%STOP>* ein.

Das *subkdo* eines %INSERT für einen bereits gesetzten *testpunkt* überschreibt nicht das bestehende *subkdo*, sondern das neue *subkdo* wird vor das bestehende gekettet. Das bedeutet, Subkommandos zu demselben *testpunkt* werden nach dem LIFO-Prinzip bearbeitet.

Mit %REMOVE löschen Sie ein Subkommando, einen *testpunkt* oder alle eingetragenen *testpunkte*.

testpunkt kann nur eine Adresse im geladenen Programm sein, deshalb muß die Basisqualifikation E=VM eingestellt sein (siehe %BASE) oder explizit angegeben werden.

%INSERT verändert den Programmzustand nicht.

testpunkt

Ist die Adresse eines ausführbaren Maschinenbefehls. *testpunkt* wird sofort durch das gezielte Überschreiben der adressierten Speicherstelle mit X'0A81' eingetragen und muß deshalb zum Zeitpunkt der %INSERT-Eingabe im virtuellen Speicher geladen sein. Ein *testpunkt*, der nicht auf die Anfangsadresse eines ausführbaren Maschinenbefehls eingetragen wird, führt im Programmablauf zu Fehlern (z.B. Daten- oder Adressierungsfehler).

Kommt der Programmablauf an den *testpunkt*, unterbricht AID das Programm und startet *subkdo*. Wird der Programmablauf fortgesetzt, beginnt er mit der Ausführung des mit *testpunkt* überschriebenen Befehls.

testpunkt-OPERAND - - - - -

[•][qua•] $\left\{ \begin{array}{l} C=csect \mid COM=common \\ V'f\dots f' \\ \text{kompl-speicherref} \end{array} \right\}$

- - - - -

- Steht der Punkt an führender Stelle, ist er das Kennzeichen für eine *vorqualifikation*. Sie muß mit einem vorhergehenden QUALIFY-Kommando definiert worden sein. Aufeinanderfolgende Qualifikationen werden durch einen Punkt getrennt. Außerdem muß zwischen der letzten Qualifikation und dem anschließenden Operandenteil ein Punkt stehen.

qua eine oder mehrere Qualifikationen geben Sie an, wenn Sie ein Speicherobjekt ansprechen wollen, das nicht im aktuellen AID-Arbeitsbereich liegt oder darin nicht eindeutig ist. Sie geben nur die Qualifikationen an, die zur eindeutigen Ansprache genügen.

E=VM

Da *testpunkt* nur im virtuellen Speicher des geladenen Programms eingetragen werden kann, geben Sie *E=VM* nur an, wenn als aktuelle Basisqualifikation eine Dump-Datei vereinbart ist (siehe %BASE).

CTX=context

geben Sie nur an, um eine CSECT oder einen COMMON im Programmsystem eindeutig ansprechen zu können.

Ein Subkommando kann einen Namen, eine Bedingung und einen Kommandoteil enthalten. Zu jedem Subkommando gehört ein Durchlaufzähler. Der Kommandoteil kann aus einem einzelnen Kommando oder einer Kommandofolge bestehen, er kann AID- und BS2000-Kommandos und Kommentare enthalten.

Wenn das Subkommando aus einem Namen oder einer Bedingung besteht, aber der Kommandoteil fehlt, erhöht AID beim Erreichen des Testpunkts nur den Durchlaufzähler.

subkdo überschreibt nicht ein bestehendes Subkommando zum selben *testpunkt*, sondern das neue Subkommando wird vor das bestehende gekettet. In *subkdo* sind die Kommandos %CONTROL*n*, %INSERT und %ON zugelassen. Sie können damit eine Schachtelung bis zu 5 Stufen vornehmen.

Die Kommandos in einem *subkdo* werden nacheinander ausgeführt. Danach wird das Programm fortgesetzt. Die Kommandos zur Ablaufsteuerung verändern auch in einem Subkommando sofort den Programmzustand. Sie brechen *subkdo* ab und starten das Programm (%CONTINUE, %RESUME, %TRACE) oder halten es an (%STOP). Sie sind nur als letztes Kommando in einem *subkdo* sinnvoll, da nachfolgende *subkdo*-Kommandos nicht mehr ausgeführt werden. Auch ein Löschen des gerade aktiven Subkommandos mit %REMOVE ist nur als letztes Kommando in *subkdo* sinnvoll.

steuerung

gibt an, ob *testpunkt* nach dem *n*-ten Durchlauf gelöscht und das Programm angehalten werden soll, damit neue Kommandos eingegeben werden können. Wird der *steuerung*-Operand nicht angegeben, setzt AID die Standardwerte $2^{31}-1$ (für *n*) und *K* ein.

Dieser Operand wird in dieser Ausgabe zum letzten Mal beschrieben.

steuerung-OPERAND - - - - -

ONLY *n* $\left[\begin{array}{c} \{K\} \\ \{C\} \\ \{S\} \end{array} \right]$

- - - - -
- n* ist eine Ganzzahl mit dem Wert $1 \leq n \leq 2^{31}-1$.
Sie gibt an, beim wievielten Durchlaufen von *testpunkt* die weiteren Vereinbarungen dieses *steuerung*-Operanden ausgeführt werden sollen.
 - K Weder *subkdo* noch *testpunkt* werden gelöscht (KEEP).
Der Programmlauf wird unterbrochen, und AID erwartet die Eingabe von Kommandos.

- S *subkdo* wird gelöscht. Ist kein weiteres Subkommando zu diesem *testpunkt* eingetragen, wird auch *testpunkt* gelöscht.
Der Programmablauf wird unterbrochen, und AID erwartet die Eingabe von Kommandos (STOP).
- C *subkdo* wird gelöscht. Ist kein weiteres Subkommando zu diesem *testpunkt* eingetragen, wird auch *testpunkt* gelöscht.
Der Programmablauf wird **nicht** unterbrochen (CONTINUE).

Beispiele

- ```
%IN V'80' <SUB1: %DISPLAY %5>
%IN V'80' <SUB2: %DISPLAY %7> ONLY 4 S
```

Bei Erreichen der Adresse V'80' wird der Inhalt der Register 7 und 5 ausgegeben. Wenn das Subkommando SUB2 viermal ausgeführt worden ist, wird es gelöscht und das Programm angehalten. SUB1 bleibt bestehen und wird im weiteren Testverlauf bei Erreichen der Adresse V'80' wieder ausgeführt.
- ```
%IN C=CS1.16 <(%0G NE -1): %SET %L=(%1G-%0G) INTO %2G>
```

Auf das 16. Byte der CSECT CS1 wird ein Testpunkt eingetragen. Kommt der Programmablauf an diese Adresse, werden die Rückmeldungen des letzten ausgeführten %FIND-Kommandos geprüft:
Gab es keinen Treffer, so ist AID-Register %0G auf -1 gesetzt, das Ergebnis der Bedingung ist FALSE, und das Subkommando wird nicht ausgeführt; gab es einen Treffer, so ist in AID-Register %0G die Trefferadresse abgelegt, das Ergebnis der Bedingung ist TRUE, und das Subkommando wird ausgeführt:
Die Länge des gesuchten Strings wird mit Hilfe der Längenfunktion errechnet und in AID-Register %2G übertragen.
- ```
%IN V'800'%SX-> <%DA 2 FROM %PC-->>
```

Auf Adresse V'800' steht ein Assemblerbefehl (RX-Format). Auf die Adresse, die sich aus Indexregister, Basisregister und Distanz ergibt, wird ein Testpunkt eingetragen. Kommt der Programmablauf an diese Adresse, werden die nächsten beiden Befehle rückübersetzt.

## %MOVE

Mit %MOVE übertragen Sie Speicherinhalte oder AID-Literale auf Speicherstellen im geladenen Programm. Die Übertragung wird linksbündig ohne Überprüfung und ohne Anpassung des Speichertyps von Sender und Empfänger durchgeführt.

- Mit *sender* bezeichnen Sie die Speicherstelle, deren Inhalt übertragen werden soll, eine Länge, eine Adresse oder ein AID-Literal. *sender* kann im virtuellen Speicher des geladenen Programms oder in einer Dump-Datei liegen.
- Mit *empfänger* bezeichnen Sie die Speicherstelle, die überschrieben werden soll. *empfänger* kann nur im virtuellen Speicher des geladenen Programms liegen.
- Mit *REP* geben Sie an, ob AID zu einer durchgeführten Änderung einen REP erzeugen soll. Mit diesem Operanden setzen Sie die globale Einstellung (siehe %AID-Kommando) für das aktuelle %MOVE-Kommando außer Kraft.

| Kommando | Operand                     |
|----------|-----------------------------|
| %MOVE]   | sender INTO empfänger [REP] |

Im Gegensatz zu %SET überprüft AID beim %MOVE nicht die Verträglichkeit der Speichertypen von *sender* und *empfänger* und paßt *sender* nicht an den Speichertyp von *empfänger* an.

AID überträgt linksbündig in der Länge von *sender*. Ist *sender* länger als *empfänger* weist AID die Übertragung mit einer Fehlermeldung ab.

Mit dem Kommando %AID, Operand *CHECK*, können Sie zur Kontrolle einen Änderungsdialog einschalten, der Ihnen vor Durchführung der Übertragung den alten und neuen Inhalt von *empfänger* zeigt und die Möglichkeit zum Abbruch des %MOVE-Kommandos bietet.

%MOVE verändert den Programmzustand nicht.

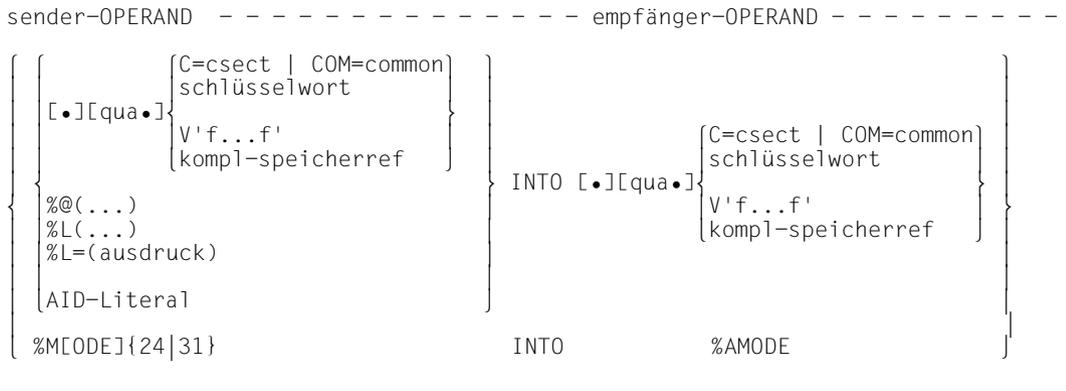
sender INTO empfänger

Für *sender* und *empfänger* können Sie eine virtuelle Adresse, eine C/COM-Qualifikation, eine komplexe Speicherreferenz oder ein Schlüsselwort angeben.

Nur als *sender* können Sie AID-Literale, Selektoren und das Schlüsselwort %MODEN angeben. Übertragen bzw. überschreiben Sie Befehlscode, kann das zu unerwünschten Ergebnissen führen, wenn *sender* oder *empfänger* in einem *control*- oder *trace-bereich* liegen, der symbolisch vereinbart wurde (siehe AID-Basishandbuch, Kapitel 12.2 [1]).

*sender* kann sowohl im virtuellen Speicher, als auch in einer Dump-Datei liegen; dagegen kann *empfänger* nur im virtuellen Speicher liegen.

Mehr als 3900 Bytes können Sie mit einem %MOVE nicht übertragen. Wenn Sie einen größeren Bereich übertragen wollen, müssen Sie mehrere %MOVE-Kommandos eingeben.



- Steht der Punkt an führender Stelle, ist er das Kennzeichen für eine *vorqualifikation*. Sie muß mit einem vorhergehenden %QUALIFY definiert worden sein. Aufeinanderfolgende Qualifikationen werden durch einen Punkt getrennt. Außerdem muß zwischen der letzten *qualifikation* und dem anschließenden Operandenteil ein Punkt stehen.

**qua** Eine oder mehrere Qualifikationen geben Sie an, wenn Sie ein Speicherobjekt ansprechen wollen, das nicht im aktuellen AID-Arbeitsbereich liegt oder darin nicht eindeutig ist. Sie geben nur die Qualifikationen an, die zur eindeutigen Ansprache genügen.

{E={VM | Dn} für *sender* | E=VM für *empfänger*}

Eine Basisqualifikation geben Sie nur an, wenn die aktuelle Basisqualifikation nicht gelten soll.

*sender* kann sowohl im virtuellen Speicher, als auch in einer Dump-Datei liegen. *empfänger* hingegen darf nur im virtuellen Speicher liegen.

{ALET={X'f...f' | %nAR | %nG} | SPID=X'f...f'}

geben Sie nur an, wenn Sie beim Testen von ESA-Programmen eine Adresse in einem Datenraum ansprechen wollen. Diesen Qualifikationen kann nur eine V-Adresse bzw. eine *kompl-speicherref* folgen.

**CTX**=kontext

geben Sie nur an, um eine CSECT oder einen COMMON im Programmsystem eindeutig ansprechen zu können.

[L=ladeeinheit.][O=objektmodul.]

geben Sie nur an, wenn Sie eine CSECT oder einen COMMON ansprechen wollen, die im aktuellen Kontext ohne diese Qualifikationen nicht eindeutig adressierbar sind. Sie geben nur die Qualifikationen an, die zur eindeutigen Ansprache nötig sind.

{C=csect | COM=common}

Endet die Adressierung mit einer C/COM-Qualifikation, so wird der ganze ausgewählte Programm-Abschnitt übertragen (*sender*), bzw. von Anfang an überschrieben (*empfänger*).

*sender* darf allerdings nicht länger als 3900 Bytes sein.

V'f...f' ist eine virtuelle Adresse.

kompl-speicherref

Folgende Operationen können darin vorkommen (siehe AID-Basishandbuch, Kapitel 7.2.4 [1] und wegen Einschränkungen beim Testen von ESA-Programmen vorliegendes Handbuch, Kapitel 4):

- Adreßversatz (•)
- indirekte Adressierung (->)
- Typmodifikation (%T(datename), %X, %C, %P, %D, %F, %A, %S, %SX)
- Längenmodifikation (%L(...), %L=(ausdruck), %Ln)
- Adreßselektion (%@(...))

Eine abschließende Typmodifikation hat keine Auswirkung auf die Übertragung, da %MOVE unabhängig vom Speichertyp von *sender* und *empfänger* stets binär überträgt.

Eine abschließende Längenmodifikation für *sender* legt fest wieviel Byte übertragen werden sollen. Bei *empfänger* kann eine Längenmodifikation nur notwendig werden, um die Bereichsgrenzen von *empfänger* nicht zu überschreiten (siehe Beispiel 3).

Sind LSD-Sätze vorhanden, können Sie innerhalb von *kompl-speicherref* auch die im Quellprogramm definierten Daten- und Anweisungsnamen verwenden.

schlüsselwort

bezeichnet ein Register, den Befehlszähler oder einen Durchlaufzähler. Im AID-Basishandbuch, Kapitel 10 [1] sind die impliziten Speichertypen der Schlüsselwörter angegeben.

|                |                                                                                                         |
|----------------|---------------------------------------------------------------------------------------------------------|
| %PC            | Befehlszähler (Program Counter)                                                                         |
| %n             | Mehrzweckregister, $0 \leq n \leq 15$                                                                   |
| %nD E          | Gleitpunktregister, einfache/doppelte Genauigkeit, $n = 0,2,4,6$                                        |
| %nQ            | Gleitpunktregister, vierfache Genauigkeit, $n = 0,4$                                                    |
| %nG            | AID-Mehrzweckregister, $0 \leq n \leq 15$                                                               |
| %nDG           | AID-Gleitpunktregister, $n = 0,2,4,6$                                                                   |
| %•[subkdoname] | Durchlaufzähler. Mit der Kurzform %• bezeichnen Sie den Durchlaufzähler des gerade aktiven Subkommandos |

Daneben können Sie noch die Schlüsselwörter für die Speicherklassen verwenden (siehe AID-Basishandbuch, Kapitel 10.5 [1]).

%@(…)

Mit dem Adreßselektor können Sie die eicherklasse (%CLASS6, %CLhend für den Klasse-5-Speicher) oder einer *kompl-speicherref* als *sender* verwenden.

Der Adreßselektor liefert eine Adreßkonstante, die Sie in einer komplexen Speicherreferenz vor einem Pointer-Operator (->) verwenden können (siehe AID-Basishandbuch, Kapitel 7.2.4.4 [1]).

%L(…)

Mit dem Längenselektor können Sie die Länge einer CSECT, eines COMMON oder einer Speicherklasse (%CLASS6, %CLASS6BELOW, %CLASS6ABOVE) als *sender* verwenden.

Der Längenselektor liefert eine Ganzzahl, die Sie für die Berechnung eines Adreßversatzes oder in einer Längenfunktion verwenden können (siehe AID-Basishandbuch, Kapitel 7.2.4.4 [1]).

**Beispiel:** %L(C=CS1) die Länge der CSECT CS1 wird als *sender* verwendet.

%L=(ausdruck)

Mit der Längenfunktion können Sie sich als *sender* einen Wert errechnen lassen. *ausdruck* wird gebildet aus dem Inhalt von Speicherreferenzen, Konstanten, Ganzzahlen und arithmetischen Operatoren. Nur Speicherreferenz-Inhalte vom Typ %F oder %A mit einer Länge  $\leq 4$  sind zugelassen.

Die Längenfunktion liefert als Ergebnis eine Ganzzahl, die Sie für die Berechnung eines Adreßversatzes, in einer weiteren Längenfunktion oder zur Längenmodifikation verwenden können (siehe AID-Basishandbuch, Kapitel 7.2.4.4 [1]).

**Beispiel:** %L=(%1) der Inhalt von Register 1 wird als *sender* verwendet.

## AID-Literal

Die folgenden AID-Literale (siehe AID-Basishandbuch, Kapitel 9 [1]) können mit %MOVE übertragen werden:

|                                 |                   |
|---------------------------------|-------------------|
| {C'x...x'   'x...x'C   '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    |

## %M[ODE]{24 | 31}

legt den Adressierungsmodus fest.

## %AMODE

ist das Systeminformationsfeld, in dem der Adressierungsmodus eingetragen ist.

Mit der Übertragung des Schlüsselwortes %M{24|31} in das Systeminformationsfeld %AMODE ändern Sie den Adressierungsmodus des Testobjekts. Wenn Sie nur die Adreßinterpretation für die indirekte Adressierung (->) ändern wollen, geben Sie das entsprechende %AINT-Kommando ein.

Wenn Sie beim Test eines 31-Bit-Adressen-Programms den Befehlszähler ändern, während der 24-Bit-Modus eingestellt ist, müssen Sie darauf achten, daß das höchstwertige Byte den Wert X'00' hat.

## REP

gibt an, ob AID zu einer durchgeführten Änderung einen REP erzeugen soll. Mit *REP* setzen Sie eine mit dem Kommando %AID getroffene Vereinbarung vorübergehend außer Kraft. Wird *REP* nicht angegeben und gibt es keine entsprechende Vereinbarung im %AID-Kommando, so wird kein REP erstellt.

rep-OPERAND - - - - -

REP = {Y[ES] | NE[O]}

- - - - -

## REP=Y[ES]

Zu der durch das aktuelle %MOVE-Kommando durchgeführten Änderung werden LMS-Korrekturanweisungen (REPs) im SDF-Format erstellt. Für den LMS-Lauf muß dann noch die MODIFY-ELEMENT-Anweisung eingefügt werden. Wenn die

Objekt-Strukturliste nicht zur Verfügung steht, erstellt AID keine Korrekturanweisungen und gibt eine Fehlermeldung aus. Auch wenn *empfänger* nicht vollständig innerhalb einer CSECT liegt, gibt AID eine Fehlermeldung aus und schreibt keinen REP. Um im zweiten Fall dennoch REPs zu erhalten, müssen Sie die Übertragung auf mehrere %MOVE-Kommandos verteilen, in denen Sie die CSECT-Grenzen beachten.

AID hinterlegt die Korrekturen mit den nötigen LMS-Korrektur-Anweisungen in einer Datei mit dem Linknamen F6. Ist keine Datei mit dem Linknamen F6 angemeldet (siehe %OUTFILE), so wird der REP in der von AID angelegten Datei AID.OUTFILE.F6 abgelegt.

Wenn Sie Rep-Sätze erstellen, sollten Sie den Linknamen F6 nur für %MOVE reservieren, um nicht versehentlich andere Testdaten in die REP-Datei schreiben zu lassen.

REP=N[O]

Zum aktuellen %MOVE-Kommando werden keine REPs erstellt.

## Beispiele

1. %MOVE %6 ->.20 INTO V'10A'

Die Anfangsadresse von *sender* errechnet sich aus dem Inhalt im Mehrzweckregister 6, auf den 20 addiert wird. Mit diesem Kommando werden 4 Bytes übertragen (Standardlänge einer Speicherstelle).

2. %MOVE E=D1.%PC INTO V'A04'

Aus dem Speicherabzug in der Dump-Datei mit dem Linknamen D1 wird der Befehlszählerstand auf die virtuelle Adresse V'A04' des aktuellen AID-Arbeitsbereichs übertragen.

3. %MOVE %7.1%L2 INTO %3.2%L2

Der Inhalt von Byte 2 und 3 des Mehrzweckregisters 7 wird in Byte 3 und 4 des Mehrzweckregisters 3 übertragen. Die Längenmodifikation bei *empfänger* ist notwendig, weil durch den Adreßversatz die Bereichsgrenzen von Register 3 überschritten würden.

4. %MOVE X'47F0' INTO V'20' REP=YES

Der Inhalt der Adresse V'20' wird mit dem Sedezimal-Literal X'47F0' überschrieben. Zu der Korrektur wird ein REP erstellt und in der Datei AID.OUTFILE.F6 bzw. der dem Linknamen F6 zugewiesenen Datei abgelegt.

5. %MOVE %MODE31 INTO %AMODE

Für das aktuelle Testobjekt wird der Adressierungsmodus auf 31-Bit-Adressen umgestellt. Gleichzeitig wird die Adreßinterpretation für indirekte Adressen in AID-Kommandos an den neuen Adressierungsmodus angepaßt, wenn kein %AINT-Kommando aktive ist. Diese Vereinbarung müßte dann evtl. angepaßt werden.

## %ON

Mit %ON legen Sie ein Ereignis fest und können dazu ein Subkommando definieren. Wenn im Programmablauf ein ausgewähltes Ereignis eintritt, bearbeitet AID das zugehörige *subkdo*.

- Mit *write-ereignis* beschreiben Sie das Ereignis des schreibenden Zugriffs auf einen Speicherbereich. Immer wenn das Programm den angegebenen Speicherbereich verändert, soll AID den Programmablauf unterbrechen, um *subkdo* zu bearbeiten.
- Mit *ereignis* beschreiben Sie eines der anderen Ereignisse (normale oder unnormale Programmbeendigung, Supervisor-Call (SVC), Programmfehler, etc.), bei dem AID den Programmablauf unterbrechen soll, um *subkdo* zu bearbeiten.
- Mit *subkdo* definieren Sie ein Kommando oder eine Kommandofolge und eventuell eine Bedingung. Bei zutreffendem Ereignis und erfüllter Bedingung wird *subkdo* ausgeführt.

| Kommando | Operand                                                                                                           |
|----------|-------------------------------------------------------------------------------------------------------------------|
| %ON      | $\left. \begin{array}{l} \text{write-ereignis} \\ \text{ereignis} \end{array} \right\} \quad [ <\text{subkdo}> ]$ |

Ohne *subkdo*-Operanden setzt AID das *subkdo* <%STOP> ein.

Das *subkdo* eines %ON für ein bereits angemeldetes *ereignis* überschreibt nicht das bestehende *subkdo*, sondern das neue *subkdo* wird vor das bestehende gekettet. Das bedeutet, Subkommandos zum selben *ereignis* werden nach dem LIFO-Prinzip bearbeitet.

Dies gilt nicht für *write-ereignis*. Die Eingabe eines neuen *write-ereignis* überschreibt ein bereits bestehendes.

Ein eingetragenes Ereignis gilt, bis es mit %REMOVE gelöscht wird oder bis zum Programmende.

Für %ON muß die Basisqualifikation E=VM eingestellt sein (siehe %BASE).

%ON verändert den Programmzustand nicht.

write-ereignis

Mit dem Schlüsselwort %WRITE wird die Schreibüberwachung eingeschaltet. Dahinter wird in Klammern der zu überwachende Speicherbereich festgelegt. Wenn das Programm ein Byte innerhalb des festgelegten Bereichs verändert, wird der Programmablauf unterbrochen und *subkdo* ausgeführt. Die Unterbrechungsstelle liegt **hinter** dem Befehl, der schreibend zugegriffen hat.

Es kann immer nur ein *write-ereignis* definiert sein. Die Eingabe eines neuen *write-ereignis* überschreibt ein bestehendes. Andere Ereignisse können jedoch gleichzeitig angemeldet sein. Trifft ein *ereignis* gleichzeitig mit *write-ereignis* ein, so bearbeitet AID das Subkommando zu *write-ereignis* als erstes. Das *write-ereignis* löschen Sie mit %REMOVE %WRITE ohne Angabe der Speicherreferenz.

**Wechselwirkungen** bestehen zwischen %ON *write-ereignis* und anderen AID-Kommandos:

- 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**.
- AID **erkennt nicht** den schreibenden Zugriff eines Maschinenbefehls, auf dessen Adresse ein mit %INSERT gesetzter Testpunkt liegt.

Für eine lückenlose Schreibüberwachung empfiehlt es sich, alle %CONTROL*n*- und %INSERT-Kommandos mit %REMOVE zu löschen. Einen eventuell noch eingetragenen %TRACE löschen Sie, wenn Sie nach dem %ON fortfahren mit %RESUME oder %TRACE 1 %INSTR.

Der zu überwachende Speicherbereich kann jedes, wie auch immer angesprochene Speicherobjekt sein, sofern es innerhalb des vom Programm belegten Speicherbereichs liegt. Register z.B. können nicht überwacht werden. Der zu überwachende Bereich, darf **64 KB nicht überschreiten**, sonst wird eine Fehlermeldung ausgegeben.

Wenn bei einem Programm mit Überlagerungsstruktur die Adresse des angegebenen Speicherobjekts überladen wird, wird anschließend der entsprechende Bereich des neu geladenen Programmteils überwacht.

Beim Testen von ESA-Programmen können nur Speicherobjekte im Programmraum überwacht werden.

*write-ereignis*-OPERAND -----

```

%WRITE([.][qua.]) {
 {C=csect | COM=common
 V'f...f'
 schlüsselwort
 kompl-speicherref
}
}

```

-----

- Steht der Punkt an führender Stelle, ist er das Kennzeichen für eine *vorqualifikation*. Sie muß mit einem vorhergehenden %QUALIFY definiert worden sein. Aufeinanderfolgende Qualifikationen werden durch einen Punkt getrennt. Außerdem muß zwischen der letzten *qualifikation* und dem anschließenden Operandenteil ein Punkt stehen.

qua Eine oder mehrere Qualifikationen geben Sie an, wenn Sie ein Speicherobjekt ansprechen wollen, das nicht im aktuellen AID-Arbeitsbereich liegt oder darin nicht eindeutig ist. Sie geben nur die Qualifikationen an, die zur eindeutigen Ansprache genügen.

E={VM | Dn}

geben Sie nur an, wenn für den zu überwachenden Speicherbereich die aktuelle Basisqualifikation nicht gelten soll.

CTX=kontext

geben Sie nur an, um eine CSECT oder einen COMMON im Programmsystem eindeutig ansprechen zu können.

[L=ladeeinheit.][O=objektmodul.]

geben Sie nur an, wenn Sie eine CSECT oder einen COMMON ansprechen wollen, deren Name im Programmsystem nicht eindeutig ist. Sie geben nur die L- und/oder O-Qualifikation an, die zur eindeutigen Ansprache nötig ist.

{C=csect | COM=common}

Endet die Adressierung mit einer C/COM-Qualifikation, so wird der ganze ausgewählte Programm-Abschnitt überwacht.

V'f...f' ist eine virtuelle Adresse. Damit werden 4 Bytes ab der angegebenen Adresse überwacht.

kompl-speicherref

bezeichnet den Überwachungsbereich ab einer errechneten Adresse in der zugehörigen impliziten oder angegebenen Länge. Folgende Operationen können in *kompl-speicherref* vorkommen (siehe AID-Basishandbuch, Kapitel 7 [1] und wegen Einschränkungen beim Testen von ESA-Programmen vorliegendes Handbuch, Kapitel 4, Maschinencode-spezifische Adressierung):

- Adreßversatz (•)
- indirekte Adressierung (->)
- Typmodifikation (%A, %S, %SX)
- Längenmodifikation (%L(...), %L=(ausdruck), %Ln)
- Adreßselektion (%@(...))

**schlüsselwort**

legt einen Speicherbereich durch Angabe eines der folgenden Schlüsselwörter (siehe AID-Basishandbuch, Kapitel 9 [1]) fest. Bei XS- bzw. ESA-Anlagen bezeichnen Sie mit %CLASS6 und %CLASS6BELOW denselben Adreßraum.

- %CLASS6                    Klasse-6-Speicher
- %CLASS6BELOW           Klasse-6-Speicher, unterhalb der 16MB-Grenze
- %CLASS6ABOVE           Klasse-6-Speicher, oberhalb der 16MB-Grenze

**ereignis**

Mit einem Schlüsselwort legen Sie fest, bei welchem Ereignis (normale oder abnormale Programmbeendigung, Supervisor-Call, Programmfehler etc.) AID das angegebene *subkdo* bearbeiten soll.

Auf ein Ereignis, daß mit einer STXIT-Routine bearbeitet wurde, kann anschließend nicht noch mit einem zu diesem *ereignis* vereinbarten *subkdo* reagiert werden.

Wird ein Subkommando zum Ereignis %ANY ausgeführt, entfällt bei der anschließenden Beendigung des Programms die Abfrage, ob ein Dump ausgegeben werden soll. Bei Bedarf müssen Sie im Subkommando mit /CREATE-DUMP selbst die Ausgabe des Dump anstoßen.

Wenn mehrere %ON-Kommandos mit unterschiedlichen *ereignis*-Vereinbarungen gleichzeitig aktiv sind und auch zutreffen, bearbeitet AID die zugehörigen Subkommandos in der Reihenfolge, in der die Schlüsselwörter in der Ereignis-Tabelle aufgeführt sind. Treffen verschiedene %TERM-Ereignisse zu, werden die zugehörigen Subkommandos nach dem LIFO-Prinzip bearbeitet. Zur Auswahl der geeigneten SVC-Nummern oder Ereigniscodes finden Sie weitere Informationen im Handbuch „Makroaufrufe an den Ablaufteil“ [7].

Wenn ein *write-ereignis* gleichzeitig mit einem anderen *ereignis* eintritt, wird zuerst das Subkommando zu *write-ereignis* bearbeitet.

| <i>ereignis</i> | <i>subkdo</i> wird bearbeitet:                                                                                                                                                                                               |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| %ERRFLG(zzz)    | nach Auftreten eines Fehlers mit dem Ereigniscode zzz und<br>vor Abbruch des Programms.                                                                                                                                      |
| %INSTCHK        | nach Auftreten eines Adressierungsfehlers, eines ungültigen<br>Systemaufrufs (SVC), nicht decodierbaren Operations-<br>Codes, Seitenwechsel-Fehlers oder einer privilegierten<br>Operation und<br>vor Abbruch des Programms. |
| %ARTHCHK        | nach Auftreten eines Datenfehlers, Divisionsfehlers, Expo-<br>nenten-Überlaufs oder einer Mantisse Null und<br>vor dem Abbruch des Programms.                                                                                |

|                   |                                                                                                                                                                                        |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>ereignis</i>   | <i>subkdo</i> wird bearbeitet:                                                                                                                                                         |
| %ABNORM           | nach Auftreten eines der Fehler, die mit den vorher beschriebenen Ereignissen erfaßt werden, sowie eines DMS-Errors (ab BS2000 V10) und %ILLSTX.                                       |
| %ERRFLG           | nach Auftreten eines Fehlers mit beliebigem Ereigniscode.                                                                                                                              |
| %SVC(z...z)       | vor Ausführung des Systemaufrufs (SVC) mit der angegebenen Nummer.                                                                                                                     |
| %SVC              | vor Ausführung eines beliebigen Systemaufrufs (SVC).                                                                                                                                   |
| %LPOV(x...x)      | nach dem Laden des Moduls bzw. Segmentes mit dem angegebenen Namen                                                                                                                     |
| %LPOV             | nach dem Laden eines beliebigen Moduls / Segmentes (der Name wird mit %D %LINK ausgegeben).                                                                                            |
| %TERM(N[ORMAL])   | vor normaler Programmbeendigung                                                                                                                                                        |
| %TERM(A[BNORMAL]) | vor abnormaler Beendigung des Programms, jedoch                                                                                                                                        |
|                   | nach Ausgabe eines Speicherabzuges                                                                                                                                                     |
| %TERM(D[UMP])     | vor Ausgabe eines Speicherabzuges mit anschließender Programmbeendigung                                                                                                                |
| %TERM(S[TEP])     | vor Beendigung eines Programms mit anschließender Verzweigung innerhalb von Prozeduren.                                                                                                |
| %TERM             | vor Beendigung eines Programms durch alle vorher beschriebenen %TERM-Ereignisse.                                                                                                       |
| %ANY              | vor der Beendigung des Programms aufgrund eines Programmfehlers bzw. durch die oben beschriebenen %TERM-Ereignisse, oder aufgrund 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).                                                                        |

zzz kann in zwei verschiedenen Formaten angegeben werden:

n maximal dreistellige vorzeichenlose Dezimalzahl

#'ff' zweistellige Sedezimalzahl

Für den Wert von zzz gilt:  $1 \leq zzz \leq 255$

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

x...x ist der Name eines Segments /einer Ladeeinheit mit maximal 32 alphanumerischen Zeichen.

subkdo

wird immer dann bearbeitet, wenn im Programmablauf das vereinbarte *ereignis* eintritt. Wird der Operand *subkdo* nicht angegeben, so setzt AID ein <%STOP> ein.

Vollständig beschrieben finden Sie *subkdo* im AID-Basishandbuch, Kapitel 6 [1].

```
subkdo-OPERAND -----
<[subkdoname:] [(bedingung):] [{ AID-kommando } { ;... }]>
 { BS2000-kommando }
```

Das Subkommando kann einen Namen, eine Bedingung und einen Kommandoteil enthalten. Zu jedem Subkommando gehört ein Durchlaufzähler. Der Kommandoteil kann aus einem einzelnen Kommando oder einer Kommandofolge bestehen, er kann AID- und BS2000-Kommandos und Kommentare enthalten.

Wenn das Subkommando aus einem Namen oder einer Bedingung besteht, aber der Kommandoteil fehlt, erhöht AID beim Eintreten des vereinbarten Ereignisses nur den Durchlaufzähler.

*subkdo* überschreibt nicht ein bestehendes Subkommando zu demselben *ereignis*, sondern das neue Subkommando wird vor das bestehende gekettet. In *subkdo* sind die Kommandos %CONTROL*n*, %INSERT und %ON zugelassen. Sie können damit eine Schachtelung über maximal 5 Stufen vornehmen. Ein Beispiel dazu finden Sie in der %INSERT-Beschreibung.

Die Kommandos in einem *subkdo* werden nacheinander ausgeführt. Danach wird das Programm fortgesetzt. Die Kommandos zur Ablaufsteuerung verändern auch in einem Subkommando sofort den Programmzustand. Sie brechen *subkdo* ab und setzen das Programm fort (%CONTINUE, %RESUME, %TRACE) oder halten es an (%STOP). Sie sind nur als letztes Kommando in einem *subkdo* sinnvoll, da nachfolgende Kommandos im Subkommando nicht mehr ausgeführt werden. Auch ein Löschen des gerade aktiven Subkommandos mit %REMOVE ist nur als letztes Kommando in *subkdo* sinnvoll.

### Beispiele

1. %ON %ERRFLG (108)  
 %ON %ERRFLG ('6C')  
 Beide Angaben überwachen den gleichen Programm-Fehler (Mantisse gleich Null), bei dem das Programm angehalten wird.
2. %ON %LPOV <%DISPLAY %LINK; %CONTINUE>  
 Nach jedem Laden eines Segments wird der Name des nachgeladenen Segments ausgegeben, und das Programm wird fortgesetzt.
3. %ON %ERRFLG <%D %AUD1; %STOP>  
 Bei Auftreten eines Fehlers mit beliebigem Fehlergewicht soll die P1-Audit-Tabelle ausgegeben und das Programm angehalten werden (siehe Benutzer-Kommandos (SDF-Format) [6]).

4. `%ON %WRITE(V'100') <%DA 4 FROM %PC->.(-6); %DA 4 FROM %PC->.(-4);%STOP>`  
Jedesmal, wenn im Programm eines der 4 Bytes von V'100' bis V'103' verändert wurde, wird das Programm angehalten und zweimal 4 disassemblierte Befehle werden ausgegeben. Einmal ab der Adresse, die 6 Bytes vor der Unterbrechungsstelle liegt und einmal ab der Adresse, die 4 Bytes vor der Unterbrechungsstelle liegt. Die Unterbrechungsstelle selbst liegt hinter dem Befehl, der den schreibenden Zugriff ausführte. Danach wird das Programm angehalten und eine STOP-Meldung ausgegeben.
5. `%ON %WRITE(V'200'%L2) <(V'200'%L2 EQ X'FFFF'):%STOP>`  
AID überwacht Byte V'200' und V'201' und wenn der Inhalt der beiden Bytes X'FFFF' ist, wird das Programm angehalten und eine STOP-Meldung ausgegeben.

## %OUT

Mit %OUT können Sie für die Ausgabe-Kommandos %DISASSEMBLE, %DISPLAY, %HELP, %SDUMP und %TRACE festlegen, über welche Medien die Daten ausgegeben werden sollen und ob in der Ausgabe Zusatzinformationen enthalten sein sollen.

- Mit *ziel-kommando* bezeichnen Sie das Ausgabe-Kommando, für das Sie *medium-u-menge* festlegen wollen.
- Mit *medium-u-menge* geben Sie an, welche Ausgabe-Medien verwendet und ob Zusatzinformationen ausgegeben werden sollen.

---

| Kommando | Operand                                 |
|----------|-----------------------------------------|
| %OUT     | [ziel-kommando [medium-u-menge][,...] ] |

---

Bei %DISPLAY und %HELP können Sie einen *medium-u-menge*-Operanden angeben, der für diese Kommandos die Vereinbarungen des %OUT-Kommandos vorübergehend außer Kraft setzt. %DISASSEMBLE und %TRACE haben keinen eigenen *medium-u-menge*-Operanden, ihre Ausgaben können Sie nur über %OUT steuern.

Bevor Sie mit %OUT das Ausgabemedium Datei wählen, müssen Sie die Datei mit %OUTFILE einem Linknamen zuweisen und öffnen; ansonsten legt AID eine Standard-Ausgabedatei mit dem Namen AID.OUTFILE.Fn an.

Die Vereinbarungen mit %OUT gelten bis sie durch ein neues %OUT-Kommando überschrieben werden oder bis /LOGOFF.

Ein %OUT-Kommando ohne Operanden setzt für alle *ziel-kommandos* den Standardwert T=MAX ein.

%OUT darf nur als Einzelkommando eingegeben werden, es darf nicht in einer Kommando- oder einem Subkommando stehen.

%OUT verändert den Programmzustand nicht.

ziel-kommando

bezeichnet das Kommando, für das die Vereinbarungen gelten sollen. Jeweils eines der folgenden Kommandos kann hier angegeben werden:

```
{
%D[IS]A[SSEMBLE]
%D[IS]P[LAY]
%H[E]L[P]
%SD[U]M[P]
%T[R]A[C]E
}
```

## medium-u-menge

legt für *ziel-kommando* fest, über welches oder über welche Medien die Ausgabe erfolgen soll und ob AID Zusatzinformationen ausgeben soll.

Wird der *medium-u-menge*-Operand nicht angegeben, so gilt für *ziel-kommando* der Standardwert T=MAX.

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

$$\left\{ \begin{array}{l} \underline{T} \\ \underline{H} \\ \underline{F}_n \\ \underline{P} \end{array} \right\} = \left\{ \begin{array}{l} \underline{MAX} \\ \underline{MIN} \end{array} \right\}$$

Der Operand *medium-u-menge* ist ausführlich im AID-Basishandbuch, Kapitel 8 [1] beschrieben.

|                      |                                                                                                             |
|----------------------|-------------------------------------------------------------------------------------------------------------|
| <u>T</u>             | Terminal-Ausgabe                                                                                            |
| <u>H</u>             | Hardcopy-Ausgabe (schließt die Terminal-Ausgabe ein und kann nicht gemeinsam mit <i>T</i> angegeben werden) |
| <u>F<sub>n</sub></u> | Datei-Ausgabe                                                                                               |
| <u>P</u>             | Ausgabe nach SYSLST                                                                                         |
| <u>MAX</u>           | Ausgabe mit Zusatzinformationen.                                                                            |
| <u>MIN</u>           | Ausgabe ohne Zusatzinformationen.                                                                           |

### Beispiel

1. %OUT %DISPLAY T=MIN, F1=MAX  
Datenausgaben des Kommandos %DISPLAY sollen auf dem Terminal in Kurzform und parallel dazu in die Datei mit dem Linknamen F1 mit Zusatzinformationen ausgegeben werden.
2. %OUT %DISPLAY  
Für das Kommando %DISPLAY wird festgelegt, daß bisherige Vereinbarungen zur Ausgabe von Daten gelöscht werden, und daß der Standardwert T=MAX gilt.

## %OUTFILE

Mit %OUTFILE können Sie den AID-Linknamen F0 bis F7 Ausgabedateien zuweisen oder Ausgabedateien schließen. In diese Dateien können Sie die Ausgaben der Kommandos %DISASSEMBLE, %DISPLAY, %HELP, %SDUMP und %TRACE schreiben lassen, indem Sie das entsprechende %OUT-Kommando eingeben oder den entsprechenden *medium-umenge*-Operanden angeben. Falls eine Datei noch nicht existiert, wird sie durch AID katalogisiert und eröffnet.

- Mit *link* wählen Sie den Linknamen für die Datei aus, die katalogisiert und geöffnet oder geschlossen werden soll.
- Mit *datei* weisen Sie dem Linknamen einen Dateinamen zu.

| Kommando | Operand           |
|----------|-------------------|
| %OUTFILE | [link [ = datei]] |

Ohne den *datei*-Operanden veranlassen Sie AID, die mit *link* bezeichnete Datei zu schließen. So können Sie auch während des Testverlaufs einen Zwischenstand der Datei ausdrucken.

Ein %OUTFILE ohne Operanden schließt alle offenen AID-Ausgabedateien. Wenn Sie eine AID-Ausgabedatei nicht explizit mit %OUTFILE schließen, bleibt sie geöffnet bis /LOGOFF.

Ohne Verwendung von %OUTFILE haben Sie zwei Möglichkeiten, AID-Ausgabedateien einzurichten und zuzuweisen:

1. Sie geben ein /SET-FILE-LINK-Kommando für einen noch nicht belegten Linknamen *Fn*. Dann eröffnet AID diese Datei beim ersten Ausgabekommando für diesen Linknamen.
2. Sie überlassen AID das Einrichten, Zuweisen und Eröffnen. Dann verwendet AID Standard-Datei-Namen mit folgendem Aufbau: AID.OUTFILE.Fn entsprechend dem Linknamen *Fn*.

%OUTFILE verändert den Programmzustand nicht.

link

bezeichnet einen der AID-Linknamen für Ausgabedateien und hat das Format  $F_n$ , wobei  $n$  eine Zahl mit einem Wert  $0 \leq n \leq 7$  ist.

Die mit %MOVE erzeugten REPs werden stets in die Ausgabedatei mit dem Linknamen  $F_6$  geschrieben. Sie sollten diesen Linknamen nicht gleichzeitig für andere Ausgaben verwenden (siehe %AID und %MOVE).

datei

gibt den vollqualifizierten Dateinamen an, mit dem AID die Ausgabedatei katalogisiert und eröffnet. Mit einem %OUTFILE ohne *datei*-Operand wird die dem Linknamen  $F_n$  zugewiesene Datei geschlossen.

## %QUALIFY

Mit %QUALIFY definieren Sie Qualifikationen oder eine Adresse, auf die Sie sich im Adreß-Operanden eines anderen Kommandos durch Voranstellen eines Punktes beziehen können.

Diese verkürzte Schreibweise ist immer dann sinnvoll, wenn Sie mehrfach Adressen ansprechen wollen, die nicht im aktuellen AID-Arbeitsbereich liegen, oder wenn Sie eine Adresse mehrfach als Anfangsadresse für einen Adreßversatz (•) einsetzen wollen.

- Mit *vorqualifikation* legen Sie Qualifikationen oder eine Adresse fest, die Sie in nachfolgenden Kommandos durch Voranstellen eines Punktes übernehmen möchten.

| Kommando | Operand            |
|----------|--------------------|
| %QUALIFY | [vorqualifikation] |

Eine mit %QUALIFY vereinbarte *vorqualifikation* gilt bis sie durch ein %QUALIFY mit neuer *vorqualifikation* überschrieben wird, bis sie durch ein %QUALIFY ohne Operanden aufgehoben wird oder bis /LOGOFF.

Bei der Eingabe eines %QUALIFY wird das Kommando nur syntaktisch überprüft. Ob dem angegebenen Linknamen in einer E-Qualifikation eine Dump-Datei zugewiesen und ob die angegebenen Bereichs-Qualifikationen geladen bzw. die Objekt-Strukturliste dazu vorhanden ist, wird erst bei der Ausführung darauf folgender Kommandos geprüft, wenn die Angaben aus *vorqualifikation* in die Adressierung einbezogen werden.

Die Vereinbarungen des %QUALIFY werden **nur** von **nachfolgend** eingegebenen Kommandos übernommen. Auf die Subkommandos in %CONTROL, %INSERT oder %ON, die vorher eingegeben wurden, hat ein neuer %QUALIFY keine Auswirkungen, auch wenn die Subkommandos erst danach ausgeführt werden.

Bei der Eingabe des %QUALIFY müssen Sie die aktuelle Einstellung für die Behandlung der Groß-/Kleinschreibung (%AID LOW={ON|OFF}) beachten, damit die *vorqualifikation* richtige Ergänzungen in den Adreß-Operanden nachfolgender Kommandos erzeugt.

%QUALIFY kann nur als Einzelkommando eingegeben werden, es darf nicht in einer Kommandofolge oder einem Subkommando stehen.

%QUALIFY verändert den Programmzustand nicht.

vorqualifikation

besteht aus einer ein- oder mehrstufigen Qualifikation und/oder der Anfangsadresse für einen Adreßversatz. In den Adreß-Operanden nachfolgender AID-Kommandos können Sie sich durch Voranstellung eines Punktes auf die in %QUALIFY definierte *vorqualifikation* beziehen.

vorqualifikation-OPERAND - - - - -

$$\left\{ \left[ E = \left\{ \begin{array}{l} \text{VM} \\ \text{Dn} \end{array} \right\} \right] \left[ [ \bullet ] \text{CTX} = \text{kontext} \right] \left[ [ \bullet ] \text{L} = \text{ladeeinheit} \right] \left[ [ \bullet ] \text{O} = \text{objektmodul} \right] \left[ [ \bullet ] \left\{ \begin{array}{l} \text{C} = \text{csect} \\ \text{COM} = \text{common} \end{array} \right\} \right] \right\}$$

$$\left\{ \left[ E = \{ \text{VM} | \text{Dn} \} \right] \left[ [ \bullet ] \left\{ \begin{array}{l} \text{ALET} = \{ \text{X}'f \dots f' | \%n\text{AR} | \%n\text{G} \} \\ \text{SPID} = \text{X}'f \dots f' \end{array} \right\} \right] \left[ [ \bullet ] \left\{ \begin{array}{l} \text{V}'f \dots f' \\ \text{kompl-speicherref} \end{array} \right\} \right] \right\}$$

- - - - -

- Aufeinanderfolgende Qualifikationen werden durch einen Punkt getrennt. Außerdem muß zwischen der letzten *qualifikation* und dem anschließenden Operandenteil ein Punkt stehen.

E={VM | Dn}

bezeichnet den virtuellen Speicher (VM) oder eine Dump-Datei mit dem Linknamen *Dn* (siehe %BASE).

{ALET={X'f...f' | %nAR | %nG} | SPID=X'f...f'}

bezeichnet einem Datenraum.

CTX=kontext

bezeichnet einen Kontext.

[L=ladeeinheit.] [O=objektmodul.]

bezeichnet die Ladeeinheit und/oder den Objektmodul.

{C=csect | COM=common}

bezeichnet eine CSECT oder einen COMMON.

V'f...f'

bezeichnet eine virtuelle Adresse, die als Anfangsadresse für einen Adreßversatz vordefiniert werden kann.

**kompl-speicherref**

bezeichnet eine errechnete Anfangsadresse für einen Adreßversatz. Folgende Operationen können darin vorkommen (siehe Kapitel 4.2 Speicherreferenzen und AID-Basishandbuch, Kapitel 7.2.4 [1]):

- Adreßversatz (•)
- indirekte Adressierung (->)
- Typmodifikation (%A, %S, %SX)
- Längenmodifikation (%L(...), %L=(ausdruck), %Ln)
- Adreßselektion (%@(...))

**Beispiele**

1. %QUALIFY E=VM.L=MOD  
%DISPLAY .C=CS

Im Kommando %DISPLAY wird vor den führenden Punkt der String E=VM.L=MOD eingesetzt, d.h. aus dem Speicherbereich des geladenen Programms wird die CSECT CS im Lademodul MOD ausgegeben. Die Qualifikation E=VM ist nur dann erforderlich, wenn der aktuelle AID-Arbeitsbereich in einer Dump-Datei liegt.

2. %QUALIFY V'100'  
%MOVE .0 INTO .#'A0'

Ohne das vorhergehende %QUALIFY müßte das %MOVE-Kommando so aussehen:  
%MOVE V'100'.0 INTO V'100'.#'A0'

3. %QUALIFY C=CSECT001.16  
%D .24

Ohne das vorhergehende %QUALIFY müßte der %DISPLAY so aussehen:  
%D C=CSECT001.16.24

4. %BASE E=VM  
%QUALIFY E=D1.%1G->  
%D .0 %FL4

Bei der Ausführung des %DISPLAY wird der Inhalt von AID-Register %1G als Adresse im Dump verwendet, der Adreßversatz dazuaddiert und ab der so errechneten Adresse 4 Bytes aus dem Dump als Ganzzahl interpretiert ausgegeben.

## %REMOVE

Mit %REMOVE heben Sie die Testvereinbarungen der Kommandos %CONTROL $n$ , %INSERT oder %ON auf.

- Mit *ziel* legen Sie fest, für welche Kommandos oder Kommandoteile die Vereinbarungen aufgehoben werden sollen.

| Kommando  | Operand |
|-----------|---------|
| %REM[OVE] | ziel    |

%REMOVE verändert den Programmzustand nicht.

ziel

bezeichnet entweder ein Kommando, für das alle Vereinbarungen gelöscht werden sollen, oder einen *testpunkt*, der gelöscht werden soll, oder ein *ereignis*, das nicht mehr überwacht werden soll, oder das zu löschende Subkommando. Liegt *ziel* in einem geschachtelten Subkommando und ist somit noch nicht eingetragen, kann es auch nicht gelöscht werden.

ziel-OPERAND - - - - -

|                                                                                                                 |                  |
|-----------------------------------------------------------------------------------------------------------------|------------------|
| {<br>%C[ONTROL]   %C[ONTROL] $n$<br>%I[N]SERT   testpunkt<br>%O[N]   %W[R]ITE   ereignis<br>%.[subkdoname]<br>} | }<br>}<br>}<br>} |
|-----------------------------------------------------------------------------------------------------------------|------------------|

- - - - -

### %C[ONTROL]

Die Vereinbarungen aller eingetragenen %CONTROL $n$  werden gelöscht.

### %C[ONTROL] $n$

Der %CONTROL $n$  mit der angegebenen Nummer ( $1 \leq n \leq 7$ ) wird gelöscht.

**%IN[*SERT*]**

Alle eingetragenen Testpunkte werden gelöscht.

**testpunkt**

Der angegebene *testpunkt* wird gelöscht. *testpunkt* wird wie bei %INSERT angegeben.

Innerhalb des eigenen Subkommandos kann der Testpunkt auch mit %REMOVE %PC-> gelöscht werden, da der Befehlszähler (%PC) zu diesem Zeitpunkt die Adresse des Testpunkts enthält.

**%ON**

Alle eingetragenen Ereignisse werden gelöscht.

**%WRITE**

Das *write-ereignis* wird gelöscht.

**ereignis**

Das angegebene *ereignis* wird gelöscht. *ereignis* wird wie bei %ON mit einem Schlüsselwort angegeben. Die *ereignis*-Tabelle mit den Schlüsselwörtern und den Erläuterungen der einzelnen Ereignisse steht in der %ON-Beschreibung.

Für die Ereignisse %ERRFLG(*zzz*), %SVC(*zzz*) und %LPOV(*zzz*) gilt:

%REMOVE *ereignis*(*zzz*) löscht nur das Ereignis mit der angegebenen Nummer.  
%REMOVE *ereignis* ohne Angabe einer Nummer löscht alle Ereignisse der entsprechenden Gruppe.

**%•[*subkdoname*]**

löscht das Subkommando eines %CONTROL*n* oder %INSERT mit *subkdoname*.

%• ist die Kurzform für einen Subkommando-Namen, die nur innerhalb des Subkommandos verwendet werden kann. %REMOVE %• löscht folglich das gerade ausgeführte Subkommando. Diese Angabe ist nur als letztes Kommando in einem Subkommando sinnvoll, da nachfolgende Kommandos von *subkdo* nicht mehr ausgeführt werden.

Da %CONTROL*n* nicht gekettet werden kann, wird auch der zugehörige %CONTROL*n* gelöscht. Das Löschen des Subkommandos entspricht folglich einer Löschung des %CONTROL*n* mit Angabe der Nummer.

An einem *testpunkt* des Kommandos %INSERT können dagegen mehrere Subkommandos gekettet sein. Mit %REMOVE %•[*subkdoname*] löschen Sie ein einzelnes Subkommando aus einer Kette, weitere Subkommandos zum selben *testpunkt* bleiben dagegen bestehen (siehe AID-Basisbuch, Kapitel 6 [1]). War zum *testpunkt* nur das Subkommando mit *subkdoname* eingetragen, so wird auch der *testpunkt* gelöscht.

Für %ON ist %REMOVE %•[*subkdoname*] nicht zugelassen.

## Beispiele

1. %C1 %BAL <CTL1: %D %.>  
%REM %C1  
%REM %.CTL1

Die beiden %REMOVE-Kommandos haben dieselbe Wirkung: %C1 wird gelöscht.

2. %IN V'100' <SUB1: %D %14, %15>  
%IN V'100' <SUB2: %D %PC; %REM %.>  
...  
%REM V'100'

Wenn der Testpunkt V'100' erreicht wird, wird der Befehlszähler ausgegeben. Danach wird das Subkommando SUB2 gelöscht. Dieses Subkommando wird also nur ein einziges Mal ausgeführt. Dann werden Register 14 und 15 ausgegeben und das Programm wird fortgesetzt. Immer wenn danach der Programmablauf an den Testpunkt V'100' kommt, wird Subkommando SUB1 ausgeführt. Mit %REM V'100' wird später der Testpunkt gelöscht. Ein %REM %•SUB1 würde dasselbe bewirken, denn zum Testpunkt V'100' ist nur noch dieses Subkommando eingetragen.

## **%RESUME**

Mit %RESUME starten Sie das geladene Programm oder setzen es an der unterbrochenen Stelle fort. Das Programm läuft ohne Ablaufverfolgung.

%RESUME beendet alle aktiven %TRACE-Kommandos, %CONTINUE hingegen hat keine Auswirkungen auf %TRACE.

---

|          |                   |
|----------|-------------------|
| Kommando | Operand <i>ff</i> |
|----------|-------------------|

---

%R[ESUME]

---

Steht %RESUME in einer Kommandofolge oder in einem Subkommando, werden nachfolgende Kommandos nicht mehr ausgeführt.

Steht in einem Subkommando nur das Kommando %RESUME, wird der Durchlaufzähler erhöht und ein eventuell aktiver %TRACE gelöscht.

%RESUME verändert den Programmzustand.

## %SDUMP

Mit %SDUMP geben Sie im maschinennahen Testen die aktuelle Aufrufhierarchie aus.

- Mit *%NEST* geben Sie an, daß AID die Programmnamen der aktuellen Aufrufhierarchie ausgeben soll.
- Mit *medium-u-menge* geben Sie an, welche Ausgabe-Medien AID verwenden soll und ob Zusatzinformationen ausgegeben werden sollen. Mit diesem Operanden setzen Sie eine mit %OUT getroffene Vereinbarung für das aktuelle %SDUMP-Kommando außer Kraft.

| Kommando | Operand                                            |
|----------|----------------------------------------------------|
| %SD[UMP] | %NEST                      [medium-u-menge][, ...] |

Um das Kommando anwenden zu können, muß beim Übersetzen des Programms eine ESD- bzw. ESV-Liste erzeugt und daraus beim Binden eine Objektstrukturliste bzw. ein Externadreßbuch aufgebaut worden sein. Außerdem muß der Modul AIDIT0 vom Runtime-System geladen sein. Das fragen Sie mit %DISPLAY C=AIDIT0 ab.

### %NEST

Mit dem Schlüsselwort %NEST stoßen Sie die Ausgabe der aktuellen Aufrufhierarchie an. Die Ausgabe von AID unterscheidet sich hier deutlich von der Ausgabe des %SDUMP %NEST bei geladem LSD. Es fehlen z.B. die Source-Referenzen.

### medium-u-menge

legt fest, über welches oder über welche Medien die Ausgabe erfolgen soll und ob zusätzlich Informationen neben der aktuellen Aufrufhierarchie ausgegeben werden sollen.

Ohne diesen Operanden und ohne eine Vereinbarung mit dem %OUT-Kommando arbeitet AID mit der Voreinstellung T=MAX.

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

$$\left\{ \begin{array}{l} T \\ H \\ \\ Fn \\ P \end{array} \right\} = \left\{ \begin{array}{l} MAX \\ \\ \\ MIN \end{array} \right\}$$

medium-u-menge ist ausführlich im AID-Basishandbuch, Kapitel 8 [1] beschrieben.

- T Terminal-Ausgabe
- H Hardcopy-Ausgabe (schließt die Terminal-Ausgabe ein und kann nicht gemeinsam mit *T* angegeben werden)
- Fn Datei-Ausgabe
- P Ausgabe nach SYSLST
  
- MAX Ausgabe mit Zusatzinformationen.
- MIN Ausgabe ohne Zusatzinformationen.

### Beispiel

Die nachfolgenden Protokollzeilen zeigen zuerst die Ausgabe der Aufrufhierarchie eines C++ - Programms, wenn die LSD geladen sind. Anschließend wird die Bibliothek mit den LSD für den aktuellen Arbeitsbereich abgemeldet und die Aufrufhierarchie nochmal ausgegeben.

```

/STOPPED AT SRC_REF: 122, SOURCE: VPTR1.C , PROC: 70::A_1::f(void)
/%sd %nest
*** TID: 0028024F *** TSN: 0308 *****
/SRC_REF: 122 SOURCE: VPTR1.C PROC: 70::A_1::f(void) *****
/SRC_REF: 127 SOURCE: VPTR1.C PROC: vptrl_main(int) *****
/SRC_REF: 105 SOURCE: BCLB.C PROC: main *****
/ABSOLUT: VT17432T SOURCE: ICS$MAIO PROC: ICS$MAIO *****

```

```

/%symlib
/%sd %nest
/ABSOLUT: V*534A* SOURCE: VPTR1@ PROC: f *****
/ABSOLUT: VT58E0T SOURCE: VPTR1@O PROC: vptrl_main *****
/ABSOLUT: V*754* SOURCE: BCLB@ PROC: main *****
/ABSOLUT: V*17432* SOURCE: ICS$MAI@ PROC: ICS$MAI@ *****

```

## %SET

Mit %SET übertragen Sie Speicherinhalte oder AID-Literale auf Speicherstellen im geladenen Programm. Vor der Übertragung werden die Speichertypen von *sender* und *empfänger* auf Verträglichkeit geprüft. Der Inhalt von *sender* wird dem Speichertyp von *empfänger* angepaßt.

- Mit *sender* bezeichnen Sie die zu übertragende Speicherstelle, geben eine Länge, eine Adresse, ein Schlüsselwort oder ein AID-Literal an. *sender* kann im virtuellen Speicher des geladenen Programms oder in einer Dump-Datei liegen.
- Mit *empfänger* bezeichnen Sie die Speicherstelle, die überschrieben werden soll. *empfänger* kann nur im virtuellen Speicher des geladenen Programms liegen.

| Kommando | Operand               |
|----------|-----------------------|
| %[SET]   | sender INTO empfänger |

Im Gegensatz zum %MOVE überprüft AID beim %SET vor der Übertragung, ob der Speichertyp von *empfänger* mit dem von *sender* verträglich ist und ob der Inhalt von *sender* zu seinem Speichertyp paßt. Andernfalls lehnt AID eine Übertragung ab und gibt eine Fehlermeldung aus.

Ist *sender* länger als *empfänger*, wird er entsprechend seinem Speichertyp links oder rechts abgeschnitten, und AID gibt eine Warnung aus. *sender* und *empfänger* können sich überlappen.

Bei der numerischen Übertragung wird *sender* bei Bedarf in den Speichertyp von *empfänger* konvertiert, und der Inhalt von *sender* wird werterhaltend in *empfänger* abgelegt. Paßt der Wert von *sender* nicht vollständig in *empfänger*, gibt AID eine Warnung aus.

Mit dem Kommando %AID CHECK=ALL können Sie zur Kontrolle einen Änderungsdialog einschalten, der Ihnen vor Durchführung der Übertragung den alten und neuen Inhalt von *empfänger* zeigt und die Möglichkeit zum Abbruch des %SET gibt.

%SET verändert den Programmzustand nicht.

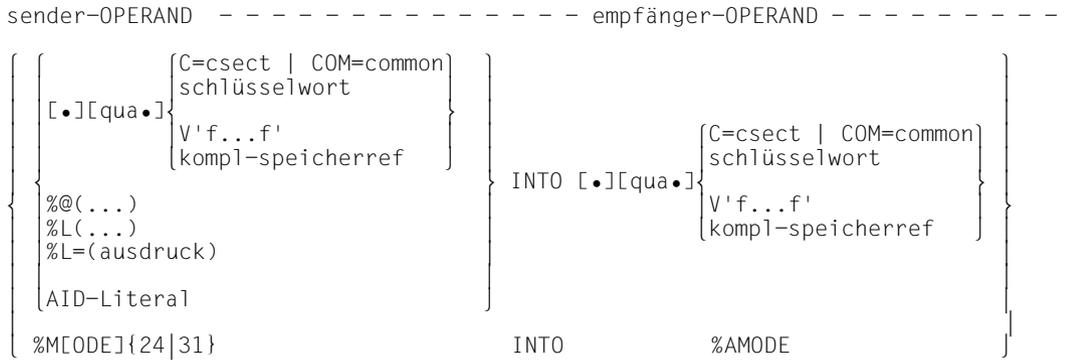


Für *sender* und *empfänger* können Sie eine virtuelle Adresse, eine C/COM-Qualifikation, eine komplexe Speicherreferenz oder ein Schlüsselwort für Mehrzweck-, Gleitpunkt-, AID-Register, Befehlszähler oder den Durchlaufzähler angeben.

Nur als *sender* können Sie Selektoren, AID-Literale und das Schlüsselwort %MODEn ange-

ben. Übertragen bzw. überschreiben Sie Befehlscode, kann das zu unerwünschten Ergebnissen führen, wenn *sender* oder *empfänger* in einem *control*- oder *trace-bereich* liegen, der symbolisch vereinbart wurde (siehe AID-Basishandbuch, Kapitel 12.2 [1]).

*sender* kann sowohl im virtuellen Speicher, als auch in einer Dump-Datei liegen, aber *empfänger* nur im virtuellen Speicher.



- Steht der Punkt an führender Stelle, ist er das Kennzeichen für eine *vorqualifikation*. Sie muß mit einem vorhergehenden %QUALIFY definiert worden sein. Aufeinanderfolgende Qualifikationen werden durch einen Punkt getrennt. Außerdem muß zwischen der letzten *qualifikation* und dem anschließenden Operandenteil ein Punkt stehen.

qua Eine oder mehrere Qualifikationen geben Sie an, wenn Sie ein Speicherobjekt ansprechen wollen, das nicht im aktuellen AID-Arbeitsbereich liegt oder darin nicht eindeutig ist. Sie geben nur die Qualifikationen an, die zur eindeutigen Ansprache genügen.

{E={VM | Dn} für *sender* | E=VM für *empfänger*}

Eine Basisqualifikation geben Sie nur an, wenn die aktuelle Basisqualifikation nicht gelten soll.

*sender* kann sowohl im virtuellen Speicher, als auch in einer Dump-Datei liegen. *empfänger* hingegen kann nur im virtuellen Speicher liegen.

{ALET={X'f...f' | %nAR | %nG} | SPID=X'f...f'}

geben Sie nur an, wenn Sie beim Testen von ESA-Programmen eine Adresse in einem Datenraum ansprechen wollen.

CTX=kontext

geben Sie nur an, um eine CSECT oder einen COMMON im Programmsystem eindeutig ansprechen zu können.

[L=ladeeinheit.][O=objektmodul.]

geben Sie nur an, wenn Sie eine CSECT oder einen COMMON ansprechen wollen, die im aktuellen Kontext ohne diese Qualifikationen nicht eindeutig adressierbar sind. Sie geben nur die Qualifikationen an, die zur eindeutigen Ansprache nötig sind.

{C=csect | COM=common}

bezeichnet eine CSECT oder einen COMMON.

Endet die Adressierung mit einer C/COM-Qualifikation, so wird der ganze ausgewählte Programm-Abschnitt übertragen (*sender*), bzw. von Anfang an überschrieben (*empfänger*).

V'f...f' ist eine virtuelle Adresse.

kompl-speicherref

Folgende Operationen können darin vorkommen (siehe AID-Basishandbuch, Kapitel 7.2.4 [1] und wegen Einschränkungen beim Testen von ESA-Programmen vorliegendes Handbuch, Kapitel 4.2, Maschinencode-spezifische Adressierung):

- Adreßversatz (•)
- indirekte Adressierung (->)
- Typmodifikation (%T(datenname), %X, %C, %P, %D, %F, %A, %S, %SX)
- Längenmodifikation (%L(...), %L=(ausdruck), %Ln)
- Adreßselektion (%@(...))

Mit einer expliziten Typ- oder Längen-Modifikation können Sie die Speichertypen von *sender* und *empfänger* einander anpassen. Mit einem Speichertyp unvereinbare Speicherinhalte jedoch werden von AID auch bei der Typmodifikation abgelehnt. Für jeden Operanden in einer komplexen Speicherreferenz darf der zugeordnete Speicherbereich durch einen Adreßversatz oder eine Längenmodifikation nicht überschritten werden, sonst führt AID das Kommando nicht aus und gibt eine Fehlermeldung aus.

Sind LSD-Sätze vorhanden, können Sie innerhalb von *kompl-speicherref* auch die im Quellprogramm definierten Daten- und Anweisungsnamen verwenden.

## schlüsselwort

bezeichnet ein Register, den Befehlszähler oder einen Durchlaufzähler. Im AID-Basishandbuch, Kapitel 9 [1] sind die impliziten Speichertypen der Schlüsselwörter angegeben.

|                |                                                                                                         |
|----------------|---------------------------------------------------------------------------------------------------------|
| %PC            | Befehlszähler (Program Counter)                                                                         |
| %n             | Mehrzweckregister, $0 \leq n \leq 15$                                                                   |
| %nD E          | Gleitpunktregister, einfache/doppelte Genauigkeit, $n = 0,2,4,6$                                        |
| %nQ            | Gleitpunktregister, vierfache Genauigkeit, $n = 0,4$                                                    |
| %nG            | AID-Mehrzweckregister, $0 \leq n \leq 15$                                                               |
| %nDG           | AID-Gleitpunktregister, $n = 0,2,4,6$                                                                   |
| %•[subkdoname] | Durchlaufzähler. Mit der Kurzform %• bezeichnen Sie den Durchlaufzähler des gerade aktiven Subkommandos |

Daneben können Sie noch die Schlüsselwörter für die Speicherklassen verwenden (siehe AID-Basishandbuch, Kapitel 10.5 [1]).

## %@(…)

Mit dem Adreßselektor können Sie die Anfangsadresse einer CSECT oder eines Speicherbereichs (%CLASS6, %CLASS6BELOW, %CLASS6ABOVE) oder von *kompl-speicherref* als *sender* verwenden.

Der Adreßselektor liefert eine Adreßkonstante, die Sie in einer komplexen Speicherreferenz vor einem Pointer-Operator (->) verwenden können (siehe AID-Basishandbuch, Kapitel 7.2.4.4 [1]).

## %L(…)

Mit dem Längenselektor können Sie die Länge einer CSECT oder eines Speicherbereichs (%CLASS6, %CLASS6BELOW, %CLASS6ABOVE) als *sender* verwenden.

Der Längenselektor liefert eine Ganzzahl, die Sie für die Berechnung eines Adreßversatzes oder in einer Längenfunktion verwenden können (siehe AID-Basishandbuch, Kapitel 7.2.4.4 [1]).

**Beispiel:** %L(C=CS1) die Länge der CSECT CS1 wird als *sender* verwendet.

## %L=(ausdruck)

Mit der Längenfunktion können Sie sich als *sender* einen Wert errechnen lassen. *ausdruck* wird gebildet aus dem Inhalt von Speicherreferenzen, Konstanten, Ganzzahlen und arithmetischen Operatoren. Nur Speicherreferenz-Inhalte vom Typ %F oder %A mit einer Länge  $\leq 4$  sind zugelassen.

Die Längenfunktion liefert als Ergebnis eine Ganzzahl, die Sie für die Berechnung eines Adreßversatzes, in einer weiteren Längenfunktion oder zur Längenmodifikation verwenden können (siehe AID-Basishandbuch, Kapitel 7.2.4.4 [1]).

**Beispiel:** %L=(%1) der Inhalt von Register 1 wird als *sender* verwendet.

## AID-Literal

Alle im AID-Basishandbuch, Kapitel 9 [1] beschriebenen Literale können Sie angeben.

Bitte beachten Sie die dort beschriebenen Anpassungen der AID-Literale an den jeweiligen *empfänger*:

|                                 |                   |
|---------------------------------|-------------------|
| {C'x...x'   'x...x'C   '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    |

## %M[ODE]{24 | 31}

legt den Adressierungsmodus fest.

## %AMODE

ist das Systeminformationsfeld, in dem der Adressierungsmodus eingetragen ist.

Mit der Übertragung des Schlüsselwortes %M{24|31} in das Systeminformationsfeld %AMODE ändern Sie den Adressierungsmodus des Testobjekts. Wenn Sie nur die Adreßinterpretation für die indirekte Adressierung (->) ändern wollen, geben Sie das entsprechende %AINT-Kommando ein.

Wenn Sie beim Test eines 31-Bit-Adressen-Programms den Befehlszähler ändern, während der 24-Bit-Modus eingestellt ist, müssen Sie darauf achten, daß das höchstwertige Byte den Wert X'00' hat.

### SET-Tabelle

Wie AID die Übertragung vornimmt, welche Speichertypen wie konvertiert werden und welche Übertragungen abgelehnt werden, entnehmen Sie der folgenden Tabelle:

| <i>sender</i>                                                             | <i>empfänger</i>                                      |      |     |
|---------------------------------------------------------------------------|-------------------------------------------------------|------|-----|
|                                                                           | %F %P %A<br>%n %nG<br>%PC %•[name]<br>%D %nE/D/Q %nGD | %C   | %X  |
| %F %P %A<br>{±}n<br>#'f...f'<br>%n %nG<br>%PC %•[name]<br>%D %nE/D/Q %nGD | num                                                   | -    | bin |
| {±}n.m<br>{±}mantE{±}exp                                                  | num                                                   | -    | -   |
| %C<br>C'x...x'                                                            | num(1)                                                | char | bin |
| %X<br>X'f...f'<br>B'b...b'                                                | bin                                                   | bin  | bin |

- bin Binäre Übertragung, linksbündig  
*sender* < *empfänger* rechts wird mit binären Nullen aufgefüllt  
*sender* > *empfänger* rechts wird abgeschnitten  
 Ein numerisches Literal entspricht bei der Übertragung in den Speichertyp %X einem Integerwert mit Vorzeichen in der Länge 4 Byte (%FL4), die binär übertragen werden.
- char Character-Übertragung, linksbündig  
*sender* < *empfänger* rechts wird mit Leerzeichen (X'40') aufgefüllt  
*sender* > *empfänger* rechts wird abgeschnitten
- num<sup>(1)</sup> Besteht ein Character-*sender* nur aus Ziffern und ist höchstens 18 Stellen lang, wird er numerisch übertragen, wenn *empfänger* vom Typ numerisch ist. Andernfalls lehnt AID eine Übertragung ab.
- num Numerische Übertragung, werterhaltend  
*sender* wird bei Bedarf dem Speichertyp von *empfänger* angepaßt.
- keine Übertragung  
 AID meldet die Unverträglichkeit der Speichertypen.

## Beispiele

1.

```

/%D V'798'
V'00000798' = M1BS + #'00000798'
00000798 (00000798) 012CF000 ...
/%SET V'798'%PL2 INTO %1G
/%D %1G, %1G%F
%1G = 0000000C
%1G = 12

```

Der Inhalt der ersten 2 Bytes an der Adresse V'798' werden als gepackte Zahl interpretiert und in das AID-Register %1G übertragen. Das AID-Register ist vom Typ %F (binär mit Vorzeichen), der Wert 12 wird entsprechend konvertiert.

2.

```

/%SET V'300'%PL1 INTO %1
/%SET V'400'%PL1 INTO %2
/%DISPLAY V'300', V'400'
V'00000300' = M1BS + #'00000300'
00000300 (00000300) 1C010080
V'00000400' = M1BS + #'00000400'
00000400 (00000400) 2D000000
/%DISPLAY %1, %2
%1 = 00000001
%2 = FFFFFFFE

```

Die beiden gepackten Inhalte an Adresse V'300' und V'400' entsprechen den Werten +1 und -2. Bei der Übertragung in ein Register werden die Werte entsprechend konvertiert.

3.

```

/%DISPLAY V'100', V'200'
V'00000100' = M1BS + #'00000100'
00000100 (00000100) F0000000 0...
V'00000200' = M1BS + #'00000200'
00000200 (00000200) 01020000
/%SET V'100'%CL1 INTO V'101'%CL2
/%SET V'200'%XL2 INTO V'202'%XL4
/%DISPLAY V'100', V'101'%CL2, V'200', V'202'%XL4
V'00000100' = M1BS + #'00000100'
00000100 (00000100) F0F04000 00 .
V'00000101' = M1BS + #'00000101'
00000101 (00000101) 0
V'00000200' = M1BS + #'00000200'
00000200 (00000200) 01020102
V'00000202' = M1BS + #'00000202'
00000202 (00000200) 01020000

```

Bei der Übertragung im Typ %C (character) wird der *empfänger* rechts mit Blanks (X'40') aufgefüllt. Bei der Übertragung im Typ %X (sedezimal) wird der *empfänger* rechts mit binär Null (X'00') aufgefüllt.

4.

```

/%SET %1 INTO V'100'%D
/%DISPLAY V'100', V'100'%D
V'00000100' = M1BS + #'00000100'
00000100 (00000100) 41100000
V'00000100' = M1BS + #'00000100'
00000100 (00000100)+.100000000000000000000000000000000000 E+001

```

In Register %1 ist der Wert 1 (X'00000001') enthalten. Dieser wird in den Speichertyp %D (Gleitpunkt) konvertiert und auf Adresse V'100' übertragen.

5.

```

/%SET X'00' INTO V'100'%L30

```

Da beim %SET im Gegensatz zum %MOVE *sender* entsprechend seinem Datentyp auf die Länge von *empfänger* ergänzt wird, werden ab V '100' 30 Bytes mit sedezimal Null überschrieben.

## %SHOW

Mit %SHOW können Sie sich informieren über die aktuellen Vereinbarungen zu einzelnen AID-Kommandos, wie gesetzte Testpunkte oder Ereignisse, feststellen, wie die letzte Eingabe eines Kommandos aussah und welches Kommando zuletzt eingegeben wurde. Ausserdem können Sie über den Subkommandonamen das Kommando anfordern, in dem es definiert wurde oder sich eine Liste aller eingetragenen Subkommandonamen mit dem zugehörigen Kommandotyp ausgeben lassen. Entsprechend der Vereinbarung zur Groß- oder Kleinschreibung im %AID-Kommando wird die Originaleingabe des Kommandos wiedergegeben oder der Eingabestring ist in Großbuchstaben umgesetzt.

- Mit *show-ziel* geben Sie ein Kommando, einen Subkommandonamen oder ein AID-Schlüsselwort für alle aktuellen Subkommandos an.

| Kommando | Operand     |
|----------|-------------|
| %SH[OW]  | [show-ziel] |

%SHOW ohne Operand gibt Ihnen das unmittelbar vorher eingegebene AID-Kommando aus. Wurde für die Task noch kein AID-Kommando gegeben, erhalten Sie eine Fehlermeldung. Ein %SHOW für eins der nicht vorgesehenen Kommandos führt zum Syntaxerror. Das Kommando ist in Kommando- und Subkommandofolgen zugelassen.

%SHOW verändert den Programmzustand nicht.

|           |
|-----------|
| show-ziel |
|-----------|

bezeichnet ein AID-Kommando, ein bestimmtes Subkommando oder alle eingetragenen Subkommandos. In *show-ziel* können Sie die für dieses Kommando zugelassenen Kommandos auch in ihrer Kurzform angeben.

| Kommando oder Subkommando | Information                                                                                                                                              |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| %AID                      | Die geltenden aktuellen Einstellungen der Kommandos %AID, %AINT, %BASE und die Version des geladenen AID.                                                |
| %BASE                     | Die aktuellen Einstellungen für %BASE, %AINT und %SYMLIB, die TSN, TID sowie die Version des Betriebssystems und der Typ des Rechners werden ausgegeben. |
| %C[ONTROL]                | Für jeden angemeldeten %CONTROL wird der Eingabestring ausgegeben.                                                                                       |

| Kommando oder Subkommando | Information                                                                                                                                                                                                                                                                                                             |
|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| %D[IS]A[SSEMBLE]          | Die aktuelle <i>anzahl</i> und <i>start</i> -Adresse (V'...') wird ausgegeben.                                                                                                                                                                                                                                          |
| %F[IND]                   | Das eingegebene Kommando und gegebenenfalls die virtuelle Adresse des letzten Treffers werden ausgegeben.                                                                                                                                                                                                               |
| %IN[INSERT] [testpunkt]   | Ohne die Angabe <i>testpunkt</i> werden alle aktiven Testpunkte ausgegeben. Sonst zeigt AID das eingegebene Kommando, in dem <i>testpunkt</i> vereinbart wurde.                                                                                                                                                         |
| %ON                       | Für jedes aktive %ON-Kommando wird der Eingabestring ausgegeben.                                                                                                                                                                                                                                                        |
| %OUT                      | Die geltenden <i>medium-u-menge</i> -Werte für die über %OUT steuerbaren Kommandos werden ausgegeben.                                                                                                                                                                                                                   |
| %OUTFILE                  | Alle implizit oder explizit angemeldeten Ausgabe-dateien werden mit ihren Linknamen aufgelistet.                                                                                                                                                                                                                        |
| %QUALIFY                  | Das letzte %QUALIFY-Kommando wird ausgegeben.                                                                                                                                                                                                                                                                           |
| %SYMLIB                   | Angemeldete Bibliotheken werden mit der zugehörigen Basisqualifikation und der evtl. vorhandenen TSN ausgegeben.                                                                                                                                                                                                        |
| %TRACE                    | Die Defaultwerte der %TRACE-Operanden werden ausgegeben. Dabei wird berücksichtigt, ob der letzte %TRACE symbolisch oder auf Maschinencod-Ebene war. In Folgezeilen gibt AID aus, wieviel Befehle oder Anweisungen schon mit dem aktuellen %TRACE bearbeitet wurden und wie das letzte aktuelle %TRACE-Kommando aussah. |
| %.*                       | Die Namen aller aktiven Subkommandos werden mit dem Typ des AID-Kommandos ausgegeben, in dem sie definiert wurden.                                                                                                                                                                                                      |
| %subkdoname               | Das Kommando, in dem <i>subkdoname</i> definiert wurde, wird ausgegeben.                                                                                                                                                                                                                                                |

## %STOP

Mit %STOP veranlassen Sie AID, das Programm anzuhalten, in den Kommandomodus zu gehen und eine STOP-Meldung auszugeben. Dieser Meldung können Sie entnehmen, an welcher Adresse und in welcher CSECT das Programm unterbrochen wurde.

Wird das Kommando am Terminal oder aus einer Prozedurdatei eingegeben, so wird der Programmzustand nicht verändert, da das Programm ja bereits steht. In diesen Fällen wird das Kommando angewendet, um mit der STOP-Meldung Lokalisierungs-Information über die Programmunterbrechungsstelle zu erhalten.

---

| Kommando | Operand |
|----------|---------|
|----------|---------|

---

%STOP

---

Steht %STOP in einer Kommandofolge oder in einem Subkommando, werden nachfolgende Kommandos nicht mehr ausgeführt.

Wenn das Programm durch Drücken der K2-Taste unterbrochen wurde, muß die Programmunterbrechungsstelle nicht unbedingt im Benutzerprogramm liegen, sondern das Programm kann auch in den Routinen des Laufzeitsystems stehen.

%STOP verändert den Programmzustand.

## %TITLE

Mit %TITLE definieren Sie einen eigenen Seitenkopf-Text. Diesen verwendet AID, wenn die Kommandos %DISASSEMBLE, %DISPLAY, %HELP, %SDUMP und %TRACE in die System-Datei SYSLST schreiben.

- Mit *seitenkopf* geben Sie den Text der Kopfzeile an und veranlassen AID, den Seitenzähler auf 1 zu setzen und vor der nächsten Druckzeile SYSLST auf Seitenanfang zu positionieren.

| Kommando | Operand      |
|----------|--------------|
| %TITLE   | [seitenkopf] |

Mit einem %TITLE ohne *seitenkopf*-Operanden wechseln Sie wieder zur AID-Standard-Überschrift. AID setzt den Seitenzähler wieder auf 1 und positioniert SYSLST vor der nächsten Druckzeile auf Seitenanfang.

%TITLE verändert den Programmzustand nicht.

seitenkopf

gibt den variablen Teil der Seitenüberschrift an. Er wird von AID mit der Uhrzeit, dem Datum und dem Seitenzähler ergänzt.

### seitenkopf

ist ein Character-Literal in der Form {C'x...x' | 'x...x'C | 'x...x'}

 und kann maximal 80 Zeichen lang sein. Ein längeres Literal wird mit einer Fehlermeldung abgewiesen, in der aber nur die ersten 52 Stellen des Literals protokolliert werden.

Auf eine Druckseite werden außer der Seitenüberschrift bis zu 58 Zeilen gedruckt.

## %TRACE

Mit %TRACE schalten Sie die AID-Ablaufverfolgung ein und starten das Programm oder setzen es an der unterbrochenen Stelle fort.

- Mit *anzahl* legen Sie fest, wieviele Befehle maximal verfolgt, d.h. ausgeführt und protokolliert werden sollen.
- Mit *kriterium* wählen Sie verschiedene Typen von Maschinenbefehlen aus. Nachdem ein Befehl des gewählten Typs ausgeführt wurde, gibt AID eine Protokollzeile aus.
- Mit *trace-bereich* legen Sie den Programmbereich fest, in dem *kriterium* berücksichtigt werden soll.

| Kommando  | Operand  |           |                    |
|-----------|----------|-----------|--------------------|
| %T[TRACE] | [anzahl] | kriterium | [IN trace-bereich] |

Ein %TRACE kann nicht gleichzeitig mit einem *write-ereignis* des %ON angemeldet sein.

%TRACE **wirkt** beim Testen auf Maschinencode-Ebene **anders** als beim Testen auf symbolischer Ebene:

Hier wird ein mit *kriterium* ausgewählter Befehl **nach** seiner Ausführung protokolliert aber **auch außerhalb** vom *trace-bereich* überwacht. *trace-bereich* **wirkt** somit **nur** auf die Protokollierung, **nicht** auf die Überwachung. Der **zusätzliche** Überwachungsaufwand verlangsamt den Programmablauf. Deshalb empfiehlt es sich, in großen Programmen mit langen Befehlsstrecken das %TRACE-Kommando gezielt an Testpunkten als Subkommando des %INSERT einzusetzen, und nach der Ausführung wieder zu löschen (siehe AID-Basis-handbuch, Kapitel 6 [1] und %INSERT).

**Unterbrochen** wird ein %TRACE durch folgende Ereignisse im Testverlauf. Er kann dann mit %CONTINUE fortgesetzt werden:

- Ein Subkommando wurde ausgeführt indem ein %STOP enthalten war.
- Ein %INSERT endet mit einer Programmunterbrechung, weil der *steuerung*-Operand K oder S lautet.
- Die K2-Taste wurde gedrückt. Siehe dazu Abschnitt „Kommandos zu Beginn einer Test-sitzung“ auf Seite 8.

**Beendet** wird der %TRACE durch folgende Ereignisse:

- Die maximale Anzahl der zu überwachenden Anweisungen wurde erreicht (deshalb kann ein %TRACE mit der Eingabe von %T 1 %INSTR gelöscht werden).
- Nach einer der oben beschriebenen Programmunterbrechung geben Sie ein %RESUME ein.

- Ein Subkommando wurde ausgeführt, das ein %RESUME- oder %TRACE-Kommando enthält.

Wurde ein %TRACE beendet, sind seine Operandenwerte noch eingetragen. Sie gelten solange bis sie durch Angaben in einem späteren %TRACE überschrieben werden oder bis Programmende. In einem neuen %TRACE-Kommando setzt AID also für einen nicht angegebenen Operanden den Wert aus dem vorhergehenden %TRACE ein. Beim *trace-bereich*-Operanden ist dies nur der Fall, wenn die aktuelle Unterbrechungsstelle in dem zu übernehmenden *trace-bereich* liegt. Gibt es keine zu übernehmenden Werte, setzt AID die Standardwerte 10 für *anzahl* und das gesamte Benutzerprogramm für *trace-bereich* ein.

Mit %OUT können Sie steuern, welche Informationen eine Protokollzeile enthält und auf welches Ausgabemedium das Protokoll ausgegeben werden soll.

Steht %TRACE in einer Kommandofolge oder in einem Subkommando, werden nachfolgende Kommandos nicht mehr ausgeführt.

%TRACE verändert den Programmzustand.

anzahl

gibt an, wieviele Befehle vom Typ *kriterium* maximal ausgeführt und protokolliert werden sollen.

anzahl

ist eine Ganzzahl mit dem Wert:

$$1 \leq \text{anzahl} \leq 2^{31} - 1$$

Standardwert ist 10. Er wird von AID in ein %TRACE-Kommando ohne *anzahl*-Operanden eingesetzt, wenn es keinen Wert aus einem vorhergehenden %TRACE gibt.

Nachdem die vorgegebene *anzahl* von Befehlen überwacht wurde, gibt AID über SYSOUT eine Meldung aus, das Programm wird angehalten, es steht und Sie können wieder AID- oder BS2000-Kommandos eingeben. Der Meldung können Sie entnehmen, an welcher Adresse und in welcher CSECT bzw. in welchem COMMON das Programm angehalten wurde.

kriterium

ist ein Schlüsselwort, das den Typ der Maschinenbefehle festlegt, die nach ihrer Ausführung protokolliert werden sollen. Standardwert ist das **symbolische** *kriterium* %STMT; deshalb muß für das maschinennahe Testen immer ein *kriterium* angegeben werden, falls nicht noch aus einem vorhergehenden %TRACE eine geeignete *kriterium*-Vereinbarung gültig ist.

|                  |                                                                                                                                          |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <i>kriterium</i> | Ausgabe der Protokollierung <u>nach</u>                                                                                                  |
| %INSTR           | jedem ausgeführten Maschinenbefehl                                                                                                       |
| %B               | jedem ausgeführten Verzweigungsbefehl (das sind die Maschinenbefehle BAL, BALR, BAS, BASSM, BASR, BC, BCR, BCT, BCTR, BSM, BXH und BXLE) |
| %BAL             | jedem Aufruf eines Unterprogramms (durch die Maschinenbefehle BAL, BALR, BAS, BASSM und BASR).                                           |

### trace-bereich

legt den Programmbereich fest, in dem die Ablaufverfolgung stattfinden soll.

Für %TRACE muß die Basisqualifikation E=VM eingestellt sein (siehe %BASE) oder explizit angegeben werden.

Eine *trace-bereichs*-Definition ist wirksam bis zu einem neuen %TRACE mit eigenem *trace-bereich*-Operanden, einem %TRACE, der außerhalb dieses Bereichs eingegeben wird, oder bis Programmende.

Wird *trace-bereich* nicht angegeben, wird die Bereichsdefinition aus einem vorhergehenden %TRACE übernommen, wenn die aktuelle Unterbrechungsstelle in diesem Bereich liegt. Sonst setzt AID als Standardwert das gesamte Benutzerprogramm ein.

Außerhalb von *trace-bereich* wird *kriterium* weiter überwacht, die durchlaufenen Befehle werden jedoch nicht protokolliert. Wegen des dadurch entstehenden zusätzlichen Überwachungsaufwands, empfiehlt es sich in großen Programmen das %TRACE-Kommando gezielt an Testpunkten zu geben und beim Verlassen des Bereichs auch wieder zu löschen (siehe %INSERT).

trace-bereich-OPERAND - - - - -

```
IN [.] [qua.] { C=csect | COM=common
 (V'f...f': V'f...f')
 schlüsselwort }
```

- Steht der Punkt an führender Stelle, ist er das Kennzeichen für eine *vorqualifikation*. Sie muß mit einem vorhergehenden %QUALIFY definiert worden sein. Aufeinanderfolgende Qualifikationen werden durch einen Punkt getrennt. Außerdem muß zwischen der letzten Qualifikation und dem anschließenden Operandenteil ein Punkt stehen.

qua Eine oder mehrere Qualifikationen geben Sie an, wenn Sie ein Speicherobjekt ansprechen wollen, das nicht im aktuellen AID-Arbeitsbereich liegt oder darin nicht eindeutig ist. Sie geben nur die Qualifikationen an, die zur eindeutigen Ansprache genügen.

E=VM

Da *trace-bereich* nur im virtuellen Speicher des geladenen Programms liegen kann, geben Sie diese Basisqualifikation *E=VM* nur an, wenn *E=Dn* vereinbart ist.

CTX=kontext

geben Sie nur an um eine CSECT oder einen COMMON im Programmsystem eindeutig ansprechen zu können.

[L=ladeinheit.][O=objektmodul.]

geben Sie nur an, wenn Sie eine CSECT oder einen COMMON ansprechen wollen, die im aktuellen Kontext ohne diese Qualifikationen nicht eindeutig adressierbar sind. Sie geben nur die Qualifikationen an, die zur eindeutigen Ansprache nötig sind.

{C=csect | COM=common}

Der *trace-bereich* umfaßt die gesamte angegebene CSECT oder den COMMON.

(V'f...f' : V'f...f')

*trace-bereich* wird durch die Angabe einer virtuellen Anfangs- und Endadresse festgelegt.

Die Adressen müssen im ausführbaren Teil eines geladenen Programms liegen. Anfangsadresse muß  $\leq$  Endadresse sein.

schlüsselwort

Der *trace-bereich* umfaßt den Speicherbereich, der durch eines der folgenden Schlüsselwörter (siehe AID-Basishandbuch, Kapitel 9 [1]) angesprochen wird. Bei XS- bzw. ESA-Anlagen bezeichnen Sie mit %CLASS6 und %CLASS6BELOW denselben Adreßraum.

|              |                                              |
|--------------|----------------------------------------------|
| %CLASS6      | Klasse-6-Speicher                            |
| %CLASS6BELOW | Klasse-6-Speicher, unterhalb der 16MB-Grenze |
| %CLASS6ABOVE | Klasse-6-Speicher, oberhalb der 16MB-Grenze  |

## Ausgabe des %TRACE-Protokolls

Das %TRACE-Protokoll wird standardmäßig in ausführlicher Form über SYSOUT ausgegeben (T=MAX). Mit %OUT können Sie die Ausgabe-Medien festlegen, und bestimmen, wie AID die Ausgabezeilen aufbauen soll (siehe AID-Basishandbuch, Kapitel 7 [1]).

Beim Testen von ESA-Programmen werden im %TRACE-Protokoll die Adressen, die in einem Datenraum liegen, mit einem Stern (\*) markiert. Außerdem werden die Zugriffsregister (ARN=...) ausgegeben.

In einem ausführlichen %TRACE-Protokoll (%OUT %T T=MAX) sind die folgenden Informationen enthalten:

1. CSECT-relative Adresse des Maschinenbefehls
2. Befehl in ASSEMBLER-Notation
3. Condition Code
4. Zu den Operanden eines Befehls werden je nach Befehlstyp unterschiedliche Informationen ausgegeben:
  - aktuelle Operandenwerte
  - zu einem SVC der zugehörige Makroname und die Parameterliste (max. 12 Bytes)
  - ein bedingter Sprungbefehl wird mit einem Stern (\*) vor dem Maskenfeldoperanden markiert, wenn die Bedingung zutraf, d.h. wenn der Sprung ausgeführt wurde. Hat das Maskenfeld den Inhalt binär Null, wird nicht das Maskenbild, sondern die Zeichenfolge "NOP" ausgegeben.
  - Der Befehl EX wird protokolliert und in derselben Aufbereitung der durch ihn ausgeführte Befehl.
  - Adreß-Operanden werden mit der errechneten Endadresse dargestellt, und diese wird noch als CSECT-relative Adresse ausgegeben.

Bei der Ausgabe nach SYSLST (%OUT %T P=MAX) sind zusätzlich enthalten:

5. Virtuelle Adresse des Maschinenbefehls
6. Befehl in sedezimaler Ausgabe

*Beispiel zum Zeilenaufbau (T=MAX)*

|         |      |                  |   |                      |              |
|---------|------|------------------|---|----------------------|--------------|
| M1BS+C  | BALR | R15, R0          | 3 | R15=7F00000E         | R0=00000000  |
| M1BS+E  | LM   | R2, R13, 56(R15) | 3 | A2=00000064=M1BS+64  |              |
|         |      |                  |   | R2=000008A0          | R3=00000798  |
|         |      |                  |   | R4=00000000          | R5=00000000  |
|         |      |                  |   | R6=00000000          | R7=00000000  |
|         |      |                  |   | R8=00000000          | R9=00000000  |
|         |      |                  |   | R10=00000000         | R11=000000A8 |
|         |      |                  |   | R12=000069D0         | R13=00000A00 |
| M1BS+12 | LA   | R0, 5(R0, R0)    | 3 | R0=00000005          |              |
|         |      |                  |   | A2=00000005=M1BS+5   |              |
| M1BS+16 | LA   | R1, 3E(R0, R15)  | 3 | R1=0000004C          |              |
|         |      |                  |   | A2=0000004C=M1BS+4C  |              |
| M1BS+1A | L    | R15, 94(R0, R11) | 3 | R15=00003868         |              |
|         |      |                  |   | A2=0000013C=M1BS+13C |              |
|         |      |                  |   | O2=00003868          |              |

Mit dem %OUT %T T=MIN baut AID die Ausgabezeilen anders auf. 1. und 6. sind nicht enthalten, 2., 3., 4. und 5. hingegen sind immer enthalten, unabhängig vom Typ des Ausgabemediums.

*Beispiel zum Zeilenaufbau (T=MIN)*

|          |      |                  |   |              |              |              |
|----------|------|------------------|---|--------------|--------------|--------------|
| 0000000C | BALR | R15, R0          | 3 | R15=7F00000E | R0=00000000  |              |
| 0000000E | LM   | R2, R13, 56(R15) | 3 | A2=00000064  | R2=000008A0  | R3=00000798  |
|          |      |                  |   | R4=00000000  | R5=00000000  | R6=00000000  |
|          |      |                  |   | R7=00000000  | R8=00000000  | R9=00000000  |
|          |      |                  |   | R10=00000000 | R11=000000A8 | R12=000069D0 |
|          |      |                  |   | R13=00000A00 |              |              |
| 00000012 | LA   | R0, 5(R0, R0)    | 3 | R0=00000005  | A2=00000005  |              |
| 00000016 | LA   | R1, 3E(R0, R15)  | 3 | R1=0000004C  | A2=0000004C  |              |
| 0000001A | L    | R15, 94(R0, R11) | 3 | R15=00003868 | A2=0000013C  | O2=00003868  |

**Beispiele**

1. %T 3 %INSTR  
Die nächsten drei Befehle in der aktuellen CSECT werden protokolliert. Danach wird das Programm angehalten, und eine STOP-Meldung und END-OF-TRACE-Meldung ausgegeben.
2. %T  
Mit diesem %TRACE werden die Operandenwerte des vorhergehenden %TRACE übernommen. Das Programm wird fortgesetzt, und 3 weitere Befehle werden protokolliert.
3. %T 10 %B IN E=VM.(V'83E4':V'8488')  
Gilt eine Basisqualifikation für eine Dump-Datei, dann muß beim *trace-bereich* die Basisqualifikation E=VM angegeben werden. Maximal 10 Verzweigungsbefehle werden im Programmbereich V'83E4' bis V'8488' protokolliert. Werden beim ersten Durchlau-

fen von *trace-bereich* noch keine 10 Befehle protokolliert, bleibt der %TRACE aktiv, und die Verzweigungsbefehle werden weiter überwacht, wenn auch nicht protokolliert. Das verlangsamt den Programmablauf. Im folgenden Beispiel finden Sie dafür eine Lösung.

4. %INSERT V'A38' <%T 20 %B IN (V'A38':V'B40')>  
%INSERT V'B40' <%R>

Von Adresse V'38' bis V'B40' soll nach jedem Verzweigungsbefehl eine %TRACE-Protokollzeile ausgegeben werden. Das Kommando wird jedoch erst aktiv, wenn der Programmablauf an die Adresse V'A38' (%INSERT) kommt. Wird *trace-bereich* über die Adresse V'B40' verlassen, wird durch das Subkommando des zweiten %INSERT der %TRACE gelöscht. In diesem Beispiel wird davon ausgegangen, daß der Trace-Bereich nur einen Ein- und Ausgang hat.



---

## 6 Anwendungsbeispiel

In diesem Kapitel wird eine AID-Testsitzung für ein kleines Assembler-Programm gezeigt. Anhand dieser Testsitzung können Sie die Anwendung und Wirkung einiger AID-Kommandos nachvollziehen, die Vorgehensweise ist bewußt einfach gehalten. Das Assembler-Programm ist in Abschnitt 6.1 dargestellt, der Testablauf in Abschnitt 6.2. Zur besseren Lesbarkeit sind Eingaben fettgedruckt.

### 6.1 Assembler-Programm

#### **Aufgabenbeschreibung**

Das Programm SUMME soll maximal 10 zweistellige Zahlen einlesen und ihre Summe ausgeben. Als Endekennzeichen wird die Zahl 00 eingegeben.  
Ist die 10. Zahl nicht das Endekennzeichen, wird eine Meldung und die errechnete Summe ausgegeben.

## Auszug aus dem Assembler-Listing

```

BERECHNUNG DER SUMME VON N ZAHLEN (N <= 10) 14:09:30 1995-02-21
 SYMBOL TYPE ID ADDR LENGTH A/R-MODE EXTERNAL SYMBOL DICTIONARY
 SUMME SD 0001 00000000 000184 ANY ANY
BERECHNUNG DER SUMME VON N ZAHLEN (N <= 10) 14:09:30 1995-02-21
LOCTN OBJECT CODE ADDR1 ADDR2 STMT M SOURCE STATEMENT
000000
1 SUMME START
2 TITLE 'BERECHNUNG DER SUMME VON N ZAHLEN (N <= 10) '
3 PRINT NOGEN
4 EQU 0
5 R0 EQU 1
6 R1 EQU 2
7 R2 EQU 3
8 R3 EQU 4
9 R4 EQU 5
10 R5 EQU 5
10 SUMME AMODE ANY
11 SUMME RMODE ANY
12 GPARMOD 31
14 2 *,VERSION 010
000000 0D 20
15 BASR R2,R0
000002 00000002
16 USING *,R2
000002
17 ANFANG WROUT MELD1,ENDE
46 2 *,@DCEI 999 921011 53531004
49 L R5,=F'1'
000026 58 50 2176 00000178
50 SCHLEIFE A R5,=F'1'
00002A 5A 50 2176 00000178
00002E 49 50 2138 0000013A
51 CH R5,ZEHN
000032 47 20 20BE 000000C0
52 BH FEHLER
000036
53 LESEN RDATA EINGABE, ENDE
88 2 *,@DCEI 999 921011 53531002
91 VERGL CLC EINGABE+4,NULL
000062 D5 05 2121213A 00000123 0000013C
92 BE AUS
000068 47 80 207A 0000007C
93 ADD PACK PACK,EINGABE+4(2)
00006C F2 11 21232121 00000125 00000123
94 AP GESAMT,PACK
000072 FA 31 213C2123 0000013E 00000125
95 B SCHLEIFE
000078 47 F0 2028 0000002A
96 AUS UNPK ERGEB,GESAMT
00007C F3 63 2131213C 00000133 0000013E
97 MVZ ERGEB+6(1),ZONE
000082 D3 00 21372140 00000139 00000142
98 WROUT MELD2,ENDE
000088
126 2 *,@DCEI 999 921011 53531004
129 ENDE TERM DUMP=Y
0000AA
132 2 *,VERSION 100
0000BE
144 FEHLER WROUT MELD3,ENDE
173 2 *,@DCEI 999 921011 53531004
0000E2 47 F0 207A 0000007C
176 B AUS
177 *
178 *
179 * DEFINITIONEN
180 *
0000E6 0039
181 MELD1 DC Y(L'M1+5)
0000E8 404001
182 DC X'404001'
0000EB C2C9E3E3C540C2C9
183 M1 DC C'BITTE BIS ZU 10 2-STELLIGE ZAHLEN EINGEBEN! ENDE: 00'
00011F 000000000000
184 EINGABE DC XL6'00'
000125 000C
185 PACK DC PL2'0'
186 *
000128 0012
187 MELD2 DC Y(L'M2+L'ERGEB+5)
00012A 404001
188 DC X'404001'
00012D E2E4D4D4C57A
189 M2 DC C'SUMME:'
000133 40404040404040
190 ERGEB DC CL7' '
191 *
00013A 000A
192 ZEHN DC H'10'
00013C F0F0
193 NULL DC C'00'
00013E 00000000C
194 GESAMT DC PL4'0'
000142 F0
195 ZONE DC X'F0'
196 *
000144 0034
197 MELD3 DC Y(L'M3+5)
000146 404001
198 DC X'404001'
000149 C5E240D2D6C5D5D5
199 M3 DC C'ES KOENNEN MAXIMAL 10 ZAHLEN VERARBEITET WERDEN'
000000
200 END SUMME
000178 00000001
201 =F'1'
00017C 9502211406561183
202 =X'9502211406561183' CONSISTENCY CONSTANT FOR AID
FLAGS IN 00000 STATEMENTS, 000 PRIVILEGED FLAGS, 000 MNOTES
HIGHEST ERROR-WEIGHT : NO ERRORS
THIS PROGRAM WAS ASSEMBLED BY ASSEMBHC V 1.2A00 ON 1995-02-21 AT 14:09:30
BERECHNUNG DER SUMME VON N ZAHLEN (N <= 10)

```

## 6.2 Testablauf

### 1. Schritt

Das Assembler-Quellprogramm SUMME in der Datei SOURCE.TEST wird mit dem ASSEMBH übersetzt. Es ist fehlerfrei.

```

/DEL-SYS-FILE OMF
/START-PROG ASSEMBH

% BLS0500 PROGRAM 'ASSEMBH', VERSION '1.2A00' OF '1994-11-11' LOADED.
% BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG. 1990. ALL RIGHTS
RESERVED
% ASS6010 V 1.2A00 OF BS2000 ASSEMBH READY

//COMPILE SOURCE=SOURCE.TEST,MODULE-LIBRARY=LMSLIB

% ASS6011 ASSEMBLY TIME: 80 MSEC
% ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
% ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
% ASS6006 LISTING-GENERATOR TIME: 102 MSEC

//END

% ASS6012 END OF ASSEMBH

```

### 2. Schritt

Das Programm SUMME soll zum Ablauf gebracht werden.

```

/START-PROG *MOD(LIB=LMSLIB,ELEM=SUMME)
% BLS0517 MODULE 'SUMME' LOADED

BITTE BIS ZU 10 2-STELLIGE ZAHLEN EINGEBEN. ENDE: 00
*05
*16
*48
*00
*0
*00
*EN
*
/%STOP
STOPPED AT V'60' = SUMME + #'60'

```

Das Programm verzweigt immer wieder zur Eingabe. Das Endekennzeichen ,00' wird nicht erkannt. Das Programm wird durch Drücken der K2-Taste unterbrochen. Durch das %STOP-Kommando wird die Ausgabe der Adresse der aktuellen Unterbrechungsstelle veranlaßt. Sie liegt in der RDATA-Behandlung. Anschließend kommt an der Adresse V'62' wieder die Abfrage auf das Ende der Eingabe mit dem CLC.

### 3. Schritt

```

/LOAD-PROG *MOD(LIB=LMSLIB,ELEM=SUMME)

% BLS0517 MODULE 'SUMME' LOADED

/%SET #'62' INTO %1G
/%INSERT %1G-> <%D V'11F'%L6;%STOP>
/%R

```

Das Programm wird erneut geladen und auf den CLC-Befehl ein Testpunkt gesetzt. Die Adresse, an der der CLC-Befehl steht, wird im AID-Register %1G abgespeichert, da die Adresse öfters gebraucht wird. Jedesmal, wenn der Programmablauf an dieser Adresse angekommen ist, soll der Inhalt des Feldes EINGABE (Anfangsadresse V'11F') ausgegeben werden. Nach der Ausgabe soll das Programm in den Zustand STOP übergehen, damit neue Kommandos eingegeben werden können.

Mit %RESUME wird das geladene Programm gestartet.

### 4. Schritt

```

BITTE BIS ZU 10 2-STELLIGE ZAHLEN EINGEBEN. ENDE: 00
*05

*** TID: 0001020D *** TSN: 2069 *****
CURRENT PC: 00000062 CSECT: SUMME *****
V'0000011F' = SUMME + #'0000011F'
0000011F (0000011F) 00060000 FOF5 05
STOPPED AT V'62' = SUMME + #'62'

/%R
*00
V'0000011F' = SUMME + #'0000011F'
0000011F (0000011F) 00060000 FOF0 00
STOPPED AT V'62' = SUMME + #'62'

```

Nach der Eingabe einer normalen Zahl wird bei der zweiten Aufforderung eine Zahl einzugeben, das Endekennzeichen „00“ eingegeben.

## 5. Schritt

```

/%T 3 %INSTR
SUMME+62 CLC 121(6,R2),13A(R2) 2 A1=00000123=SUMME+123
 A2=0000013C=SUMME+13C
 01=F0F0055D 0000
 02=F0F00000 041D

SUMME+68 BC B'1000',7A(R0,R2) 2 M=0
 A1=0000007C=SUMME+7C

SUMME+6C PACK 123(2,R2),121(2,R2) 2 A1=00000125=SUMME+125
 A2=00000123=SUMME+123
 01=000F 02=F0F0

STOPPED AT VT72T = SUMME + #T72T , END OF TRACE

/%D V'64'%S->%L6
V'00000123' = SUMME + #'00000123'
00000123 (00000123) F0F0005F 0000 00.)..

/%D V'66'%S->%L6
V'0000013C' = SUMME + #'0000013C'
0000013C (0000013C) F0F00000 005C 00...*

```

Die Endeabfrage wird mit %TRACE verfolgt. Nach der Protokollierung von 3 Assemblerbefehlen wird der %TRACE mit der Stopmeldung beendet. Das Endekennzeichen b00T wurde nicht erkannt und das Programm im Zweig, caddieren des EingabewertesR fortgesetzt.

Mit den zwei nachfolgenden %DISPLAY-Kommandos werden die im CLC verglichenen Speicherinhalte ausgegeben. Die Adressen werden durch die Typmodifikation %S mit dem anschließenden Pointer-Operator unmittelbar aus Teilen des Befehls errechnet.

Da der CLC mit der impliziten Länge 6 des ersten Adreoperanden arbeitet (EINGABE), das Vergleichsfeld NULL aber nur 2-stellig definiert wurde und auch die Eingabe nur 2-stellig ist, kann der CLC nicht auf Gleichheit kommen.

Der Assembler-Befehl müßte korrekt lauten:

```
VERGL CLC EINGABE+4(2),NULL
```

## 6. Schritt

Dieser Fehler kann vorläufig mit dem %MOVE-Kommando behoben werden. Das Programm wird erneut geladen und mit %AID wird der Änderungsdialog eingeschaltet.

```

/LOAD-PROG *mod(lib=bib,elem=summe)
% BLS0517 MODULE 'SUMME' LOADED

/%AID CHECK=ALL
/%MOVE X'01' INTO %1G->.1

OLD CONTENT:
05
NEW CONTENT:
01
% AID0274 Change desired? Reply (Y=Yes; N=No)?
/Y
/%R

```

Mit dem %MOVE wird die Längenangabe des CLC-Befehls von '05' in '01' geändert. Dabei wird zur Kontrolle ein Änderungsdialog geführt. Mit %RESUME wird das Programm gestartet.

```
*05
*16
*48
*12
*10
*15
*17
*19
*29
ES KOENNEN MAXIMAL 10 ZAHLEN VERARBEITET WERDEN
SUMME:0000171
```

Es liegt ein weiterer Programmfehler vor, da der Benutzer nur 9 Zahlen eingegeben hat. Er sollte jedoch 10 Zahlen eingeben können.

## 7. Schritt

```
/LOAD-PROG *MOD(LIB=BIB,ELEM=SUMME)
% BLS0517 MODULE 'SUMME' LOADED

/%AID CHECK
/%M X'01' INTO %1G->.1
/%T 4 %INSTR IN (V'26':V'36')
BITTE BIS ZU 10 2-STELLIGE ZAHLEN EINGEBEN! ENDE: 00
SUMME+26 L R5,176(R0,R2) 0 R5=00000001
 A2=00000178=SUMME+178
 O2=00000001
SUMME+2A A R5,176(R0,R2) 2 R5=00000002
 A2=00000178=SUMME+178
 O2=00000001
SUMME+2E CH R5,138(R0,R2) 1 R5=00000002
 A2=0000013A=SUMME+13A
 O2=000A
SUMME+32 BC B'0010',BE(R0,R2) 1 M=2
 A1=000000C0=SUMME+C0
STOPPED AT V'36' = SUMME + #'36' , END OF TRACE
```

Mit %AID wird der Änderungsdialog wieder ausgeschaltet.

Mit %MOVE erfolgt wieder die temporäre Korrektur des ersten Fehlers.

Mit %TRACE wird der Programmabschnitt V'26' bis V'36' überwacht und protokolliert. Im %TRACE-Protokoll ist zu sehen, daß das Register 5 mit dem Anfangsstand 1 geladen wird und so schon bevor die erste Zahl gelesen wurde den Wert '2' enthält. Der Befehl mit der Adresse V'26' müßte korrekt lauten:

```
L R5,=F'0'
```

## 8. Schritt

Dieser Fehler kann vorläufig mit Hilfe eines %SET-Kommandos behoben werden. Das Programm wird erneut geladen.

Die Korrektur des CLC-Befehls mit %MOVE wird wiederholt, diesmal mit direkter Adresse.

```

/LOAD-PROG *MOD(LIB=BIB,ELEM=SUMME)
% BLS0517 MODULE 'SUMME' LOADED

/%M X'01' INTO V'63'
/%INSERT V'2A' <%SET 0 INTO %5; %REM %INSERT>
/%R

```

Mit %INSERT wird ein Testpunkt hinter den fehlerhaften Ladebefehl mit der Adresse V'26' gesetzt. Im Subkommando des %INSERT setzt ein %SET das Register 5 auf '0'. Danach wird der %INSERT mit %REMOVE gelöscht. Das Programm läuft dann weiter.

Eine alternative Lösung wäre, den Ladebefehl direkt zu ändern, wie vorher die Länge im CLC, damit das Register 5 gleich richtig geladen wird. Da das Feld EINGABE (Adresse V'11F') mit X'00' geladen ist, können Sie dann schreiben:

```
%MOVE X'211F' INTO V'28'
```

```

BITTE BIS ZU 10 2-STELLIGE ZAHLEN EINGEBEN. ENDE: 00
*05
*16
*48
*12
*10
*15
*17
*19
*29
*11
ES KOENNEN MAXIMAL 10 ZAHLEN VERARBEITET WERDEN
SUMME:0000182

```

Nach dieser Korrektur läuft das Programm fehlerfrei ab. Die Fehler können endgültig im Quellprogramm behoben werden.



# 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 (Maschinencode-Ebene) oder Anweisungen (symbolisches Testen) protokolliert werden sollen. Den Programmablauf können Sie am Bildschirm mitverfolgen, wenn die Ausgabe nicht mit `%OUT %TRACE` auf ein anderes Ausgabemedium umgeleitet wurde.

## Adressierungsmodus

bestimmt, wie Adressen für die Ausführung der Maschinenbefehle umgesetzt werden sollen. AID übernimmt standardmäßig den Adressierungsmodus des Testobjekts. Das trifft sowohl für die Adreßbreite (24- oder 31-Bit) von Programmen auf XS-Anlagen (`%AMODE`), als auch für die Adressierung von Datenräumen auf ESA-Anlagen (`%ASC`) zu.

Mit dem Schlüsselwort `%AMODE` sprechen Sie die Systeminformation Adreßbreite an. Sie kann mit `%DISPLAY` abgefragt werden und mit `%MOVE %MODE{24|31} INTO %AMODE` geändert werden.

Mit dem Schlüsselwort `%ASC` (access space control mode) sprechen Sie die Systeminformation AR-Modus (access register mode) an. Sie gibt Auskunft darüber, ob Zugriffsregister für die Adressierung von Datenräumen in die Adreßumsetzung einbezogen werden. Sie kann mit `%DISPLAY` abgefragt werden.

## 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 liegt 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 Bereichs-Qualifikationen oder über eine *kennzeichnung* (COBOL) eindeutig der gewünschten Adresse zuordnen.

## ALET

gibt es nur auf ESA-Anlagen. Ein ALET (access list entry token) ist eine Art Zeiger auf die Zugriffsliste (AL), über die die Zugriffe auf einen Datenraum verwaltet werden. Die ALETs sind in den Zugriffsregistern enthalten und können in der ALET-Qualifikation angegeben werden.

## 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 Speicherstellen ohne Angabe einer Basisqualifikation ansprechen können.

Beim Testen auf Maschinencode-Ebene ist das der nicht-privilegierte Teil des virtuellen Speichers in Ihrer Task, der vom Programm samt allen seinen konnektierten Subsystemen belegt ist, oder der entsprechende Bereich in einem Speicherabzug.

In einem Kommando können Sie vom AID-Arbeitsbereich abweichen, indem Sie im Adreßoperanden eine Basisqualifikation angeben. Mit dem Kommando %BASE können Sie den AID-Arbeitsbereich vom geladenen Programm in einen Speicherabzug verlegen oder umgekehrt.

## AID-Ausgabedateien

sind die Dateien, in die Sie die Ausgaben der Kommandos %DISASSEMBLE, %DISPLAY, %HELP, %MOVE, %SDUMP und %TRACE schreiben lassen können. Die Dateien werden 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 (%DUMPFIL)E)
2. PLAM-Bibliotheken, in denen sich Bindemodule (OMs) oder Bindeladmodule (LLMs) mit LSD-Sätzen 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 AID-Basishandbuch, Kapitel 9 [1]):

|                                 |                   |
|---------------------------------|-------------------|
| {C'x...x'   'x...x'C   '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 des Testobjekts 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

Beim Testen auf Maschinencode-Ebene ist das der nicht-privilegierte Teil des virtuellen Speichers in Ihrer Task, der vom Programm samt allen seinen 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.

### **Anweisungsname**

ist ein Name, der im Quellprogramm für eine Anweisung vergeben wurde. Das sind die Namen von Marken, Entrys, Paragraphen, Kapiteln etc.. Zu ihnen ist in den LSD-Sätzen eine Adreßkonstante hinterlegt, mit der die entsprechende Speicherstelle adressiert werden kann.

### **AR-Modus**

Den AR-Modus (access register mode) gibt es auf ESA-Anlagen. Er entscheidet, ob die Zugriffsregister bei der Adreßumsetzung mit ausgewertet werden oder nicht. Wenn %DISPLAY %ASC den sedezimalen Wert X'01' ausgibt, ist der AR-Modus gesetzt, die Zugriffsregister werden ausgewertet, und somit können Adressen in Datenräumen angesprochen werden.

### **Attribute**

Jedes Speicherobjekt hat bis zu sechs Attribute:

Adresse, Name (opt), Inhalt, Länge, Speichertyp, Ausgabetyt

Mit Selektoren können Sie auf Adresse, Länge und Speichertyp zugreifen. Über den Namen findet AID in den LSD-Sätzen alle zugehörigen Attribute, um damit zu arbeiten.

Adreßkonstanten und Konstanten aus dem Quellprogramm haben nur bis zu fünf Attribute: Name (opt), Wert, Länge, Speichertyp, Ausgabetyt.

Sie haben keine Adresse. Beim Ansprechen einer Konstanten greift AID nicht auf ein Speicherobjekt zu, sondern setzt nur den dafür vorgemerkten Wert ein.

### **Ausgabetyt**

Attribut eines Speicherobjekts, das bestimmt, wie der Speicherinhalt von AID ausgegeben wird. Jedem Speichertyp ist ein Ausgabetyt zugeordnet. Im AID-Basishandbuch, Kapitel „Allgemeine Speichertypen“ [1] sind die Speichertypen mit den zugehörigen Ausgabetyten aufgelistet. Für die Datentypen in der jeweiligen Programmiersprache gilt eine entsprechende Zuordnung. Eine Typmodifikation in %DISPLAY und %SDUMP bewirkt eine Änderung des Ausgabetyts.

### 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'). Für eine CSECT bzw. einen COMMON als Speicherobjekt ergeben sich die Bereichsgrenzen aus Anfangs- und Endadresse der CSECT oder des COMMON (siehe AID-Basishandbuch, Kapitel 7 [1]).

### 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, also für alle Variablen und Konstanten. Auf Elemente von Tabellen können Sie sich wie in der jeweiligen Programmiersprache über einen Index beziehen.

### Datenraum

Auf ESA-Anlagen gibt es die Möglichkeit neben dem Programmraum, der dem bisherigen Adreßraum entspricht, weitere Adreßräume für Daten, die Datenräume (data spaces), zu nutzen. Speicherobjekte, die sich in einem Datenraum befinden, können auf Maschinencode-Ebene über eine virtuelle Adresse mit ALET- oder SPID-Qualifikation angesprochen werden.

## Datentyp

Gemäß dem im Quellprogramm deklarierten Datentyp ordnet AID allen Datenelementen einen AID-Speichertyp zu:

- binär ( $\hat{=}$  %X)
- character ( $\hat{=}$  %C)
- 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).

Jedem Speichertyp entspricht ein Ausgabotyp, wie das Datenelement von %DISPLAY bzw. %SDUMP (symbolisches Testen) ausgegeben 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. Sie müssen dann das Kommando oder die Kommandofolge kürzen bzw. die Funktionen auf mehrere Kommandos aufteilen.

## ESA

(**E**nterprise **S**ystem **A**rchitecture) ist ein neues Hardware-Konzept, mit dem zusätzliche Adreßräume für Daten nutzbar gemacht werden. Die Adressierung innerhalb eines solchen Datenraums ist über die Zugriffsregister realisiert. Bei eingeschaltetem AR-Modus wird bei der Adreßumsetzung in einem Befehl das zum Basisregister korrespondierende Zugriffsregister mit ausgewertet. Anlagen mit dieser Hardware werden ESA-Anlagen genannt; z.Zt. sind das die Anlagen H130 und Z5. Zum Speicherkonzept auf ESA-Anlagen siehe Handbuch "Makroaufrufe an den Ablaufteil".

## ESD

(**E**xternal **S**ymbol **D**ictionary) ist das Verzeichnis der Externbezüge eines Objektmoduls/Bindemoduls (OM). 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/Externadreßbuch erzeugt.

## ESV

(**E**xternal **S**ymbol **V**ector) ist das Verzeichnis der Externbezüge eines Bindeladmoduls (LLM). (siehe auch ESD).

## Externadreßbuch

Auf Basis des ESV (**External Symbols Vector**) erstellt der BINDER das Externadreßbuch, wenn seine Erzeugung nicht unterdrückt wird.

Wird das Externadreßbuch später mitgeladen, ohne daß auch die LSD mitgeladen werden, so kann das Kommando %SDUMP %NEST benutzt werden, um die aktuelle Aufrufhierarchie auszugeben

## 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, %OUT und %QUALIFY) bzw. bis zu ihrer nächsten Änderung.

## Index

ist ein Teil eines Adreßoperanden. Mit einem Index wird die Position eines Tabellenelements 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. Nicht zugelassen in Kommandofolgen sind die AID-Kommandos %AID, %BASE, %DUMPFIL, %HELP, %OUT, %QUALIFY und die im AID-Basishandbuch, 14 Anhang [1], 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 symbolischen 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 Subkommandokettung, AID-Basishandbuch, Kapitel 6 [1].

### Lokalisierungsinformation

Die **statische** Programmverschachtelung zu der angegebenen Speicherstelle gibt Ihnen AID aus, mit

%DISPLAY %HLLOC(speicherref) für die **symbolische Ebene**

%DISPLAY %LOC(speicherref) für die **Maschinencode-Ebene**.

Im Gegensatz dazu erhalten Sie die **dynamische** Programmverschachtelung, die sogenannte Aufrufhierarchie zur aktuellen Programmunterbrechungsstelle mit %SDUMP %NEST.

### 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 und im Bindemodul hinterlegt. AID holt sich hieraus die Informationen zur symbolischen Adressierung.

### Namensraum

umfaßt alle zu einem Programmteil in den LSD-Sätzen verzeichneten Daten- und Anweisungsnamen.

## Objekt-Strukturliste

Auf Basis des ESD (**E**xternal **S**ymbol **D**ictionary) erstellt der TSOSLNK die Objekt-Strukturliste, wenn SYMTEST=MAP (Standardeinstellung) oder SYMTEST=ALL gilt. Wird die Objekt-Strukturliste später mitgeladen, ohne daß auch die LSD mitgeladen werden, so kann das Kommando %SDUMP %NEST benutzt werden, um die aktuelle Aufrufhierarchie auszugeben (siehe auch Externadreßbuch). Dabei ist zu berücksichtigen, daß dieser Ausgabe alle Informationen fehlen, für die LSD notwendig wären wie z.B. Source-Referenzen.

## Pointer-Operator

heißt die Zeichenfolge ->, die Sie in einem Adreßoperanden schreiben, wenn der Inhalt eines Speicherobjektes oder der Wert einer Konstanten zur indirekten Adressierung herangezogen wird (siehe AID-Basishandbuch, Kapitel 7.2.4.2 [1]). Bei indirekter Adressierung wird der Adressierungsmodus berücksichtigt.

## Programmraum

ist der Adreßraum, in dem das Programm abläuft, der aber auch Daten enthält. Auf ESA-Anlagen gibt es noch zusätzlich den Datenraum.

## Programmzustand

AID unterscheidet drei Programmzustände, in denen sich das zu testende Programm befinden kann:

1. Das Programm steht.  
%STOP-Kommando oder K2-Taste unterbrechen ein laufendes Programm. Außerdem wird das Programm unterbrochen, wenn ein %TRACE abgearbeitet ist oder wenn die Bedingung, die mit dem Operanden *steuerung* (%INSERT, %ON) vereinbart wurde, zur Unterbrechung des Programms geführt hat. Die Task befindet sich im Kommandomodus.
2. Das Programm läuft **ohne** Ablaufverfolgung.  
Das Programm wurde mit START-PROGRAM geladen und gestartet oder mit %RESUME gestartet oder fortgesetzt. Ist kein %TRACE vereinbart kann dazu auch %CONTINUE verwendet werden.
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 Qualifikationen können Sie eine Speicherstelle ansprechen, die nicht im AID-Arbeitsbereich liegt oder darin nicht eindeutig ist. Die **Basisqualifikation** legt fest, ob die Speicherreferenz im geladenen Programm oder in einem Speicherabzug liegt. **Bereichsqualifikationen** geben den Pfad zu dem Programm-

teil an, in dem eine Speicherreferenz liegt oder begrenzen die Wirkung eines Kommandos auf den damit bezeichneten Bereich. Wenn ein Operand durch eine Qualifikation überbestimmt ist (d.h. die Qualifikation ist überflüssig oder widersprüchlich), wird sie 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.

### SPID

(Space Identification) ist ein systemweit eindeutiges Kennzeichen für einen Datenraum. Datenräume und damit SPIDs gibt es nur auf ESA-Anlagen. Die SPID wird beim Anlegen eines Datenraumes vergeben.

### Speicherobjekt

besteht aus einer bestimmten 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, eine abschließende C/COM-Qualifikation, 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 es nur Adreßkonstanten sind.

Mit **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
- Adreßselektion

### Speichertyp

ist entweder der Datentyp, der im Quellprogramm festgelegt wurde oder der durch Typmodifikation gewählte. AID kennt die allgemeinen Speichertypen %X, %C, %P, %D, %F, %A und die Speichertypen zur Interpretation von Maschinenbefehlen %SX und %S (siehe %SET und AID-Basishandbuch, Kapitel 7 und Kapitel 10 [1]).

### Subkommando

ist ein Operand der Überwachungskommandos %CONTROLn, %INSERT, %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 im AID-Basishandbuch, Kapitel 6 [1] 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.

### 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 COBOL85- und FOR1-Programme können Sie mit %JUMP eine andere Fortsetzungsadresse vereinbaren.

### Zugriffsregister

%nAR,  $0 \leq n \leq 15$ . Die Zugriffsregister sind auf ESA-Anlagen parallel zu den Mehrzweckregistern vorhanden. Sie werden zur Adressierung von Datenräumen benötigt. In ihnen ist ein ALET (access list entry token) eingetragen, der über die Zugriffsliste auf einen Datenraum verweist. Wenn der AR-Modus eingeschaltet ist, werden die Zugriffsregister bei der Adreßumsetzung mit ausgewertet.



# Literatur

## Bestellen von Handbüchern

Die aufgeführten Handbücher finden Sie mit ihren Bestellnummern im *Druckschriftenverzeichnis* der Siemens Nixdorf Informationssysteme AG. Neu erschienene Titel finden Sie in den *Druckschriften-Neuerscheinungen*.

Beide Veröffentlichungen erhalten Sie regelmäßig, wenn Sie in den entsprechenden Verteiler aufgenommen sind. Wenden Sie sich bitte hierfür an Ihre zuständige Geschäftsstelle. Dort können Sie auch die Handbücher bestellen.

- [ 1]    **AID (BS2000)**  
Advanced Interactive Debugger  
**Basishandbuch**  
Benutzerhandbuch

*Zielgruppe*

Programmierer im BS2000

*Inhalt*

Überblick über AID, Beschreibung der Sachverhalte und Operanden, die für alle Programmiersprachen gleich sind.

Meldungen

Gegenüberstellung von AID - IDA

*Einsatz*

Testen von Programmen im Dialog- und Stapelbetrieb

- [ 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-Pro-  
    grammen;  
    Anwendungsbeispiel
- Einsatz*  
    Testen von PL/I-Programmen im Dialog- und Stapelbetrieb

- [ 5] **AID (BS2000)**  
Advanced Interactive Debugger  
**Testen von ASSEMBH-Programmen**  
Benutzerhandbuch
- Zielgruppe*  
Assembler-Programmierer
- Inhalt*  
Beschreibung der AID-Kommandos für das symbolische Testen von ASSEMBH-XT-Programmen;  
Anwendungsbeispiel
- Einsatz*  
Testen von ASSEMBH-XT-Programmen im Dialog- und Stapelbetrieb
- [ 6] BS2000  
**Benutzer-Kommandos (SDF-Format)**  
Benutzerhandbuch
- Zielgruppe*  
BS2000-Anwender
- Inhalt*  
Benutzer-Kommandos des BS2000 in der Syntax der Dialogschnittstelle SDF (System Dialog Facility)
- Einsatz*  
BS2000-Dialogbetrieb und -Stapelbetrieb mit SDF
- [ 7] BS2000  
**Makroaufrufe an den Ablaufteil**  
Benutzerhandbuch
- Zielgruppe*  
BS2000-Assembler-Programmierer (nicht privilegiert); Systemverwalter
- Inhalt*  
Alle Makroaufrufe an den Ablaufteil in lexikalischer Reihenfolge mit Hinweisen und Beispielen, einschließlich ausgewählter Makroaufrufe für das DVS und für TIAM;  
Zusammenstellung der Makroaufrufe nach Anwendungsgebieten.  
Ausführlicher Lernteil über Ereignissteuerung, Serialisation, Inter-Task-Kommunikation, Contingencies;
- Einsatz*  
BS2000-Anwendungsprogramme

- [ 8]    **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, Laders, Binde-Laders der Test- und Diagnosehilfen des Programmbibliothekssystems
- Bestellnummer*  
          U3216-J-Z53-1
- 
- [ 9]    **ASSEMBH**  
          Beschreibung
- Zielgruppe*  
          Assembler-Anwender im BS2000
- Inhalt*  
          Beschreibung des Sprachumfangs des Assemblers ASSEMBH;  
          Struktur der Assemblersprache, Assembleranweisungen;  
          Struktur und Elemente sowie Instruktionen der Makrosprache;  
          Strukturierte Programmierung mit ASSEMBH-XT, vordefinierte Makros für die strukturierte Programmierung
- 
- [10]    **ASSEMBH**  
          Benutzerhandbuch
- Zielgruppe*  
          Assembler-Anwender im BS2000
- Inhalt*  
          Aufruf und Steuerung des ASSEMBH;  
          Übersetzen, Binden, Laden und Starten;  
          Eingabequellen und Ausgaben des ASSEMBH;  
          Laufzeitsystem, Listenausgabe der strukturierten Programmierung;

Sprachverknüpfungen;  
Assembler-Diagnoseprogramm ASSDIAG;  
Dialogtesthilfe AID;  
Meldungen des ASSEMBH;  
Format der Assemblerbefehle

[ 11] **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

Gegenüberstellung von AID und IDA

*Einsatz*

Testen von Programmen im Dialog- und Stapelbetrieb

[ 12] **AID (BS2000)**

Advanced Interactive Debugger

**Testen von C/C++ - Programmen**

Benutzerhandbuch

*Zielgruppe*

C/C++ - Programmierer

*Inhalt*

Beschreibung der AID-Kommandos für das symbolische Testen von  
C/C++ - Programmen;

Anwendungsbeispiel

*Einsatz*

Testen von C/C++ - Programmen im Dialog- und Stapelbetrieb



# Stichwörter

%? 56  
%\*subkdoname 41  
%0G 51  
%1G 51  
%AID 18  
%AID CHECK 89  
    %MOVE 63  
%AID REP  
    %MOVE REP 67  
%AINT 22  
%AMODE 22, 93  
    Schlüsselwort Systeminformation 67  
%BASE 25, 33, 49, 51  
%CLASS6 31, 35, 54, 72, 104  
%CONTINUE 27  
%CONTROL aufheben 83  
%CONTROLn 28  
%CONTROLn an Testpunkten 32  
%DISASSEMBLE 33, 76, 78, 100  
%DISASSEMBLE-Protokoll 36  
%DISPLAY 38, 76, 78, 100  
    abfragen Adressierungsmodus 22  
%DISPLAY %DS 12  
%DS 41  
%DUMPFILe 25, 46, 49  
%FIND 51  
    im Subkommando 51  
    in Prozeduren 51  
%H%? 56  
%H? 56  
%HELP 56, 76, 78, 100  
%HELP, deutsch - englisch 18  
%INSERT 28, 58, 101  
%INSERT aufheben 83  
%LOC 10, 45

%M[ODE]24 23  
%M[ODE]31 23  
%MAP 40  
%MODEn 67  
%MOVE 18, 22, 63  
    %PC ändern 67  
    ändern Adressierungsmodus 22  
    Änderungsdialo g 63  
    REP 18  
%MOVE REP  
    %AID %REP 67  
%MOVE-Grenze 3900 64  
%nAR 12  
%NEST 87  
%nG 12  
%ON 69  
%ON aufheben 83  
%OUT 33, 36, 38, 43, 76, 87, 105  
%OUT %T 105  
%OUTFILE 78  
    AID-Linkname F6 68  
%PC 40, 93  
    löschen %INSERT 84  
%PC ändern, %MOVE 67  
%PCB 45  
%QUALIFY 30, 34, 39, 53, 64, 70, 80, 90, 103  
    CTX-Qualifikation 10  
%REMOVE 28, 31, 74, 83  
    %CONTROL[n] 28  
%REMOVE aktives Subkommando 61  
%RESUME 86, 101  
%S 113  
%SDUMP 87  
%SET 18, 89  
%STOP 99, 101  
    im Subkommando 99  
%TITLE 100  
%TRACE 76, 78, 100, 113  
    fortsetzen 27  
    maschinennah - symbolisch 101  
    Überwachungsaufwand 103  
%TRACE-Protokoll 105  
%TRACE-Standardwert  
    anzahl (10) 102

trace-bereich(Benutzerprogramm) 102  
%WRITE, %REMOVE 84

24-Bit-Adresse 22  
31-Bit-Adresse 22

**A**

Ablaufsteuerung 31, 61, 74  
Ablaufüberwachung 117  
Ablaufverfolgung 117  
Ablaufverfolgung einschalten 101  
Adresse  
    als Vorqualifikation 80  
    im Datenraum 14  
    im Programmraum 14  
Adressierung  
    %MODE{24|31} 117  
    Datenraum 117  
    XS-Anlage 117  
Adressierungsmodus 93, 117  
    abfragen, %DISPLAY 22  
    ändern, %MOVE 22  
    indirekte AID-Adresse 22  
    Testobjekt 22  
    XS-Rechner 67  
Adressierungsmodus ändern 67  
Adreßoperand 117  
Adreßselektion 35, 41, 54, 60, 65  
Adreßselektion, kompl-speicherref 13  
Adreßselektor 13, 41, 42, 54, 66, 71, 82, 91, 92  
Adreßversatz 13, 35, 41, 54, 60, 65, 71, 82, 91  
AID-Adreßinterpretation 22  
AID-Arbeitsbereich 10, 22, 25, 118, 119  
AID-Ausgabedateien 118  
AID-Eingabedateien 119  
AID-Linkname  
    Dn 49  
    F6 68  
AID-Literal 38, 42, 67, 93, 119  
aid-mode 22  
AID-Register 12, 40, 51  
    %0G 51  
    %1G 51

AID-Standard-Adreßinterpretation 22, 119  
Aktuelle  
    Aufrufhierarchie 119  
    CSECT 120  
ALET 48, 118  
ALET/SPID-Informationen 12  
ALET=  
    %nAR 12  
    %nG 12  
    X'f...f' 12  
ALET-Qualifikation 12, 39, 41, 53, 64, 81, 90  
alignment 51, 54  
ALL 51  
ändern  
    %PC 67  
    Befehlszähler 67  
ändern von Speicherinhalten 89  
Änderungsdialog 63, 89, 118  
    %MOVE 18  
    %SET 18  
Anfangsadresse 54, 104  
    für Adreßversatz 81  
Anfangsadresse CSECT  
    %DISASSEMBLE 11  
    %INSERT 11  
Anweisungsname 12, 41, 65, 91, 120  
anzahl 33  
    %DISASSEMBLE 34  
    %TRACE 102  
Arbeitsunterlage  
    Assembler-Listing 7  
    Binder-Listing 7  
    Locator map 7  
    Objectlisting 7  
AR-Modus 120  
Attribut  
    Ausgabetyt 120  
    Inhalt 120  
    Länge 120  
    Name 120  
    Speichertyp 120  
Attribute  
    Adresse 120  
    Adreß-Konstanten 120

- Konstanten 120
- Speicherobjekt 120
- Ausführungsbedingung 31, 74
- Ausgabe 43, 78, 87
  - Ablaufprotokoll 105
  - Adressen 38
  - Begrenzer 18
  - Längen 38
  - rückübersetzen Befehle 33
  - Speicherinhalt 38
  - Systeminformationen 38
- Ausgabe SYSLST 57
- Ausgabe von Treffern, %FIND 51
- Ausgabedatei
  - F6 79
  - katalogisieren 79
  - öffnen 79
  - schließen 78
  - zuweisen 78
- Ausgabe-Kommando
  - %DISASSEMBLE 33, 76, 78, 100
  - %DISPLAY 76, 78, 100
  - %HELP 56, 76, 78, 100
  - %TRACE 76, 78, 100, 101
- Ausgabemedium 38, 43, 56, 57, 87
- Ausgabesteuerung 36, 76
- Ausgabetyt 120
- Auswirkung %BASE
  - Subkommando 25
- automatische Änderung im Speicher 51

## B

- basis 25
- Basisqualifikation 10, 25, 26, 30, 39, 59, 64, 71, 90, 104, 121
- Bedeutung
  - Fehlermeldung 56
- Bedienung von AID 56
- Befehlszähler 66
- Befehlszähler als start 35
- Begrenzer der AID-Ausgabefelder 18
- Bereichsgrenzen 121
  - COMMON 121
  - CSECT 121
  - V-Adresse 121

Bereichsgrenzen überschreiten

CSECT/COMMON 13

Bereichsqualifikation

COMMON 10

CSECT 10

Ladeeinheit 10

Objektmodul 10

Bereichsqualifikationen 10

Bereichsüberprüfung 121

Bereichsüberschreitung 65

Bindemodul (OM) 11

Binden und Laden mit DLL/DBL 8

## C

C/COM-Qualifikation bezeichnet Speicherstelle 12

C=csect 10, 30, 35, 40, 53, 60, 65, 71

Speicherreferenz 13

C=N'...'

CSECT-Name mit Sonderzeichen 11

Character-Literal, Sonderfall 94

CHECK 18, 19

COM=common 10, 11, 13, 30, 35, 40, 53, 60, 65, 71, 81, 91, 104

Speicherreferenz 13

COMMON 11

control-bereich 30

COMMON-Anfangsadresse

testpunkt 60

COM-Qualifikation 11, 30, 40

control-bereich 28, 29

Anfangsadresse-Endadresse 30

COMMON 30

CSECT 30

C-Qualifikation 30, 40

CSECT 30, 81, 104

aktuelle 120

control-bereich 30

in mehreren Kontexten 10

nicht aktuelle Unterbrechungsstelle 10

CSECT, gesamt

%CONTROLn 11

%DISPLAY 11

%FIND 11

%MOVE 11

%SET 11

- %TRACE 11
- CSECT-Anfangsadresse
  - testpunkt 60
- CSECT-Name aus
  - BINDER-Lauf 11
  - LMS-Lauf 11
  - Quellprogramm 11
  - TSOSLNK-Lauf 11
- CSECT-Name mit Sonderzeichen
  - C=N'...' 11
- CSECT-Namen im Testobjekt
  - %DISPLAY %MAP 11
  - %DISPLAY %SORTEDMAP 11
- CTX=kontext 10, 30, 34, 39, 53, 59, 64, 71, 81, 90, 104
- CTXPHASE
  - Standardkontextname (TSOSLNK) 10
- CTX-Qualifikation 30
  - %QUALIFY 10

**D**

- datei 49, 79
- Datei-Ausgabe 43, 57, 77, 88
- daten 38
- Datenname 12, 41, 65, 91
  - Konstanten 121
  - Variablen 121
- Datenraum 121, 122
- Datenraum bezeichnen
  - ALET 12
  - SPID 12
- Datentyp 122
- DELIM 18, 21
- Dn, Dump-Datei 10
- Doppelwortgrenze, suchen an 55
- dump-bereich 87
- Dump-Datei 10, 25, 49
  - öffnen 49
  - schließen 49
- Durchlaufzähler 40, 61, 66, 74, 92
  - erhöhen 31, 61

### E

E=VM 69  
einfügen MODIFY-ELEMENT 19  
Eingabepuffer 122  
einschalten  
    Ablaufverfolgung 101  
    Protokollierung 101  
Einzelkommando  
    %DUMPFIL 49  
    %HELP 56  
empfänger 63, 89  
Endadresse 54, 104  
E-Qualifikation 10, 23, 30, 34, 39, 53, 59, 64, 71, 81, 90  
ereignis 69  
    festlegen 69  
    löschen 69  
    Tabelle 73  
erhöhen Durchlaufzähler 61  
ESA 122  
ESA-Anlagen  
    Datenräume 121  
ESA-Programm  
    ALET-Qualifikation 12  
    kompl-speicherref 14  
    SPID-Qualifikation 12  
ESA-Unterstützung 41, 47, 48  
ESD, external symbol dictionary 122  
ESD-Liste 87

### F

F6, Linkname 19  
Fachwörter 117  
Fehlermeldung, Information 56  
festlegen globaler Vereinbarungen 22  
find-bereich 51  
Fortsetzungsadresse,%1G 51

### G

globale Einstellung 123  
globale Einstellungen vereinbaren 18  
Grenze  
    %FIND %CLASS6 54  
    %FIND 64 KB 54  
    %MOVE 3900 64

%ON 64 KB 70

## H

Halbwortgrenze, suchen an 55

Hardcopy-Ausgabe 43, 57, 77, 88

## I

Index 123

indirekte Adressierung 13, 35, 41, 54, 60, 65, 71, 82

Informationen über

ALET-Qualifikationen 12

SPID-Qualifikationen 12

Informieren über

AID-Bedienung 56

AID-Meldung 56

info-ziel 56

Interpretation

des Bindestrichs 18

von indirekten Adressen 22

## K

K2-Taste

drücken 99, 101

Suche abbrechen 51

Kommandofolge 31, 74, 123

Kommando-Kurzbeschreibung 56

Kommandomodus 123

komplexe Speicherreferenz 13, 35, 41, 54, 60, 65, 71, 82, 91

ALET/SPID-Qualifikation 14

bezeichnet Speicherstelle 12

kompl-speicherref

Adreßselektion 13

ESA-Programme 14

indirekte Adressierung 13

Längenmodifikation 13

Typmodifikation 13

Konstante 124

Kontext 10

Shared-Code-Programm 10

Kontrolle Änderungsdialog 89

kriterium 28, 102

### L

L=ladeeinheit 10, 30, 35, 39, 53, 60, 65, 71, 81, 91, 104  
Laden mit ELDE 8  
LANG 18  
Längenmodifikation 13, 35, 41, 54, 60, 65, 71, 82, 91  
    kompl-speicherref 13  
Längenselektor 42, 66, 92  
Laufzeitsystem 99  
LIFO-Prinzip 69, 72, 124  
    subkdo-ketten 58  
link 49, 78  
Linkname 49, 78  
    Dn 49  
    Dump-Datei 49  
    F6 19  
Linkname F6  
    %MOVE 68  
    REP 68  
Literal  
    ausgeben 51  
    suchen 51  
LMS-Korrekturanweisung 19  
LOCAL#DEFAULT 10  
    Standardkontextname 10  
Lokalisierungsinformation 124  
löschen  
    %.subkdoname 84  
    %CONTROL 83  
    %INSERT 83  
    ereignis 84  
    testpunkt 84  
    write-ereignis 84  
löschen %ON 83  
löschen ereignis  
    %ERRFLG 84  
    %LPOV 84  
    %SVC 84  
löschen testpunkt  
    %REMOVE 58  
    Programmende 58  
    steuerung-Vereinbarung 58  
LOW 18, 20  
    Klein-/Großbuchstaben 18  
L-Qualifikation 10, 11, 30

LSD 12, 38, 41, 65, 91, 124  
List for Symbolic Debugging 124

## M

Maschinenbefehle  
  Testpunkt setzen 59  
  Typen 29  
  überwachen 29  
medium-u-menge 38, 43, 76, 87  
  %HELP 57  
Meldungen  
  AID 56  
  von AIDSYS 56  
Meldungsnummer  
  AID0n 56  
  IDA0n 56  
  In 57  
Metasyntax 5  
MODIFY-ELEMENT-Anweisung 19

## N

Namensraum 124  
numerische Übertragung 89

## O

O=objektmodul 10, 30, 35, 39, 53, 60, 65, 71, 81, 91, 104  
Objekt-Strukturliste 87, 125  
Objekt-Strukturliste fehlt 11  
öffnen Dump-Datei 49  
O-Qualifikation 10, 11, 30  
OV 18, 20  
Overlay 18

## P

Pointer-Operator 113  
Programm  
  anhalten 99  
  fortsetzen 27, 31, 61, 74  
  mit Überlagerungsstruktur 70  
  starten 27, 86, 101  
Programm mit Überlagerungsstruktur 18  
Programmbereich  
  überwachen 29  
Programmzustand 125

ändern 27

Programmzustand ändern 86, 99

Protokollierung einschalten 101

Punkt 23, 41, 70, 81, 90, 103

## Q

Qualifikation 125

## R

Register 40, 66, 92

REP 18, 19, 63, 67

%MOVE 18, 63

Linkname F6 68

## S

schließen Dump-Datei 49

Schlüsselwort 13

%MODEn 22, 67

%PCB 45

bezeichnet Speicherstelle 12

schlüsselwort 35, 40, 66, 92, 104

kriterium 29, 102

Speicherklasse 31, 54, 72

Schreibüberwachung 69

Sedezimal-Literal 51

seitenkopf 100

Seitenkopf-Text definieren 100

Seitenüberschrift 100

Seitenvorschub 43

Seitenzähler 100

sender 63, 89

show-ziel 97

Source-Referenz 126

Speicherbereich 40, 52, 54, 104

Speicherinformation rückübersetzen 33

Speicherobjekt 30, 34, 39, 53, 59, 64, 70, 71, 90, 104, 120

Bereichsgrenzen 121

Datenraum 14

Programmraum 14

Speicherreferenz

C=csect 13

COM=common 13

einfach 126

komplex 126

Speicherstelle bezeichnen  
  C/COM-Qualifikation 12  
  komplexer Speicherreferenz 12  
  Schlüsselwort 12  
  V-Adresse 12  
Speichertyp 91, 127  
Speichertyp Character, Sonderfall 94  
Speichertyp, Schlüsselwörter 66, 92  
SPID 48  
SPID-Qualifikation 12, 39, 41, 53, 64, 81, 90  
  SPID='X'f...'f' 12  
Standard-Adreßinterpretation 22  
Standardkontextname  
  CTXPHASE 10  
  LOCAL#DEFAULT 10  
start 33, 34  
steuerung-Operand, %INSERT 27, 101  
STOP-Meldung 99  
subkdo 28, 31, 69, 73  
  %CONTROL 31  
  %INSERT 60  
  %ON 73  
Subkommando 27, 31, 61, 69, 72, 101, 127  
  ausführen 58  
  definieren 69  
  Name 31, 74  
  verketteten 58, 61, 69, 74  
  verschachteln 61, 74  
suchbegriff 51  
  Character-Literal 52  
  Sedezimal-Literal 52  
suchen an  
  Doppelwortgrenze 55  
  Halbwortgrenze 55  
  Wortgrenze 55  
Suchen von Zeichenfolgen 51  
SYMCHARS 18, 20  
SYSLST 38, 43, 77, 88  
SYSOUT 51  
Systeminformation 40  
Systeminformation ausgeben 38

### T

Terminal-Ausgabe 43, 57, 77, 88  
testpunkt  
    COMMON 60  
    CSECT 60  
    festlegen 58  
    löschen 61  
    V-Adresse 60  
Testpunkt aufheben 83  
Testvereinbarungen aufheben 83  
trace-bereich 103  
Trefferadresse,%0G 51  
Trefferausgabe 51  
Typmodifikation 13, 35, 41, 54, 60, 65, 71, 82, 91, 113  
Typmodifikation ohne Auswirkung 65

### U

Überlagerungsstruktur 70  
überschreiten Bereichsgrenzen  
    CSECT/COMMON 13  
übertragen  
    linksbündig 63  
    numerisch 89  
Übertragung 89  
    %MOVE 63  
    %SET 89  
Überwachen  
    an Testpunkten 58  
überwachen  
    Ereignisse 69  
    Programmbereich 29, 103  
    Schreibzugriff 69  
    Typ Maschinenbefehl 28, 102  
Überwachungsaufwand minimieren 28, 101  
Überwachungsaufwand, %TRACE 103  
unterbrechen %TRACE 101  
Unterbrechungsstelle 29, 127

**V**

- V-Adresse 12, 35, 40, 60, 65, 71, 81, 91
  - Basisqualifikation 12
  - Bereichsqualifikation 12
  - ESA-Programm 12
- V-Adresse bezeichnet Speicherstelle 12
- verketten
  - Subkommando 61
- verschachteln
  - Subkommando 61
- virtuelle Adresse 12
- virtueller Speicher 10
  - geladenes Programm 25
  - Speicherabzug 25
- Vorqualifikation 23, 30, 39, 53, 64, 70, 81, 90, 103
- vorqualifikation 34, 41
- vorqualifikation definieren 80
- vorschubsteuerung 43

**W**

- Wildcard-Symbol
  - %FIND 52
  - suchbegriff 52
- Wortgrenze, suchen an 55
- write-ereignis 69
- write-ereignis löschen 84

**X**

- XS-Rechner 93
- XS-Rechner, Adressierungsmodus 67

**Z**

- Zeichen-Literal 51
- Zeilenvorschub 43
- ziel 83, 87
- ziel-kommando 76
- Zugriffsregister 12, 122, 127
- zulässige Kombinationen bei %SET 94
- Zusatzinformation 76
- zusätzlicher Überwachungsaufwand 28, 101



# Inhalt

|          |                                                     |           |
|----------|-----------------------------------------------------|-----------|
| <b>1</b> | <b>Einleitung</b> .....                             | <b>1</b>  |
| 1.1      | Zielgruppe .....                                    | 1         |
| 1.2      | Konzept der AID-Handbücher .....                    | 2         |
| 1.3      | Änderungen gegenüber AID V2.0A .....                | 3         |
| 1.4      | Verwendete Darstellungsmittel .....                 | 4         |
| <b>2</b> | <b>Metasyntax</b> .....                             | <b>5</b>  |
| <b>3</b> | <b>Voraussetzungen zum Testen</b> .....             | <b>7</b>  |
| 3.1      | Übersetzen, Binden, Laden .....                     | 7         |
| 3.2      | Kommandos zu Beginn einer Testsitzung .....         | 8         |
| <b>4</b> | <b>Maschinencode-spezifische Adressierung</b> ..... | <b>9</b>  |
| 4.1      | Qualifikationen .....                               | 9         |
|          | Basisqualifikation .....                            | 10        |
|          | Bereichsqualifikationen .....                       | 10        |
| 4.2      | Speicherreferenzen .....                            | 12        |
| <b>5</b> | <b>AID-Kommandos</b> .....                          | <b>17</b> |
|          | %AID .....                                          | 18        |
|          | %AINT .....                                         | 22        |
|          | %BASE .....                                         | 25        |
|          | %CONTINUE .....                                     | 27        |
|          | %CONTROLn .....                                     | 28        |
|          | %DISASSEMBLE .....                                  | 33        |
|          | %DISPLAY .....                                      | 38        |
|          | %DUMPFILe .....                                     | 49        |
|          | %FIND .....                                         | 51        |
|          | %HELP .....                                         | 56        |
|          | %INSERT .....                                       | 58        |
|          | %MOVE .....                                         | 63        |
|          | %ON .....                                           | 69        |
|          | %OUT .....                                          | 76        |
|          | %OUTFILE .....                                      | 78        |
|          | %QUALIFY .....                                      | 80        |
|          | %REMOVE .....                                       | 83        |

|          |                                 |            |
|----------|---------------------------------|------------|
|          | %RESUME .....                   | 86         |
|          | %SDUMP .....                    | 87         |
|          | %SET .....                      | 89         |
|          | %SHOW .....                     | 97         |
|          | %STOP .....                     | 99         |
|          | %TITLE .....                    | 100        |
|          | %TRACE .....                    | 101        |
| <b>6</b> | <b>Anwendungsbeispiel .....</b> | <b>109</b> |
| 6.1      | Assembler-Programm .....        | 109        |
| 6.2      | Testablauf .....                | 111        |
|          | <b>Fachwörter .....</b>         | <b>117</b> |
|          | <b>Literatur .....</b>          | <b>129</b> |
|          | <b>Stichwörter .....</b>        | <b>135</b> |

---

# AID V2.1A (BS2000/OSD)

## Testen auf Maschinencode-Ebene

### *Zielgruppe*

Programmierer und Tester

### *Inhalt*

- Beschreibung der AID-Kommandos für das Testen auf Maschinencode-Ebene
- Anwendungsbeispiel

Neu 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.

**Ausgabe: April 1995**

**Datei: AID\_MC.PDF**

BS2000 ist ein eingetragenes Warenzeichen der Siemens Nixdorf Informationssysteme AG  
Copyright © Siemens Nixdorf Informationssysteme AG, 1995.

Alle Rechte vorbehalten.  
Liefermöglichkeiten und technische Änderungen vorbehalten.

Alle verwendeten Hard- und Softwarenamen sind Handelsnamen und/oder Warenzeichen der jeweiligen Hersteller.



## 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](mailto:ts.fujitsu.com).

The Internet pages of Fujitsu Technology Solutions are available at

[http://ts.fujitsu.com/...](http://ts.fujitsu.com/)

and the user documentation at <http://manuals.ts.fujitsu.com>.

Copyright Fujitsu Technology Solutions, 2009

## 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](mailto:ts.fujitsu.com).

Die Internetseiten von Fujitsu Technology Solutions finden Sie unter

[http://de.ts.fujitsu.com/...](http://de.ts.fujitsu.com/), und unter <http://manuals.ts.fujitsu.com> finden Sie die Benutzerdokumentation.

Copyright Fujitsu Technology Solutions, 2009