Deutsch

FUJITSU Software

BS2000 OSD/BC V11.0

Makroaufrufe an den Ablaufteil

Benutzerhandbuch

Ausgabe Juli 2017

Kritik… Anregungen… Korrekturen…

Die Redaktion ist interessiert an Ihren Kommentaren zu diesem Handbuch. Ihre Rückmeldungen helfen uns, die Dokumentation zu optimieren und auf Ihre Wünsche und Bedürfnisse abzustimmen.

Sie können uns Ihre Kommentare per E-Mail an manuals@ts.fujitsu.com senden.

Zertifizierte Dokumentation nach DIN EN ISO 9001:2008

Um eine gleichbleibend hohe Qualität und Anwenderfreundlichkeit zu gewährleisten, wurde diese Dokumentation nach den Vorgaben eines Qualitätsmanagementsystems erstellt, welches die Forderungen der DIN EN ISO 9001:2008 erfüllt.

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Copyright und Handelsmarken

Inhalt

1 Einleitung

Das vorliegende Handbuch beschreibt alle Makroaufrufe, die Sie an den Ablaufteil und die Systemdienste von BS2000 richten können, sowie Makros für den Zugriff zu Datenstationen.

1.1 Zielsetzung und Zielgruppen des Handbuches

Das Handbuch wendet sich an alle Assembler-Programmierer von BS2000.

Sie sollten mit der Assemblersprache und der Verwendung eines Makros vertraut sein. Als Unterlage dafür dienen Ihnen die Handbücher „Assemblerbefehle (BS2000)“ [1] und „ASSEMBH“ [2]. Das vorliegende Handbuch enthält in einer Zusammenfassung das Wichtigste, was Sie über die Benutzung eines Makros wissen sollten (siehe Kapitel „Benutzung von Makroaufrufen“ auf Seite 17).

Über den Umgang mit BS2000 sollten Sie ebenfalls Kenntnisse haben. Darüber können Sie sich im Handbuch „Einführung in die Systembetreuung“ [10] informieren.

1.2 Konzept des Handbuchs

Das Kapitel „Komponenten von BS2000“ zeigt Ihnen die Einbettung des Ablaufteils in die Komponenten von BS2000. Es erläutert auch die Unterschiede zwischen Auftrag, Task und Prozess.

Die formalen Grundlagen, die Sie bei der Erstellung von Makros wissen müssen, vermittelt das Kapitel „Benutzung von Makroaufrufen“.

Das Kapitel „Anwendungsgebiete und Kurzbeschreibungen“ enthält eine Zusammenstellung der Aufgabengebiete, zu denen Makros in diesem Handbuch beschrieben werden. Die Makros sind entsprechend ihrem Zweck, der durch eine Kurzbeschreibung angegeben ist, zu Funktionsgruppen zusammengestellt. Wo es erforderlich ist, wird die Anwendung einer Funktionsgruppe, also das Zusammenspiel mehrerer Makros, näher erklärt.

Im Kapitel „Beschreibung der Makroaufrufe“ werden alle Makros des Ablaufteils und einige ausgewählte Makros anderer Komponenten in alphabetischer Reihenfolge beschrieben.

In einigen Fällen wird zum besseren Verständnis die DSECT des Makros abgebildet. Kleine Beispielprogramme ergänzen die Beschreibung.

Einen Makro, der hier beschrieben ist, können Sie nach folgenden Kriterien suchen:

- alphabetisch über seine Bezeichnung (Kolumnentitel, Inhalts- und Stichwortverzeichnis und Anhang (Seite 1182)).
- über seine Funktion (Inhaltsverzeichnis, sowie Seite 47);
- über seine SVC-Nummer (Kapitel „Anhang“ auf Seite 1188).

Im Anhang werden Makros beschrieben, die nur noch aus Kompatibilitätsgründen unterstützt werden. Es folgen Zusammenstellungen von Makros sowie von Funktionstastencodes.

Readme-Datei

Funktionelle Änderungen der aktuellen Produktversion und Nachträge zu diesem Handbuch entnehmen Sie bitte ggf. der produktspezifischen Readme-Datei.

Readme-Dateien stehen Ihnen online bei dem jeweiligen Produkt zusätzlich zu den Produkthandbüchern unter *http://manuals.ts.fujitsu.com* zur Verfügung. Alternativ finden Sie Readme-Dateien auch auf der Softbook-DVD.

Informationen unter BS2000

Wenn für eine Produktversion eine Readme-Datei existiert, finden Sie im BS2000-System die folgende Datei:

`SYSRME.<product>.<version>.<lang>`

Diese Datei enthält eine kurze Information zur Readme-Datei in deutscher oder englischer Sprache (<lang>=D/E). Die Information können Sie am Bildschirm mit dem Kommando `SHOW-FILE` oder mit einem Editor ansehen.
Das Kommando `/SHOW-INSTALLATION-PATH INSTALLATION-UNIT=<product>` zeigt, unter welcher Benutzerkennung die Dateien des Produkts abgelegt sind.

Ergänzende Produkt-Informationen

Aktuelle Informationen, Versions-, Hardware-Abhängigkeiten und Hinweise für Installation und Einsatz einer Produktversion enthält die zugehörige Freigabemitteilung. Solche Freigabemitteilungen finden Sie online unter *http://manuals.ts.fujitsu.com*.

1.3 Änderungen gegenüber dem Vorgänger-Handbuch

Das Handbuch wurde an BS2000 OSD/BC V11.0 angepasst.

Die Makros von Subsystemen mit eigener Version haben folgenden Beschreibungsstand: BLSSERV V2.8A, TIAM V13.2A, VTSU V13.3A, siehe Seite 171.

Weitere wichtige Änderungen

Makro	Änderungen
SHOWMP	Neuer Makro zur Anzeige von Memory Pools
NDGUINF	Der Makro ist obsolet (Globalspeicher wird nicht mehr bedient). Seine Beschreibung wurde aus diesem Handbuch entfernt.

1.4 Darstellungsmittel

In diesem Handbuch werden folgende Kurzbezeichnungen verwendet:

- **BS2000-Server** für die Server mit /390-Architektur und die Server mit x86-Architektur. Diese Server werden mit dem entsprechenden BS2000-Betriebssystem betrieben.
- **/390-Server** für die Server Unit /390 der Fujitsu Server BS2000 SE Serie und die Business Server der S-Serie
- **x86-Server** für die Server Unit x86 der Fujitsu Server BS2000 SE Serie
- **SE Server** für die Fujitsu Server BS2000 der SE Serie (Server-Units /390 und x86)
- **S-Server** für die Business Server der S-Serie (/390-Architektur)

Die Zeichenfolgen `<date>`, `<time>` und `<ver>` bezeichnen in den Beispielen die aktuellen Ausgaben für Datum, Uhrzeit und Version, wenn die Beispiele sonst Datums-, Zeit- und Versions-unabhängig sind.

In diesem Handbuch werden folgende Darstellungsmittel verwendet:

MAKRO	Makronamen werden im Fließtext fett ausgezeichnet
`eingabe`	Eingaben in Beispielen werden in halbfetter Schreibmaschinenschrift dargestellt
`Ausgabe`	DSECTS, Übersetzungslisten oder Ausgaben in Beispielen werden in Schreibmaschinenschrift dargestellt
	für Hinweise auf besonders wichtige Informationen

Literaturhinweise werden im Text in Kurztiteln und eckigen Klammern angegeben.
Der vollständige Titel jeder Druckschrift, auf die verwiesen wird, ist im Literaturverzeichnis aufgeführt.

2 Komponenten von BS2000

Das Betriebssystem BS2000 besteht aus zwei Hauptgruppen:
einem zentralen Teil und den Benutzerprogrammen
(beispielsweise Sprachübersetzer, Dateibearbeiter, Dienstprogramme).

Der zentrale Teil lässt sich mit der Kommandosprache oder der Makros von BS2000 vom Anwender beeinflussen und steuern.

Der zentrale Teil untergliedert sich in:

- Ablaufteil
- Kommunikations-Zugriffssystem
- Datenverwaltungssystem
- Systemdienste

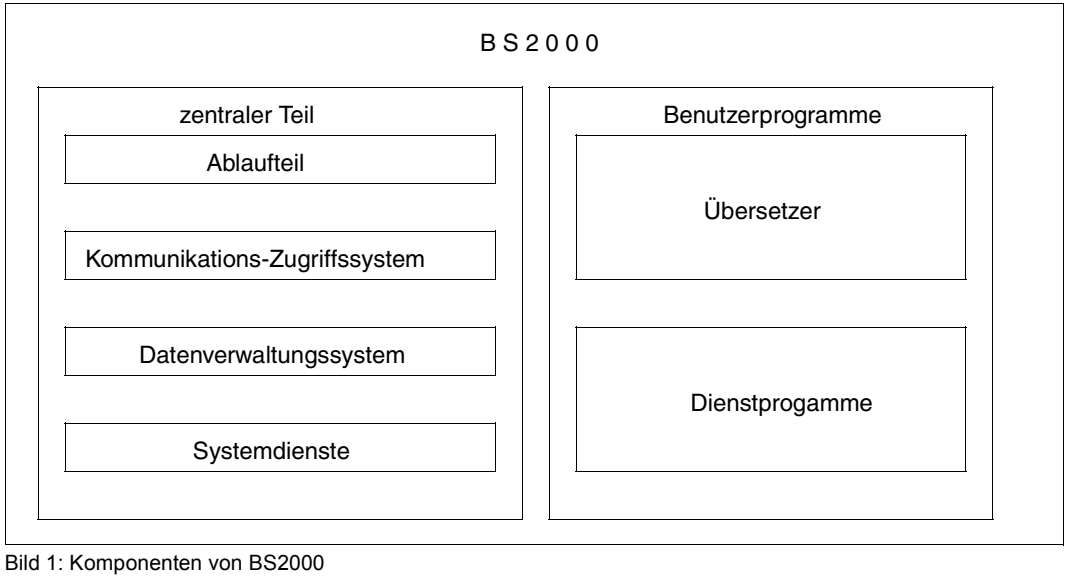


Bild 1: Komponenten von BS2000

Der **Ablaufteil** enthält die zentralen Steuerungsroutinen des Betriebssystems und erfüllt folgende Aufgaben:

– Steuerung des Ablaufs sämtlicher Aufträge (Jobs), zum Beispiel aller Dialog-, Batch- oder Spool-Aufträge
– Verwaltung von virtuellem und realem Speicher
– Formale Analyse der Kommandos
– Durchführung des Spulbetriebs
– Ein-, Ausgabe zu den Konsolen
– Systemabrechnung

BS2000-Makroaufrufe an den Ablaufteil ermöglichen eine programmspezifische Nutzung der zentralen Steuerungsroutinen.
Dieses Handbuch beschreibt u.a. alle Makroaufrufe an den Ablaufteil.

Das **Kommunikations-Zugriffssystem** (DCM) führt folgende Aufgaben durch:

– Datenübermittlung zwischen Programm und Datenstationen bzw. anderen Programmen
– Verwaltung der hierfür notwendigen Betriebsmittel

BS2000-Makroaufrufe an das Kommunikationszugriffssystem steuern u.a. die Arbeitsweise der Datensichtgeräte. Von den Makros des Kommunikationszugriffsystems ist in diesem Handbuch eine Beschreibung der Makros für den Zugriff zu Datensichtstationen enthalten. Das Handbuch „TIAM“ [16] enthält den Funktionsumfang dieser Makros. Weitere Makros sind im Handbuch „DCAM“ [15] beschrieben.

Zum **Datenverwaltungssystem** (DVS) gehören Routinen, die für folgende Funktionen zuständig sind:

– Dateiverwaltung, zum Beispiel Katalogisieren, Speichern, Wiederauffinden und Löschen von Dateien
– Unterstützung der Dateizugriffsmethoden
– Ein-/Ausgabe zu den peripheren Geräten (außer Bedien- und Datenstationen)

BS2000-Makroaufrufe an das Datenverwaltungssystem haben die Datei-, Datenträger- und Gerätebehandlung zum Ziel. Makroaufrufe an das Datenverwaltungssystem finden Sie im Handbuch „DVS Makros“ [7].

Als **Systemdienste** werden zusätzliche Funktionen des zentralen Teils bezeichnet, wie

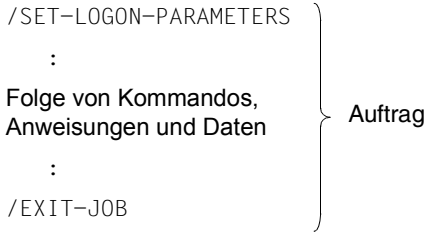
- Dialogtesthilfe AID
- Dynamischer Bindelader (DBL)
- Binder (BINDER)

Die BS2000-Makroaufrufe an die Dialogtesthilfe dienen der Fehlerbehandlung in geladenen Programmen, da mit dieser Systemfunktion Programme überwacht und in ihrem Ablauf beeinflusst werden können.
Dieses Handbuch beschreibt neben den Makroaufrufen an den Ablaufteil alle Makroaufrufe an die Systemdienste.

Auftrag, Task, Prozess

BS2000 kennt u.a. die Begriffe Auftrag (Job), Task und Prozess. Mit diesen Begriffen werden bestimmte Zustände und Aktivitäten einer dem Betriebssystem übergebenen Aufgabe bezeichnet.

Auftrag: (Job) Folge von Kommandos, Anweisungen und Daten, die zwischen den Kommandos SET-LOGON-PARAMETERS und EXIT-JOB eingeschlossen ist. Es wird zwischen Batch-Auftrag (Batch-Job) und Dialogauftrag unterschieden. In einem Batch-Auftrag wird die Folge von Kommandos, Anweisungen und Daten aus einer Datei gelesen; in einem Dialogauftrag wird die Folge interaktiv an der Datensichtstation eingegeben. Ein Auftrag (Job) wird vom Job-Management einer Jobklasse zugeordnet und in die entsprechende Warteschlange eingereiht. Der Auftrag erhält eine Auftragsnummer (TSN), mit deren Hilfe er während seiner Verweilzeit im System angesprochen werden kann.



Task: Aus der Sicht des Betriebssystems wird ein Auftrag zur Task, wenn ihm Systemressourcen (CPU, Speicher, Geräte) zugeteilt werden. Die Task wird vom Task-Management verwaltet, ein Task-Control-Block (TCB) wird eingerichtet.

Prozess: Die innerhalb der Task ablaufenden Aktivitäten auf Programm- bzw. Modulebene werden als die Prozesse der Task bezeichnet. Jeder Prozess besitzt einen Process-Control-Block (PCB), der der Aufnahme des Programmkontexts bei Programmunterbrechungen dient. Die Prozesse einer Task werden über den Task-Control-Block verwaltet.

Das folgende Bild soll den Zusammenhang zwischen den Begriffen Auftrag, Task und Prozess nochmals verdeutlichen:

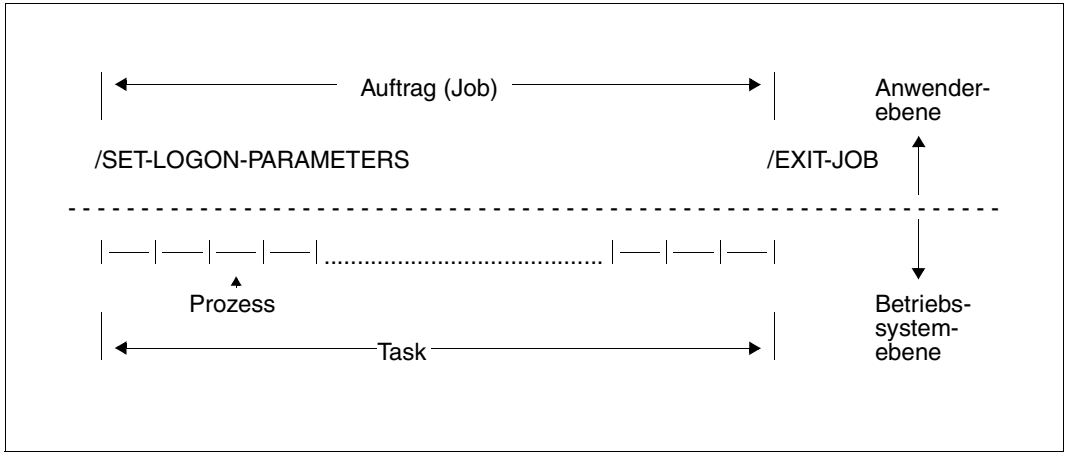


Bild 2: Auftrag, Task und Prozess

3 Benutzung von Makroaufrufen

3.1 Bearbeitung eines Makroaufrufs durch den Assembler

Ein Makroaufruf ist eine Anweisung im Primärprogramm. Er besteht aus einem mnemotechnischen Operationscode (Makroname), mit dem die auszuführende Funktion bezeichnet wird. Zusätzlich können als Operanden Werte angegeben sein, die die zur Ausführung nötige Information darstellen, oder die die gewünschte Funktion genauer beschreiben (z.B. **EXIT** CONTINU=YES). Die „Beschreibung der Makroaufrufe“ auf Seite 171 informiert über die Operanden, die jeweils angegeben werden können oder müssen.

Jede Primärprogrammanweisung, also auch ein Makroaufruf, wird beim Übersetzen des Programms vom Assembler verarbeitet. Als Resultat dieser Verarbeitung werden Makroaufrufe im Primärprogramm durch die zugehörigen Makroauflösungen ersetzt. Eine Makroauflösung besteht aus Befehlen und Assembleranweisungen, die zusammen die Funktion ausführen, die von dem Makroaufruf erwartet wird. Wenn man von Anweisungen spricht, die durch einen Makroaufruf generiert werden, ist die Makroauflösung gemeint.

Die Makroauflösung wird vom Assembler über die Makrodefinition generiert. Zu jedem Makroaufruf gehört eine Makrodefinition. Makroaufrufe können also nur gegeben werden, wenn eine entsprechende Makrodefinition existiert. Makrodefinitionen können vom Benutzer geschrieben werden (siehe Handbuch „ASSEMBH“ [2]) oder sie werden als Teil des Betriebssystems dem Benutzer zur Verfügung gestellt wie diejenigen, deren Aufrufe im vorliegenden Handbuch genannt werden. Alle zur Verfügung stehenden Makrodefinitionen sind in der Makrobibliothek gesammelt. Eine Makrodefinition ist die Vorlage, nach der der Assembler die Makroauflösung generiert.

Die Makrodefiniton wird gemäß den Operandenangaben im Aufruf modifiziert, und aktuelle Werte werden eingesetzt.

Das Primärprogramm, in dem alle Makroaufrufe durch die Makroauflösung ersetzt sind, wird dann in Maschinensprache übersetzt. Im Übersetzungsprotokoll sind die Makroauflösungen mit ausgedruckt, sofern dies nicht durch die Assembleranweisung PRINT NOGEN (siehe Handbuch „ASSEMBH“ [2]) unterdrückt wurde. Die durch den Makroaufruf geforderte Funktion wird erst während des Programmlaufs ausgeführt.
Mit der Bezeichnung Makro meint man die Funktion, die auf die beschriebene Weise, also durch das Zusammenwirken von Makroaufruf und Makrodefinition, realisiert wird.

3.2 Syntaxdarstellung der Makroaufrufe

Format eines Makroaufrufs

Das Makroaufruf-Format ist aus zwei Feldern aufgebaut.
Das obere Feld enthält das optionale Namensfeld und den Makronamen.
Das untere Feld enthält die möglichen Operanden.

[name] Makroname
<operand$_1$> ,<operand$_2$>

Ein Eintrag im **Namensfeld** ist grundsätzlich erlaubt. Der hier angegebene Name stellt die symbolische Adresse des ersten Befehls der Makroauflösung dar. Der Benutzer kann diese Adresse z.B. als Sprungziel oder als Haltepunkt für die Dialogtesthilfe benutzen.
Die Aufruf-Formate in diesem Handbuch verzichten allgemein auf die Darstellung des Namensfeldes außer in den Fällen, wo einer Angabe im Namensfeld besondere Bedeutung zukommt. Ein Beispiel dafür ist der Aufruf eines Makros vom Typ S in der L-Form (siehe Seite 30): die im Namensfeld angegebene symbolische Adresse ist nötig zur Verknüpfung des Datenbereichs mit dem Befehlsteil des Makros (E-Form). Weitere Beispiele sind die Makros **ARDS**, **CUPAB**, **DCSTA** und **TMODE**, wo der Standardname des generierten Pseudoabschnittes durch die Angabe im Namensfeld ersetzt werden kann. Die Formatdarstellung dieser Makros umfasst auch das Namensfeld.

Der **Makroname** bezeichnet den gewünschten Makro. Das Zeichen „$“ als erstes Zeichen kommt bei Makros für Benutzer nicht vor, da es den privilegierten Makroaufrufen vorbehalten ist.

Das **Operandenfeld** kann eine beliebige Anzahl von Operanden enthalten, die durch Kommas getrennt sind. Es kann auch leer sein. Die Art und Anzahl der Operanden, die angegeben werden können oder müssen, ist der Formatdarstellung des jeweiligen Makroaufrufs zu entnehmen.

Beim Aufruf des Makros im Assemblerprogramm müssen das Namensfeld, der Makroname und der erste Operand mit mindestens einem Leerzeichen getrennt sein. Mehrere Operanden müssen durch Kommas getrennt werden.

Fehler im Format eines Makroaufrufs, die bereits bei der Verarbeitung durch den Assembler erkannt werden, sind durch MNOTE-Meldungen (siehe die Kapitel zur Makrosprache im Handbuch „ASSEMBH“ [2]) im Übersetzungsprotokoll gekennzeichnet.

Operandenformen

Stellungsoperanden

Stellungsoperanden werden vom Assembler anhand ihrer Stellung im Operandenfeld identifiziert, weshalb in ihrer Schreibweise eine bestimmte Reihenfolge eingehalten werden muss. Beispiel: `MAKRO A,B,C`

Soll der zweite Operand (B) entfallen, so muss dennoch das zweite Komma (unmittelbar hinter dem ersten Komma) geschrieben werden, um die Stellung des dritten Operanden (C) beizubehalten: `MAKRO A,,C`

Entfallen dagegen die letzten Stellungsoperanden, so müssen die trennenden Kommas nicht geschrieben werden. Im vorliegenden Beispiel bedeutet dies, dass beim Wegfall der Operanden B und C der Makroaufruf wie folgt geschrieben werden kann: `MAKRO A`

Schlüsselwortoperanden

Ein Schlüsselwortoperand wird vom Assembler durch das ihm eindeutig zugeordnete Schlüsselwort identifiziert. Deshalb ist auch die Reihenfolge dieser Operanden beliebig. Einem Schlüsselwortoperanden wird durch ein Gleichheitszeichen ein Operandenwert zugewiesen, welcher aus einem definierten Wertevorrat stammt.
Die Schreibfolge ist: `<Schlüsselwort>=<gewünschter Wert>`
Beispiel: `MAKRO AREA=X,LENGTH=100`

Gemischte Operanden

Ein Operandenfeld kann beide Arten, Stellungs- und Schlüsselwortoperanden, enthalten. Dabei müssen jedoch alle Stellungsoperanden den Schlüsselwortoperanden vorangehen.
Beispiel: `MAKRO A,B,C,AREA=X,LENGTH=100`

Die Regeln für das Weglassen von Stellungs- und Schlüsselwortoperanden gelten entsprechend auch für gemischte Operandenfelder. Sollen z.B. die Operanden B, C und AREA entfallen, so lautet obiges Beispiel wie folgt: `MAKRO A,LENGTH=100`

Operanden-Unterlisten

Eine Unterliste besteht aus einem oder mehreren durch Kommas getrennten Stellungsoperanden, wobei die gesamte Liste in Klammern eingeschlossen ist. Sie wird als ein Operand betrachtet, d.h. sie besetzt entweder eine einzelne Stelle im Operandenfeld oder ist einem einzelnen Schlüsselwort zugeordnet. Die Verarbeitung des Inhalts einer Unterliste geschieht ähnlich wie bei Stellungsoperanden. Beispiel: `(A,B,C)` oder `(A)`
Wenn die Unterliste aus nur einem Operanden besteht (wie im zweiten Beispiel), so müssen die einschließenden Klammern dennoch geschrieben werden, da hierdurch die Unterliste als solche gekennzeichnet ist.

Metasyntax

Bei der Darstellung des Makroaufruf-Formats werden bestimmte Metazeichen verwendet und Vereinbarungen vorausgesetzt, die im Folgenden erläutert werden.

Formale Darstellung	Erläuterung	Beispiel
GROSSBUCH-STABEN	Großbuchstaben bezeichnen Schlüsselwörter oder Konstanten, die in dieser Form vom Benutzer angegeben werden müssen. Schlüsselwörter müssen mit * beginnen, falls alternativ sowohl Schlüsselwörter als auch Namen von Konstanten oder Variablen angegeben werden können	DIB FORCED=*YES
GROSSBUCH-STABEN in Halbfett	Großbuchstaben in Halbfett kennzeichnen erlaubte Abkürzungen der Schlüsselwörter.	GLOBAL=**Y**ES Anzugeben ist: GLOBAL=YES oder GLOBAL=Y
kleinbuch-staben	Kleinbuchstaben bezeichnen Datentypen der Werte, die vom Benutzer angegeben werden können, oder Variablen, die bei der Eingabe vom Benutzer durch aktuelle Werte ersetzt werden müssen.	DIB=<var: pointer> FILE dateiname
< >	Spitze Klammern kennzeichnen Variablen, deren Wertevorrat durch die Datentypen beschrieben wird.	<var: pointer>
{ }	Geschweifte Klammern schließen Alternativen ein, d.h. aus den eingeschlossenen Größen muss eine Angabe ausgewählt werden. Ausnahme: Standardwerte	TAPE={ YES / NO } Anzugeben ist: TAPE=YES oder TAPE=NO
/	Der Schrägstrich trennt alternativ zu verwendende Angaben und hat dieselben Funktionen wie geschweifte Klammern.	FORCED=*NO/*YES Anzugeben ist: FORCED=*NO oder FORCED=*YES
Unter-streichung	Die Unterstreichung hebt einen Standardwert hervor. Das ist der Wert, den das System einsetzt, wenn der Benutzer keine Angabe macht (= Voreinstellung). Hat ein Operand keinen Standardwert, so ist die Angabe eines Operanden Pflicht.	FORCED=*NO/*YES Anzugeben ist: FORCED=*NO oder FORCED=*YES (keine Angabe impliziert FORCED=*NO)

Tabelle 1: Makro-Syntax

Formale Darstellung	Erläuterung	Beispiel
`[ ]`	Eckige Klammern schließen Wahlangaben ein, d.h. Angaben, die weggelassen werden können. Steht bei Wahlangaben das Komma innerhalb der Klammer, so muss es nur bei Verwendung dieser Wahlangabe geschrieben werden. Steht es hingegen außerhalb der Klammer, so muss es stets geschrieben werden, auch wenn die Wahlangabe nicht gemacht wird. (Runde Klammern müssen eingegeben werden!)	`dateiname[,ERASE]` Anzugeben ist z.B.: `FILE,ERASE` oder `FILE` oder `XYZ,ERASE` etc.
`list-poss(n)`	Aus den list-poss folgenden Operandenwerten kann eine Liste gebildet werden. n gibt die maximale Anzahl der Listenelemente an. Enthält die Liste mehr als ein Element, muss sie in runden Klammern eingeschlossen werden.	`FLAG=list-poss(3): *SLI/*SKIP/*DC` Anzugeben ist: `FLAG=*SKIP` `FLAG=(*SLI,*DC)`
`...`	Punkte bedeuten eine Wiederholung. Sie zeigen an, dass die davor stehende Einheit mehrmals hintereinander wiederholt werden kann.	`(dateiname,...)` Anzugeben ist: `FILE` oder `(FILE,XYZ)` oder `(FILE1,FILE2,FILE3)` etc.
␣	Dieses Zeichen kennzeichnet ein Leerzeichen (X'40')	`STD␣` Anzugeben ist: `'STD '` (ohne Hochkommas)
=	Das Gleichheitszeichen verbindet den Operandennamen mit den dazugehörenden Operandenwerten.	`DATA=<var:pointer>`

Tabelle 1: Makro-Syntax

Datentypen der Operandenwerte

Datentyp	Zeichenvorrat	Anmerkungen
`c-string`	EBCDIC-Zeichen	ist in Hochkommata einzuschließen
`integer`	[+-] 0..2147483647	ist eine dezimale Zahl
`var:`	leitet eine variable Angabe ein. Nach dem Doppelpunkt folgt der Typ der Variablen (siehe Tabelle „Datentypen der Variablen“).	`<var: var-type>`
`reg:`	Register 0..15	`(<reg: var-type>)`

Zusätze zu Datentypen

Zusatz	Bedeutung
`n..m`	für Datentyp `integer` bedeutet `n..m` eine Intervallangabe; `n`: Mindestwert `m`: Maximalwert
	für Datentyp `c-string` bedeutet `n..m` eine Längenangabe in Bytes; `n`: Mindestlänge `m`: Maximallänge mit `n < m`
`n`	bei Datentyp `c-string` bedeutet `n` eine Längenangabe in Bytes; `n` muss exakt eingehalten werden.

Die Operandenwerte können direkt als Zeichenkette oder Integer-Zahl (siehe Datentypen `c-string` und `integer`) eingegeben werden oder indirekt über eine Variable (siehe Datentyp `var:`) bezeichnet werden. Die nachfolgende Tabelle enthält die Datentypen, die für Variablen möglich sind.

Datentypen der Variablen

Datentyp	Beschreibung	Definition im Programm
`char: n`	Die Variable ist eine Zeichenkette von n Zeichen. Fehlt die Längenangabe, wird `n = 1` angenommen.	`CLn`
`int: n`	Die Variable ist eine Integer-Zahl, die `n` Bytes belegt. Fehlt die Längenangabe, wird `n = 1` angenommen. Bedingung: $n \leq 4$	`FLn`
`enum-of E: n`	Die Variable ist die Aufzählung `E`, die n Bytes belegt. Fehlt die Längenangabe, wird `n = 1` angenommen ($n \leq 4$).	`XLn`
`pointer`	Die Variable ist eine Adresse oder ein Adresswert.	`A`

3.3 Registerverwendung

Register werden in den Textstellen des Handbuchs mit R0, R1, ... bezeichnet. Diese Bezeichnungsweise soll die Registerbezeichnung gegen sonstige Zahlenangaben im Text absetzen und entspricht andererseits einer häufig benutzten Schreibweise für Registerangaben in Programmen.

Von den Makroaufrufen des Organisationsprogramms werden standardmäßig die Mehrzweckregister R0, R1 und R15 verwendet, im **SAVE**- und **RETRN**-Makroaufruf zusätzlich Register R13 und R14.

Register R0 und R1 können beim Aufruf Operanden oder die Adresse von Operanden enthalten.

Register R14 wird manchmal als Rücksprungregister genutzt. Es enthält die Adresse desjenigen Befehls im Benutzerprogramm, welcher dem Makroaufruf folgt.

Bei einer Reihe von Makros enthält Register R1 nach der Ausführung die Adresse des Datenbereichs statt des ursprünglichen Inhalts.

Register R15 enthält die Fehleranzeige (Returncode). Makros, bei deren Ausführung Fehler auftreten können, speichern in Register R15 Informationen über die Ausführung des Makros ab, bevor das Benutzerprogramm fortgesetzt wird.
Bei neueren Makros wird das Register R15 nicht oder nur zusätzlich zur Aufnahme des Returncodes benutzt. In diesen Fällen ist im Standardheader (siehe Seite 43) ein Feld zur Aufnahme des Returncodes reserviert.

3.4 Rückinformation und Fehleranzeigen (Returncodes)

Rückinformation

Bei einigen Makros ist die Übergabe von Daten an das aufrufende Programm wesentlicher Teil ihrer Funktion. Diese Information wird gelegentlich in Register R1 übergeben oder auch in einen Bereich des Programms übertragen, dessen Adresse im Makroaufruf angegeben wurde. Die Übergabeart solcher Informationen ist in der Funktionsbeschreibung der jeweiligen Makros angegeben.

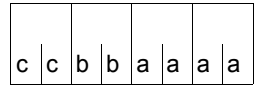
Fehleranzeigen (Returncodes)

Nach der Ausführung eines Makros wird das aufrufende Programm über den Erfolg oder Misserfolg des Makroaufrufs informiert. Dies geschieht durch die Übergabe eines Returncodes.

In Abhängigkeit von der jeweiligen Makroschnittstelle werden zwei Fälle unterschieden: die Übergabe im Register R15 und die Übergabe im Standardheader. Bei einigen Makros ist auch eine Kombination beider Fälle möglich.

1. **Übergabe des Returncodes im Standardheader**

(Teil vom Standard-header)

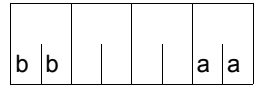


`aaaa` = Maincode
`bb` = Subcode1
`cc` = Subcode2

In den zwei rechten Byte wird der sog. **Maincode** übergeben. Er kennzeichnet das Ergebnis der Funktionsausführung. **Subcode1** dient der Fehlerklassifizierung.
Subcode2 dient der weiteren Unterteilung des Fehlers in Fehlerklassen oder enthält zusätzliche Diagnoseinformationen.
Alle Teile des Returncodes werden sedezimal angegeben. Zu Aufbau und Inhalt des Standardheaders siehe Seite 43.

2. **Übergabe des Returncodes im Register R15**

R15:



`aa` = primärer Returncode
`bb` = sekundärer Returncode

Im rechtsbündigen Byte des Registers R15 wird der **primäre Returncode** übergeben. Er besagt, ob die Funktion erfolgreich ausgeführt werden konnte oder nicht. Trat während der Ausführung kein Fehler auf, so enthält das rechtsbündige Byte den Code X'00'. Wenn ein Fehler während der Ausführung eines Makros aufgetreten ist, bringt der Ablaufteil als Fehlerkennzeichen einen anderen sedezimalen Code in dieses Byte. (Die drei linksbündigen Byte enthalten jeweils X'00', solange nicht ausdrücklich etwas anderes vereinbart wird).

In einigen Fällen wird die Information des primären Returncodes durch einen **sekundären Returncode** im linksbündigen Byte des Register R15 ergänzt. Mit dieser zusätzlichen Informationen ist es möglich, die Fehlerursache genauer zu bestimmen. Der sekundäre Returncode wird ebenfalls sedezimal dargestellt. Es liegt in der Verantwortung des Benutzers, diesen Code zu prüfen und geeignete Maßnahmen zu ergreifen.
Die Werte des Returncodes und deren Bedeutung sind in der jeweiligen Makrobeschreibung im Abschnitt „Rückinformation und Fehleranzeigen“ angegeben.

Sind die Codewerte des primären Returncodes im Abstand X'04' und ein garantierter Maximalwert festgelegt, kann der Returncode durch eine Sprungtabelle (bestehend aus 4 Byte langen Sprungbefehlen) verarbeitet werden.
Die Returncodes vieler Makros haben keine solche feste Strukturierung. In diesen Fällen ist eine Returncode-Verarbeitung mit expliziten Abfragen (Compare-Befehle) erforderlich.

Für beide Arten ist im Folgenden ein Beispiel aufgeführt.

Returncode des Makros **OPCOM**

X'aa'	**Erläuterung**
X'00'	Die ITC-Teilnahme ist eröffnet
X'04'	Fehler in der Operandenangabe. Die ITC-Teilnahme ist nicht eröffnet
X'08'	ITC-Name ist schon vergeben. Die ITC-Teilnahme ist nicht eröffnet
X'0C'	Kein Systemspeicher zur Eröffnung der ITC verfügbar oder die systeminterne Größe für Empfangswarteschlangen ist überschritten. Die ITC-Teilnahme ist nicht eröffnet.
X'10'	Die ITC-Teilnahme wurde bereits erklärt.

Beispiel für Returncode-Verarbeitung mit Sprungtabelle

```
  RCTAB    START
           PRINT NOGEN
           BALR  3,0
           USING *,3
  *        :
           OPCOM ITCNAME       * Declare ITC participation
1                 *,MACRO: OPCOM, VERSION: VER041
           B     RS00(15)
  CONTINUE EQU   *
  *        :
  END      TERM
  RS00     B     CONTINUE      * R15=00: No error handling
           B     OPERR         * R15=04
           B     NAMEERR       * R15=08
           B     MEMERR        * R15=0C
  EXIST    NOP   EXIST         * R15=10:
  *                              EXISTING ITC PARTICIPATION handling
           B     END
  OPERR    NOP   OPERR         *         OPERAND ERROR handling
           B     END
  NAMEERR  NOP   NAMEERR       *         DUPLICATE NAME handling
           B     END
  MEMERR   NOP   MEMERR        *         MEMORY ERROR handling
           B     END
           END
```

Beispiel für Returncode-Verarbeitung mit expliziter Abfrage

```
  RCEXPL   START
           PRINT NOGEN
           BALR  3,0
           USING *,3
  *         :
           OPCOM ITCNAME       * Declare ITC participation
1                 *,MACRO: OPCOM, VERSION: VER041
           LTR   15,15
           BZ    CONTINUE      * R15 = X'00'
           C     15,=F'4'
           BE    OPERR         * R15 = X'04'
           C     15,=F'8'
           BE    NAMEERR       * R15 = X'08'
           C     15,=F'12'
           BE    MEMERR        * R15 = X'0C'
           C     15,=F'16'
           BE    EXIST         * R15 = X'10'
  *        :                    ***
  *        :                    * Handling of other return codes
  *        :                    ***
  CONTINUE EQU   *
  *        :
  END      TERM
  *
  OPERR    NOP   OPERR         * OPERAND ERROR handling
           B     END
  NAMEERR  NOP   NAMEERR       * DUPLICATE NAME handling
           B     END
  MEMERR   NOP   MEMERR        * MEMORY ERROR handling
           B     END
  EXIST    NOP   EXIST         * EXISTING ITC PARTICIPATION handling
           B     END
  *        :
           END
```

3.5 Makroauflösung

Makroaufrufe werden durch Organisationsaufrufe (Supervisorcall, SVC) verwirklicht. In einigen Fällen, wie zum Beispiel beim Makro **CALL**, enthält die Makroauflösung keinen SVC. Der Organisationsaufruf bewirkt eine SVC-Unterbrechung, durch die die Unterbrechungsanalyse aktiviert wird. Diese ermittelt über eine Indextabelle der SVCs den Bearbeitungsprogrammbaustein, der zu dem Makro gehört. Die Prüfung der gültigen Operandenwerte und die Informationsübergabe an das aufrufende Programm über Fehler oder über die Ausführung der geforderten Arbeit wird von dem Bearbeitungsprogrammbaustein selbst gesteuert.

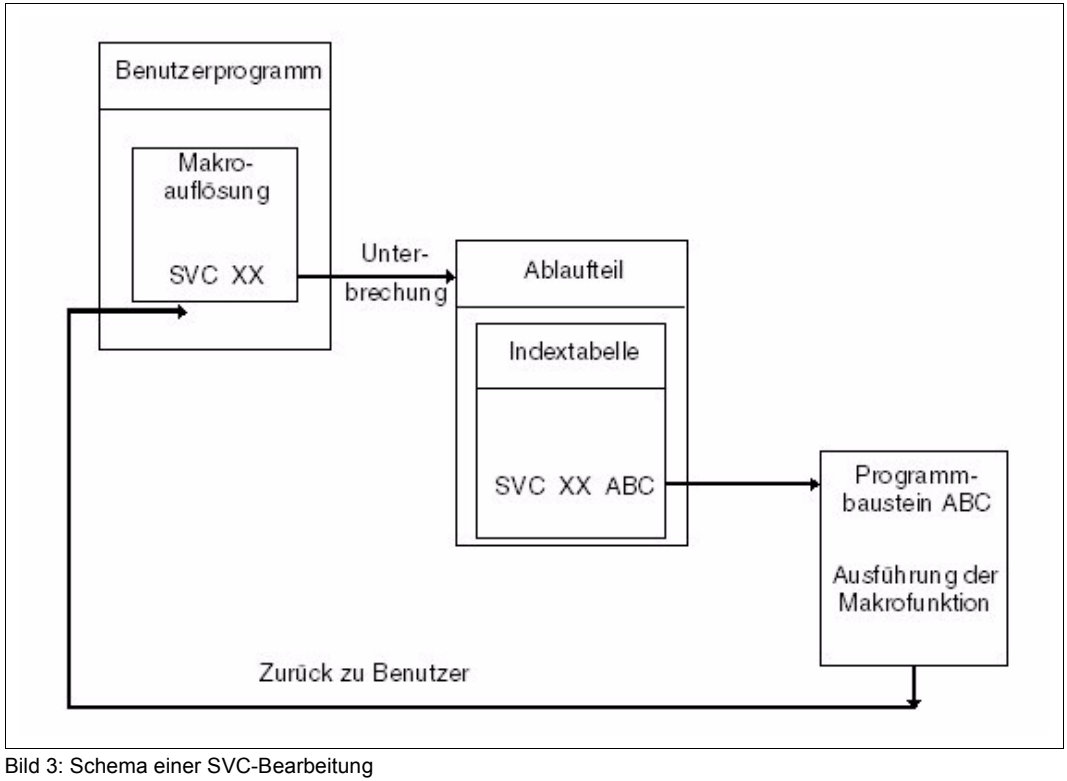


Bild 3: Schema einer SVC-Bearbeitung

3.6 Typen von Makroaufrufen

Der Eindeutigkeit wegen sei hier erwähnt, dass mit der Bezeichnung „Makroaufruftyp“ nicht die Begriffe „Aktionsmakro“ bzw. „Definitionsmakro“ gemeint sind. Diese Begriffe nehmen Bezug auf die Funktion eines Makros:

Ein **Aktionsmakro** ist ein Makro, von dem die Ausführung bestimmter Handlungen erwartet wird. Z.B. wird das Schreiben eines Abrechnungssatzes durch den **AREC**-Makro gesteuert (siehe Seite 209).

Ein **Definitionsmakro** ist ein Makro, von dem keine Handlungen, sondern Definitionen (Adressierungshilfen, DSECTS) erwartet werden. Der Makro **CUPAB** (Seite 387) ist ein Beispiel für die Generierung von symbolischen Namen zur Adressierung von Operandentabellen.

In **Typen** werden die Makroaufrufe abhängig von der Art ihrer **Operandenübergabe** eingeteilt. Es gibt den R-Typ (Übergabe in Registern), den S-Typ (Übergabe im Speicher) und den sog. O-Typ (Makros ohne Typzuordnung).
Makros vom S-Typ können sowohl Aktionsmakros (mit dem Operanden MF=E) als auch Definitionsmakros (mit dem Operanden MF=D) sein.

3.6.1 O-Typ-Makroaufrufe

Eine Reihe von Makroaufrufen kann weder dem R-Typ noch dem S-Typ zugeordnet werden. Sie sind Makroaufrufe ohne Typ-Zuordnung und werden in den Beschreibungen als O-Typ bezeichnet.

Vom O-Typ sind z.B. jene Makroaufrufe, die im Operandenfeld die Angabe **eines** Registers vorsehen (häufig R1), das die Anfangsadresse eines Parameterbereichs enthält.

Der Parameterbereich wird im Datenteil des Programms definiert (DC-Anweisungen) und enthält die Operandenwerte.

3.6.2 R-Typ-Makroaufrufe

MAKRO
operand1 / (r1) ,operand2 / (r2)

Ein Makroaufruf ist vom Typ R, wenn alle erforderlichen Operandenwerte in die zwei für diesen Zweck verwendeten Register R0 und R1 geladen werden können. Durch einen R-Typ-Makroaufruf wird also kein Parameterbereich generiert.

Es werden nicht bei allen Makroaufrufen beide Register verwendet (z.B. Makros **DELFEI** und **RSOFEI**). Die Parameter können direkt als Operanden des Makroaufrufs angegeben werden oder in den Registern R0 und R1 enthalten sein.

Bei Angabe von `operand1` und `operand2` werden in der Makroauflösung die angegebenen Werte in die Register R0 und R1 geladen.
Bei Angabe der Register müssen vor Aufruf des Makros die Werte der Operanden `operand1` und `operand2` in die Register R1 und R0 geladen werden. Diese Vorgehensweise wird als „Registerschreibweise“ bezeichnet.

Adressoperanden in R-Typ-Makroaufrufen können immer als explizite oder implizite Adressen geschrieben werden, also auch im Format Basis-Index-Distanz.
Hat der Makro nur einen wahlweisen Operanden, so muss einem evtl. gewünschten Kommentar ein „,“ (Komma) vorangestellt werden.

Beispiel: `CLCOM ,Kommentar`

3.6.3 S-Typ-Makroaufrufe

Beim S-Typ werden die im Makroaufruf angegebenen Operandenwerte in Form eines Datenbereichs an den Funktionsbaustein übergeben. Der Datenbereich ist Teil der Makroauflösung. Er erhält die für die Übergabe der Operandenwerte notwendigen Daten- und Speicherdefinitionen (DC- und DS-Anweisungen).

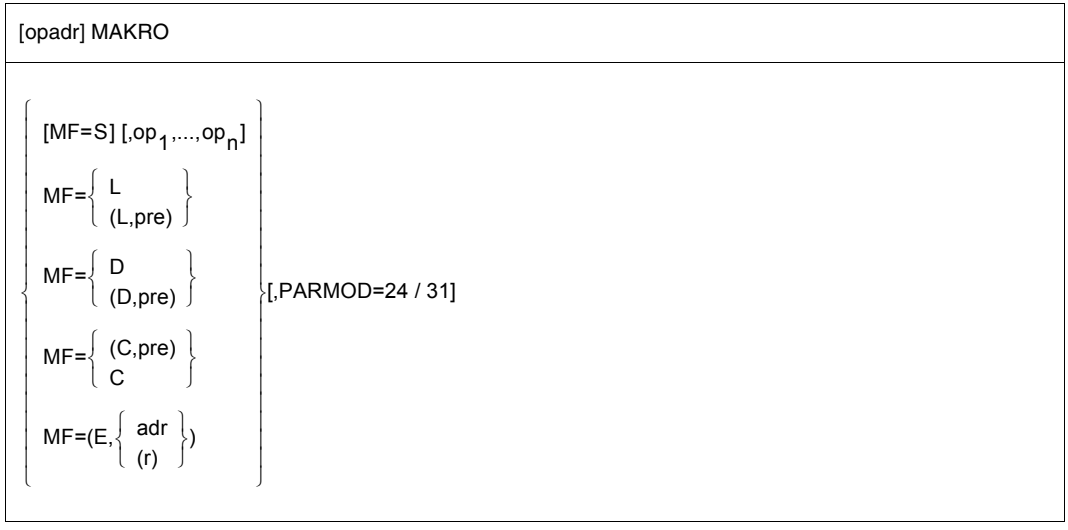
Der S-Typ unterstützt die Angabe des Operanden MF (siehe Seite 31). In Abhängigkeit von der Funktionalität der verschiedenen Makros sind mehrere Arten der Darstellung von MF möglich. Es werden drei **MF-Formate** unterschieden:

MF-Format	**für**	**besondere Eigenschaften**
MF-Format 1	Makros mit 24-Bit-Schnittstelle: Returncode in Register R15 Makros mit 31-Bit-Schnittstelle: Returncode in Register R15 oder (wenn vorhanden) im Standardheader	Es gibt die Voreinstellung MF=S (Standardform) für den MF-Operanden.
MF-Format 2	Makros mit 31-Bit-Schnittstelle: Returncode im Standardheader	
MF-Format 3	Makros mit 31-Bit-Schnittstelle: Returncode im Standardheader	Es gibt keine bestimmte Voreinstellung für den MF-Operanden.

Tabelle 2: MF-Formate für die S-Typ-Makroaufrufe

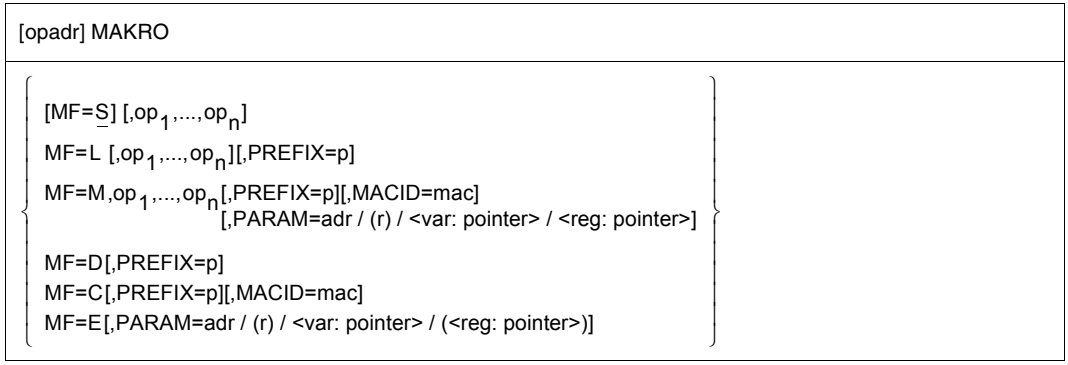
Im Folgenden sind alle drei MF-Formate ausführlich dargestellt. An die Formatdarstellungen schließt sich die Beschreibung der Operanden und Operandenwerte an.
Ab Seite 35 folgen Beispiele für Makroaufrufe in der S-, D-, E-, M- und L-Form.
Die verschiedenen Formen werden ab Seite 31 beschrieben.

MF-Format 1



Die explizite Angabe der Standardform mit MF=S ist für die meisten Makros im MF-Format 1 nicht erlaubt. Ausnahmen werden beim jeweiligen Makro kenntlich gemacht. (Siehe auch Operandenbeschreibung zu MF=S, Seite 31)
Bei der 24-Bit-Schnittstelle ist i.a. die Präfix-Schreibweise (z.B. (C,pre)) bei der C-/D-/L-Form nicht erlaubt. Ausnahmen werden beim jeweiligen Makro kenntlich gemacht.

MF-Format 2



MF-Format 3

```
[opadr] MAKRO

{ MF=L [,op1,...,opn][,PREFIX=p]
  MF=M,op1,...,opn[,PREFIX=p][,MACID=mac]
  MF=R,op1,...,opn[,PREFIX=p][,MACID=mac]
  MF=D[,PREFIX=p]
  MF=C[,PREFIX=p][,MACID=mac]
  MF=E[,PARAM=adr / (r) / <var: pointer> (<reg: pointer>)] }
```

opadr
Assembler-Name; bezeichnet bei MF=L die Adresse des Datenbereichs, sonst wahlfrei. opadr kann bei MF=(E,adr) oder MF=M,PARAM= zur Adressierung des Datenbereichs verwendet werden.

op_1,...,op_n
stellen anzugebende Funktionsoperanden dar.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit-Schnittstelle oder die 31-Bit-Schnittstelle generiert.

24
Die 24-Bit-Schnittstelle wird generiert. Datenbereiche und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenbereiche und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenbereiche beginnen mit dem Standardheader.

MF=
bestimmt die Art der Makrogenerierung. In Abhängigkeit von den Operandenwerten für MF unterscheidet man sieben Formen des Makroaufrufs:

S (Standardform: ist Voreinstellung bei MF-Format 1 und 2):
Die Angabe dieses Operandenwertes ist bei Format 3 nicht erlaubt.
Für die meisten Makros im MF-Format 1 darf MF=S *nicht explizit* angegeben werden, d.h. die Auswahl der Standardform erfolgt durch Weglassen des MF-Operanden. Im Aufrufformat des betreffenden Makros wird MF=S *nicht* dargestellt.
Bei Makros, die die explizite Angabe von MF=S unterstützen, wird MF=S auch im Aufrufformat dargestellt.

Es werden zuerst der Befehlsteil und anschließend der Datenbereich generiert, unter Beachtung der im Makroaufruf angegebenen Operandenwerte. Der Datenbereich enthält keine Feldnamen und keine erläuternden Equates. Der Standardheader ist initialisiert.

C (C-Form)
Es wird nur der Datenbereich generiert. Jedes Feld hat einen Feldnamen und erläuternde Equates, falls erforderlich. Der Datenbereich wird durch ein Längenequate beendet. Der Standardheader muss i.d.R. vom Anwender initialisiert werden.

(C,pre)
Diese Angabe ist nur bei Format 1 erlaubt. Die ersten Zeichen der Feldnamen und Equates können vom Anwender durch Angabe eines Präfix pre bestimmt werden.
pre = 1..4 Zeichen.

C [,PREFIX=p][,MACID=mac]
Diese Angabe ist nur bei MF-Format 2 und 3 erlaubt. Mit dem Operanden PREFIX kann der Anwender das erste Zeichen der Feldnamen und Equates bestimmen.
p = 1 Buchstabe.
Mit dem Operanden MACID kann der Anwender das zweite, dritte und vierte Zeichen der Feldnamen und Equates bestimmen. mac = 1..3 Zeichen.
Weitere Operanden werden in der C-Form nicht ausgewertet.

D (D-Form)
Es wird eine DSECT generiert. Jedes Feld hat einen Feldnamen und erläuternde Equates, falls erforderlich. Die DSECT wird durch ein Längenequate beendet. Es wird nicht auf den anfänglichen Adresspegel umgeschaltet.
Die **DSECT** beschreibt die Struktur eines Speicherbereiches, ohne selbst Speicherplatz zu belegen. Der bei DSECT angegebene symbolische Name wird in einen ESD-Satz (External Symbol Dictionary-Satz) eingetragen. Der Adresspegel wird auf Null gesetzt.

(D,pre)
Diese Angabe ist nur bei MF-Format 1 erlaubt. Die ersten Zeichen der Feldnamen und Equates können vom Anwender durch Angabe eines Präfix pre bestimmt werden.
pre = 1..4 Zeichen.

D [,PREFIX=p]
Diese Angabe ist nur bei MF-Format 2 und 3 erlaubt. Mit dem Präfix p kann der Anwender das erste Zeichen der Feldnamen und Equates bestimmen. p = 1 Buchstabe.
Weitere Operanden werden in der D-Form nicht ausgewertet.

L (L-Form)
Es wird nur der Datenbereich generiert, unter Beachtung der im Makroaufruf angegebenen Operandenwerte. Der Datenbereich enthält keine Feldnamen und keine erläuternden Equates. Der Standardheader ist initialisiert. Der Makroaufruf wird im Definitionsteil des Programms abgesetzt.

Bei Shared-Code-Programmierung darf dieser Aufruf nicht im invarianten Programmteil liegen, wenn er variable Daten enthält. Der Datenbereich wird im invarianten Programmteil mit konstanten Werten initialisiert, vor dem E-Form-Aufruf in einen ablauflokalen Datenbereich kopiert und dort ggf. modifiziert. Die Modifizierung wird z.B. mit der M-Form realisiert, falls sie für die betreffende Schnittstelle angeboten wird.

E (E-Form)
Es werden nur die zum Aufruf des Funktionsbausteins notwendigen Befehle generiert. Der Befehlsteil endet i.d.R. mit einem SVC. Im Makroaufruf muss die Adresse des Datenbereichs mit den Operandenwerten angegeben sein.

(E,adr) / (E,(r))
Diese Angabe ist nur bei MF-Format 1 erlaubt.
adr = Assembler-Name (Adresse des Datenbereichs).
r = Register, das die Adresse des Datenbereichs enthält. Vor dem Makroaufruf muss das Register mit diesem Adresswert geladen werden.

E [,PARAM=adr / (r)]
E [,PARAM=<var: pointer> / (<reg: pointer>)]
Der Operand PARAM bezeichnet die Adresse des Datenbereichs. Diese Angabe ist nur bei MF-Format 2 und 3 erlaubt.
adr / <var: pointer> = Assembler-Name (Adresse des Datenbereichs).
r / (<reg: pointer>) = Register, das die Adresse des Datenbereichs enthält. Vor dem Makroaufruf muss das Register mit diesem Adresswert geladen werden.
Wenn nicht anders angegeben, ist die Voreinstellung: PARAM = (1)
Weitere Operanden werden in der E-Form nicht ausgewertet.

M (M-Form)
Die Angabe dieses Operandenwertes ist nur bei MF-Format 2 und 3 erlaubt.
Es werden Befehle (z.B. MVCs) generiert, die während des Programmlaufs in einem mit MF=L bereits initialisierten Datenbereich bzw. bei Shared-Code-Programmierung in einer ablauflokalen Kopie des mit MF=L initialisierten Datenbereichs Felder mit den Operandenwerten überschreiben, die im Makroaufruf angegeben werden. Damit bietet die M-Form eine komfortable Möglichkeit, die Operandenwerte, mit denen ein Makro aufgerufen wird, **dynamisch** dem Programmlauf anzupassen.
Bei MF=M werden für Funktionsoperanden keine Standardwerte angenommen, d.h. alle Operanden müssen explizit angegeben werden.
Da die bei MF=M generierten Befehle die symbolischen Adressen und Equates der C-Form oder der D-Form benutzen, ist bei der Verwendung der M-Form sicherzustellen, dass diese Namen für die Adressierung des zu modifizierenden Datenbereichs zur Verfügung stehen. Insbesondere ist darauf zu achten, dass bei einem Makroaufruf mit MF=M ggf. die Operanden PREFIX und MACID mit den gleichen Werten angegeben werden wie im zugehörigen MF=C- bzw. MF=D-Aufruf.

M [,PREFIX=p][,MACID=mac]
Mit dem Operanden PREFIX kann der Anwender das erste Zeichen der Feldnamen und Equates bestimmen. p = 1 Buchstabe.
Mit dem Operanden MACID kann der Anwender das zweite, dritte und vierte Zeichen der Feldnamen und Equates bestimmen. mac = 1..3 Zeichen.

M [,PARAM=adr / (r)]
M [,PARAM=<var: pointer> / (<reg: pointer>)]
Der Operand PARAM bezeichnet die Adresse des Datenbereichs. Diese Angabe ist nur bei MF-Format 2 erlaubt.
adr / <var: pointer> = Assembler-Name (Adresse des Datenbereichs).
r / (<reg: pointer>) = Register, das die Adresse des Datenbereichs enthält. Vor dem Makroaufruf muss das Register mit diesem Adresswert geladen werden.
Voreinstellung: PARAM = (1)

R (R-Form)
Die Angabe dieses Operandenwertes ist nur bei MF-Format 3 erlaubt.
Es werden die durch Funktionsoperanden angegebenen Operandenwerte (von Ausgabeparametern) aus dem Datenbereich gelesen und in Variablen des Anwenderprogramms abgespeichert.
Da die dafür generierten Befehle die symbolischen Adressen und Equates der C-Form oder der D-Form benutzen, ist bei der Verwendung der R-Form sicherzustellen, dass diese Namen für die Adressierung des zu modifizierenden Datenbereichs zur Verfügung stehen. Insbesondere ist darauf zu achten, dass bei einem Makroaufruf mit MF=R ggf. die Operanden PREFIX und MACID mit den gleichen Werten angegeben werden wie im zugehörigen MF=C- bzw. MF=D-Aufruf.

R [,PREFIX=p][,MACID=mac]
Mit dem Operanden PREFIX kann der Anwender das erste Zeichen der Feldnamen und Equates bestimmen.
p = 1 Buchstabe.
Mit dem Operanden MACID kann der Anwender das zweite, dritte und vierte Zeichen der Feldnamen und Equates bestimmen.
mac = 1..3 Zeichen.

Beispiel 1: RDATA-Makroaufruf (MF-Format 1) mit S-Form (Standardform)

```
  RDATA1   START
           LDBASE R3,0
1                *,MACRO: LDBASE, VERSION: VER021                    021
1          ##BALR R3,0                                               020
2          BASR  R3,0                                                012
           USING *,R3
  RDATA1   AMODE ANY
  RDATA1   RMODE ANY
           GPARMOD 31
1                *,MACRO: GPARMOD, VERSION: VER121
           PRINT GEN
           RDATA INAREA,STOP ──────────────────────────────────────────  (1)
1          ##SPASS  S0004S,S0004D                                   A312
2          CNOP  0,4
2          BAS   1,S0004S           ADDRESS AND SKIP PARAMS
1 S0004D     DS 0F                                                  A340
1          FHDR  UNIT=36,FUNCT=18,VERS=2
2          DS    0A
2          DS    0XL8                GENERAL OPERAND LIST HEADER
2          DC    AL2(36)             FUNCTION UNIT NUMBER
2          DC    AL1(18)             FUNCTION NUMBER
2          DC    AL1(2)              FUNCTION INTERFACE VERSION NUMBER
2          DC    X'FFFFFFFF'          Returncode is virgin
1 *
1          DC    A(STOP)                  ERROR ADDRESS
1          DC    AL4(INAREA)              READ IN AREA ADDRESS
1          DS    AL1(0)                   PLACE FOR I.EDIT BYTE 1
1          DS    AL1(0)                   PLACE FOR I.EDIT BYTE 2
1          DC    AL1(0)                   SYSDTA ASSIGNMENT
1          DC    AL1(0)                   FLAG BYTE 1
1          DC    AL2(L'INAREA)           LENGTH OF READ
1          DC    AL1(0)                  FLAG TABLE BYTE
1          DC    AL1(0)                  ASSIGNMENT CHANGE INDICATOR
1          DC    H'0'                    KEY-POSITION
1          DC    H'0'                    KEY-LENGTH
1          DC    AL4(0)                  VTSUCB ADDRESS
1          DC    AL2(0)                  INPUT TIMER VALUE           009
1          DC    H'0'                    RES_FOR_TIAM                007
1 *
1          @DCEI DCEDIT=,MODE=,IGETFC=,ICFD=,                           C
1                ITRSUP=,ILINEND=,IGETBS=,                              C
1                IMANUAL=,ILCASE=,IHDR=,                                C
1                IGETIC=,RDA1=-20,RDA2=-19
2          ORG   *-20
2          DC    AL1(0)
2          ORG   *+20-1
```

```
2          ORG   *-19
2          DC    AL1(0)
2          ORG   *+19-1
2               *,@DCEI      999    921011   53531002
1 S0004S     DS 0Y                                                  A340
1          SVC   39                     SYSFILE SVC
1 *
           PRINT NOGEN
  STOP     TERM

  *
  INAREA   DS    CL104
  R3       EQU   3
           END
```

(1) Beim Makro **RDATA** ist die Standardform Voreinstellung. Die Auswahl der Standardform erfolgt durch Weglassen des MF-Operanden. Alle benötigten Operanden müssen angegeben werden. Es wird der Befehlsteil und der Datenbereich generiert.

Beispiel 2: RDATA-Makroaufruf (MF-Format 1) mit E- und L-Form

```
  RDATA2   START
           LDBASE R3,0
1                *,MACRO: LDBASE, VERSION: VER021                   021
1          ##BALR R3,0                                              020
2          BASR  R3,0                                               012
           USING *,R3
  RDATA2   AMODE ANY
  RDATA2   RMODE ANY
           GPARMOD 31
1                *,MACRO: GPARMOD, VERSION: VER121
           PRINT GEN
           RDATA MF=(E,PARLIST) ------------------------------------  (1)
1          LA    1,PARLIST              LOAD ADDR PARAM LIST INTO R1
1          SVC   39                     SYSFILE SVC
1 *
           PRINT NOGEN
  STOP     TERM
  *
  INAREA   DS    CL104
           PRINT GEN
  PARLIST  RDATA INAREA,STOP,MF=L ------------------------------------  (2)
1 S0007D     DS 0F                                                  A340
1 PARLIST  FHDR  UNIT=36,FUNCT=18,VERS=2
2          DS    0A
```

```
2 PARLIST  DS    0XL8               GENERAL OPERAND LIST HEADER
2          DC    AL2(36)            FUNCTION UNIT NUMBER
2          DC    AL1(18)            FUNCTION NUMBER
2          DC    AL1(2)             FUNCTION INTERFACE VERSION NUMBER
2          DC    X'FFFFFFFF'         Returncode is virgin
1 *
1          DC    A(STOP)                 ERROR ADDRESS
1          DC    AL4(INAREA)             READ IN AREA ADDRESS
1          DS    AL1(0)                  PLACE FOR I.EDIT BYTE 1
1          DS    AL1(0)                  PLACE FOR I.EDIT BYTE 2
1          DC    AL1(0)                  SYSDTA ASSIGNMENT
1          DC    AL1(0)                  FLAG BYTE 1
1          DC    AL2(L'INAREA)          LENGTH OF READ
1          DC    AL1(0)                 FLAG TABLE BYTE
1          DC    AL1(0)                 ASSIGNMENT CHANGE INDICATOR
1          DC    H'0'                   KEY-POSITION
1          DC    H'0'                   KEY-LENGTH
1          DC    AL4(0)                 VTSUCB ADDRESS
1          DC    AL2(0)                 INPUT TIMER VALUE           009
1          DC    H'0'                   RES_FOR_TIAM                007
1 *
1          @DCEI DCEDIT=,MODE=,IGETFC=,ICFD=,                           C
1                ITRSUP=,ILINEND=,IGETBS=,                              C
1                IMANUAL=,ILCASE=,IHDR=,                                C
1                IGETIC=,RDA1=-20,RDA2=-19
2          ORG   *-20
2          DC    AL1(0)
2          ORG   *+20-1
2          ORG   *-19
2          DC    AL1(0)
2          ORG   *+19-1

2                *,@DCEI      999    921011   53531002
1 *
           PRINT NOGEN
  R3       EQU   3
           END
```

(1) Durch die E-Form des Makroaufrufs wird der Befehlsteil des Makros **RDATA** generiert. Der Datenbereich mit den gewünschten Operanden beginnt ab der symbolischen Adresse PARLIST.

(2) In der L-Form des Makroaufrufs werden alle gewünschten Operanden angegeben. Der Datenbereich wird generiert.

Beispiel 3: GTIME-Makroaufruf (MF-Format 3) mit D-, E-, M- und L-Form

```
  GTIME    START
           PRINT NOGEN
           BALR  R3,0
           USING *,R3
  GTIME    AMODE ANY
  GTIME    RMODE ANY
           LA    R5,GLIST ------------------------------------------------  (1)
           USING DGLIST,R5
           LA    R13,SAVE
  *
  E1       GTIME MF=E,PARAM=GLIST,LINKADR=*NONE ------------------------  (2)
           MVC   TEXT,='Date: ' ------------------------------------------  (3)
           MVC   DATE,NTIGDTIC
           WROUT OUTPUT,STOP
2                *,@DCEO      999    921011   53531004
  CLEAR    MVC   DATE,=CL10' '
  *
  M        GTIME MF=M,PARAM=GLIST,DAY=YES ------------------------------  (4)
  *
  E2       GTIME MF=E,PARAM=GLIST,LINKADR=*NONE ------------------------  (5)
           MVC   TEXT,='Day:  ' ------------------------------------------  (6)
           MVC   DAY,NTIGDYID
           WROUT OUTPUT,STOP
2                *,@DCEO      999    921011   53531004
  STOP     TERM
  *
  OUTPUT   DC    Y(OUTPUTE-OUTPUT)
           DC    X'404001'
  TEXT     DS    CL6
  DATE     DS    CL10
           ORG   DATE
  DAY      DS    CL2
           ORG
  OUTPUTE  EQU   *
  SAVE     DS    18F
  GLIST    GTIME MF=L,DATE=YES ------------------------------------------  (7)
           PRINT GEN
  DGLIST   GTIME MF=D ------------------------------------------------  (8)
1 DGLIST   MFTST MF=D,PREFIX=N,MACID=TIG,ALIGN=F,                        C
1                DMACID=TIG,SUPPORT=(E,D,C,M,L),DNAME=TIG_MDL
2 DGLIST   DSECT ,
2                *,##### PREFIX=N, MACID=TIG #####
1 *   subcodes
1 NTIGERROR_IN_CALL      EQU   1          Error in Call
1 NTIGRNAP               EQU   32         no Action possible
1 NTIGWARNING_SITUATION  EQU   512        Warning Situation (SPL)
```

```
1 NTIGRWCS              EQU   2          Warning Situation (ASS)
1 *
1 *   GTIME-Parameter-Area
1 NTIGFHDR FHDR  MF=(C,NTIG),EQUATES=NO                  Standardheader
2 NTIGFHDR DS    0A
2 NTIGFHE  DS    0XL8           0   GENERAL PARAMETER AREA HEADER

2 *
2 NTIGIFID DS    0A             0   INTERFACE IDENTIFIER
2 NTIGFCTU DS    AL2            0   FUNCTION UNIT NUMBER
2 *                                 BIT 15    HEADER FLAG BIT,
2 *                                 MUST BE RESET UNTIL FURTHER NOTICE
2 *                                 BIT 14-12 UNUSED, MUST BE RESET
2 *                                 BIT 11-0  REAL FUNCTION UNIT NUMBER
2 NTIGFCT  DS    AL1            2   FUNCTION NUMBER
2 NTIGFCTV DS    AL1            3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 NTIGRET  DS    0A             4   GENERAL RETURN CODE
2 NTIGSRET DS    0AL2           4   SUB RETURN CODE
2 NTIGSR2  DS    AL1            4   SUB RETURN CODE 2
2 NTIGSR1  DS    AL1            5   SUB RETURN CODE 1
2 NTIGMRET DS    0AL2           6   MAIN RETURN CODE
2 NTIGMR2  DS    AL1            6   MAIN RETURN CODE 2
2 NTIGMR1  DS    AL1            7   MAIN RETURN CODE 1
2 NTIGFHL  EQU   8              8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 *   main return codes
1 NTIGRNIN              EQU   1          GTIME function not
1 *                                      initialized
1 NTIGRNSI              EQU   2          no season information
1 NTIGRPRV              EQU   16         no previous change date known
1 NTIGRPST              EQU   17         no later change date in past
1 *                                      known
1 NTIGRNCD              EQU   18         no change date known
1 NTIGRXIE              EQU   8          internal error concerning
1 *                                      xcs_mode
1 *
1 NTIGIB1               DS    AL1        indicator byte 1
1 NTIGIMU               EQU   X'80'      MODE = UTC
1 NTIGIFB               EQU   X'40'      FORMAT = BIN
1 NTIGICS               EQU   X'20'       not used
1 NTIGIFT               EQU   X'10'      FORMAT = TODR
1 NTIGIDW               EQU   X'08'      date wanted
1 NTIGIWW               EQU   X'04'      day wanted
1 NTIGITW               EQU   X'02'      TOD wanted
1 NTIGIZW               EQU   X'01'      zone wanted
1 NTIGIB2               DS    AL1        indicator byte 2
1 NTIGIRM               EQU   X'80'      resolution = microsec.
```

```
1 NTIGICN               EQU   X'40'       next change date demanded
1 NTIGICP               EQU   X'20'       previous ch.date demanded
1 NTIGIRF               EQU   X'10'       time reference
1 NTIGIMX               EQU   X'08'       global XCS-time on
1 NTIGICA               EQU   X'04'       announcement of chdate
1 *                                       demanded
1 NTIGRESERVED_2BITS    EQU   X'03'       not yet used
1 NTIGIRES              DS    XL2          indicator byte 3 & 4
1 NTIGDATE_UNION        DS    0XL16       date_union
1 NTIGDATE_SPL          DS    XL16        for SPL
1          ORG   NTIGDATE_UNION
1 *
1 NTIGDTI               DS    0XL16       date_iso4
1 NTIGDATE_UN           DS    0XL10       date union

1 *
1 NTIGDATE_1            DS    0XL10       date struct
1 NTIGDTIY              DS    CL4         year
1 NTIGDTI1              DS    CL1         hyphen1
1 NTIGDTIM              DS    CL2         month
1 NTIGDTI2              DS    CL1         hyphen2
1 NTIGDTID              DS    CL2         day
1 *
1          ORG   NTIGDATE_UN
1 NTIGDTIC              DS    CL10        date_char
1          ORG   NTIGDATE_UN+10
1 NTIGDTIJ              DS    CL3         julian date
1 NTIGDTIB              DS    CL1         blank
1 NTIGDYID              DS    CL2         weekday in ISO4
1 *
1          ORG   NTIGDATE_UNION
1 *
1 NTIGDTB               DS    0XL16       date_bin
1 *
1 NTIGDATE_2            DS    0XL6        date
1 NTIGDTBY              DS    H           year
1 NTIGDTBM              DS    H           month
1 NTIGDTBD              DS    H           day
1 *
1 NTIGDTBJ              DS    H           Julian date
1 NTIGFILL_6            DS    XL6         fill for weekday
1 NTIGDYBD              DS    H           weekday bin.: MO=0, DI=1, ...
1 *                                       SO=6
1 *
.
.
.
```

Ablaufprotokoll

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,gtime), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,gtime))
%  ASS6011 ASSEMBLY TIME: 538 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 135 MSEC
//end
%  ASS6012 END OF ASSEMBH
/start-executable-program library=macexmp.lib,element-or-symbol=gtime
%  BLS0523 ELEMENT 'GTIME', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'GTIME', VERSION ' ' OF '<date <time>' LOADED
Date: 2012-01-20
Day:  FR
```

(1) Das Register R5 wird mit der Adresse des mit MF=L erzeugten Datenbereichs geladen und zur Adressierung der Parameterleiste benutzt.

(2) Mit dem Aufruf **GTIME MF=E** wird der Befehlsteil generiert.
Der Datenbereich mit den Operandenwerten beginnt ab der symbolischen Adresse GLIST: Es soll das aktuelle Datum (DATE=YES) ermittelt werden (siehe auch Punkt (7)).

(3) Der Ausgabebereich wird mit dem Text „Date:“ und dem Inhalt des Feldes NTIGDTIC versorgt. Das Feld NTIGDTIC ist Bestandteil der DSECT und des Datenbereichs GLIST und enthält das aktuelle Datum. Mit dem Makro **WROUT** wird die gewünschte Information ausgegeben.

(4) Mit dem Aufruf **GTIME MF=M** wird der Datenbereich GLIST dynamisch verändert. Es soll nun zusätzlich der aktuelle Wochentag (DAY=YES) ausgegeben werden.

(5) Mit dem Aufruf **GTIME MF=E** wird der Befehlsteil generiert.
Der Datenbereich mit den Operandenwerten beginnt wiederum ab der symbolischen Adresse GLIST: Nun soll der aktuelle Wochentag (DAY=YES) ermittelt werden (siehe auch Punkt (7)).

(6) Der Ausgabebereich wird mit dem Text „Day:“ und dem Inhalt des Feldes NTIGDYID versorgt. Das Feld NTIGDYID ist Bestandteil der DSECT und des modifizierten Datenbereichs GLIST und enthält nach dem Aufruf mit MF=E den aktuellen Wochentag. Mit dem Makro **WROUT** wird die gewünschte Information ausgegeben.

(7) Mit dem Aufruf **GTIME MF=L** wird der Datenbereich für die Operandenwerte generiert. Dieser Datenbereich beginnt ab der symbolischen Adresse GLIST. Beim ersten Aufruf von **GTIME MF=E** wird die Information DATE abgefragt (siehe Punkt (2)). Beim zweiten **GTIME MF=E** wurde der Datenbereich mit einem vorhergehenden **GTIME MF=M** dahingegend geändert, dass die gewünschte Information nun auch DAY ist (siehe Punkte (4) und (5)).

(8) Die DSECT für GTIME wird generiert.
Der Datenbereich kann unter Benutzung der Feldnamen der DSECT versorgt werden. Die Feldnamen beginnen (standardmäßig) mit den Zeichen NTIG.
Die Adressrechnung beginnt bei DGLIST wieder mit X'000000'. Die nachfolgenden Distanzen werden zusammen mit dem Basisregister R5 adressiert (z. B. in MVCs).

3.7 Standardheader

Alle neuen Makros und i.d.R. die Makros, die um die 31-Bit-Schnittstelle erweitert wurden, benutzen zur Identifikation ihrer Schnittstelle den Standardheader.

Der Standardheader ist ein 8 Byte langes Feld am Anfang des Datenbereichs mit der (normierten) Bezeichnung der Schnittstelle und 4 Byte zur Aufnahme eines Returncodes. Der Standardheader wird vom jeweiligen Makro erzeugt und initialisiert, d.h. mit den gültigen Werten für UNIT, FUNCTION und VERSION versorgt. Bei Makroaufrufen in der E-Form unter Bezug auf den Datenbereich muss unter Umständen der Aufrufer den Standardheader initialisieren. Näheres ist beim jeweiligen Makro angegeben.

Aufbau des Standardheaders:

Byte	Feldinhalt und Bedeutung
0 - 1	Bezeichnung der Funktionseinheit (UNIT) mit der verlangten Funktion
2	Bezeichnung der Funktion (FUNCTION) innerhalb der Funktionseinheit
3	Bezeichnung des Änderungsstandes (VERSION) der Funktion
4	Unterwert 2 des Returncodes (SUBCODE2)
5	Unterwert 1 des Returncodes (SUBCODE1)
6 - 7	Hauptwert des Returncodes (MAINCODE)

Tabelle 3: Standardheader

Folgende Werte des Returncodes sind durch Konvention makroübergreifend festgelegt:

SUB-CODE2	SUB-CODE1	MAIN-CODE	Bedeutung
X'00'	X'00'	X'0000'	Erfolgreiche Funktionsausführung. Es gibt keine zusätzlichen Informationen zum MAINCODE.
X'01'	X'00'	X'0000'	Erfolgreiche Funktionsausführung. Es waren keine Aktionen erforderlich.
X'00'	X'01'	X'FFFF'	Die angeforderte Funktion wird nicht unterstützt (falsche Angabe für UNIT oder FUNCTION im Standardheader). Nicht behebbarer Fehler.
X'00'	X'02'	X'FFFF'	Die angeforderte Funktion ist nicht verfügbar. Nicht behebbarer Fehler.
X'00'	X'03'	X'FFFF'	Die angegebene Version der Schnittstelle wird nicht unterstützt (falsche Versionsangabe im Standardheader). Nicht behebbarer Fehler.
X'00'	X'04'	X'FFFF'	Der Datenbereich ist nicht auf Wortgrenze ausgerichtet.
X'00'	X'41'	X'FFFF'	Das Subsystem ist nicht vorhanden; es muss explizit erzeugt werden.

Tabelle 4: Standard-Returncodes (Teil 1 von 2)

SUB-CODE2	SUB-CODE1	MAIN-CODE	Bedeutung
X'00'	X'42'	X'FFFF'	Die aufrufende Task ist mit dieser Schnittstelle nicht konnektiert; sie muss explizit konnektiert werden.
X'00'	X'81'	X'FFFF'	Das Subsystem ist zurzeit nicht verfügbar.
X'00'	X'82'	X'FFFF'	Das Subsystem ist im DELETE- oder HOLD-Zustand.

Tabelle 4: Standard-Returncodes (Teil 2 von 2)

MAINCODE kennzeichnet das Ergebnis der Funktionsausführung. SUBCODE1 dient der Klassifizierung des Hauptwertes. SUBCODE2 dient der weiteren Unterteilung des Fehlers in Fehlerklassen oder enthält zusätzliche Diagnoseinformationen.
Bei allen neuen Makros sollte der Returncode ausschließlich im Standardheader übergeben werden. Der Returncode kann bei manchen Makroschnittstellen aber auch im Register R15 oder im Standardheader und im Register R15 übergeben werden. Um zu prüfen, ob im Standardheader ein Returncode übergeben wurde, sollte das Returncode-Feld mit X'FFFFFFFF' vorbesetzt werden.

Beispiel für die Erzeugung des Standardheaders

```
           WROUT    START
           PRINT NOGEN
           BALR  3,0
           USING *,3
  WROUT    AMODE ANY
  WROUT    RMODE ANY
           GPARMOD 31
1                *,MACRO: GPARMOD, VERSION: VER121
           PRINT GEN
           WROUT OUTPUT,STOP
1          ##SPASS S0002S,S0002D                                    A312
2          CNOP  0,4
2          BAS   1,S0002S           ADDRESS AND SKIP PARAMS
1 S0002D     DS 0F                                                  A340
1          FHDR  UNIT=36,FUNCT=17,VERS=2
2          DS    0A
2          DS    0XL8                GENERAL OPERAND LIST HEADER
2          DC    AL2(36)             FUNCTION UNIT NUMBER
2          DC    AL1(17)             FUNCTION NUMBER
2          DC    AL1(2)              FUNCTION INTERFACE VERSION NUMBER
2          DC    X'FFFFFFFF'          Returncode is virgin
1 *
1          DC    AL4(STOP)                ERROR ADDRESS
1          DC    AL4(OUTPUT)              MESSAGE AREA ADDRESS
  *        :
```

3.8 Makroaufrufe an den Kommandosprachübersetzer

Der Aufruf des Kommandosprachübersetzers MCLP (Macro Command Language Processor) ermöglicht im Programmodus die Eingabe eines (System-) Kommandos. Der Aufruf des MCLP (SVC 58_{16}) und die Übergabe des Kommandonamens und der Kommandooperanden erfolgt durch den Makro **CMD**. Der MCLP führt eine Syntaxprüfung durch und verzweigt zur eigentlichen Verarbeitungsroutine des Kommandos. Nach Ausführung des Kommandos wird das Programm fortgesetzt.

Einige der aufrufbaren Kommandos beenden das aufrufende Programm (siehe Tabelle 13 auf Seite 324). Das aufrufende Programm wird auch beendet, wenn mittels SDF-A definierte und durch Kommandoprozeduren implementierte (Anwender-)eigene Kommandos aufgerufen werden.

Neben SDF-Kommandos können auch ISP-Kommandos aufgerufen werden. Für fehlerhaft eingegebene SDF-Kommandonamen kann im Dialogbetrieb ein Korrekturdialog geführt werden

Einige BS2000 Kommandos können nicht über den Makro **CMD** aufgerufen werden, siehe Tabelle 12 auf Seite 322.

Für einige Kommandos gibt es eigene Makros. Die folgende Tabelle stellt diesen Makros die entsprechenden Kommandos gegenüber (Makros im Anhang werden nicht berücksichtigt):

Makro	**Kommando**	**Funktion**
CDUMP2	CREATE-DUMP	Dump erstellen
CHKPRV	SHOW-PRIVILEGE	eigene Jobprivilegien abfragen
ENTER	ENTER-JOB	Auftrag (Job) einleiten
LGOFF	EXIT-JOB	Auftrag (Job) beenden
MSGSHOW	SHOW-MSG-FILE-ASSIGNMENT	Informationen über System- oder taskspezifische Meldungsdateien ausgeben
MSGSINIT	MODIFY-MSG-FILE-ASSIGNMENT	Meldungsdatei sperren oder dem Meldungssystem hinzufügen
MSGSMOD	MODIFY-MSG-FILE-ASSIGNMENT	Meldungsdatei sperren oder dem Meldungssystem hinzufügen

Tabelle 5: Kommandos mit eigenen Makros (Teil 1 von 2)

Makro	Kommando	Funktion
NKDINF	SHOW-DEVICE-DEPOT SHOW-DEVICE-CONFIGURATION SHOW-DEVICE-STATUS SHOW-DISK-DEFAULTS SHOW-DISK-STATUS SHOW-MOUNT-PARAMETERS SHOW-RESOURCE-ALLOCATION SHOW-RESOURCE-REQUEST SHOW-TAPE-STATUS	Informationen über den Belegungs- und Verfügbarkeitsstand der Konfiguration und der montierten Datenträger ausgeben
NSIINF	SHOW-SYSTEM-INFORMATION	Systeminformationen ausgeben
NSIOPT	SHOW-SYSTEM-PARAMETERS	Systemparameter ausgeben
RDUID	SHOW-JOB-STATUS	Benutzerkennung abfragen
SINF	SHOW-SYSTEM-INFORMATION SHOW-SYSTEM-PARAMETERS	Systeminformationen und Systemparameter ausgeben
SRMUINF	SHOW-USER-ATTRIBUTES	Informationen aus dem Benutzerkatalog ausgeben
STAMCE	SHOW-MASTER-CATALOG-ENTRY SHOW-PUBSET-PARAMETERS	MRSCAT-Einträge ausgeben Pubset-Parameter ausgeben
SWITCH	MODIFY-JOB-SWITCHES MODIFY-USER-SWITCHES SHOW-JOB-SWITCHES SHOW-USER-SWITCHES	Auftragsschalter setzen Benutzerschalter setzen Auftragsschalter abfragen Benutzerschalter abfragen
SYSFL	ASSIGN-SYSDTA ASSIGN-SYSLST ASSIGN-SYSOUT REMOVE-TASKLIB SET-TASKLIB	Systemdateien zuordnen
SYSTA	SHOW-SYSTEM-FILE-ASSIGNMENTS	Zuordnungen der Systemdateien ausgeben
TCHNG	MODIFY-TERMINAL-OPTIONS	Eigenschaften der Datenstation ändern

Tabelle 5: Kommandos mit eigenen Makros (Teil 2 von 2)

4 Anwendungsgebiete und Kurzbeschreibungen

4.1 Binden und Laden

Makro	Kurzbeschreibung
ASHARE	bindet und lädt Shared Code, der aus einer Menge von Modulen bestehen kann, in einen Memory Pool
BIND	ruft den Dynamischen Bindelader DBL auf, um einen oder mehrere Module zu binden und zu laden und setzt den Prozess wahlweise mit dem aufrufenden Programm oder mit dem geladenen Modul fort
CALL	lädt ein Segment, sofern es noch nicht im Speicher steht, und setzt den Prozess mit dem geladenen Segment fort. Weitere Segmente innerhalb desselben Astes der Überlagerungsstruktur werden automatisch nachgeladen
DSHARE	entlädt Shared Code aus einem Memory Pool
ETABIT	erzeugt oder ändert einen Eintrag für eine Symboltabelle, die beim Makroaufruf ETABLE an den DBL übergeben wird
ETABLE	übergibt dem DBL eine Symboltabelle, die in die Symboltabelle des angegebenen Kontextes eingemischt wird
GETPRGV	gibt die Programmversion aus, die vom Benutzer vorher mit dem Makroaufruf SELPRGV oder mit dem Kommando SELECT-PROGRAM-VERSION ausgewählt wurde
ILEMGT	verwaltet eine Liste von ILEs (Indirect Linkage Entries)
ILEMIT	erzeugt einen Listeneintrag für eine ILE-Liste, die beim Makroaufruf ILEMGT verwendet wird
LDSLICE	lädt die angegebene Slice, die vom Benutzer in einem Bindelademodul (LLM) definiert wurde, in den Hauptspeicher
LPOV	lädt das angegebene Segment, auch wenn es schon im Speicher steht, und setzt den Prozess wahlweise mit dem aufrufenden Programm oder einem beliebigen Modul fort
PINF	informiert über Programme, die mit den Kommandos LOAD-PROGRAM oder START-PROGRAM geladen wurden
SELPRGV	legt fest, welche Programmversion der DBL verwendet, wenn mehrere Versionen eines Programms geladen sind

Makro	Kurzbeschreibung
SEGLD	lädt ein Segment, auch wenn es schon im Speicher steht, und setzt den Prozess wahlweise mit dem aufrufenden Programm oder einem beliebigen Modul fort. Weitere Segmente innerhalb desselben Astes der Überlagungsstruktur werden automatisch nachgeladen
UNBIND	gibt während des Programmlaufs Speicherplatz frei, den ein nicht mehr benötigtes Objekt belegt und entbindet innerhalb des Objekts wahlweise CSECTs und ENTRYs (d.h. Externverweise zu diesen Symbolen behandelt der DBL dann als unbefriedigte Externverweise). Ein solches Objekt kann eine Ladeeinheit, ein Bindelademodul (LLM) oder ein Bindemodul (OM) sein
VSVI1	informiert den Anwender über Einträge in den Tabellen des DBL, insbesondere über die Namen der Kontexte sowie die Namen, Ladeadressen, Längen und Attribute von CSECTs, ENTRYs und COMMON-Bereichen

Der Makro **TABLE** wird nur noch aus Kompatibilitätsgründen unterstützt. Er wird im Anhang ab Seite 1177 beschrieben.

Ausführliche Informationen zum Bindelader-Starter DBL und den BINDER-Funktionen enthalten die Handbücher „BLSSERV“ [4] und „BINDER“ [5].

4.2 Virtuelle Adressräume

In diesem Abschnitt wird der Aufbau des virtuellen Adressraums in BS2000 und die Umsetzung von virtuellen in reale Adressen (Adressumsetzung) beschrieben. Des Weiteren sind die Kurzbeschreibungen der Makroaufrufe und detaillierte Beschreibungen der Anwendungsgebiete „Arbeiten mit virtuellem Adressraum“, „Memory Pool Technik“ und „Erweiterung durch Datenräume“ enthalten.

4.2.1 Aufbau des virtuellen Adressraums

Der virtuelle Adressraum ist eine Folge von lückenlos aufsteigenden virtuellen Adressen, beginnend bei 0.

Die Größe des virtuellen Adressraums kann für /390-Server eingestellt werden und beträgt maximal 2 GByte, siehe Bild 4 auf Seite 51. Der Benutzeradressraum für x86-Server ist in der Größe von zwei Gbyte generiert und kann nicht verändert werden.

Da Befehle nur ausführbar sind, wenn sie und ihre Operanden im Hauptspeicher stehen, ist es notwendig, die virtuellen Adressen des Adressraums in reale Hauptspeicheradressen umzuwandeln (siehe Seite 53). Die Umsetzung geschieht zum Zeitpunkt der Programmausführung.

BS2000 verwaltet den virtuellen Adressraum seitenweise. Eine Seite umfasst 4 KByte (4096 Byte), d.h. der Anwender kann mit seinen Programmen Speicherplatz in Einheiten zu je 4 KByte anfordern und belegen.
Jeder Task wird bei ihrer Erzeugung ein eigener virtueller Adressraum zugeordnet.
Der virtuelle Adressraum wird in sechs Speicherklassen unterteilt, die unterschiedliche Attribute (Merkmale) haben. Jede Seite kann eindeutig einer dieser Klassen zugeordnet werden.
Die Klassen 5 und 6 stellen zusammen den tasklokalen Anteil des virtuellen Adressraums dar, wobei jedoch nur Klasse-6-Speicher vom Anwender für seine Programme und Daten adressierbar ist (Benutzeradressraum). Im Klasse-5-Speicher legt das System Tabellen an, die es zur Kommunikation mit der Anwendertask benötigt. Auf diesen Speicherbereich kann in der Regel vom nichtprivilegierten Anwender nicht zugegriffen werden. Eine Ausnahme bilden z.B. nichtprivilegierte DSSM-Subsysteme, die ebenfalls in den Klasse-5-Speicher geladen werden können. Die Klassen 1 bis 4 sind privilegiert, wobei die Klasse 4 neben Systemtabellen auch Subsysteme und ablaufinvariante Programme des Benutzers aufnehmen kann. Die Klassen 1, 2 und 3 sind nur für das System verfügbar.

Einteilung in Speicherklassen

Speicher-klasse	Attribute	Inhalt und Verfügbarkeit
6	– nicht resident (Voreinstellung) – dynamisch zuweisbar – tasklokal	– Programme und Arbeitsbereiche des (nichtprivilegierten) Anwenders – vom nichtprivilegierten Anwender lesend und schreibend zugreifbar
5	– nicht resident – dynamisch zuweisbar – tasklokal	– Systemtabellen für die Verbindung der Task zum System und zu nicht privilegierten Subsystemen – vom nichtprivilegierten Anwender i.d.R. nicht zugreifbar
4	– nicht resident – dynamisch zuweisbar – systemglobal	– Seitenwechselbare Tabellen und nachladbare Teile des Systems sowie ablaufinvariante Programme (Shared Code) – vom nichtprivilegierten Anwender ist Lesezugriff auf Shared Code möglich
3	– resident – dynamisch zuweisbar – systemglobal	– Residente Tabellen und nachladbare Teile des Systems sowie Arbeitsbereiche des Systems – vom nichtprivilegierten Anwender nicht zugreifbar
2	– nicht resident – statisch – systemglobal	– Seitenwechselbare Tabellen und Module des Systems – vom nichtprivilegierten Anwender nicht zugreifbar
1	– resident – statisch – systemglobal	– Residente Tabellen und Module des Systems – vom nichtprivilegierten Anwender nicht zugreifbar

Tabelle 6: Definition der Speicherklassen

Virtueller Adressraum auf /390-Servern

Bild 4 zeigt die Aufteilung des virtuellen Adressraums bei einer Größe von 2 GByte (Standard-Einstellung).

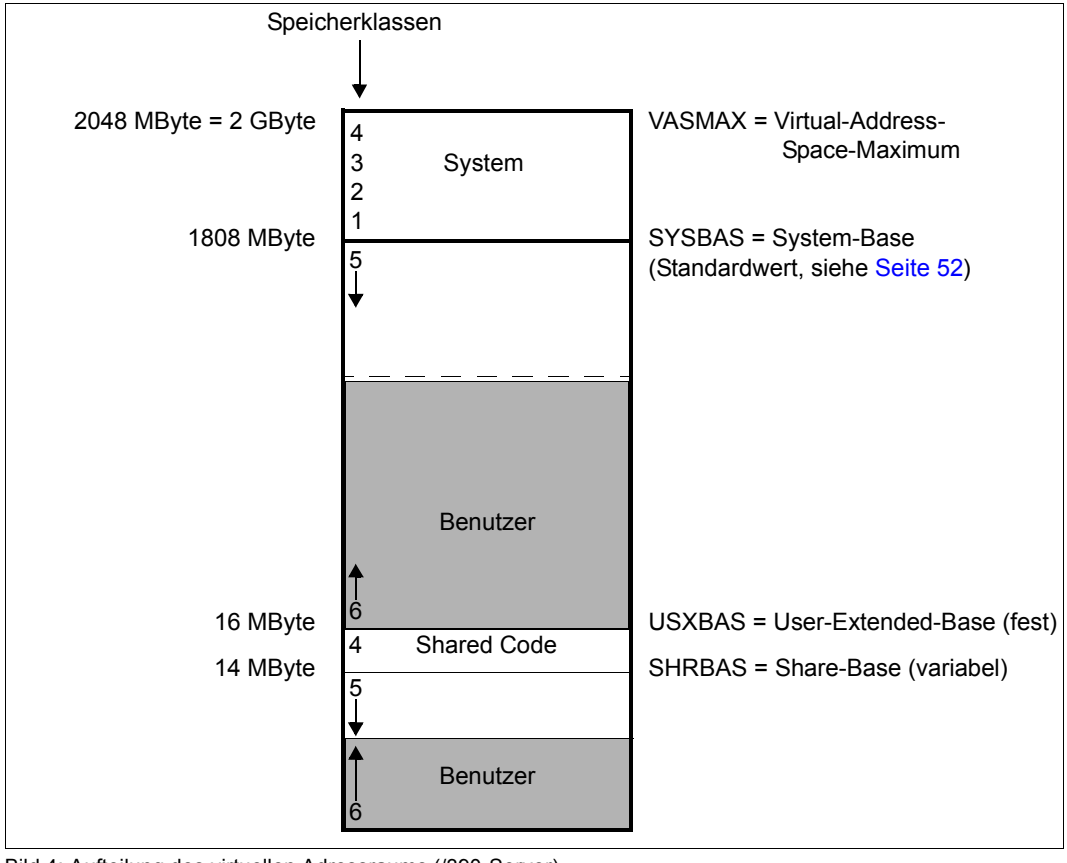


Bild 4: Aufteilung des virtuellen Adressraums (/390-Server)

Größe des Benutzeradressraums

Benutzeradressraum (Klasse-6-Speicher) und tasklokaler Systemadressraum (Klasse-5-Speicher) teilen sich ober- und unterhalb der 16-MByte-Grenze je einen Adressraumbereich. Zwischen Klasse-5- und Klasse-6-Speicher gibt es keine festen Grenzen. Die jeweilige Grenze variiert in Abhängigkeit von den Speicheranforderungen für jede der beiden Speicherklassen und ist daher in beide Richtungen verschiebbar. Die maximale Größe des Benutzeradressraums ist also zeitlich veränderlich und hängt vom aktuellen Klasse-5-Speicherbedarf ab, der sich aus den vorangegangenen Aktivitäten der Anwendertask ergibt. Allerdings ist stets je 1/8 des gemeinsamen Bereichs für Klasse-5-Speicher reserviert; Klasse-6-Speicher kann nie den gesamten Bereich einnehmen.

Abgesehen davon gibt es drei Faktoren, die die Größe des Benutzeradressraums bestimmen:

1. die Einstellung der Größe des Benutzeradressraums (/390-Server)
   Der Benutzeradressraum ist in der Größe von 1808 Mbyte generiert.
   Mit der Prozedur SYSPRC.BS2000-EXEC.version kann vom ausgelieferten BS2000-Standard-EXEC ein BS2000-EXEC mit anderem Benutzer- und Gesamtadressraum abgeleitet werden, siehe Handbuch „Systeminstallation“ [11].
   Bei dem Standardwert von 1808 MByte (bzw. 896 MByte oder 448 MByte) für die Größe des tasklokalen Adressraums (das entspricht einem Gesamtadressraum von 2048 MByte (bzw. 1024 MByte oder 512 MByte)), wird 1/8 für den Klasse-5-Speicher reserviert. Damit verbleibt für den Klasse-6-Speicher oberhalb 16 MByte eine Größe von 1568 MByte (bzw. 770 MByte oder 378 MByte).

2. die Systemeinleitung
   Die Größe des Shared-Code-Bereichs unterhalb 16 MByte kann über den Parameterservice bei der Systemeinleitung eingestellt werden. Die Standardeinstellung beträgt 2 MByte. Ferner ermittelt DSSM während der Systemeinleitung über alle Subsysteme hinweg den Gesamtbedarf an Klasse-5-Speicher mit SCOPE=*GLOBAL. Ein entsprechend großer Bereich wird (zusätzlich zu dem allgemeinen Anteil von 1/8) für Klasse-5-Speicher reserviert, kann aber tasklokal mit /RELEASE-SUBSYSTEM-SPACE wieder freigegeben werden. Insgesamt verbleibt für den Klasse-6-Speicher bei SHRSIZE=2 MByte eine Größe von etwa 12 MByte.

3. der Benutzerkatalog
   Im Benutzereintrag ist das Kontingent (in MByte) festgelegt, das dem Benutzer für Allokierungen im virtuellen Adressraum zur Verfügung steht. Es umfasst die Speicheranforderungen im Klasse-6-Speicher des Benutzeradressraums und in den Data Spaces, die vom Benutzer angelegt wurden. Dieser Wert bezieht sich jedoch nur auf den Umfang des angeforderten Klasse-6-Speichers, nicht auf seine Lage (ober- oder unterhalb 16 MByte).
   In der Ausgabe des Kommandos SHOW-USER-ATTRIBUTES enthält das Feld ADDRESS-SPACE-LIMIT die maximal erlaubte Größe des Klasse-6-Speichers für die Benutzerkennung (siehe Handbuch „Kommandos“ [19]).

4.2.2 Adressumsetzung

Jeder virtuelle Adressraum bildet eine Domäne. Soweit diese nicht zum Systemadressraum gehört oder explizit als mehrfach benutzbar definiert wurde, sind die Seiten eines Adressraums nur innerhalb dieses Adressraums zugänglich, d.h. vor Übergriffen von anderen Adressräumen her geschützt. Dieser Schutz wird durch die Adressumsetzung erreicht, die sich bei der Transformation einer virtuellen in eine reale Adresse stets auf den eingeschalteten Adressraum bezieht. Bei der Taskinitiierung wird ein spezielles Hardwareregister (CR1) eingeschaltet, das die Basisadresse und die Länge der für diesen Adressraum gültigen Umsetzungstabelle enthält. (Bei der Deinitiierung einer Task wird ihr Adressraum wieder ausgeschaltet, Task- und Adressraumwechsel sind also aneinander gekoppelt.)

Eine virtuelle Adresse setzt sich aus Segmentnummer, Seitennummer (Index) und Bytenummer zusammen, die BS2000 in einer zweistufigen Adressumsetzung in eine reale Adresse umrechnet. Zuerst wird die über die entsprechende Segmenttabelle (Umsetzungstabelle) eine Seitentabelle ausgewählt. In dieser steht dann die (reale) Hauptspeicher-Seitennummer, die zu der virtuellen Seitennummer gehört. Die Adressumsetzung erfolgt nur für den Seitenanteil (Segment- und Seitennummer) einer Adresse, die Distanz innerhalb einer Seite bleibt virtuell und real gleich.

Die Größe der virtuellen und der realen Seite beträgt für alle BS2000-Server 4 KByte.

Dementsprechend wird eine virtuelle 31-Bit-Adresse eingeteilt in 11 Bit Segmentnummer, 8 Bit Seitennummer und 12 Bit Distanz (siehe Bild 5 auf Seite 54).

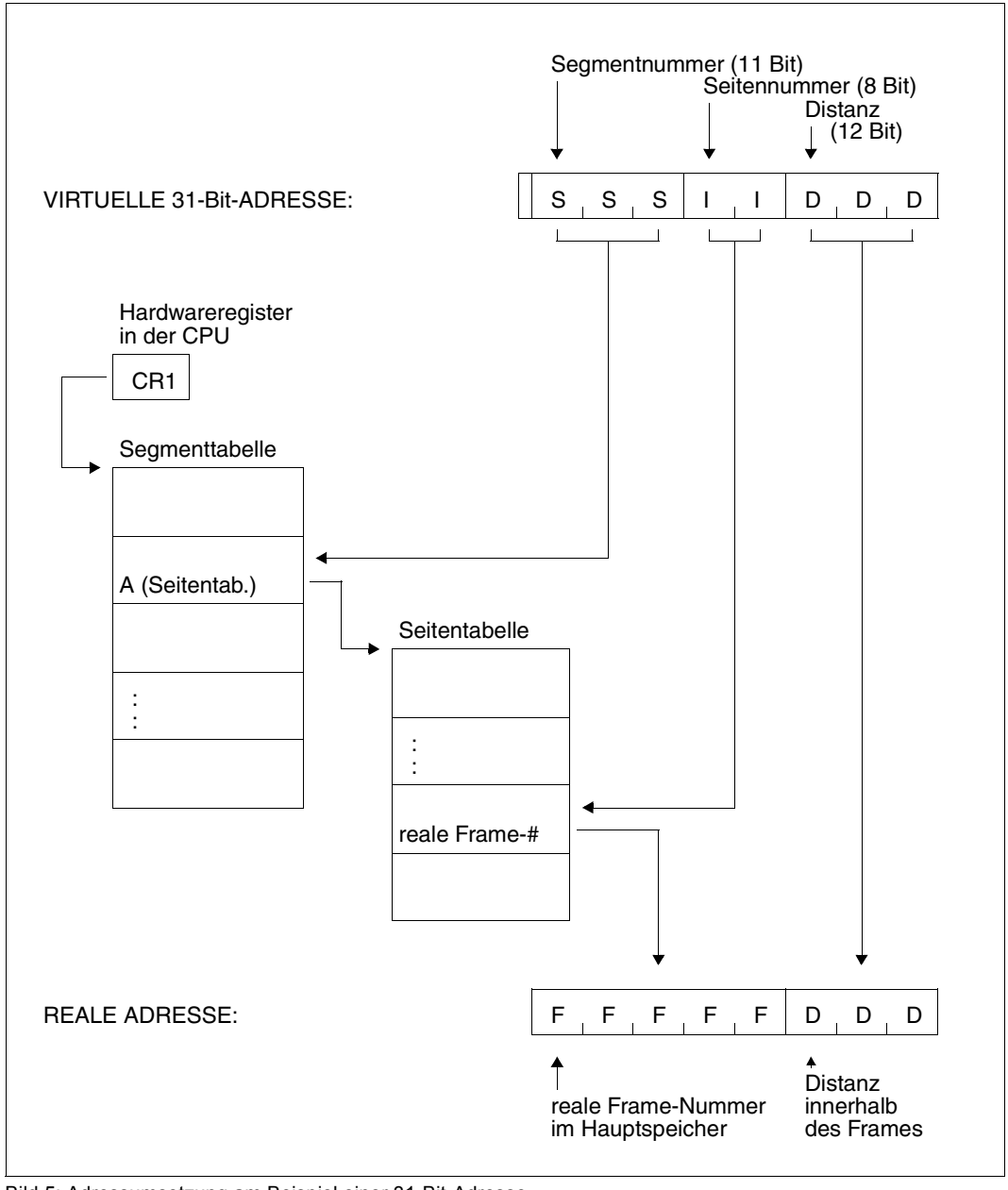


Bild 5: Adressumsetzung am Beispiel einer 31-Bit-Adresse

4.2.3 Arbeiten mit virtuellem Speicher

Makro	Kurzbeschreibung
CSTAT	ändert den Status von Speicherseiten eines Programms (Seitenwechsel und Lese-/Schreibzugriff)
MINF	informiert über Speicherbelegung und Größe des Klasse-6-Speichers (oder eines Memory Pools)
RELM	gibt einen zusammenhängenden Speicherbereich des aufrufenden Programms frei
REQM	fordert (zusätzlichen) Speicherplatz für das Programm an

Seiten eines Memory Pools (siehe Seite 56) können mit den Makros **REQMP** bzw. **RELMP** angefordert bzw. freigegeben werden.

4.2.4 Gemeinsamer Speicherbereich für mehrere Anwender (Memory Pool)

Makro	Kurzbeschreibung
CSTAT	ändert den Status von Speicherseiten eines Programms (Seitenwechsel und Lese-/Schreibzugriff)
CSTMP	ändert den Lese-/Schreibstatus auf einen Memory Pool
DISMP	beendet die Teilnahme an einem Memory Pool. Der Memory Pool wird aufgelöst, wenn das aufrufende Programm der letzte (einzige) Teilnehmer ist
ENAMP	richtet einen Memory Pool ein oder ermöglicht die Teilnahme an einem bestehenden Memory Pool
MINF	informiert über Seitenbelegung und Größe eines Memory Pools
RELMP	gibt (zusammenhängenden) Speicherplatz eines Memory Pools frei
REQMP	fordert (zusammenhängenden) Speicherplatz für einen Memory Pool an
SHOWMP	informiert über Common Memory Pools, die aktuell im System angelegt sind

Eigenschaften eines Memory Pools

Ein Memory Pool ist ein zusammenhängender Speicherbereich im Klasse-6-Speicher, der von mehreren Anwendern gemeinsam benutzt werden kann. Jeder Pool-Teilnehmer darf in jeder Seite des Memory Pools lesen oder schreiben. Ein Anwender, der kein Teilnehmer ist, hat keinen Zugriff auf die Pool-Seiten.

Wenn ein Pool-Teilnehmer den Inhalt oder den Status der Pool-Seiten ändert, sind alle Pool-Teilnehmer davon betroffen. Schreibt ein Teilnehmer in eine Seite des Pools, so sind diese Daten allen anderen Teilnehmern zugänglich. Schützt ein Teilnehmer eine Pool-Seite gegen Überschreiben (Makro **CSTAT**), so kann kein Pool-Teilnehmer mehr in diese Seite schreiben. Unterprogramme, die im Pool liegen, sollen ablaufinvariant sein.

Um die Zugriffe auf den Memory Pool zu koordinieren und eine sinnvolle Zusammenarbeit zu sichern, sollten die teilnehmenden Tasks synchronisiert ablaufen.
Dafür steht dem Benutzer z.B. das Verfahren der (Task-)Serialisierung zur Verfügung (siehe Seite 92).

Informationen über die aktuell im System angelegten Memory Pools und die jeweils angeschlossenen Tasks liefert der Makro **SHOWMP** und das BS2000-Kommando `/SHOW-MEMORY-POOL-STATUS`, siehe Handbuch „Kommandos“ [19].

Eröffnen eines Memory Pools

Mit dem Makroaufruf **ENAMP** kann ein Anwender einen Memory Pool einrichten oder sich einem bestehenden Memory Pool anschließen.

Richtet ein Anwender den Pool ein, so legt er dabei dessen Namen, Geltungsbereich und Größe für alle weiteren Teilnehmer verbindlich fest. Vom Geltungsbereich hängt es ab, ob und welche Tasks noch an dem Pool teilnehmen können.

Die Pool-Größe ist begrenzt durch den verfügbaren Benutzerspeicher der Teilnehmer. Der Memory Pool wird in 64 KByte- oder 1 MByte-Einheiten angelegt. Memory Pools aus 64 KByte-Einheiten werden immer unterhalb der 16 MByte-Grenze angelegt und langfristig nicht unterstützt. Es wird empfohlen, nur Memory Pools aus 1 MByte-Einheiten anzulegen (bessere Performance). Der **ENAMP**-Aufruf bewirkt, dass das System die entsprechende Anzahl von Einträgen in der Segmenttafel für den Pool reserviert (siehe Bild 6 auf Seite 59).

Schließt sich ein Anwender einem bestehenden Memory Pool an, so muss er dessen Namen, Geltungsbereich und Größe akzeptieren. An Stelle des Namens stellt das System eine Kurzkennung zur Verfügung, die die Verarbeitung beschleunigt. Jeder Teilnehmer kann bei der Pool-Eröffnung eine Adresse angeben, an der ihm das System die Kurzkennung einträgt.
Bei der Eröffnung kann jeder Anwender individuell festlegen, welcher Teil seines verfügbaren Adressraums dem Memory Pool zugeordnet sein soll. Er gibt für den Memory Pool eine Anfangsadresse an, die in seinem Adressraum liegt. Jeder Teilnehmer kann den Pool mit

dem selbstgewählten Bereich seines Adressraums adressieren. Die Adresse muss nicht einheitlich von allen Teilnehmern gewählt werden. Das System übergibt den Teilnehmern die virtuelle Adresse des ersten Pool-Byte in Register R1.

Schließen eines Memory Pools

Mit dem Makroaufruf **DISMP** wird die Teilnahme an dem angegebenen Memory Pool beendet. Der Teilnehmer kann danach wieder über den Teil seines Adressraums, der dem Pool zugeordnet war, neu verfügen. Der Memory Pool wird aufgelöst, wenn der Aufrufer von **DISMP** der letzte oder einzige Pool-Teilnehmer war. (Der den Memory Pool auflösende Teilnehmer muss nicht derjenige sein, der den Memory Pool eingerichtet hat).

Speicheranforderung im Memory Pool

Mit dem Makroaufruf **REQMP** kann ein Pool-Teilnehmer seitenweise (4 KByte) Speicherplatz für einen Memory Pool anfordern. Das System belegt die angeforderten Seiten in dem Bereich des virtuellen Speichers, der für den Pool (durch **ENAMP**) reserviert ist.
In den belegten Seiten kann nun jeder Pool-Teilnehmer schreiben und lesen. Der Teilnehmer, der die Belegung (verbindlich für alle) vornimmt, legt fest, welche und wie viele Seiten im reservierten Pool-Bereich belegt werden. Der Makro **MINF** informiert über Seitenbelegung und Größe des Memory Pools.

Speicherfreigabe im Memory Pool

Jeder Teilnehmer an einem Memory Pool kann mit den Makroaufruf **RELMP** seitenweise Speicher freigeben, der für den Memory Pool belegt war. Dabei spielt es keine Rolle, welcher der Teilnehmer den Speicher belegt hatte, und in welchen Portionen er belegt worden war. Der Aufrufer von **RELMP** legt fest, wie viele und welche der belegten Seiten freigegeben werden. Diese Freigabe gilt für alle Pool-Teilnehmer.

Seitenstatus im Memory Pool

Im Makro **ENAMP** kann festgelegt werden, ob der Speicherbereich resident sein soll. Der Aufrufer muss jedoch dazu berechtigt sein: Im Kommando START-PROGRAM für das Programm, das den **ENAMP** aufruft, muss vereinbart werden, wie viele Seiten des Prozesses resident sein dürfen. Diese Angabe muss der Benutzer berücksichtigen, wenn er im **ENAMP**-Aufruf die Größe des Memory Pools festlegt und gleichzeitig angibt, dass der Pool resident sein soll.

Mit dem Makroaufruf **CSTAT** kann ein Pool-Teilnehmer den Seitenstatus im Benutzerspeicher ändern. Der Seitenstatus eines Memory Pools kann mit dem Makro **CSTAT** nur von seitenwechselbar zu resident geändert werden. Eine Änderung des Seitenstatus von resident zu seitenwechselbar wird bei Memory Pools ignoriert, die über **ENAMP** resident festgelegt worden sind.

Der Makro **CSTAT** berücksichtigt keine Memory-Pool-Grenzen. Der Seitenbereich, der mit **CSTAT** beeinflusst wird, darf außerhalb eines Pools beginnen und im Pool enden und umgekehrt. Bei einer Änderung des Seitenstatus von resident zu seitenwechselbar werden nur die Seiten geändert, die außerhalb des residenten Memory Pools liegen.
Wird der Seitenstatus von seitenwechselbar zu resident geändert, so können alle Pool-Teilnehmer auch auf die residenten Seiten zugreifen. Diese Statusänderung kann jedoch nur von einem Pool-Teilnehmer durchgeführt werden, der dazu berechtigt ist.
(Im START-PROGRAM-Kommando für das Programm, das **CSTAT** aufruft, ist festgelegt, ob und wie viele Speicherseiten resident sein dürfen.)
Der Anwender kann mit **CSTAT** auch die Zugriffsart (Lese-/Schreibzugriff) auf die angegebenen Speicherseiten festlegen. Die Speicherseiten können sowohl in einem Memory Pool als auch außerhalb desselben liegen.

Mit dem Makro **CSTMP** kann der (dazu berechtigte) Anwender die Zugriffsart für einen Memory Pool festlegen. Die Angabe betrifft immer alle Speicherseiten des Memory Pools. Folgendes ist zu beachten:

– Bezüglich der Änderung der Zugriffsart (Lese-/Schreibzugriff) hat **CSTAT** eine niedrigere Priorität als **CSTMP**.
  Mit **CSTMP** eingerichteter Schreibschutz kann mit **CSTAT** nicht aufgehoben werden.
– **CSTAT** wird abgewiesen, wenn der Schreibschutz mit **CSTMP** eingerichtet wurde.
– Mit **CSTAT** eingerichteter Schreibschutz kann mit **CSTMP** auf alle Pool-Seiten erweitert werden.

Einschränkungen bei Verwendung von Memory Pools

– Fixpunktverarbeitung (Makro **WRCPT**/Kommando RESTART-PROGRAM) ist nicht zugelassen, wenn ein Memory Pool eröffnet ist.
– Das HOLD-TASK-Kommando (siehe Handbuch „Kommandos“ [19]) wird abgewiesen.
– Wenn ein nicht behebbarer Hauptspeicherfehler in einer Pool-Seite auftritt, wird dasjenige Programm beendet, das auf die fehlerhafte Speicherseite zugreift.

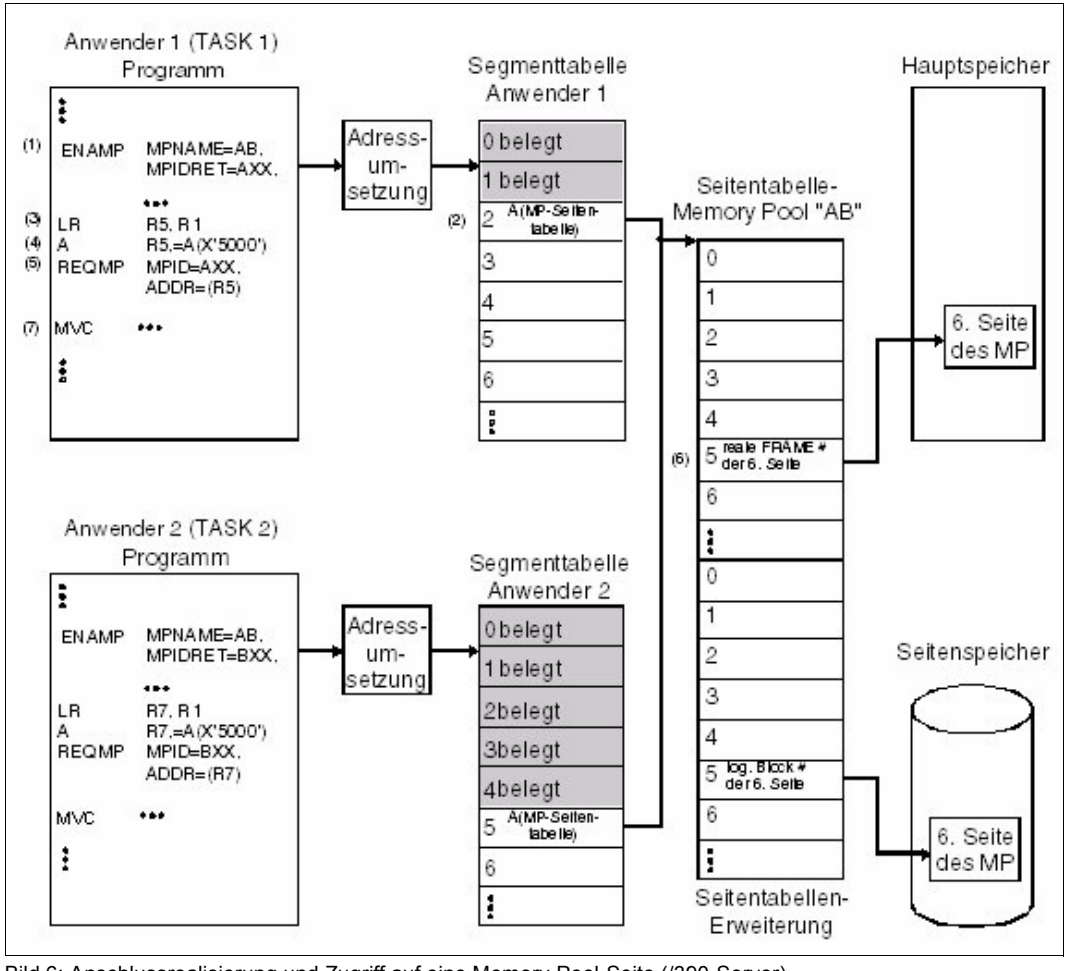


Bild 6: Anschlussrealisierung und Zugriff auf eine Memory-Pool-Seite (/390-Server)

(1) Mit dem Makro **ENAMP** erfolgt der Anschluss an den Memory Pool AB.

(2) In den nächsten freien Eintrag der Segmenttabelle wird die reale Anfangsadresse der Memory-Pool-Seitentabelle eingetragen. In diesem Fall ist der nächste freie Eintrag der Eintrag für das 3. Segment. Für Anwender 1 beginnt also der Memory Pool AB an der virtuellen Adresse X'00200000'. Diese virtuelle Anfangsadresse wird im Register R1 abgespeichert.

(3) Der Inhalt von Register R1 wird in Register R5 geladen. Beide Register enthalten nun die virtuelle Anfangsadresse des Memory Pools AB für den Anwender 1.

(4) Zur virtuellen Anfangsadresse des Memory Pools AB werden 5 Seiten (X'5000') addiert und das Ergebnis in Register R5 abgelegt.

(5) Mit dem **REQMP**-Makro wird nun diese 6. Seite des Memory Pools AB angefordert.

(6) Der 6. Eintrag in der Memory-Pool-Seitentabelle enthält die reale Frame-Nummer der 6. Seite des Memory Pools AB im Hauptspeicher. Der 6. Eintrag in der Seitentabellen-Erweiterung enthält die logische Block-Nummer der 6. Seite im Seitenspeicher.

(7) Zugriff auf die 6. Seite des Memory Pools AB.

4.2.5 Erweiterung durch Datenräume

Makro	Kurzbeschreibung
ALESRV	realisiert den Anschluss eines Programms an einen Datenraum und löst diese Verbindung wieder.
ALINF	informiert über die Zugriffslisten, mit denen Datenräume und deren Verbindungen verwaltet werden.
DSPSRV	erzeugt, erweitert, löscht virtuellen Adressraum für Datenadressierung (Datenraum), informiert über einen Datenraum und gibt ihn wieder frei.

Das Prinzip des Erweiterten Adressraums

Der erweiterte Adressierungsmodus steht auf allen BS2000-Servern zur Verfügung.

Im erweiterten Adressierungsmodus werden neben dem bisherigen Adressraum weitere Adressräume für Daten zur Verfügung gestellt. Da diese neuen virtuellen Adressräume z.T. andere Eigenschaften haben als der bisherige virtuelle Adressraum, werden folgende Begriffe eingeführt:

– **Programmraum** für den bisherigen virtuellen Adressraum
– **Datenraum** für die neuen virtuellen Adressräume

Die Eigenschaften eines Datenraumes im Überblick:
– Adressraum nur für Daten, d.h. adressierter Programmcode kann nicht ausgeführt werden
– Nutzung im 24-Bit- und 31-Bit-Adressierungsmodus möglich
– Größe je nach Adressierungsmodus von 4 KByte bis 16 MByte/2 GByte
– homogener Adressraum, d.h. keine Speicherklassen, gleiche Seitenattribute
– enthält keine reservierten Systemadressbereiche
– Mehrbenutzbarkeit möglich (ähnlich der Memory-Pool-Technik)
– Abschottung von Daten möglich

Vorteile, die sich bei Nutzung von Datenräumen für den Anwender ergeben:
– eine Erweiterung der gesamten adressierbaren Datenmenge um Größenordnungen
– eine bessere Möglichkeit zur Strukturierung des Adressraumes, der konsequenten Trennung von Programmcode und Daten und der Abschottung sicherheitsrelevanter oder sonstiger kritischer Daten
– eine Vergrößerung des verfügbaren Programmraums für den Anwender durch Auslagerung seiner eigenen Daten in einen Datenraum

Programmraum und Datenraum

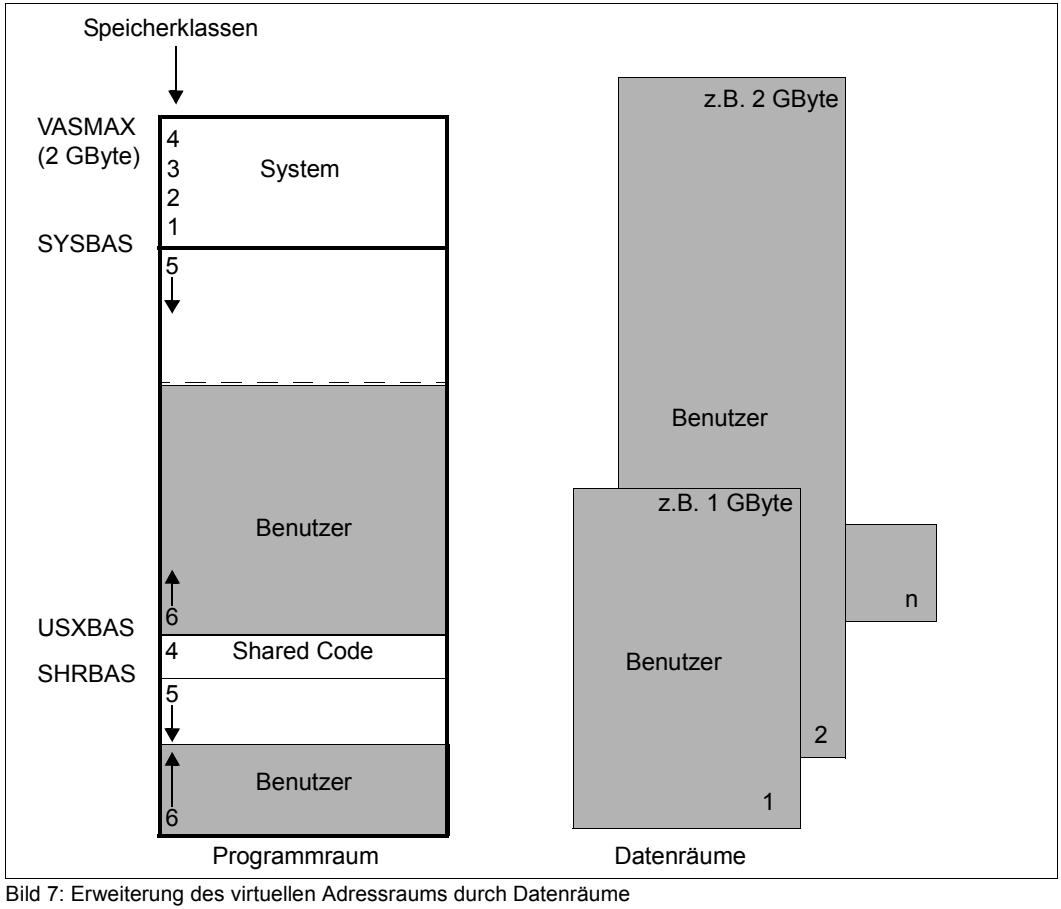


Bild 7: Erweiterung des virtuellen Adressraums durch Datenräume

Der **Programmraum** entspricht dem bisherigen virtuellen Adressraum, beginnt also an der virtuellen Adresse 0 (Null) und ist maximal 2 GByte groß. In ihm können sowohl ablauffähige Programme als auch reine Daten adressiert werden.

Ein **Datenraum** ist ein Bereich von zusammenhängendem virtuellen Adressraum in der Größe von 4 KByte bis 2 GByte.
Ein Datenraum beginnt an der virtuellen Adresse 0. Er ist in seiner gesamten Größe vom Anwender nutzbar, da er (im Gegensatz zum Programmraum) keine für das System reservierten Bereiche enthält.

In einem Datenraum können nur Daten und als Daten abgelegte Programme adressiert werden, d.h. in einem Datenraum adressierter Programmcode kann nicht zum Ablauf gebracht werden. Der Datenraum wird von einem Anwender durch Angabe von Typ, Name, Geltungsbereich und gewünschter Größe eingerichtet. Dieser Anwender wird damit zum Eigentümer des Datenraums.

Ein Datenraum wird als homogener Adressraum realisiert, d.h. alle Seiten erhalten, sobald sie angefordert werden, dieselben Attribute. Diese Attribute werden beim Anlegen eines Datenraums definiert. Die beim Einrichten angegebenen Attribute sind für die Lebensdauer des Datenraumes festgeschrieben. Alle Teilnehmer, die sich an einen Datenraum anschließen, müssen diese Eigenschaften akzeptieren.
Der Geltungsbereich legt fest, welche Tasks sich an den Datenraum anschließen und Zugriffe ausführen dürfen. Der Name eines Datenraumes ist nur innerhalb seines Geltungsbereiches eindeutig. Sessionweit wird der angeforderte Datenraum eindeutig durch die SPID (space identification) gekennzeichnet. Die SPID wird vom System vergeben.

Datenraum-Typen

Der Datenraum-Typ bestimmt die Art der Allokierung/Deallokierung des Speichers innerhalb des Datenraums. Er wird zum Zeitpunkt der Erzeugung bestimmt.

Es gibt folgende Datenraum-Typen:

– STACK
  Ein Datenraum vom Typ STACK ist ein virtuell zusammenhängender allokierter Bereich, beginnend ab Adresse 0 bis zur aktuellen Größe. Es stehen Funktionen zum Erweitern und Reduzieren der aktuellen Größe (EXTEND/REDUCE) und zum Löschen des Inhalts eines Bereichs innerhalb der aktuellen Größe (CLEAR) zur Verfügung.

– HEAP
  Ein Datenraum vom Typ HEAP ist ein virtueller Adressraum, in dem dynamisch beliebig große Bereiche bis zur maximalen Größe des Datenraums allokiert werden können. Die verfügbaren Allokierungsfunktionen sind GETAREA und RETAREA.

Die Größe wird in Einheiten zu je 4KB angegeben.

Adressierung in Datenräumen

Zugriffslisten
Bevor ein Programm auf Daten in einem Datenraum zugreifen kann, muss es eine Verbindung zu diesem Datenraum herstellen.
Jede Task besitzt eine taskeigene Zugriffsliste (access list, AL), die alle Verbindungen repräsentiert, die ein Programm aktuell zu Datenräumen aufgebaut hat. Zugriffslisten liegen im privilegierten Speicherbereich und werden vom System verwaltet.
Bei Auf- und Abbau von Verbindungen durch ein Programm werden Einträge (access list entry, ALE) in der taskeigenen Zugriffsliste hinzugefügt oder gelöscht.

ALET und SPID
Beim Aufbau einer Verbindung erhält das Programm einen Wert zurück (access list entry token, ALET), der als Verweis auf den neuen Eintrag in der Zugriffsliste dient.
Während die SPID (die dem Datenraum bei seiner Erzeugung vom System vergeben wurde) von der Software zur sessionweiten Identifizierung des Datenraums benutzt wird, dient der ALET der hardwaremäßigen Adressierung eines Datenraums. Der Wert des ALET ist taskspezifisch: wenn mehrere Tasks eine Verbindung zu ein und demselben Datenraum aufbauen, bekommt jede von ihnen einen anderen ALET zugewiesen, da dieser einen Eintrag in der taskeigenen Zugriffsliste identifiziert.

Zugriffsregister
Der Zugriff auf die Datenräume erfolgt durch einen zusätzlichen Registersatz von 16 Zugriffsregistern (access register, AR). Die 16 Zugriffsregister sind den 16 Mehrzweckregistern (MZR) eineindeutig zugeordnet. Die bei der Adressberechnung verwendeten MZR können Basis- oder Indexregister sein.
Wird zur Adressberechnung nur ein Indexregister und kein Basisregister verwendet, wird immer der Programmraum adressiert.
Wird jedoch bei der Adressberechnung (auch) ein Basisregister zur Adressierung verwendet und ist der ALET des mit ihm korrespondierenden Zugriffsregisters nicht Null, wird der entsprechende Datenraum adressiert. Somit kann jeder Adresse, die über ein Basisregister angesprochen wird, ein eigener Datenraum zugeordnet werden. Da MZR0 nicht als Basisregister verwendet werden darf, kann auch AR0 nicht zur Adressierung eines Datenraums verwendet werden.
Indem das entsprechende Zugriffsregister mit der Identifikation der Verbindung (ALET) zu einem Datenraum geladen wird, wird die Adressumsetzung für den Datenzugriff mit den Adress-Umsetztabellen des Datenraums durchgeführt.
So wie ein anderer Bereich innerhalb des Programmraums durch Umladen des Basisregisters mit der Anfangsadresse dieses Bereichs adressiert werden kann, kann durch Umladen des Zugriffsregisters mit einem anderen ALET aus der Zugriffsliste ein anderer Datenraum adressiert werden. Damit repräsentiert die Zugriffsliste für ein Programm die Menge aller adressierbaren Datenräume zu einem bestimmten Zeitpunkt.

Der Mechanismus des Zugriffs auf einen Datenraum über die Zugriffsliste wird nur dann wirksam, wenn es sich um einen Operandenzugriff auf Daten handelt. Bei Sprungbefehlen erfolgt der Operandenzugriff immer im Programmraum. Dadurch wird verhindert, dass Programmcode in einem Datenraum ablaufen kann.

AR-Modus

Um die Möglichkeit der Adressraumerweiterung zu nutzen, muss dem Programm mitgeteilt werden, dass es mit dem zusätzlichen Registersatz arbeiten, d.h. im **AR-Modus** (access register mode) laufen soll. Ist der AR-Modus nicht eingeschaltet, werden virtuelle Adressen des Programmraums angesprochen.
Mit dem Assemblerbefehl SAC kann der AR-Modus ein- und ausgeschaltet werden.

```
SAC   512      * set address space control  -  einschalten des AR-Modus *
SAC   0        *                            -  ausschalten des AR-Modus *
```

Während also ohne eingeschaltetem AR-Modus der Zugriff auf Programmcode und Daten im Programmraum erfolgt, kann im AR-Modus der Datenzugriff auf durch Zugriffsregister indizierte Datenräume erfolgen.
Ist der AR-Modus eingeschaltet, berechnet sich eine reale Adresse wie bisher durch zweistufige Adressumsetzung der virtuellen Adresse (Indexregister + Basisregister + Distanz), jedoch unter Berücksichtigung des Zugriffsregisters. Ist das zum Basisregister korrespondierende Zugriffsregister $\neq 0$, verweist es auf einen Datenraum und die virtuelle Adresse dieses Datenraumes wird umgesetzt. Ist das Zugriffsregister = 0, wird eine virtuelle Adresse des Programmraums umgesetzt.
Wenn ein Programm im AR-Modus läuft und ein MZR als Basisregister zur Adressierung von Daten verwendet, dann muss das entsprechende Zugriffsregister einen gültigen ALET erhalten, anderenfalls kommt es zum Programmabbruch.
Um im AR-Modus auch Daten im Programmraum adressieren zu können, steht ein spezieller ALET-Wert zur Verfügung: Der Wert ALET = 0 adressiert immer den Programmraum.

Für die Arbeit mit den Zugriffsregistern sind einige neue Assemblerbefehle eingeführt worden (siehe Handbuch „Assemblerbefehle (BS2000)“ [1]).

LAM	Load Access Multiple
STAM	STore Access Multiple
LAE	Load Address Extended
SAR	Set Access Register
CPYA	CoPY Access register
EAR	Extract Access Register
TAR	Test Access Register
SAC	Set Address space Control
IAC	Insert Address space Control

Schritte zur Herstellung der Adressierbarkeit von Datenräumen:

1. Anlegen eines Datenraumes (Makro **DSPSRV** FCT=CREATE)
   oder
   Besorgen der Identifikation (SPID) eines existierenden, mehrbenutzbaren Datenraums (Makro **DSPSRV** FCT=INFORM).
2. Herstellen der Verbindung zu diesem Datenraum
   (Makro **ALESRV** FCT=CONNECT), es wird ein ALET zurückgeliefert.
3. Laden des ALET in das Zugriffsregister, dessen entsprechendes MZR als Basisregister die Daten innerhalb des Datenraums adressiert
   (Befehl **LAM AR**x,**AR**x,**alet**, wenn ARx EQU x definiert ist).
4. Einschalten des AR-Modus (Befehl **SAC 512**).

Sind mehrere Tasks berechtigt, auf denselben Datenraum zuzugreifen und ist die Adressierbarkeit über die Zugriffsliste hergestellt, so muss die korrekte Serialisierung von den Anwendern realisiert werden.
Die max. Anzahl einzurichtender Datenräume pro Task ist 32. Pro Task kann jedoch auf weit mehr (bereits von anderen Tasks eingerichtete) Datenräume zugegriffen werden. Es können maximal 125 ALETs pro Task verwaltet werden, d.h. zu
125 Datenräumen kann eine Verbindung hergestellt werden.
Alle Zugriffsregister sind bei Programmbeginn mit Null initialisiert.

Zugriffsregister und AR-Modus sind nur prozedurlokal zu verwenden, da die Linkage Routinen nicht im AR-Modus ablauffähig sind und beim Aufruf von Subroutinen auch keine Sicherung der Zugriffsregister durchführen.

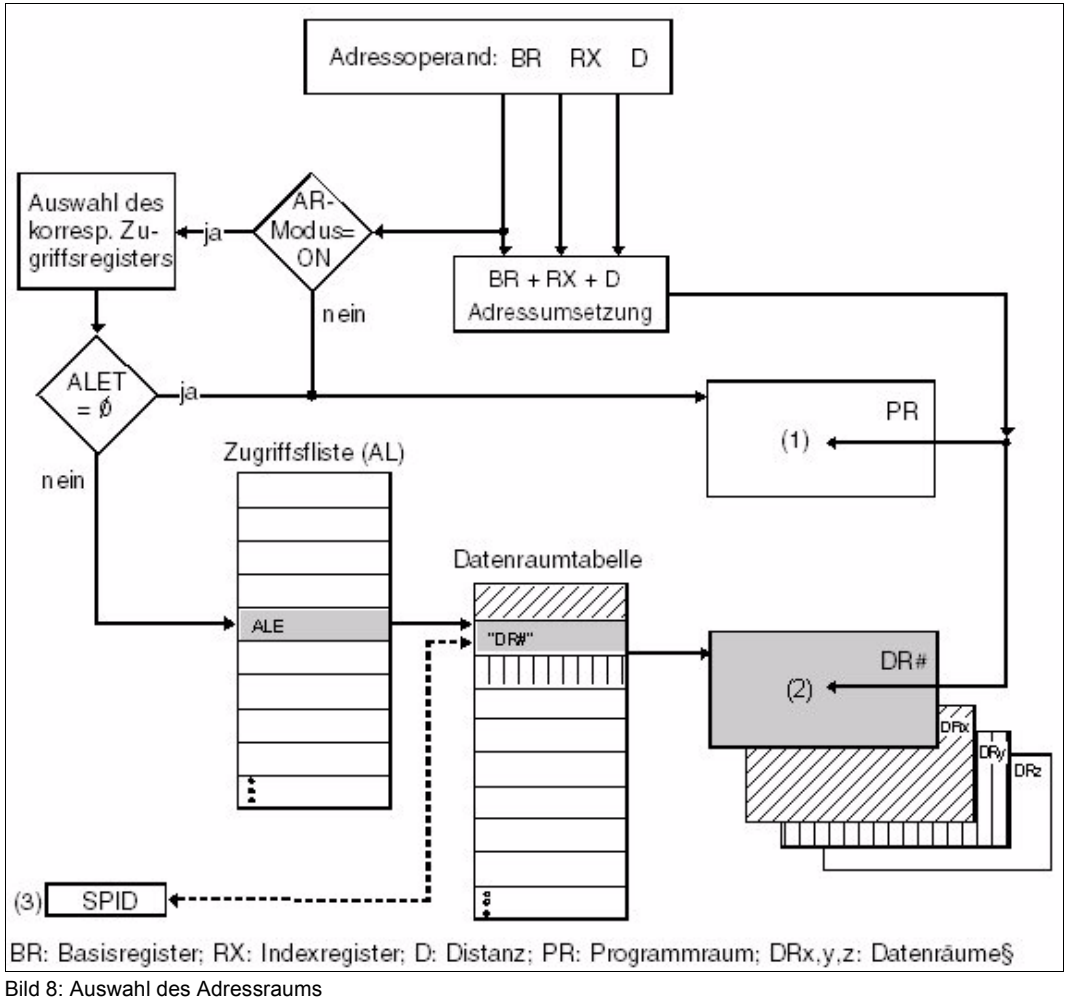


Bild 8: Auswahl des Adressraums

(1) Die bei der Adressumrechnung berechnete Adresse bezieht sich auf den Programmraum, wenn
- der AR-Modus nicht eingeschaltet ist oder
- der ALET im (zum Basisregister korrespondierenden) Zugriffsregister Null ist.

(2) Die bei der Adressumrechnung berechnete Adresse bezieht sich auf einen Datenraum. Über den ALET des korrespondierenden Zugriffsregisters wird ein Eintrag (ALE) in der Zugriffsliste adressiert. Dieser ALE zeigt wiederum auf einen Eintrag in der Datenraumtabelle, durch den der Datenraum identifiziert wird.

(3) Ein Datenraum kann auch durch die vom System vergebene SPID angesprochen werden. Durch die SPID wird er sessionweit eindeutig identifiziert.

Makros für die Nutzung von Datenräumen

Für den nichtprivilegierten Anwender stehen drei Makros für die Arbeit mit erweitertem virtuellen Adressraum zur Verfügung.
Mit dem Makro **DSPSRV** kann ein Datenraum angefordert oder freigegeben werden. In einem schon bestehenden Datenraum kann der Anwender (zusätzliche) Speicherseiten anfordern und freigeben. Es können Informationen über ihn angefordert werden.
Der Makro **ALESRV** verwaltet die Einträge in der Zugriffsliste. Er realisiert den Anschluss einer Task an einen Datenraum und kann diese Verbindung auch wieder lösen.
Bei Angabe des ALET gibt der Makro die zum Datenraum gehörende SPID aus.
Mit dem Makro **ALINF** kann sich der Anwender darüber informieren, welche Einträge in der Zugriffsliste auf welche Datenräume verweisen.

Beispiel

Folgendes Beispiel soll das Arbeiten mit einem Datenraum vom Typ STACK veranschaulichen. (@-Ausdrücke sind vordefinierte Makros des ASSEMBH-Assemblers.) Nachstehend sind nur die zum Arbeiten mit Datenräumen relevanten Teile des vollständigen Programms aufgeführt.

```
DATASPAC START
         PRINT NOGEN
*        :
DSPMFD   DSPSRV MF=D
ALEMFD   ALESRV MF=D
*
DATASPAC @ENTR TYP=M
*        :
CREATE   LA    1,DSPPL ------------------------------------------------------ (1)
         @DATA DSECT=DSPMFD,BASE=1
         MVC   DSPPL(NVDD#),DSPMFL
         DSPSRV MF=M,FCT=CREATE,INISIZE=25
         DSPSRV MF=E,PARAM=(1)
         DSPSRV MF=R,SPID=DSPSPID
*
CONNECT  LA    1,ALEPL ------------------------------------------------------ (2)
         @DATA DSECT=ALEMFD,BASE=1
         MVC   ALEPL(NVDA#),ALEMFL
         ALESRV MF=M,SPID=DSPSPID
         ALESRV MF=E,PARAM=(1)
         ALESRV MF=R,ALET=DSPALET
*
         SAC   512 ---------------------------------------------------------- (3)
*
```

```
WRITEDS  LAM   8,8,DSPALET ------------------------------------------------ (4)
         SR    8,8
         MVC   0(100,8),DATA
*
         SAC   0 ---------------------------------------------------------- (5)
*
INFORM   LA    1,DSPPL ---------------------------------------------------- (6)
         @DATA DSECT=DSPMFD,BASE=1
         MVC   DSPPL(NVDD#),DSPMFL
         DSPSRV MF=M,FCT=INFORM,IDENT=NAME,NAME='SHARED#DS',           C
               SCOPE=GLOBAL
         DSPSRV MF=E,PARAM=(1)
         DSPSRV MF=R,SPID=DSPSPID2
*
CONNECT2 LA    1,ALEPL ---------------------------------------------------- (7)
         @DATA DSECT=ALEMFD,BASE=1
         MVC   ALEPL(NVDA#),ALEMFL
         ALESRV MF=M,SPID=DSPSPID2
         ALESRV MF=E,PARAM=(1)
         ALESRV MF=R,ALET=DSPALET2
*
         SAC   512 -------------------------------------------------------- (8)
*
COPYDS   LAM   7,7,DSPALET2 ------------------------------------------------ (9)
         SR    7,7
         MVC   0(100,7),0(8)
*
         SAC   0 --------------------------------------------------------- (10)
*
CLEAR    LA    1,DSPPL --------------------------------------------------- (11)
         @DATA DSECT=DSPMFD,BASE=1
         DSPSRV MF=M,FCT=CLEAR,SPID=DSPSPID2,AREA=1000,SIZE=100
         DSPSRV MF=E,PARAM=(1)
*
EXTEND   DSPSRV MF=M,FCT=EXTEND,SPID=DSPSPID,SIZE=1000 ---------------- (12)
         DSPSRV MF=E,PARAM=(1)
         DSPSRV MF=R,EXTADDR=DSPEXTND
*
*** Further access to the extended data space (read, write, etc.) ***
*
DISCONN  LA    1,ALEPL --------------------------------------------------- (13)
         @DATA DSECT=ALEMFD,BASE=1
         MVC   ALEPL(NVDA#),ALEMFL
         ALESRV MF=M,FCT=DISCONN,ALET=DSPALET
         ALESRV MF=E,PARAM=(1)
         ALESRV MF=M,FCT=DISCONN,ALET=DSPALET2
         ALESRV MF=E,PARAM=(1)
*
```

```
DESTROY  LA    1,DSPPL ------------------------------------------------- (14)
         @DATA DSECT=DSPMFD,BASE=1
         DSPSRV MF=M,FCT=DESTROY,SPID=DSPSPID
         DSPSRV MF=E,PARAM=(1)
         @EXIT
*
*** Definitions ***
*
DSPMFL   DSPSRV MF=L,NAME='SPACE1',MAXSIZE=2000
ALEMFL   ALESRV MF=L,FCT=CONNECT
*
DSPSPID  DS    D                 * SPID of data space created by
*                                * program
DSPALET  DS    F                 * ALET of this data space
DSPSPID2 DS    D                 * SPID of another program's
*                                * data space
DSPALET2 DS    F                 * ALET of this data space
DSPEXTND DS    A                 * Extension address
DSPPL    DS    XL(NVDD#)         * Dynamic data area for DSPSRV
ALEPL    DS    XL(NVDA#)         * Dynamic data area for ALESRV
DATA     DS    XL100             * Data to be transferred
         @END
```

(1) Es wird ein Datenraum in der Größe von 100 KByte erzeugt:
In das Register 1 wird die Adresse des Datenbereichs für den Makro **DSPSRV** geladen (LA), danach die DSECT über den Datenbereich gelegt (@DATA) und der Datenbereich initialisiert (MVC). Mit den **DSPSRV**-Aufrufen wird der Datenraum erzeugt. In das Feld `DSPSPID` soll die vom System vergebene SPID eingetragen werden.

(2) Das Programm schließt sich an den Datenraum an:
In das Register 1 wird die Adresse des Datenbereichs für den Makro **ALESRV** geladen (LA), danach die DSECT über den Datenbereich gelegt (@DATA) und der Datenbereich initialisiert (MVC). Mit den **ALESRV**-Aufrufen schließt sich das Programm unter Angabe der SPID an den Datenraum an. Im Feld `DSPALET` soll der dazugehörende ALET abgespeichert werden.

(3) Der AR-Modus wird eingeschaltet. Damit ist jedem Mehrzweckregister ein Zugriffsregister eineindeutig zugeordnet.

(4) Der Datenraum wird mit Daten beschrieben:
Das Basisregister ist Register 8. Das Zugriffsregister 8 wird mit dem ALET des Datenraums geladen (LAM).
Danach wird das Basisregister gelöscht (SR).
Aus dem Feld `DATA` werden Daten in der Länge von 100 Byte an den Anfang des Datenraums geschrieben (MVC).

(5) Der AR-Modus wird ausgeschaltet:
Der AR-Modus sollte immer so früh wie möglich ausgeschaltet werden, um nicht-erwünschten Datenraum-Zugriff zu vermeiden (gilt besonders bei Verzweigung in Unterprogramme).

(6) Es wird die SPID eines fremden, jedoch mehrbenutzbaren Datenraums abgefragt:
Mit den **DSPSRV**-Aufrufen wird der fremde Datenraum durch Angabe seines Namens und seines Geltungsbereichs angesprochen. In das Feld `DSPSPID2` soll die SPID des fremden Datenraums eingetragen werden.

(7) Das Programm schließt sich an den fremden Datenraum an:
Mit den **ALESRV**-Aufrufen schließt sich das Programm unter Angabe der SPID an den fremden Datenraum an. Im Feld `DSPALET2` soll der dazugehörende ALET abgespeichert werden.

(8) Der AR-Modus wird eingeschaltet. Damit ist jedem Mehrzweckregister ein Zugriffs-register eineindeutig zugeordnet.

(9) Kopieren von Daten aus einem Datenraum in einen anderen:
Das Zugriffsregister 7 wird mit dem ALET des fremden Datenraums geladen (LAM). Aus dem eigenen Datenraum, der durch das Zugriffsregister 8 repräsentiert wird, werden 100 Byte an den Anfang des fremden Datenraums geschrieben.

(10) Der AR-Modus wird ausgeschaltet (siehe (5)).

(11) Löschen von Daten im fremden Datenraum:
Mit den **DSPSRV**-Aufrufen werden ab Adresse X'1000' des fremden Datenraums 400 KByte gelöscht, also mit binären Nullen überschrieben.

(12) Erweitern eines Datenraums:
Mit den **DSPSRV**-Aufrufen wird der eigene Datenraum um 4000 KByte erweitert.

(13) Es wird die Verbindung zu beiden Datenräumen gelöst:
Mit den **ALESRV**-Aufrufen löst das Programm unter Angabe der jeweiligen ALETs die Verbindung zu beiden Datenräumen.

(14) Der vom Programm erzeugte (eigene) Datenraum wird zerstört. Der korrespondierende Eintrag in der Zugriffsliste wird gelöscht.

4.3 Steuerung von Task und Programmlauf

4.3.1 Starten, Unterbrechen und Beenden

Makro	Kurzbeschreibung
BKPT	übergibt die Steuerung an das System. Der Benutzer kann dann an seiner Datenstation Kommandos eingeben
ENTER	leitet einen Batch-Auftrag ein (Funktion des Kommandos ENTER-JOB)
EXIT	beendet einen STXIT-Prozess
LGOFF	beendet den Auftrag; (Funktion des Kommandos EXIT-JOB)
PASS	lässt das aufrufende Programm eine Sekunde warten
RETRN	springt aus dem laufenden Programm zurück in dasjenige, das es (z.B. mit Befehl BR) aufgerufen hat. Dabei können Registerinhalte zurückgeladen werden, die das laufende Programm mit dem Makro SAVE zwischengespeichert hatte
SAVE	sichert Registerinhalte durch Zwischenspeichern; der Aufruf ist sinnvoll zu Beginn eines Unterprogramms. Die Registerinhalte können mit dem Makro RETRN zurückgeladen werden
SETIC	startet den Intervallzeitgeber oder setzt ihn zurück. (Verwendung in Verbindung mit dem Makro STXIT)
STXIT	legt benutzereigene Routinen zur Unterbrechungsbehandlung fest, mit denen das System die Verarbeitung fortsetzt, wenn eine Programmunterbrechung auftritt
TERM	beendet Programm und Auftragsabschnitt und veranlasst einen Speicherabzug
TINF	verändert die Runpriorität der Task, den Jobtyp (Batch-, Dialog-, Transaktionsauftrag) oder die Operanden für das Deaktivierungsverbot der Task
VPASS	setzt die Anwendertask für eine bestimmte Zeit in den Wartezustand

4.3.2 Benutzer- und Auftragsschalter

Makro	Kurzbeschreibung
SWITCH	liest oder verändert die 32 Auftragsschalter des Auftrags bzw. die 32 Benutzerschalter der eigenen oder einer fremden Benutzerkennung

Der Makro **SWITCH** ersetzt die Makros **GETSW**, **GETUS**, **SETSW** und **SETUS**. Diese Makros werden noch aus Kompatibilitätsgründen unterstützt und sind im Anhang ab Seite 1147 beschrieben.

Verwendung von Auftragsschaltern in BS2000

Wenn man Auftragsschalter verwendet, muss man beachten, dass einige Systemkomponenten und Software-Produkte bestimmte Auftragsschalter verändern oder sich durch sie steuern lassen. Die folgende Tabelle zeigt in der Reihenfolge der nachfolgenden Beschreibungen, welche Software-Produkte und Systemkomponenten standardmäßig Auftragsschalter verwenden:

Systemkomponente/Software-Produkt	Schalter
ARCHIVE	30, 31
BCAMDEF	0, 4, 5, 31
DAMP	5, 30
DBL/ELDE (Binder/Lader)	4
EDT	4 - 7
STEP/SET-JOB-STEP-Kommando	16 - 31
TSOSLNK	4

Es gilt:
Durch Einschalten des Schalters 4 wird die Meldung `BLS0500` des Laders unterdrückt.

Nach Ausführung des Kommandos SET-JOB-STEP sind alle Auftragsschalter größer 15 ausgeschaltet.

Die Wirkung der Schalterstellung auf die oben angegebenen Systemkomponenten bzw. Software-Produkte wird auf den nächsten Seiten beschrieben.

ARCHIVE
Das Software-Produkt ARCHIVE kann sowohl in Prozeduren als auch in einem ENTER-Auftrag aufgerufen werden. Informationen über den Verlauf kann man der Stellung der Schalter entnehmen, die ARCHIVE während des Laufs bzw. nach dem Lauf ein-/ausschaltet.

Schalter 30 von ARCHIVE eingeschaltet: Warnmeldung in Prozeduren
Schalter 30 wird von ARCHIVE eingeschaltet, wenn die ARCHIVE-Anweisung zwar ausgeführt, aber eine Warnmeldung ausgegeben wurde.

Schalter 31 von ARCHIVE eingeschaltet: Fehler in Prozeduren
Schalter 31 wird von ARCHIVE eingeschaltet, wenn ein Fehler erkannt wurde und die ARCHIVE-Anweisung dennoch ausgeführt wurde.

BCAMDEF
Schalter 0, 4, 5, 31 werden benutzt:
Innerhalb der Prozedur BCAMDEF werden die Schalter 0, 4, 5, 31 eingeschaltet und wieder ausgeschaltet.

DAMP

Schalter 5 für DAMP gesetzt

Vor dem Aufruf des Programms DAMP in einer Prozedur muss der Auftragsschalter 5 gesetzt werden.
Folgt im Prozedurbetrieb auf eine DAMP-Anweisung ein Systemkommando, das nicht in der DAMP-Programmebene zugelassen ist, geht DAMP vom Prozedur- in den Dialogbetrieb über und setzt den Auftragsschalter 5 zurück. Am Bildschirm wird dann der letzte DAMP-Ausgabeschirm angezeigt. Im Batchbetrieb führt das nicht zugelassene Systemkommando zum Abbruch des Auftrags.

Schalter 30 für DAMP gesetzt

Ist der Auftragsschalter 30 gesetzt und läuft DAMP im Dialogmodus, so erfolgt eine Rückfrage, ob ein Speicherauszug veranlasst werden soll.
Ist der Auftragsschalter 30 gesetzt und läuft DAMP im Batchmodus, so unterbleibt der Speicherauszug.

DBL/ELDE (Binder/Lader)
Schalter 4 eingeschaltet: Die Systemmeldungen (`BLS0500`, `BLS0517`,...) über das Laden eines Moduls werden unterdrückt.

EDT

Schalter 4 für EDT eingeschaltet: EDT unterdrückt Anfangsmeldung, Endemeldung

– Dialogaufträge, Batch-Aufträge:
  Einschalten von Auftragsschalter 4 vor dem Laden des EDT unterdrückt die Ladermeldung BLS0500 und bei Beendigung des EDT die Meldung `EDT800`.
  Folgende Meldung wird ebenfalls unterdrückt:
  `EDITED FILE(S) NOT SAVED! TERMINATE (Y/N)?`

– Batch-Aufträge:
  Setzen von Auftragsschalter 4 bedingt, dass während des EDT-Laufs kein Protokoll geschrieben wird (≙ @LOG=NONE).

Schalter 5 für EDT eingeschaltet: EDT arbeitet im L-Modus
Die Eingaben werden (zeilenweise) von SYSDTA gelesen. An der Datensichtstation wird an Stelle der aktuellen Zeilennummer das Zeichen * ausgegeben. Mit der Anweisung @EDIT FULL SCREEN wird der F-Modus eingestellt.
Das Ein-/Ausschalten des Schalters 5 während des EDT-Laufs verändert den eingestellten Modus nicht.

Schalter 6 für EDT eingeschaltet: EDT schreibt 160 Schreibstellen
Ist Schalter 6 eingeschaltet, so schreibt EDT 160 Schreibstellen in eine Zeile nach SYSLST und einen eventuellen Rest in den nächsten Satz. Die Anwendung ist sinnvoll, wenn die (System-)Datei SYSLST auf einen Drucker mit max. 160 Zeichen Druckzeilenlänge ausgegeben wird. Der Schalter muss vor Aufruf des EDT eingeschaltet werden.
Normalerweise benutzt EDT 132 Schreibstellen und schreibt einen eventuellen Rest in den nächsten Satz.

Schalter 7 für EDT eingeschaltet: Überflüssiger Speicherplatz wird nicht freigegeben.
Einschalten von Schalter 7 verhindert die automatische Freigabe von vorab zugewiesenem überschüssigem Speicherplatz durch den EDT. EDT gibt sonst nicht belegten Speicherplatz frei (negative Angabe im SPACE-Operanden des FILE-Makros, siehe Handbuch „DVS Makros“ [7]). Der Schalter kann auch während des EDT-Laufs eingeschaltet werden.

SET-JOB-STEP-Kommando

Schalter 16 bis 31 werden ausgeschaltet:
Setzt ein Anwender ein SET-JOB-STEP-Kommando ab, so werden die Auftragsschalter 16 bis 31 ausgeschaltet.

TSOSLNK

Schalter 4 für TSOSLNK eingeschaltet: Seitenvorschub unterdrückt.
Einschalten des Auftragsschalters 4 bewirkt, dass bei der Ausgabe des Binderprotokolls auf SYSLST alle Seitenvorschübe entfallen.

4.3.3 Intertaskkommunikation (ITC)

Makro	Kurzbeschreibung
CLCOM	beendet die Teilnahme an der Intertaskkommunikation
OPCOM	erklärt die Teilnahme an der Intertaskkommunikation und übergibt einen ITC-Namen
RELBF	löscht die erste Nachricht in der ITC-Empfangswarteschlage des ITC-Teilnehmers
REVNT	empfängt eine Nachricht für den ITC-Namen des Teilnehmers
SEVNT	sendet eine Nachricht an einen ITC-Teilnehmer

Verfahren der Intertaskkommunikation ITC

Die Intertaskkommunikation (ITC) ermöglicht den Nachrichtenaustausch zwischen Programmen, die in voneinander verschiedenen Tasks ausgeführt werden. Jeder ITC-Teilnehmer muss sich durch einen Namen gegenüber der ITC ausweisen. ITC-Teilnehmer sind aus der Sicht des Betriebssystems die Tasks, die mit einem ITC-Namen in der Kommunikationstabelle geführt werden; aus der Sicht des Anwenders sind es die (Anwender-) Programme, die in diesen Tasks ausgeführt werden und **OPCOM** aufgerufen haben. Durch die Teilnahme an der Intertaskkommunikation können alle Tasks im System untereinander Nachrichten austauschen und - wenn nötig - auf deren Eintreffen warten, sofern sie gegenseitig ihre ITC-spezifischen Namen kennen. Jeder ITC-Teilnehmer kann Sender und Empfänger von Nachrichten sein. Ein Teilnehmer (Programm) kann an einen anderen Teilnehmer eine Nachricht senden, aber erfährt nicht automatisch, ob der Empfänger die Nachricht auswertet. Der Empfänger kann zu diesem Zweck seinerseits eine Nachricht zurücksenden.

Ein Teilnehmer kann nicht nur Nachrichten senden, sondern er kann auch Nachrichten anfordern, die andere Teilnehmer an ihn gesendet haben oder senden werden. Liegt die Nachricht zum Zeitpunkt der Anforderung noch nicht vor, dann kann der Programmlauf für eine gewählte Zeitspanne angehalten werden. Sobald die Nachricht eintrifft oder spätestens nach Ablauf der Wartezeit setzt ihn das System fort und übergibt die Nachricht. Der Empfänger kann den Teilnehmer festlegen, dessen Nachricht seine Wartezeit beenden soll; er kann aber auch einen beliebigen Teilnehmer als Sender zulassen.

Die Intertaskkommunikation wird mit folgenden Makros durchgeführt:

OPCOM ITC-Teilnahme erklären
SEVNT Nachricht senden
REVNT Nachricht anfordern
RELBF Empfangswarteschlange löschen
CLCOM ITC-Teilnahme beenden

Teilnahme an der ITC erklären und beenden

Soll ein Programm an der Intertaskkommunikation teilnehmen, so muss der Makro **OPCOM** aufgerufen und ein ITC-Name übergeben werden. Das System prüft diesen Namen und weist ihn ab, wenn er schon an einen anderen ITC-Teilnehmer vergeben ist. Ist der Name eindeutig und ist genügend Systemspeicher (für ITC-Listen) verfügbar, so nimmt das System die Task unter den ITC-Namen in die Teilnehmerliste auf. Ist die Task der erste Teilnehmer an der ITC, so wird mit dem Aufruf **OPCOM** zusätzlich die Eröffnung des ITC-Verfahrens ausgelöst (das System richtet eine Kommunikationstabelle ein). **OPCOM** wird abgewiesen, wenn für die Listen und Tabellen nicht genügend Systemspeicher zur Verfügung steht.

Sobald der **OPCOM**-Aufruf vom System angenommen ist, kann der Teilnehmer an andere ITC-Teilnehmer, deren ITC-Namen er kennt, Nachrichten senden oder sie von ihnen empfangen, sofern sie seinen ITC-Namen kennen.

Mit dem Makro **CLCOM** beendet der Anwender seine Teilnahme an der Intertaskkommunikation. Gibt er dabei den Operanden NOKEEP an, dann wird seine Teilnahme vollständig beendet, und er kann weder Nachrichten senden noch welche empfangen. Das System löscht alle evtl. noch vorhandenen Nachrichten in der Empfangswarteschlange und streicht seinen ITC-Namen aus der Teilnehmerliste.

Ruft der Anwender den **CLCOM** mit dem Operanden KEEP auf, dann wird nur seine Teilnahme als Empfänger beendet. Das System lässt keine Nachrichten an ihn zu, aber er kann seine Empfangswarteschlange noch auswerten und auch selbst Nachrichten absenden. Ist die Empfangswarteschlange jedoch leer, wenn der Aufruf **CLCOM** KEEP gegeben wird, dann verfährt das System wie bei dem Aufruf **CLCOM** NOKEEP.

Es wird empfohlen, die ITC-Teilnahme mit der Aufruffolge **CLCOM** KEEP oder **CLCOM** NOKEEP zu beenden, um einen Nachrichtenverlust zu vermeiden. Nachrichten, die zwischen den Aufrufen **REVNT** und **CLCOM** eintreffen, gehen sonst verloren.

Wird das Programm oder die Task beendet, ohne dass **CLCOM** aufgerufen wurde, so beendet das System die ITC-Teilnahme mit einem systeminternen Aufruf **CLCOM** NOKEEP. Dabei können Nachrichten verloren gehen. Um das zu vermeiden, kann man in STXIT-Routinen (siehe Seite 133) die Empfangswarteschlange noch auswerten. Dort gibt man den Aufruf **CLCOM** KEEP, prüft die vorliegenden Nachrichten, sendet evtl. eine entsprechende Quittung zurück und schließt mit **CLCOM** NOKEEP. Damit kommt man der systeminternen Programmbeendigung zuvor (einschließlich **CLCOM** NOKEEP).

Die Empfangswarteschlange

Für jeden Teilnehmer an der ITC richtet das System im Systemspeicher eine Empfangswarteschlange ein. Alle Nachrichten an einen Teilnehmer werden in der Reihenfolge ihres Eintreffens in die Empfangswarteschlange eingereiht. Ausgewertet und gelöscht werden die Nachrichten in der Warteschlange von dem Teilnehmer.

Fordert ein Teilnehmer eine Nachricht an (**REVNT**), so übergibt ihm das System diejenige, die als erste Nachricht eingetroffen ist und die auch als erste Nachricht in der Warteschlange steht (FIFO-Prinzip). Wünscht der Teilnehmer eine Nachricht von einem bestimmten Sender, dann bekommt er die erste Nachricht, die dieser Sender geschickt hat. Solange eine Nachricht als erste Nachricht in der Warteschlange steht, übergibt das System immer nur diese. Will der Teilnehmer eine zweite Nachricht anfordern, so muss er die erste Nachricht vorher löschen. Bereits beim Anfordern einer Nachricht (**REVNT**) kann festgelegt werden, dass das System sie nach der Übergabe aus der Warteschlange löscht. Das kann - je nach Anforderung - die erste Nachricht in der Warteschlange oder die erste Nachricht eines bestimmten Senders sein. Explizit kann die erste Nachricht eines bestimmten Senders nicht gelöscht werden, sondern nur zusammen mit der Anforderung (**REVNT**). Die erste Nachricht in der Warteschlange kann jedoch explizit mit dem Makro **RELBF** gelöscht werden. Die ganze Empfangswarteschlange kann der Teilnehmer dadurch löschen, dass er den Makro **RELBF** in einer Schleife aufruft und über den Returncode prüft, ob die Empfangswarteschlange schon leer ist.

Sobald die Teilnahme an der ITC beendet wird (**CLCOM**), nimmt das System keine Nachrichten mehr in die Empfangswarteschlange dieses Teilnehmers auf. Gleichzeitig mit dem Beenden der ITC-Teilnahme kann der Teilnehmer veranlassen, dass das System seine Empfangswarteschlange auflöst. Er kann aber auch die bestehenden Nachrichten noch auswerten und danach die Empfangswarteschlange durch einen weiteren Aufruf **CLCOM** freigeben.

Senden von Nachrichten

Ein Teilnehmer an der ITC kann an jeden anderen ITC-Teilnehmer beliebig oft eine Nachricht schicken (**SEVNT**), solange dafür genug Systemspeicher verfügbar ist. Er muss dazu nur den ITC-Namen des Empfängers kennen. Das System meldet es einem Teilnehmer jedoch nicht, wenn eine Nachricht für ihn eintrifft. Der Teilnehmer muss sie aus eigener Initiative anfordern. Nachdem er eine Nachricht angefordert und erhalten hat, ist es sinnvoll, dass er eine Nachricht als Quittung zurückschickt. Diese kann der Sender zu einem für ihn geeigneten Zeitpunkt mit **REVNT** anfordern (z.B. nachdem alle Nachrichten abgeschickt sind).

Der sendende Teilnehmer hält die Nachricht in einem eigenen Programmbereich bereit. Die Nachricht kann bis zu 64 KByte lang sein. Nach Aufruf des Makros **SEVNT** prüft das System, ob der ITC-Name des Empfängers syntaktisch richtig ist, der Name in der Teilnehmerliste steht und in der Empfangswarteschlange nicht bereits Nachrichten mit einer Gesamtlänge > 128 KByte stehen. Verläuft die Prüfung positiv, dann überträgt das System die Nachricht in die Empfangswarteschlange des Empfängers. Die Task wird nach dem Aufruf **SEVNT** ohne Warten fortgesetzt und kann weitere Nachrichten an beliebige Empfänger absenden.

Anfordern und Empfangen von Nachrichten

Jeder ITC-Teilnehmer kann mit dem Makro **REVNT** eine Nachricht anfordern und wenn sie noch nicht da ist - auf sie warten. Dabei kann er festlegen, ob sie von einem bestimmten Absender kommen soll, oder ob er eine Nachricht von einem beliebigen Teilnehmer annimmt. Wenn die Empfangswarteschlange leer ist oder wenn sie keine Nachricht von dem gewünschten Absender enthält, unterbricht das System die Task, bis die Nachricht eintrifft oder längstens bis die Wartezeit abgelaufen ist (siehe Bild 9 auf Seite 80). Die Dauer der Wartezeit kann im **REVNT**-Aufruf zwischen einer Sekunde und 6 Stunden festlegt werden. Sobald die Task fortgesetzt wird, kann mit dem **REVNT**-Returncode überprüft werden, ob die Nachricht eingetroffen ist.

Die Anforderung einer Nachricht (Makro **REVNT**) kann mit der Ereignissteuerung gekoppelt werden. Das Eintreffen der Nachricht stellt für die Ereignissteuerung ein ITC-Ereignis dar (siehe folgenden Abschnitt).

Das System überträgt die Nachricht aus der Empfangswarteschlange in einen Programmbereich des Teilnehmers, zusammen mit dem ITC-Namen des Absenders und der Länge. Ist das Empfangsfeld nicht groß genug, um die vollständige Nachricht aufzunehmen, so überträgt das System nur ihren Kopf, der aus Absender, Länge der vollständigen Nachricht und den ersten 4 Byte der Nachricht besteht.

Fordert ein Teilnehmer mit **REVNT** eine Nachricht an, so kann er festlegen, dass das System sie nach der Übertragung aus der Empfangswarteschlange löschen soll (Operand REL=YES). Er kann die Nachricht auch bestehen lassen (REL=NO), um sie evtl. noch einmal anzufordern. Solange die erste Nachricht in der Warteschlange nicht gelöscht ist, wird immer diese übertragen. Um die folgende Nachricht übertragen zu lassen, muss der Teilnehmer die erste Nachricht löschen (implizit bei **REVNT** oder explizit mit **RELBF**).

Empfängt ein Teilnehmer Nachrichten verschiedener Länge, so ist es unter Umständen nicht sinnvoll, ein Empfangsfeld in voller Länge (64K + 7 für Sendername und Nachricht) bereitzuhalten. Es wird nur der Kopf der Nachricht übergeben (Sendername, Satzlängenfeld und 4 Byte Information), wenn **REVNT** mit der Länge 16 (und REL=NO) aufgerufen wird. Der Teilnehmer kann dann entscheiden, ob er die Nachricht notfalls löscht, oder er kann Speicherplatz anfordern (**REQM**), um sein Empfangsfeld zu vergrößern. Anschließend kann er die Nachricht wieder anfordern, diesmal in voller Länge.

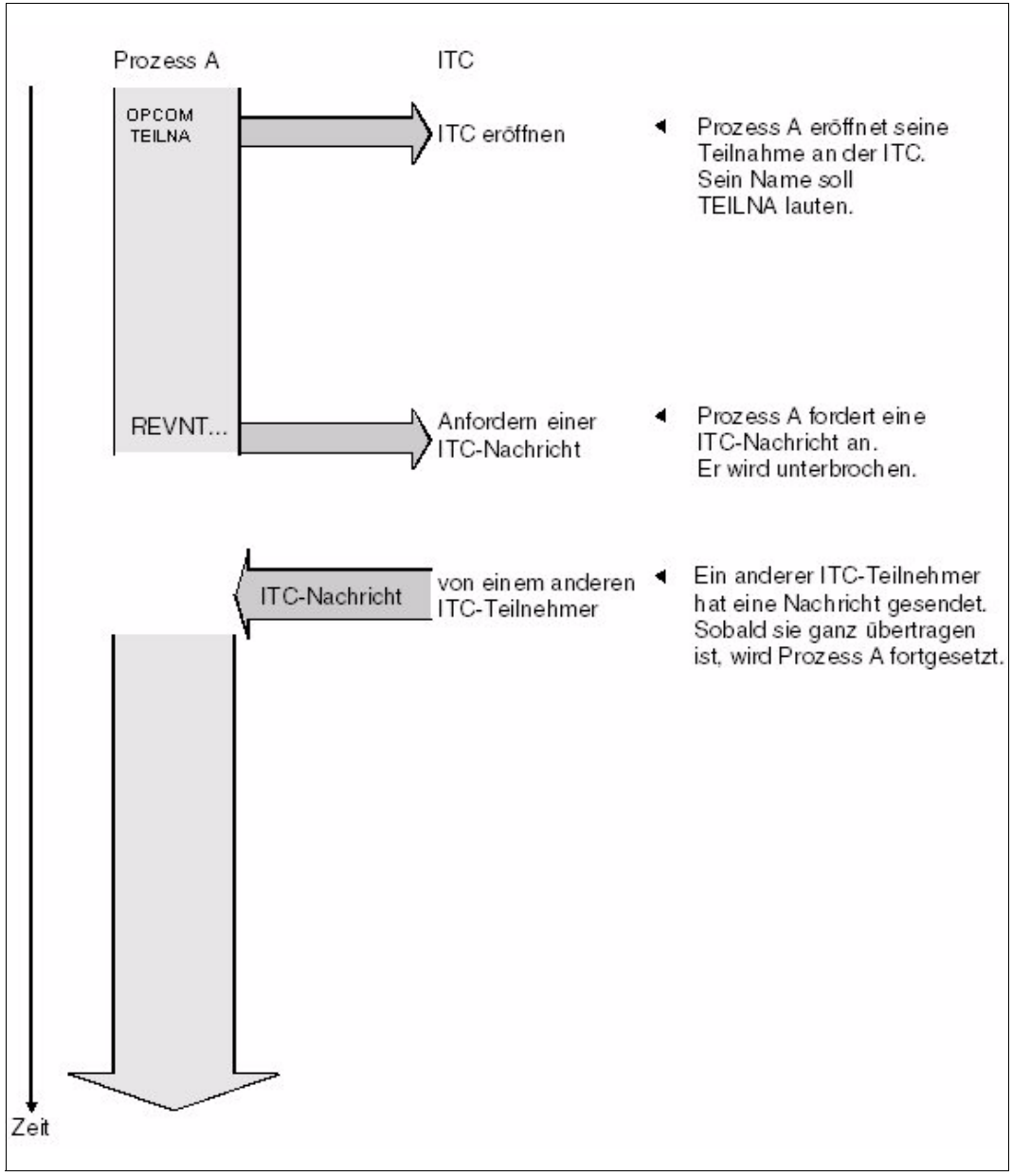


Bild 9: Intertaskkommunikation ITC: Programm, das eine Nachricht anfordert

Koppelung von ITC und Ereignissteuerung

In diesem Abschnitt wird die Kenntnis der Ereignissteuerung vorausgesetzt. Sie ist auf Seite 95 und Seite 111 des vorliegenden Handbuchs beschrieben.

Ereignisklassen:

Das Verfahren der Ereignissteuerung gibt dem Anwender die Möglichkeit, den Ablauf seines Programms unterbrechen zu lassen, bis eines von mehreren verschiedenen Ereignissen eintritt (siehe Seite 95). Die Ereignisse, die die Ereignissteuerung beeinflussen können, sind nach ihrer Herkunft in Klassen eingeteilt (siehe Bild 10 auf Seite 82). Bisher gibt es folgende Ereignisklassen:

Wahlfreie Ereignisse
UPAM-Ereignisse
ITC-Ereignisse
DCAM-Ereignisse
CJC-Ereignisse

In der Klasse der wahlfreien Ereignisse legt der Anwender fest, wann und warum er der Ereignissteuerung ein Ereignis signalisiert (**POSSIG**, siehe Seite 104). Ereignisse aus allen anderen Klassen (ITC, UPAM usw.) haben zwar ihre Ursache in dem Programmablauf, aber das System (nicht das Programm) signalisiert sie der Ereignissteuerung (systeminterner **POSSIG**). Außerdem ist festgelegt, was als Ereignis gilt:
Ein UPAM-Ereignis z.B. tritt ein, wenn eine Ein-/Ausgabeoperation abgeschlossen ist.
Als ITC-Ereignis gilt das Eintreffen einer ITC-Nachricht (oder Ablauf der Wartezeit).

Ein Programm, das von der Ereignissteuerung ein Signal anfordert (**SOLSIG**), kann auch dasjenige sein, in dem das erwartete Ereignis eintritt. Zum Beispiel wird mit einem PAM-Aufruf eine Ein-/Ausgabeoperation eingeleitet und dann ein Ereignis-Signal angefordert (**SOLSIG**). Die Ereignissteuerung unterbricht daraufhin (synchroner Fall) den Programmlauf und setzt ihn fort, sobald ihr ein systeminterner **POSSIG** anzeigt, dass die Ein-/Ausgabeoperation abgeschlossen ist.

Ein Teilnehmer kann die Signalanforderung nicht auf eine bestimmte Ereignisklasse beschränken. Das Signal, das er auf seine Anforderung erhält, kann sich auf jedes Ereignis beziehen, das im Geltungsbereich der Ereignissteuerung eintreten kann. Die Ereignissteuerung informiert den Teilnehmer durch den Post Code (siehe Seite 97) über die Klasse und die näheren Umstände des Ereignisses.

Der Programmablauf kann also von verschiedenen Ereignissen abhängig gemacht werden. Das ist sinnvoll, wenn mehrere Ereignisse erwartet werden, man aber nicht weiß, welches zuerst eintreten wird (z.B. ITC-Nachricht oder Lesen eines PAM-Blockes beendet).

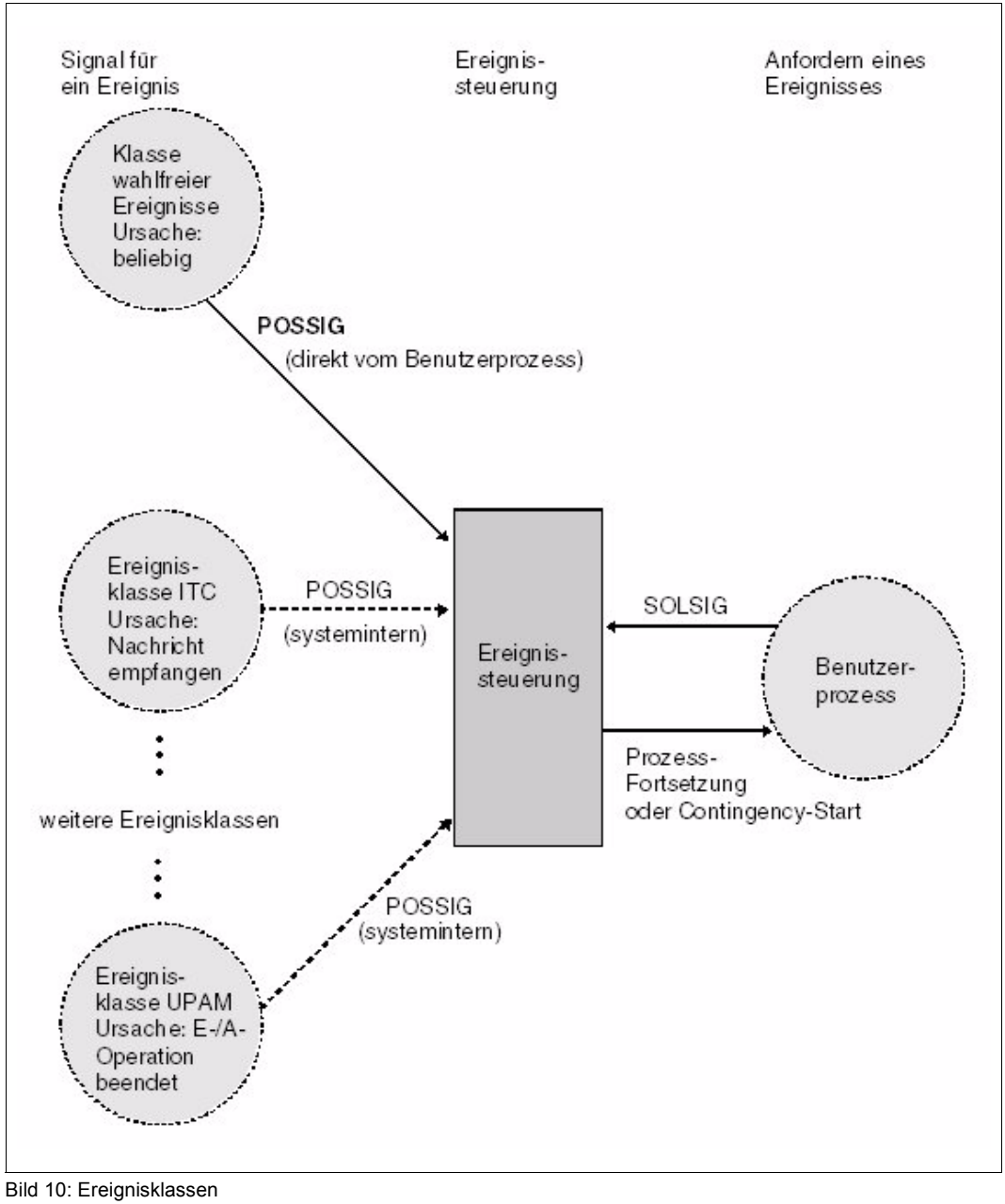


Bild 10: Ereignisklassen

ITC-Nachricht als Ereignis der ITC-Klasse

Die Verbindung von ITC und Ereignissteuerung bedeutet, dass ein (ITC-)Ereignis auftritt, wenn eine ITC-Nachricht eintrifft oder wenn die Wartezeit abgelaufen ist. Die Koppelung entsteht dadurch, dass im Aufruf **REVNT** die Kurzkennungsadresse einer Ereigniskennung angegeben wird. Der Programmablauf wird dann durch den **REVNT**-Aufruf nicht unterbrochen. Der Sender der Nachricht braucht der Ereignissteuerung nicht anzugehören.

Die Koppelung von ITC und Ereignissteuerung gestaltet den Programmablauf flexibler:

- Der Teilnehmer kann das Warten auf eine ITC-Nachricht mit dem Warten auf ein anderes Ereignis kombinieren.

  Beispiel

  Der Teilnehmer gibt einen **PAM**- (siehe Handbuch „DVS Makros“ [7]) und einen **REVNT**-Aufruf, beide gekoppelt mit der Ereignissteuerung. Er gibt dann einen **SOLSIG**-Aufruf und wird unterbrochen. Nach seiner Fortsetzung prüft er den Post Code, ob die Nachricht eingetroffen war oder ob die Ein-/Ausgabeoperation beendet war.

- Der Teilnehmer kann seine Warte-Unterbrechung auf einen späteren Zeitpunkt verschieben: Die Unterbrechung tritt nicht nach dem **REVNT**-Aufruf, sondern nach dem **SOLSIG**-Aufruf ein (siehe Bild 11 auf Seite 87 und Bild 12 auf Seite 88).

  Beispiel

  Der Teilnehmer fordert mit einem gekoppelten **REVNT**-Aufruf eine ITC-Nachricht an und setzt seine Verarbeitung fort. Später gibt er einen **SOLSIG**-Aufruf, um „nachzuschauen“, ob die Nachricht da ist. Ist sie immer noch nicht da, so setzt erst jetzt die Unterbrechung ein.

- Der Teilnehmer braucht gar keine Warte-Unterbrechung in Kauf zu nehmen, wenn er im **SOLSIG**-Aufruf eine Contingency-Routine angibt (siehe Bild 13 auf Seite 89). Bei diesem asynchronen Fall der Ereignissteuerung kann der **SOLSIG**-Aufruf auch vor dem **REVNT**-Aufruf folgen.

  Beispiel

  In einem Programm ist eine Contingency-Routine definiert, die die ITC-Nachricht auswertet und alle davon abhängigen Aktionen durchführt. Diese Contingency-Routine wird im **SOLSIG**-Aufruf angegeben. Vor oder nach nach dem **SOLSIG**-Aufruf wird die Nachricht mit einem **REVNT**-Aufruf angefordert. Der Programmablauf wird weder durch den **REVNT** noch durch den **SOLSIG** unterbrochen (um zu warten), sondern nur durch den Ablauf der Contingency-Routine. Deren Start erfolgt, sobald ein Ereignis eintrifft.

Mit dem **SOLSIG**-Aufruf kann ein Teilnehmer kein bestimmtes Ereignis anfordern. Die Ereignissteuerung ordnet ihm das erste Ereignis aus ihrem Geltungsbereich zu, das noch nicht durch andere (frühere) **SOLSIG**-Aufrufe angefordert wurde. Der Teilnehmer muss mit dem Post Code die Ereignisse herausfiltern, die für ihn maßgebend sind.
Der Geltungsbereich kann eine Task, alle Tasks einer Benutzerkennung, alle Tasks einer Benutzergruppe oder alle Tasks im System umfassen. Er wird bei der Eröffnung der Ereigniskennung festgelegt.
Mehrere Ereigniskennungen (mit verschiedenen Geltungsbereichen) können gleichzeitig bestehen, und eine Task kann sich gleichzeitig verschiedenen Ereigniskennungen anschließen (siehe Seite 95).

Programmaufbau für gekoppelte Anwendung von ITC und Ereignissteuerung ohne Contingency-Prozess (siehe Bild 11 auf Seite 87 und Bild 12 auf Seite 88).

– Sowohl die Teilnahme an der ITC als auch an der Ereignissteuerung müssen erklärt werden (**OPCOM** und **ENAEI**). Unter der im **ENAEI** angegeben Adresse trägt das System die Kurzkennung der Ereignissteuerung ein.
  Ist die Ereignissteuerung schon eröffnet, dann darf sich der Teilnehmer nur anschließen, wenn er dem festgelegten Geltungsbereich angehört. Rufen auch andere Teilnehmer des Geltungsbereiches den Makro **SOLSIG** auf, dann ist nicht vorauszusehen, welchem Teilnehmer die Ereignissteuerung ein Ereignis zuordnet.

– Eine Nachricht wird mit dem Makro **REVNT** angefordert. Zusätzlich zu den bekannten Operanden wird mit dem Operanden EIID die Kurzkennungsadresse angegeben, die im **ENAEI**-Aufruf verwendet worden war.

– Der Programmablauf ist nicht unterbrochen. Das Programm muss den **REVNT**-Returncode in R15 prüfen, ob der Aufruf akzeptiert ist oder wegen formaler Fehler abgewiesen wurde. Ein abgewiesener **REVNT**-Aufruf führt zu keinem ITC-Ereignis. Der Returncode für einen gekoppelten **REVNT** enthält noch keine Angaben über die Nachricht, sondern erst der Post Code.
  Solange ein gekoppelter **REVNT**-Aufruf nicht abgeschlossen ist (d.h. Nachricht eingetroffen oder Wartezeit abgelaufen), darf kein weiterer **REVNT** gekoppelt oder ungekoppelt - aufgerufen werden. Andere Aufrufe, die sich an die Ereignissteuerung koppeln, z.B. ein PAM-Aufruf, sind jedoch erlaubt.

– Das Programm ruft zu einem Zeitpunkt, der für seinen Ablauf sinnvoll ist, den Makro **SOLSIG** auf und gibt dabei eine Adresse für den Post Code an. Der Programmlauf wird nicht unterbrochen, wenn die Signalanforderung sofort befriedigt wird. Ansonsten wird im synchronen Fall der Programmlauf unterbrochen - längstens bis Ablauf der angegebenen Wartezeit.

– Nach seiner Fortsetzung prüft das Programm zuerst den Returncode des **SOLSIG**-Aufrufs, ob er keine formalen Fehler anzeigt oder ob die SOLSIG-Wartezeit abgelaufen ist. (Der Ablauf der REVNT-Wartezeit wird im Post Code angegeben.)

– Der Post Code wird geprüft. Das System hat ihn unter der Adresse gespeichert, die im **SOLSIG**-Aufruf angegeben war (ein Contingency-Prozess erhält ihn in Register R3). Das linksbündige Byte des Post Codes gibt an, zu welcher Klasse das eingetretene Ereignis gehört (ITC-Ereignis: X'08'). Das Programm geht zu dem Verarbeitungsteil über, der für die Ereignisklasse vorgesehen ist.

– Verarbeitung eines ITC-Ereignisses: Das rechtsbündige Byte des Post Codes wird geprüft. Es gibt an, ob die Wartezeit des **REVNT** abgelaufen war oder ob die Nachricht bzw. nur der Nachrichtenkopf übertragen wurde. Ab jetzt können wieder **REVNT**-Aufrufe gegeben werden.

  Bei vielen ITC-Anwendungen ist es wahrscheinlich, dass einer Nachricht sofort noch weitere folgen. Dann empfiehlt es sich, die weiteren **REVNT**-Aufrufe nicht mehr an die Ereignissteuerung zu koppeln. Die nötigen **SOLSIG**-Aufrufe sind zeitaufwendig, und sie bieten keinen Vorteil, wenn die Nachricht bereits da ist. Stattdessen gibt das Programm solange ungekoppelte **REVNT**-Aufrufe mit WTIME=0, bis alle anstehenden Nachrichten empfangen sind. Danach ist ein gekoppelter **REVNT**-Aufruf wieder sinnvoll, um weitere Nachrichten zu erwarten.

– Das Programm beendet die Teilnahme an der ITC und an der Ereignissteuerung mit **CLCOM** und **DISEI** (siehe Hinweis).

Hinweis zum Beenden der ereignisgesteuerten ITC

Solange für einen gekoppelten **REVNT**-Aufruf kein Ereignis signalisiert wurde (d.h. Nachricht da oder Wartezeit abgelaufen), darf der Teilnehmer die zugehörige Ereignissteuerung nicht beenden. Bei asynchroner Ereignissteuerung wird sonst der Contingency-Prozess auf Grund des **DISEI**-Aufrufs gestartet, im synchronen Fall kann kein Ereignissignal mehr empfangen werden, auch wenn die Nachricht übertragen wurde. Folgende Vorgehensweise wird empfohlen:
Der Teilnehmer muss intern kontrollieren, ob der letzte gekoppelte **REVNT** abschlossen ist (z.B. mit einem Zähler; bei lokalem Geltungsbereich mit **CHKEI**). Danach kann er den **DISEI**-Aufruf geben. Mit dem Aufruf **CLCOM** KEEP unterbindet er den Empfang weiterer Nachrichten. Diejenigen, die noch in seiner Nachrichtenwarteschlange eingetragen sind, holt er mit ungekoppelten **REVNT**-Aufrufen ab. Danach kann die Teilnahme an der ITC mit dem Aufruf **CLCOM** NOKEEP beendet werden.

Hinweis zu Wartezeiten

Sowohl der **REVNT**-Aufruf als auch der **SOLSIG**-Aufruf ermöglichen die Angabe einer Wartezeit.
Die REVNT-Wartezeit begrenzt die Lebensdauer der Nachrichtenanforderung. Läuft sie ab, ohne dass eine Nachricht eintraf, so führt das zu einem ITC-Ereignis. Das System gibt intern einen **POSSIG**-Aufruf an die Ereignissteuerung, die dem Teilnehmer ein ITC-Ereignis signalisiert. Das rechte Byte des Post Codes enthält bei Ablauf der REVNT-Wartezeit den Wert X'10'.

Die SOLSIG-Wartezeit begrenzt die Lebensdauer der Signalanforderung. Läuft sie ab, ohne dass die Ereignissteuerung ein Ereignis signalisiert, so wird das unterbrochene Programm fortgesetzt (synchroner Fall) oder die Contingency-Routine wird gestartet (asynchroner Fall). Der Returncode des **SOLSIG**-Aufrufs hat bei Ablauf der SOLSIG-Wartezeit den Wert X'20000004'. (Ist die REVNT-Wartezeit abgelaufen, so gilt das als ITC-Ereignis. Der **SOLSIG**-Returncode ist dann X'00000000').

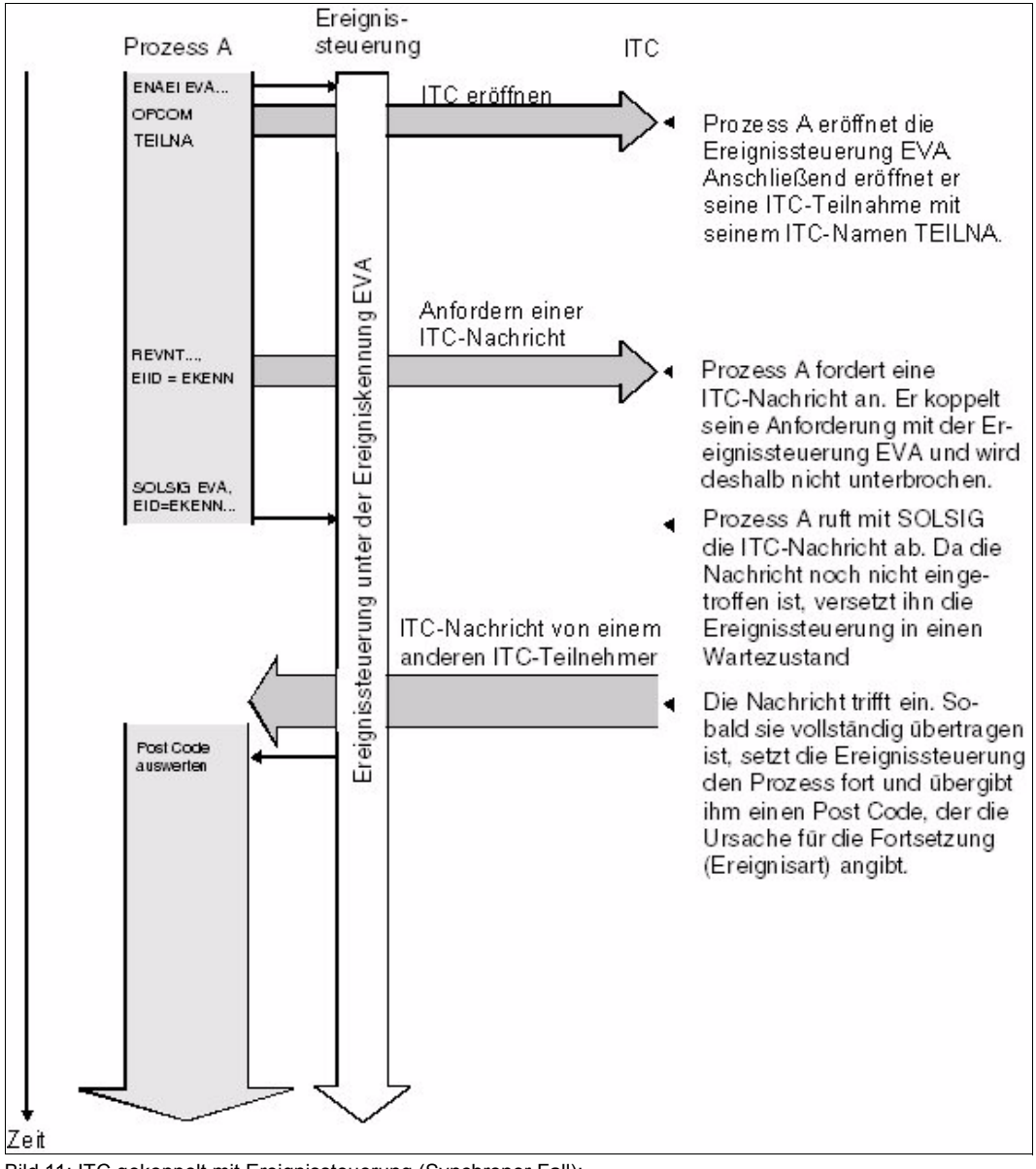


Bild 11: ITC gekoppelt mit Ereignissteuerung (Synchroner Fall):
Die angeforderte Nachricht trifft nach dem SOLSIG-Aufruf ein

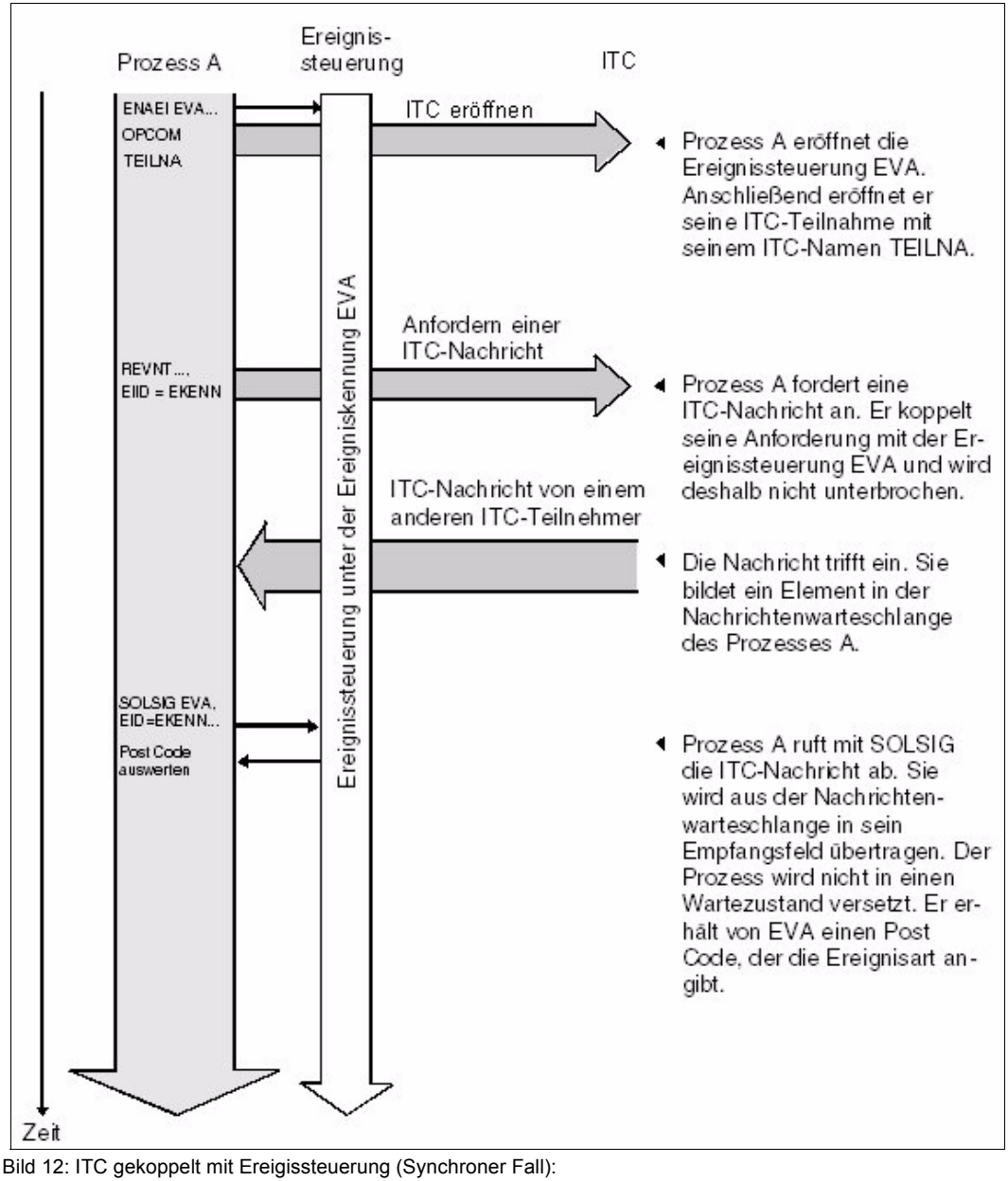


Bild 12: ITC gekoppelt mit Ereigissteuerung (Synchroner Fall):
Die angeforderte Nachricht trifft vor dem SOLSIG-Aufruf ein

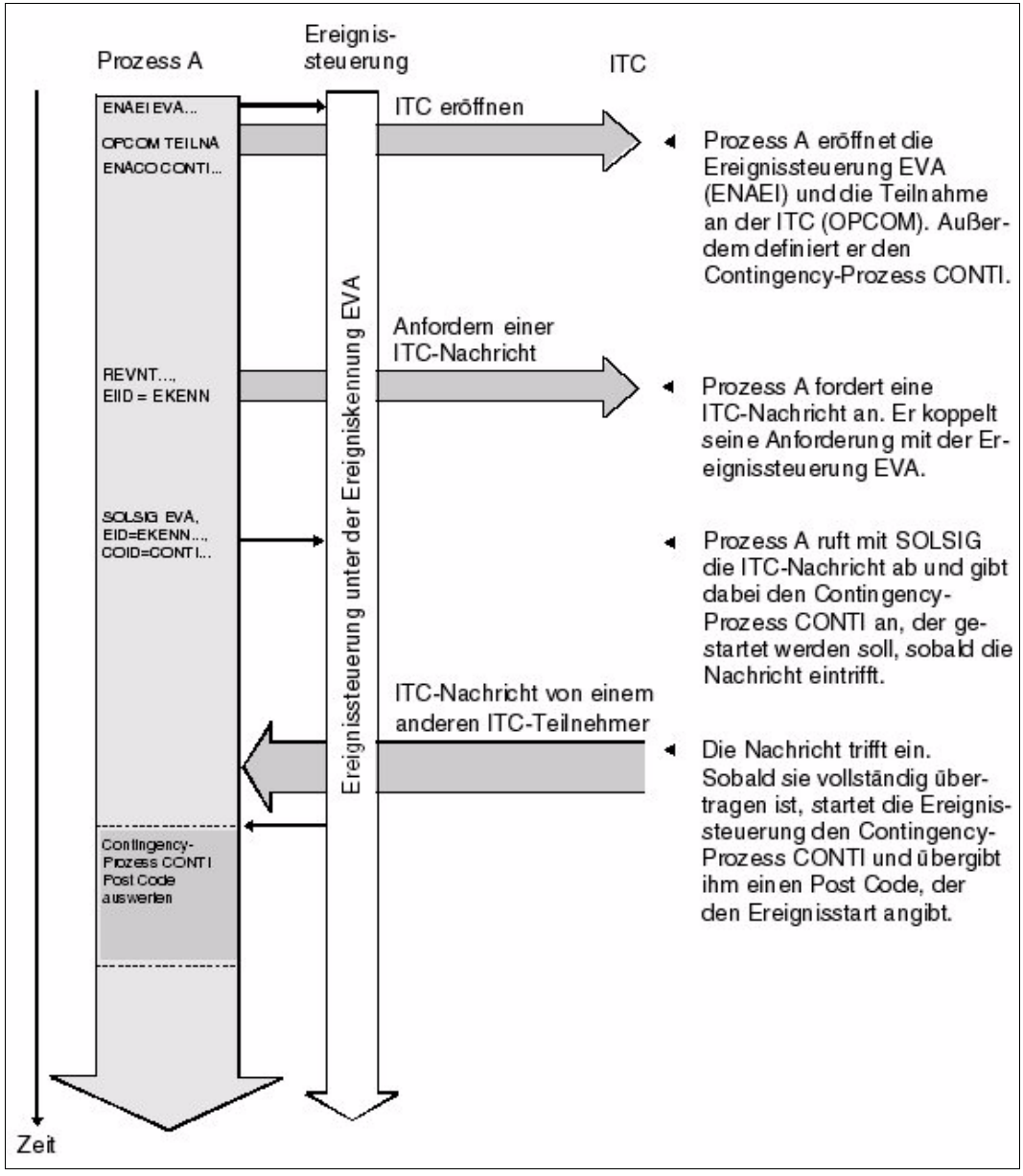


Bild 13: ITC gekoppelt mit Ereignissteuerung (Asynchroner Fall):
Ein Contingency-Prozess läuft ab, wenn die Nachricht eintrifft

Beispiel

Programmaufbau für gekoppelte Anwendung von ITC und Ereignissteuerung mit einem Contingency-Prozess (siehe Bild 13 auf Seite 89)

```
BEISPIEL START
         BALR   6,0
         USING  *,6
         :
         :
         ENAEI  EINAME=EVA,EIIDRET=EKENN ------------   (1)
         ENACO  CONAME=CONTEVA,COADAD=CONTADR,
                COIDRET=CONTI
         OPCOM  TEILNA
         :
         :
         SOLSIG EIID=EKENN,COID=CONTI,LIFETIM=3600 --   (2)
         C      15, RCOO
         BNE    SOLSFEHL
         :
         :
         REVNT  EMPFANG,100,WTIME=600,EIID=EKENN ----   (3)
         C      15,RCOO
         BNE    REVFEHL
         :
         :
         CLCOM ------------------_________----------    (4)
         DISCO  COID=CONTI
         DISEI  EIID=EKENN
         TERM

CONTANF  BALR   5,0 ---------------------------------   (5)
         USING  *,5
         ST     3,POSTCODE
         :                      Prüfung des Post Codes auf
         :                      verschiedene Ereignisklassen
         CLI    POSTCODE,ITCEREIG
         BE     ITCBEARB
         :                      Bearbeitung von Ereignissen
         :                      anderer Klassen
         :
ITCBEARB :                      Prüfung des Post Codes
         :                      auf Nachrichtenempfang
         :                      Auswertung der Nachricht
         RETCO
```

Programmteil mit ITC-Verarbeitung (Basisprozess)

Contingency-Routine (Contingency-Prozess)

```
REVFEHL  (Behandlung der Fehler im REVNT-Aufruf)
         :
         B      ....
         :
SOLSFEHL (Behandlung der Fehler im SOLSIG-Aufruf)
         :
         B      ....
         :
EKENN    DS     F
CONTADR  DC     A(CONTANF)
CONTI    DS     F
EMPFANG  DS     CL100
RC00     DC     A(0)
POSTCODE DC     A(0)
ITCEREIG EQU    X'08'
         END
```

Fehlerroutinen

Daten

(1) Die Teilnahme an der Ereignissteuerung wird erklärt. Name der Ereigniskennung ist EVA. Die Adresse für die Kurzkennung ist ERKENN. Der Geltungsbereich ist lokal. Eine Contingency-Routine wird definiert. Die Kurzkennung hat die Adresse CONTI. Die Teilnahme an der ITC wird erklärt. ITC-Name ist TEILNA.

(2) Ein Ereignis-Signal wird mit **SOLSIG** angefordert. Im Aufruf wird eine Contingency-Routine angegeben. Diese wird gestartet, wenn eine Nachricht eintrifft oder spätestens, wenn die Wartezeit abgelaufen ist. Der **SOLSIG**-Aufruf kann auch nach dem **REVNT**-Aufruf erfolgen.
Der Basisprozess läuft weiter (sofern die Nachricht nicht schon vorlag und der Contingency-Prozess ihn bereits unterbrochen hat). Der **SOLSIG**-Returncode wird geprüft. Wenn er ungleich Null ist, hat das System den Aufruf nicht akzeptiert. Die Routine SOLSFEHL prüft die Ursache.

(3) Eine ITC-Nachricht wird angefordert. Die Angabe EIID=EKENN koppelt diese Anforderung an die Ereignissteuerung EVA. Der Programmablauf wird nicht unterbrochen. Vor oder nach diesem **REVNT**-Aufruf sind Aufrufe für andere Ereignisklassen möglich (z.B. PAM).
Ist der Returncode in R15 ungleich Null, so wurde der Aufruf nicht akzeptiert. Die Routine REVFEHL untersucht die Ursache.

(4) Der Teilnehmer will die gekoppelte ITC-Verarbeitung beenden. Mit **CLCOM** beendet er die ITC-Teilnahme, mit **DISCO** löscht er die Contingency-Definition und mit **DISEI** beendet er die Ereignissteuerung EVA (zur Vermeidung von Nachrichtenverlust siehe Hinweis vor dem Beispiel).

(5) Die Contingency-Routine speichert den Post Code ab, der in Register R3 übergeben wird und wertet dessen linksbündiges Byte aus. Dieses Byte gibt an, zu welcher Klasse das Ereignis gehört, das den Start der Contingency-Routine ausgelöst hat. Im Verarbeitungsteil für die entsprechende Ereignisklasse wird in der Contingency-Routine das rechtsbündige Byte des Post Codes ausgewertet. Es enthält den klassenspezifischen Returncode. Bei ITC-Ereignissen sind es Angaben, ob die Wartezeit abgelaufen ist oder die Nachricht übertragen werden konnte (siehe **REVNT**). Bei abgelaufener Wartezeit kann z.B. noch einmal ein gekoppelter **REVNT** aufgerufen werden (Rücksprung zum ersten **REVNT**-Aufruf ist nicht möglich!). Der Makro **RETCO** beendet die Contingency-Routine.

4.3.4 (Task-)Serialisierung

Makro	Kurzbeschreibung
CHKSI	prüft die Warteschlangenbelegung einer Serialisierungskennung
DEQAR	hebt die (exklusive) Belegung einer Serialisierungskennung durch die Task des Aufrufers auf
DISSI	löst die Zuordnung der Task zu der Serialisierungskennung. Die Serialisierungskennung wird gelöscht, wenn sie von keiner weiteren Task benutzt wird
ENASI	ordnet die Task einer Serialisierungskennung zu. Die Serialisierungskennung wird eingerichtet, wenn sie nicht schon von einer anderen Task benutzt wird
ENQAR	fordert die (exklusive) Belegung einer Serialisierungskennung. Die Anforderung wird in eine Warteschlange eingetragen

Einleitung

Dem Anwender wird ein Verfahren (semaphore-type mechanism) zur Verfügung gestellt, das serielle Zugriffe zu bestimmten Serialisierungskennungen ermöglicht.

Serialisierungskennung

Eine Serialiserungskennung ist mit keiner speziellen „Eigenschaft“ assoziiert. Es ist Aufgabe des Anwenders, eine Serialisierungskennung mit einer bestimmten „Eigenschaft“ mit einem bestimmten „Dasein“ zu verknüpfen.

Die Serialisierungskennung wird identifiziert mittels eines Namens oder mittels einer Kurzkennung, die dem Anwender zur Verfügung gestellt wird und in weiteren Aufrufen zur Beschleunigung der Verarbeitung verwendet werden kann.

Geltungsbereich einer Serialisierungskennung (SCOPE-Operand)

Der Operand SCOPE definiert den Geltungsbereich (Teilnehmerkreis) einer Serialisierungskennung. Der Geltungsbereich kann eine Task, alle Tasks einer Benutzerkennung oder alle Tasks im System umfassen.

Verwendung einer Serialisierungskennung ermöglichen

Eine Serialisierungskennung muss der Task zugeordnet sein, bevor eine (exclusive) Belegungsanforderung (**ENQAR**) für die Serialisierungskennung angenommen wird.
Eine Serialisierungskennung wird der Task mittels eines **ENASI**-Makroaufrufs (ENAble Serialization Item) explizit oder mittels eines **ENQAR**-Makroaufrufs (mit Namen) implizit zugeordnet. Wenn eine Serialisierungskennung mit dem angegebenen Namen in dem definierten Geltungsbereich bereits besteht (eingerichtet durch einen **ENASI**-Aufruf in einer anderen Task), bewirkt der Aufruf nur eine Zuordnung zwischen der Task mit dem aufrufenden Programm und der Serialisierungskennung. Andernfalls wird die Serialisierungskennung vom System erstellt und zugeordnet. Nur bei Verwendung des **ENASI**-Aufrufs wird eine Kurzkennung für die Serialisierungskennung zur Verfügung gestellt.

Einer Task können maximal 2000 Serialisierungskennungen gleichzeitig zugeordnet sein.

Zuordnung zu einer Serialisierungskennung aufheben

Die Zuordnung zu einer Serialisierungskennung wird aufgehoben, entweder explizit mittels eines **DISSI**-Aufrufs (DISable Serialization Item) oder implizit mittels eines **DEQAR**-Aufrufs (DEQueue Access Request) mit dem Operanden DISSI. Verwendet keine weitere Task diese Serialisierungskennung, so wird sie gelöscht.

Ein **ENASI**-Aufruf für eine Serialisierungskennung, deren Verwendung zuvor bereits beendet wurde, führt nicht unbedingt zu derselben Kurzkennung wie wie im vorhergehenden **ENASI**-Aufruf.

Programmbeendigung

Mit Programmbeendigung wird die Zuordnung der Task zu allen verwendeten Serialisierungskennungen aufgehoben.

Belegungsanforderung für eine Serialisierungskennung (ENQAR-Makro)

Der Makro **ENQAR** (ENQueue Access Request) fordert die (exclusive) Belegung der angegebenen Serialisierungskennung. Die Anforderung wird in die Warteschlange dieser Serialisierungskennung eingetragen und die Task solange in den Wartezustand versetzt, bis sie die ersteTask in dieser Warteschlange ist. Danach läuft die Task weiter und belegt die Serialisierungskennung solange, bis ein **DEQAR**-Aufruf (DEQueue Access Request) für diese Serialisierungskennung gegeben wird.

Ist keine Serialisierungskennung mit dem angegebenen Namen im definierten Geltungsbereich vorhanden, so wird sie eingerichtet und zugeordnet (implizite Enable-Funktion).

Durch den COND-Operanden bestimmt der Anwender, ob die Belegungsanforderung unmittelbar befriedigt werden soll oder ob eine Wartezeit akzeptiert wird. Der LIFETIM-Operand spezifiziert die Wartezeit.

Belegung einer Serialisierungskennung beenden (DEQAR-Makro)

Der Makro **DEQAR** (DEQueue Access Request) beendet die Belegung der Serialisierungskennung durch die Task.

Die Beendigung kann in der Task erfolgen, die die Belegung angefordert hat, oder in jeder anderen Task, die dieser Serialisierungskennung auch zugeordnet ist. (Operand HOLDER=ANY im **DEQAR**-Aufruf).

Wahlweise (DISSI-Operand) kann zugleich auch die Zuordnung der Task zu der Serialisierungskennung aufgehoben werden (implizite Disable Funktion).

Prüfen der Serialisierungskennung (CHKSI-Makro)

Der Makro **CHKSI** (CHeck Serialisation Item) informiert über Verfügbarkeit und Belegung (Task des Aufrufers oder andere Tasks) der angegebenen Serialisierungskennung. Die Information wird als Returncode (Register R15) übergeben.

4.3.5 Ereignisgesteuerte Verarbeitung (Eventing)

Makro	Kurzbeschreibung
CHKEI	prüft die Warteschlangenbelegung für eine Ereigniskennung
DELFEI	löscht einen SOLSIG-/POSSIG-Eintrag in der EVENTLST. (Optimiertes Eventing, Forward Eventing)
DISEI	beendet die Teilnahme an der Ereignissteuerung. Die angegebene Ereigniskennung wird gelöscht, wenn sie von keiner weiteren Task benutzt wird
DPOFEI	erzeugt einen POSSIG-Eintrag in der EVENTLST. (Optimiertes Eventing, Forward Eventing)
DSOFEI	erzeugt einen SOLSIG-Eintrag in der EVENTLST. (Optimiertes Eventing, Forward Eventing)
ENAEI	ermöglicht die Teilnahme an der Ereignissteuerung. Die Task wird der angegebenen Ereigniskennung zugeordnet
POSSIG	sendet ein Signal an die Ereignissteuerung
RPOFEI	sendet ein Signal (Ereignis) an die Ereignissteuerung. (Optimiertes Eventing, Forward Eventing)
RSOFEI	fordert von der Ereignissteuerung das Eintreffen eines Signals an. (Optimiertes Eventing, Forward Eventing)
SOLSIG	fordert von der Ereignissteuerung das Eintreffen eines Signals an

Allgemeines

Die Ereignissteuerung (Eventing) ermöglicht die Koordinierung der Abläufe von zwei oder mehreren (Anwender-)Programmen in voneinander verschiedenen Tasks. Zum Beispiel soll ein Programm B erst dann Datensätze aus einer Datei lesen, wenn das Programm A die Datei aktualisiert hat, oder Programm A soll erst dann einen Datensatz in einen gemeinsam benutzten Speicherbereich schreiben, wenn Programm B den vorhergehenden Satz gelesen hat. Die Koordinierung der Programme erfolgt durch Nutzung (Steuerung) einer gemeinsamen Ereignisvariablen und daraus abgeleiteten Aktionen des Betriebssystems (Task in den Wartezustand setzen, Wartezustand beenden oder Contingency-Prozess starten). Die Ereignisvariable wird über die Ereigniskennung angesprochen. Die Gesamtheit - Ereigniskennung und daraus abgeleitete Aktionen - wird als Ereignissteuerung bezeichnet. Nach außen wird die Ereignissteuerung durch die Ereigniskennung repräsentiert.

Teilnehmer an einer Ereignissteuerung sind aus der Sicht des Betriebssystems die Tasks, die sich einer gemeinsamen Ereignisvariablen (Ereigniskennung) zuordnen; aus der Sicht des Anwenders sind es die (Anwender-)Programme, die in diesen Tasks ausgeführt werden.

Grundfunktionen der Ereignissteuerung (Eventing)

- Teilnahme an der Ereignissteuerung erklären
  Programme, deren Abläufe koordiniert werden sollen, müssen die Teilnahme an der Ereignissteuerung erklären und den Namen (und Geltungsbereich) der gemeinsamen Ereignisvariablen (Ereigniskennung) angeben. Die Vereinbarungen über die Teilnahme an der Ereignissteuerung und die Bezeichnung der Ereigniskennung gelten nur für das gerade ausgeführte Programm.
  Für jede Ereigniskennung werden zwei Warteschlangen (POSSIG-/SOLSIG-Warteschlange) eingerichtet. Einer Task können gleichzeitig maximal 2000 Ereigniskennungen zugeordnet sein.
- Signal senden
  Ein Programm, das den Abschluss einer bestimmten Verarbeitung melden will, sendet ein POSSIG-Signal an die Ereignissteuerung. Das Signal wird in die POSSIG-Warteschlange eingereiht (Einreihungsprinzip = FIFO).
- Signal anfordern
  Ein Programm, das Kenntnis über den Abschluss einer bestimmten Verarbeitung in einem anderen Programm benötigt, sendet eine SOLSIG-Anforderung an die Ereignissteuerung. Die Forderung wird in die SOLSIG-Warteschlange eingereiht (Einreihungsprinzip = FIFO/LIFO, je nach Angabe).
- Warteschlange abarbeiten (siehe Bild 14 auf Seite 99)
  Jeweils das erste POSSIG-Signal in der POSSIG-Warteschlange und die erste SOLSIG-Anforderung in der SOLSIG-Warteschlange werden einander zugeordnet - unabhängig davon, aus welchen Tasks sie stammen. Kann die SOLSIG-Anforderung nicht befriedigt werden (d.h. die POSSIG-Warteschlange ist leer) sind zwei Abläufe möglich:
  - synchroner Ablauf (siehe Bild 15 auf Seite 100 und Bild 16 auf Seite 101)
    Die Task mit der SOLSIG-Anforderung wird in den Wartezustand versetzt, bis ein POSSIG-Signal bei der Ereignissteuerung eintrifft oder bis eine spezifizierte Wartezeit abgelaufen ist. (Der Programmlauf wird mit dem POSSIG-Signal synchronisiert).
  - asynchroner Ablauf (siehe Bild 17 auf Seite 102)
    Die Task mit der SOLSIG-Anforderung wird nicht in den Wartezustand versetzt. Kann die SOLSIG-Anforderung im weiteren Programmablauf befriedigt werden oder ist eine für die Befriedigung angegebene Wartezeit abgelaufen, wird eine im Programm spezifizierte Contingency-Routine gestartet.

  Kann die SOLSIG-Anforderung unmittelbar befriedigt werden (die POSSIG-Warteschlange ist nicht leer), wird entweder der Programmlauf mit dem den Makro **SOLSIG** folgenden Befehl fortgesetzt oder eine im **SOLSIG**-Aufruf spezifizierte Contingency-Routine gestartet.

Jeder Teilnehmer kann den Zustand der Warteschlangen prüfen. Er erhält Auskunft, ob POSSIG-Signale in der POSSIG-Warteschlange und ob SOLSIG-Signale in der SOLSIG-Warteschlange stehen.

- Teilnahme an der Ereignissteuerung beenden
  Die Teilnahme an der Ereignissteuerung kann zu jedem beliebigen Zeitpunkt beendet werden. Der Teilnehmer wird aus der Liste der teilnehmenden Tasks gelöscht. Noch anstehende SOLSIG-Anforderungen in der SOLSIG-Warteschlange werden gelöscht. Nicht zugeordnete POSSIG-Signale verbleiben in der POSSIG-Warteschlange.
  Implizit endet die Teilnahme immer mit Programmbeendigung (nicht Taskende!).
  Die Ereigniskennung und alle internen Verarbeitungslisten werden gelöscht, wenn der letzte (einzige) Teilnehmer die Teilnahme beendet.

- Post Code (siehe Bild 18 auf Seite 103)
  Die Ereignissteuerung registriert nur das Eintreffen eines POSSIG-Signals. Es wird keine Information über das mit dem Signal verknüpfte Ereignis (Datei geschlossen, Datensatz geschrieben, ...) mitgeteilt. Da immer die ersten Elemente in den beiden Warteschlangen einander zugeordnet werden, ist eine Adressierung des Empfängers oder Senders nicht möglich. Der Sender des POSSIG-Signals hat aber die Möglichkeit, zusammen mit dem Signal eine Kurznachricht (Post Code) an die Ereignissteuerung zu senden. Diese Nachricht wird der Task übergeben, deren SOLSIG-Anforderung durch dieses POSSIG-Signal befriedigt wird. Der Empfänger der Nachricht kann so entscheiden, ob das POSSIG-Signal mit seinem Programmablauf in Verbindung steht oder nicht.

Der **Post Code** ist 4 oder 8 Byte lang (bei Verwendung der 24-Bit-Schnittstelle nur 4 Byte). Die folgende Tabelle gibt eine Übersicht über wichtige Anwendungen für verschiedene Ereignisklassen:

Anwendung/ Ereignisklasse	**Post Code**	**Bemerkung**
(POSSIG)	X'aa....aa'	aa....aa: vom Anwender anzugebender String; (4 oder 8 Byte)
ITC	X'08000000' X'08000004' X'0800000C' X'08000010'	REVNT-Ereignis abgeschlossen. Operandenfehler Empfangsfeld zu klein Timeout
DCAM	X'0Caa...a'	aa...a: vom Anwender anzugebende Zeichenfolge; (3 oder 7 Byte)
UPAM	X'1000i..a'	31-Bit-Adressierungsmodus i..a = iiiiaaaaaaaa. Es bedeuten: iiii: Identifier, dem Auftrag vom Anwender mitgegeben (2 Byte) a...a: PAM-Datenbereichsadresse (4 Byte) Der Anwender kann mit der Adresse des Datenbereichs arbeiten oder nur das erste Wort verwenden und mit dem Identifier iiii arbeiten
	X'10aaaaaa'	24-Bit-Adressierungsmodus. a...a: PAM-Datenbereichsadresse (3 Byte)
CJC	X'1400iiii' X'1404iiii' X'1408iiii'	Bedingung erfüllt Jobvariable gelöscht Katalog exportiert } iiii: Zeichenfolge zur Identifizierung des ONEVT

Tabelle 7: Anwendungen/Ereignisklassen und deren Post Codes

Der Makro **ETCNAM** (ohne Operanden) generiert symbolische Namen für die einzelnen Ereignisklassen:

```
           ETCNAM
1                *,MACRO: ETCNAM, VERSION: VER040
1 *
1 *                                EVENT TYPE CODES WHICH MAY BE
1 *                                OUTPUT FROM THE SYSTEM TO THE USER
1 *
1 ETCTCS   EQU   X'04'             TCS-EVENT
1 ETCITC   EQU   X'08'             ITC-EVENT
1 ETCDCM   EQU   X'0C'             DCAM-EVENT
1 ETCUPM   EQU   X'10'             UPAM-EVENT
1 ETCCJC   EQU   X'14'             CONDITIONAL JOB CONTROL-EVENT
```

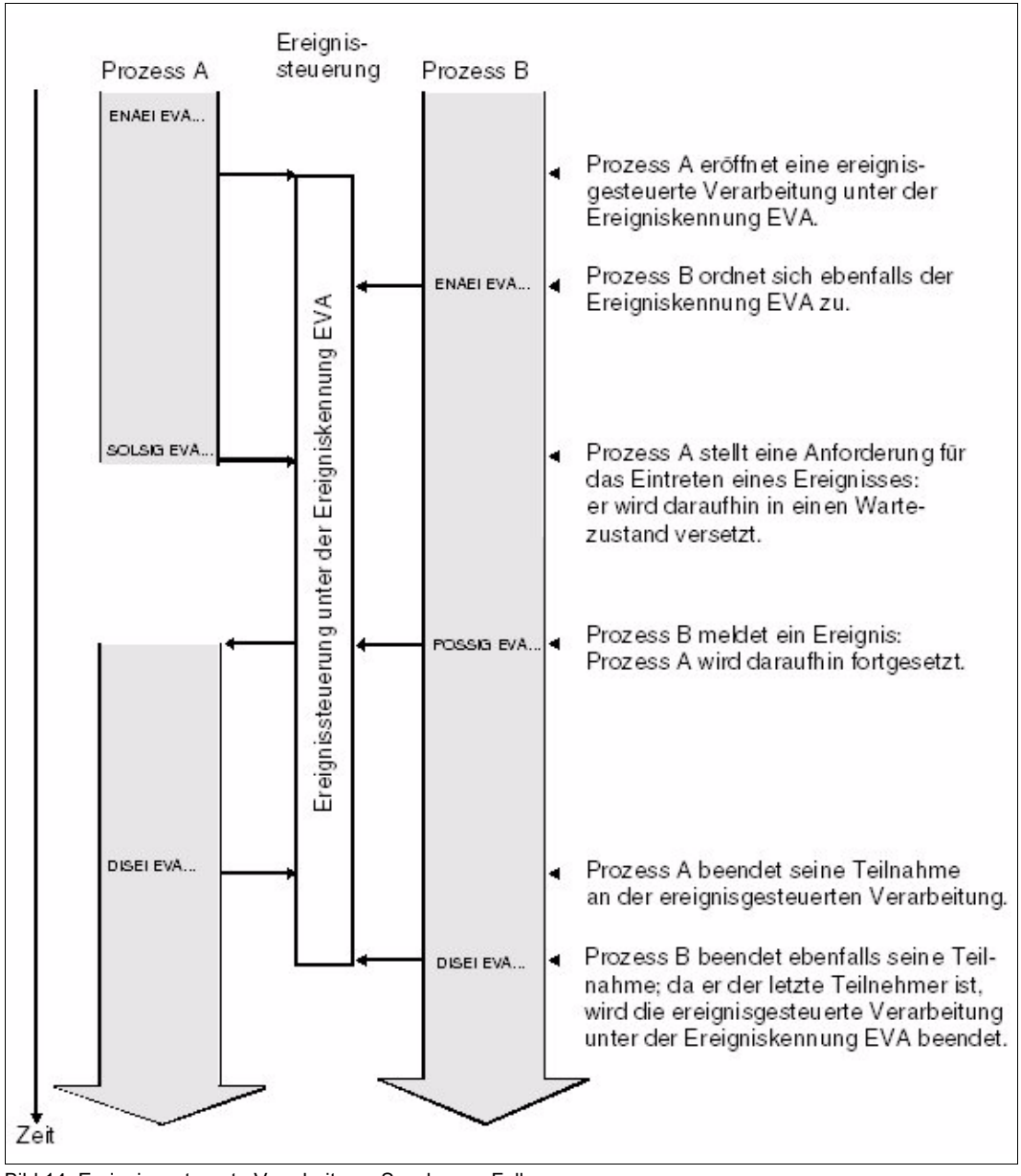


Bild 14: Ereignisgesteuerte Verarbeitung: Synchroner Fall:
Das Ereignis wird vor Ablauf der Wartezeit signalisiert.

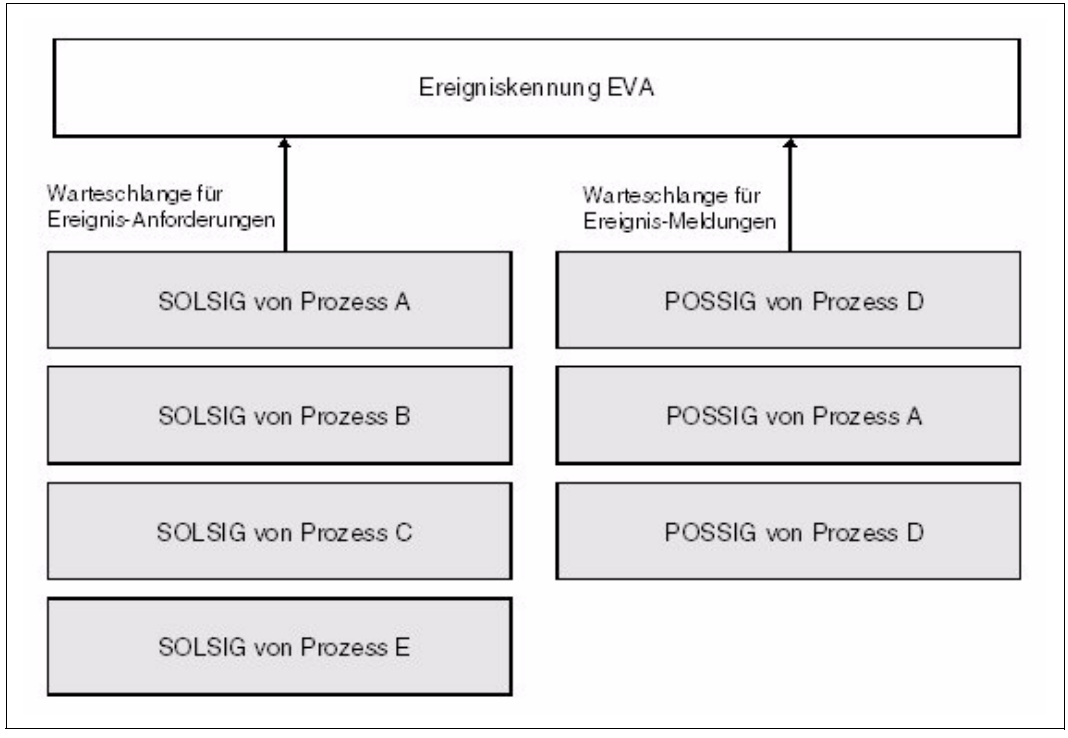


Bild 15: Warteschlangen einer Ereigniskennung

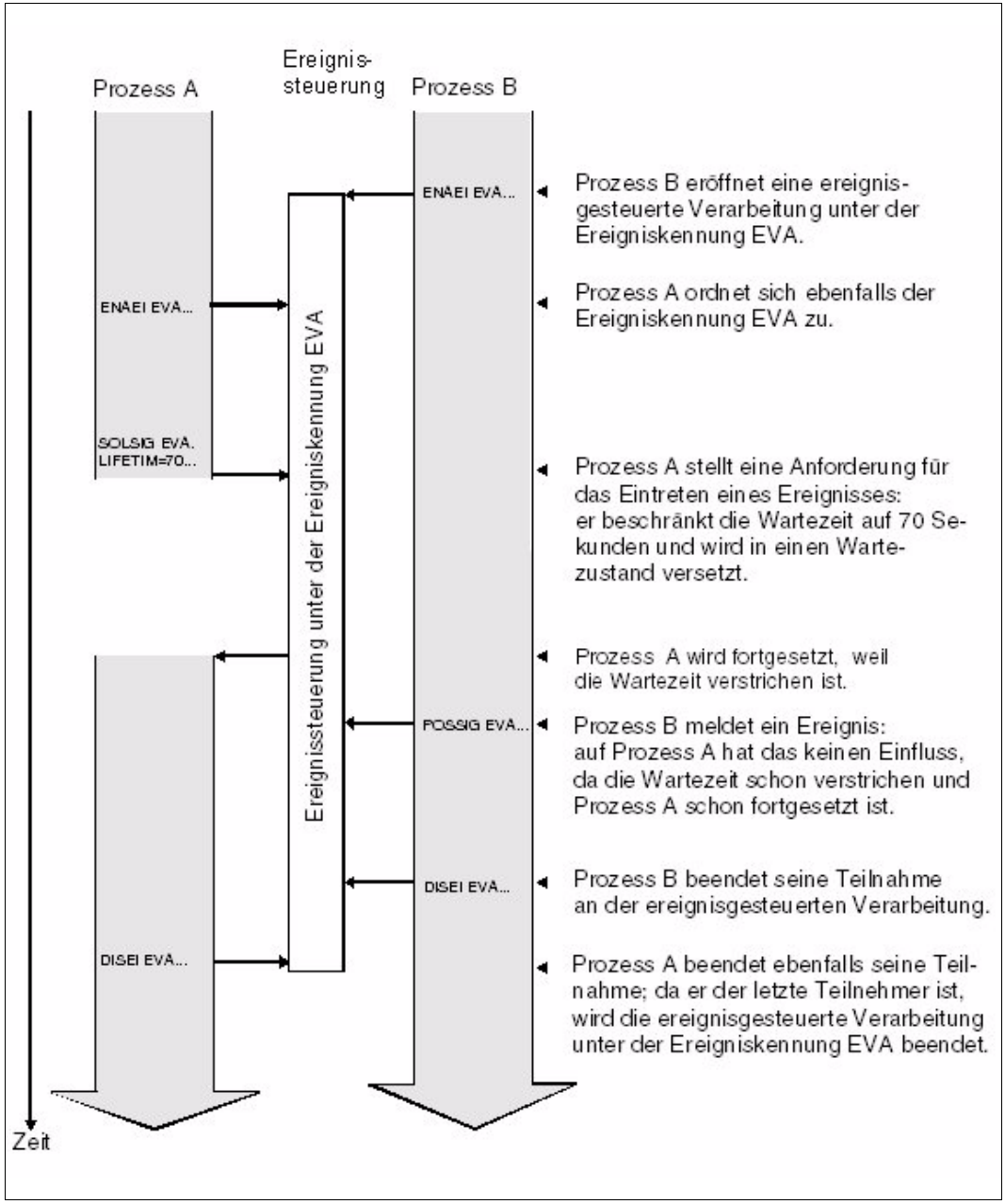


Bild 16: Ereignisgesteuerte Verarbeitung: Synchroner Fall:
Das Ereignis wird nach Ablauf der Wartezeit signalisiert

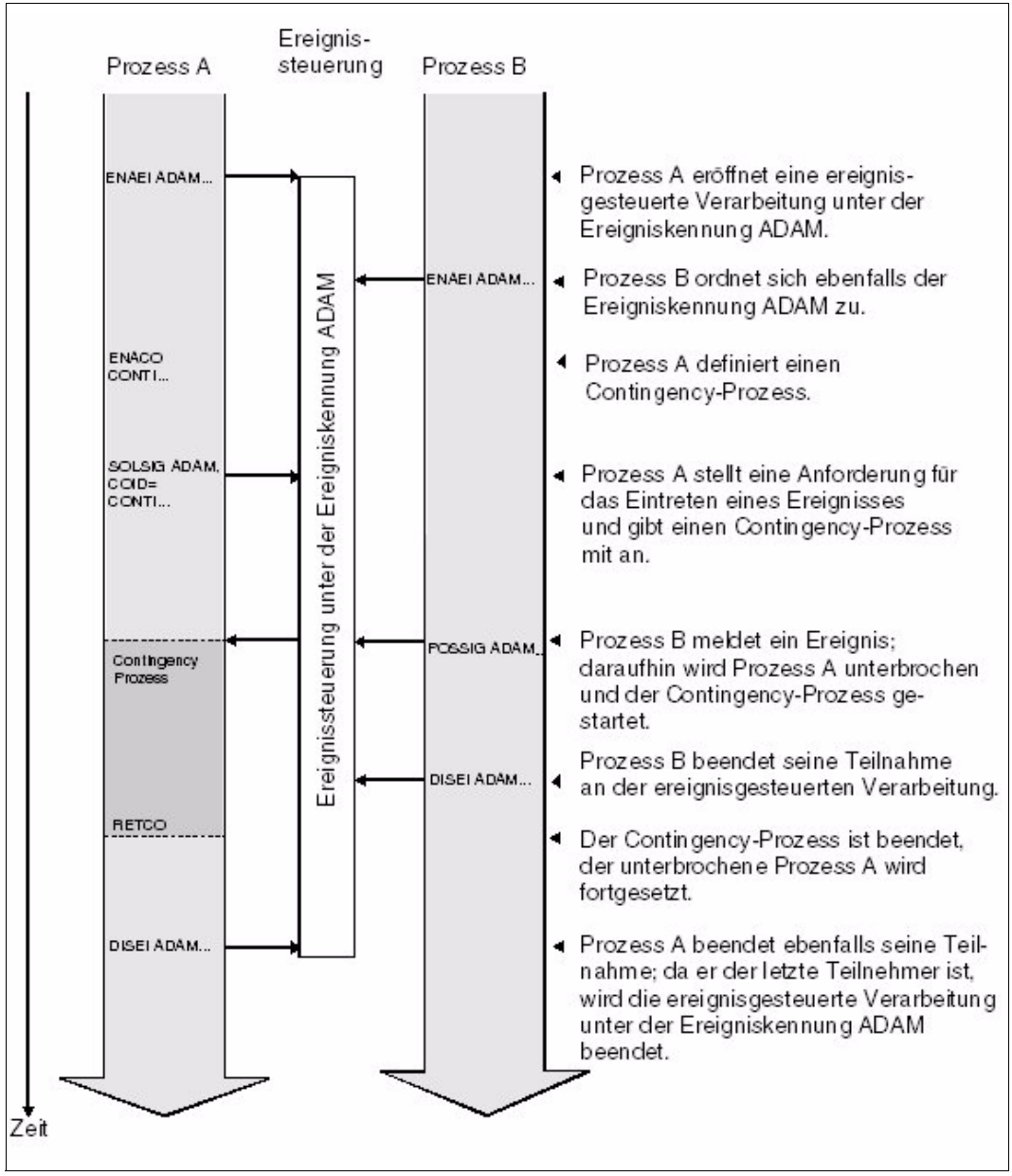


Bild 17: Ereignisgesteuerte Verarbeitung: Asynchroner Fall:
Das Ereignis wird vor Ablauf der Wartezeit signalisiert

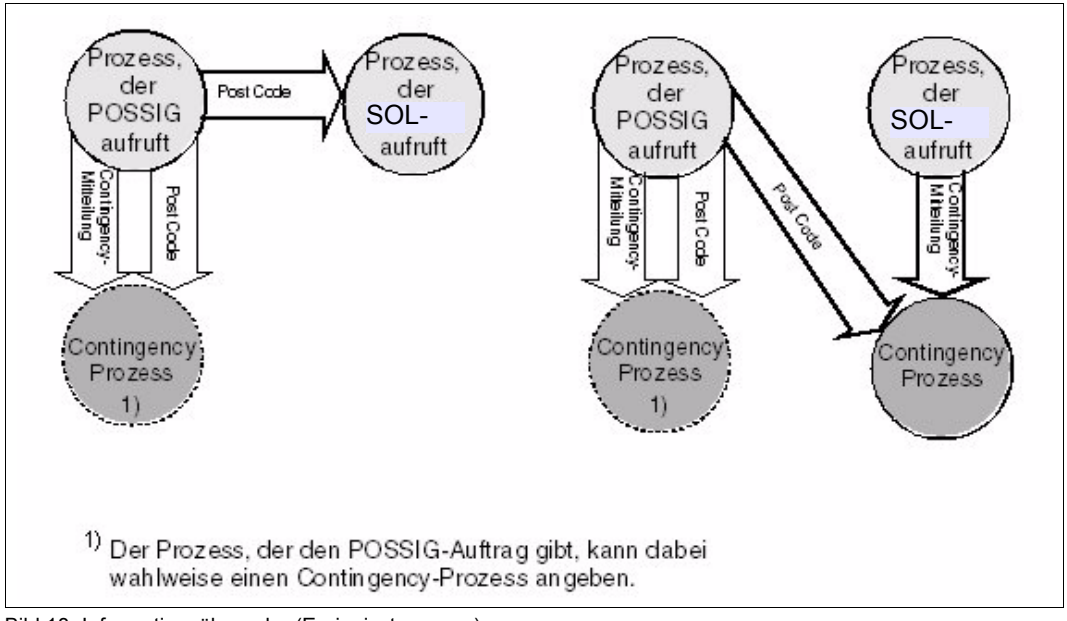


Bild 18: Informationsübergabe (Ereignissteuerung)

Die aufgeführten Grundfunktionen lassen sich mit zwei Gruppen von Makros realisieren. Die Makrogruppe **User Eventing** ermöglicht die umfassende Ausnutzung der Grundfunktionen. Die Makrogruppe **Forward Eventing** (FEV) ergänzt User Eventing um eine im Ablauf optimierte Variante. Forward Eventing ist nur für den synchronen Fall anwendbar. Die Makros von User Eventing und Forward Eventing können nebeneinander verwendet werden.

User Eventing

User Eventing wird mit folgenden Makros realisiert:

ENAEI	Teilnahme an der Ereignissteuerung erklären
POSSIG	POSSIG-Signal senden
SOLSIG	SOLSIG-Anforderung melden
CHKEI	Warteschlange prüfen
DISEI	Teilnahme an der Ereignissteuerung beenden

- Die Teilnahme an der Ereignissteuerung wird durch Aufruf des Makros **ENAEI** erklärt. Im Makroaufruf muss ein Name für die Ereigniskennung und der Geltungsbereich (Teilnehmerkreis an der Ereignissteuerung) angegeben werden. Der Name ist nur im angegebenen Geltungsbereich gültig. Aus den beiden Angaben wird intern eine Kurzbezeichnung für die Ereigniskennung gebildet. Die Kurzbezeichnung wird dem Aufrufer übergeben und kann in weiteren Makroaufrufen zur Ereignissteuerung verwendet werden (beschleunigt die Makroausführung). Jeder Teilnehmer an einer Ereignissteuerung erhält eine eigene Kurzbezeichnung. Beendet ein Anwender die Teilnahme an einer Ereignissteuerung und erklärt sie in demselben Programmlauf etwas später von neuem, wird ihm nicht notwendig dieselbe Kurzbezeichnung übergeben. Der erste Teilnehmer bewirkt das Einrichten der Ereigniskennung und der intern benötigten Verarbeitungslisten; die weiteren Teilnehmer werden der Ereigniskennung zugeordnet.
  Der Geltungsbereich kann eine Task, alle Tasks einer Benutzerkennung oder alle Tasks im System umfassen (derselbe Name mit einem anderen Geltungsbereich bezeichnet auch eine andere Ereignissteuerung).
- Mit dem Makro **POSSIG** wird ein Signal an die Ereignissteuerung gesendet. Die Task wird durch das Senden des POSSIG-Signals nicht unterbrochen. Im Makroaufruf kann eine Contingency-Routine spezifiziert werden, die dann gestartet wird, wenn das POSSIG-Signal einer SOLSIG-Anforderung zugeordnet werden konnte oder wenn eine spezifizierte Wartezeit für das Zuordnen des POSSIG-Signals abgelaufen ist (SOLSIG-Warteschlange ist leer). Nach Ablauf der Wartezeit wird das POSSIG-Signal aus der Warteschlange gelöscht.
  Der Contingency-Routine wird bei ihrem Ablauf in Register R2 ein Ereignis-Informationscode übergeben (siehe Seite 116). Der Ereignis-Informationscode informiert, ob das erwartete Ereignis (POSSIG-Signal wurde zugeordnet) eingetreten ist oder nicht; er stellt somit eine Art Quittung für das POSSIG-Signal dar.

Im **POSSIG**-Aufruf kann außerdem eine Contingency-Mitteilung spezifiziert werden, die der Contingency-Routine nach dem Start in Register R1 übergeben wird.

Mehrere aufeinander folgende **POSSIG**-Aufrufe können verkettet werden. Die Kette kann auch mit dem Makro **SOLSIG** abschließen.

- Mit dem **SOLSIG**-Makro wird eine SOLSIG-Anforderung an die Ereignissteuerung gesendet. Die Task wird im synchronen Fall solange in den Wartezustand versetzt, bis innerhalb einer angegebenen Wartezeit der SOLSIG-Anforderung ein POSSIG-Signal zugeordnet werden konnte. Es kann auch vereinbart werden, dass die Task nicht auf das Eintreffen des POSSIG-Signals warten soll.
  Im asynchronen Fall wird die Task nicht in den Wartezustand versetzt. Eine im Makroaufruf spezifizierte Contingency-Routine wird gestartet, wenn

  a) innerhalb einer angegebenen Wartezeit der SOLSIG-Anforderung ein POSSIG-Signal zugeordnet werden konnte, oder
  b) die Wartezeit abgelaufen ist.

  Für Fall a) kann vereinbart werden, dass sofort eine neue SOLSIG-Anforderung gesendet wird.

  An die Contingency werden folgende Informationen übergeben:

  – Register R1 der Contingency enthält die im **SOLSIG**-Aufruf angegebene Contingency-Mitteilung.
  – Register R2 der Contingency enthält den Ereignisinformationscode.
  – Register R3 (+ Register R4) enthält den Post Code des **POSSIG**.

  Sowohl im synchronen als auch im asynchronen Fall wird nach Ablauf der Wartezeit die SOLSIG-Anforderung aus der SOLSIG-Warteschlange gelöscht.

- Mit dem Makro **CHKEI** kann die Belegung der Warteschlangen geprüft werden. Dem Teilnehmer wird die Belegung über den Returncode mitgeteilt.

- Der Makro **DISEI** beendet die Teilnahme an der Ereignissteuerung. Noch in den Warteschlangen anstehende SOLSIG-Anforderungen werden gelöscht, ebenso die zugeordneten Forward-Events (siehe unten). Nicht zugeordnete POSSIG-Signale verbleiben in der POSSIG-Warteschlange. Die Ereigniskennung wird gelöscht, wenn der letzte (einzige) Teilnehmer die Teilnahme beendet.

Forward Eventing

Forward Eventing ergänzt User Eventing mit folgenden Makros:

DPOFEI	POSSIG-Eintrag in der EVENTLST erzeugen
DSOFEI	SOLSIG-Eintrag in der EVENTLST erzeugen
RPOFEI	POSSIG-Signal senden
RSOFEI	SOLSIG-Anforderung melden
DELFEI	POSSIG-/SOLSIG-Eintrag in der EVENTLST löschen

Forward Eventing (FEV) ist eine optimierte Form der synchronen Ereignissteuerung. FEV vermeidet für wiederholte **POSSIG**- bzw. **SOLSIG**-Aufrufe an eine bestimmte Ereigniskennung die wiederholte Validation der angegebenen Operanden. Stattdessen wird eine Ereignisliste EVENTLST angelegt und z.B. für SOLSIG-Anforderungen ein SOLSIG-Eintrag eingetragen. Im weiteren Programmfortschritt wird bei der (realen) SOLSIG-Anforderung nur noch auf diesen Eintrag Bezug genommen. Der Eintrag kann explizit wieder gelöscht werden. Gleiches gilt auch für das Senden von POSSIG-Signalen.
Pro Teilnehmer können maximal 2047 Einträge in der EVENTLST erzeugt werden.
Die Teilnahme an der Ereignissteuerung muss mit dem Makro **ENAEI** erklärt und mit **DISEI** beendet werden. Mit dem Makro **CHKEI** kann die POSSIG- bzw. die SOLSIG-Warteschlange überprüft werden. Aus der Sicht der Ereignissteuerung besteht kein Unterschied, ob z.B. ein POSSIG-Signal mit **POSSIG** oder **RPOFEI** gesendet wird.
Gleiches gilt für das Senden einer SOLSIG-Anforderung. Forward Eventing ist nur für den synchronen Fall anwendbar. Ein Post Code kann angegeben werden.

- Der Makro **DPOFEI** erzeugt einen POSSIG-Eintrag in der EVENTLST. Dem Aufrufer wird eine Referenznummer für den Eintrag zurückgegeben. Analog dem Makro **POSSIG** können mehrere aufeinander folgende Aufrufe **DPOFEI** miteinander verkettet werden. Die Angaben im Makroaufruf werden in den EVENTLST-Eintrag übernommen. Ein Post Code kann angegeben werden. Eine Contingency-Routine (Makro **POSSIG**) kann nicht spezifiziert werden.
- Der Makro **DSOFEI** erzeugt einen SOLSIG-Eintrag in der EVENTLST. Dem Aufrufer wird eine Referenznummer für den Eintrag zurückgegeben. Die Angaben im Makroaufruf werden in den EVENTLST-Eintrag übernommen. Es kann nur der synchrone Fall spezifiziert werden.
- Mit dem Makro **RPOFEI** wird unter Bezugnahme auf den POSSIG-Eintrag in der EVENTLST ein POSSIG-Signal an die Ereignissteuerung gesendet.
- Mit dem Makro **RSOFEI** wird unter Bezugnahme auf den SOLSIG-Eintrag in der EVENTLST eine SOLSIG-Anforderung an die Ereignissteuerung gemeldet.
- Mit dem Makro **DELFEI** kann ein POSSIG- oder SOLSIG-Eintrag in der EVENTLST wieder gelöscht werden.

Beispiel: Synchroner Fall

Die Programme EV1 und EV2 werden in voneinander verschiedenen Dialogaufträgen (an verschiedenen Datensichtstationen) gestartet. EV2 liest aus einer Datei, die EV1 kopieren will. Das Schließen der Datei signalisiert EV2 an EV1; EV1 kopiert daraufhin die Datei. Beide Programme sollen im 31-Bit-Adressierungsmodus ablaufen; EV1 unterhalb und EV2 oberhalb der 16-MByte-Grenze.

Programm EV1

```
EV1      START
EV1      AMODE ANY
EV1      RMODE ANY
         GPARMOD 31
         PRINT NOGEN
         BALR  3,0
         USING *,3
         ENAEI EINAME=EVE,SCOPE=GROUP,EIIDRET=ABRID1 ——————————————————  (1)
         GDATE TOD=TIME1
         WROUT MESS1,ERROR
         SOLSIG EIID=ABRID1,COND=UNCOND,RPOSTAD=PCEMPF,RPOSTL=2,       -
               LIFETIM=800 ———————————————————————————————————————————  (2)
M1       GDATE TOD=TIME2
         WROUT MESS2,ERROR ———————————————————————————————————————————  (3)
         CMD   'COPY-FILE','TEST.FILE.1,TEST.FILE.2'
         WROUT TEXT,ERROR
         DISEI EIID=ABRID1 ———————————————————————————————————————————  (4)
         TERM
ERROR    TERM  DUMP=Y
***      DEFINITIONS     ***
ABRID1   DS    F
MESS1    DC    Y(MESS1END-MESS1)
         DS    CL2
         DC    X'01'
         DC    C'PROGRAM WAITING SINCE '
TIME1    DS    CL8
MESS1END EQU   *
MESS2    DC    Y(MESS2END-MESS2)
         DS    CL2
         DC    X'01'
         DC    C'POSSIG SIGNAL RECEIVED AT '
TIME2    DS    CL8
         DC    C'; POSTCODE = '
PCEMPF   DS    CL8
MESS2END EQU   *
```

```
TEXT     DC    Y(TEXTEND-TEXT)
         DS    CL2
         DC    X'01'
         DC    C'COPY-FILE COMMAND EXECUTED'
TEXTEND  EQU   *
         END
```

(1) Die Ereignissteuerung EVE wird eröffnet.

(2) EV1 sendet ein SOLSIG-Signal an EVE und stellt ein Empfangsfeld für einen Post Code (2 Worte) zur Verfügung. Anschließend wartet EV1 auf das Eintreffen eines POSSIG-Signals (höchstens 800 Sekunden).

(3) Nach Eintreffen des POSSIG-Signals (bzw. nach Ende der Wartezeit) wird EV1 fortgesetzt und gibt eine Meldung aus. EV2 hat die Datei TEST.FILE.1 geschlossen; EV1 kann die Datei kopieren.

(4) EV1 beendet die Teilnahme an der Ereignissteuerung EVE.

Programm EV2

```
EV2      START
EV2      AMODE ANY
EV2      RMODE ANY
         GPARMOD 31
         PRINT NOGEN
         BALR  3,0
         USING *,3
         ENAEI EINAME=EVE,SCOPE=GROUP,EIIDRET=ABRID2 ------------------  (5)
         OPEN  TESTFCB,INPUT ------------------------------------------  (6)
         GET   TESTFCB,INAREA
CLOSE    CLOSE ALL
         GDATE TOD=TIME1
         POSSIG EIID=ABRID2,SPOSTAD=PCSEND,SPOSTL=2 -------------------  (7)
         WROUT MESS1,ERROR
         DISEI EIID=ABRID2 --------------------------------------------  (8)
         TERM
ERROR    TERM  DUMP=Y
***      FILE     ***
         DS    OF
TESTFCB  FCB   FCBTYPE=SAM,LINK=FILIN,EXIT=E1
E1       EXLST EOFADDR=CLOSE,COMMON=CLOSE
***      DEFINITIONS     ***
ABRID2   DS    F
PCSEND   DC    X'C5E5F2606OC5E5F14O'  FIELD ALIGNED ON WORD BOUNDARY!
```

```
MESS1    DC    Y(MESS1END-MESS1)
         DS    CL2
         DC    X'01'
         DC    C'POSSIG SIGNAL SENT AT '
TIME1    DS    CL8
MESS1END EQU   *
INAREA   DS    CL200
         END
```

(5) EV2 schließt sich der Ereignissteuerung EVE an.

(6) EV2 öffnet die Datei TEST.FILE.1 und liest einen Datensatz ein. Die Datei wird geschlossen.

(7) EV2 sendet ein POSSIG-Signal und übergibt einen Post Code (2 Worte).

(8) EV2 schließt die Ereignissteuerung EVE.

Ablaufprotokoll des Dialogauftrags mit dem Programm EV1

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,ev1), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,ev1))
%  ASS6011 ASSEMBLY TIME: 525 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 86 MSEC
//end
%  ASS6012 END OF ASSEMBH
/start-executable-program library=macexmp.lib,element-or-symbol=ev1 ——   (1)
%  BLS0523 ELEMENT 'EV1', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'EV1', VERSION ' ' OF '<date> <time>' LOADED
PROGRAM WAITING SINCE 13:14:41 ————————————————————————————————————   (2)
POSSIG SIGNAL RECEIVED AT 13:20:22; POSTCODE = EV2--EV1
COPY-FILE COMMAND EXECUTED ————————————————————————————————————————   (3)
```

(1) EV1 wird geladen und gestartet.

(2) EV1 wartet auf das POSSIG-Signal.

(3) Das POSSIG-Signal wurde empfangen, EV1 kopiert die Datei.

Ablaufprotokoll des Dialogauftrags mit dem Programm EV2

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,ev2), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,ev2))
%  ASS6011 ASSEMBLY TIME: 871 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 86 MSEC
//end
%  ASS6012 END OF ASSEMBH
/add-file-link link-name=filin,file-name=test.file.1
/start-executable-program library=macexmp.lib,element-or-symbol=ev2 ---  (4)
%  BLS0523 ELEMENT 'EV2', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'EV2', VERSION ' ' OF '<date> <time>' LOADED
POSSIG SIGNAL SENT AT 13:20:22 ------------------------------------------  (5)
```

(4) EV2 wird geladen und gestartet.

(5) EV2 sendet das POSSIG-Signal, nachdem es die Datei TEST.FILE.1 geschlossen hat.

Weitere **Beispiele** enthalten der Abschnitt „Contingency-Prozesse“ (Seite 119) sowie die Beschreibung der Makros **POSSIG** (Seite 760) und **SOLSIG** (Seite 850).

4.3.6 Contingency-Prozesse

Makro	Kurzbeschreibung
CONTXT	liest oder schreibt in den Registern des unterbrochenen Contingency-Prozesses oder des Basisprozesses
DISCO	schließt die Definition einer Routine als Contingency-Prozess
ENACO	eröffnet eine Routine als Contingency-Prozess und weist ihr einen Contingency-Namen und eine Priorität zu
LEVCO	ändert die Priorität des aufrufenden Basis- oder Contingency-Prozesses während des Ablaufs
RETCO	beendet den aufrufenden Contingency-Prozess. Das System setzt den Prozess mit dem Basisprozess oder einem anderen Contingency-Prozess fort
SUSPEND	setzt den aufrufenden Basis- oder Contingency-Prozess in einen unterbrechbaren Wartezustand

Einsatz von Contingency-Prozessen

Contingency-Prozesse werden in Zusammenhang mit der ereignisgesteuerten Verarbeitung und auch mit dem STXIT-Verfahren verwendet.
Die Verwendung von Contingency-Prozessen im STXIT-Verfahren stellt einen speziellen Anwendungsfall dar. Daraus sich ergebende Einschränkungen gegenüber der folgenden Beschreibung gehen aus dem Abschnitt „STXIT-Verfahren“, (Seite 133) hervor.
Es handelt sich dabei um benutzereigene Abläufe, die vom System als abhängiger Prozess verarbeitet werden.

Folgende Merkmale kennzeichnen einen Contingency-Prozess und unterscheiden ihn von einer Task einerseits und von einer Routine andererseits:

– Ein Contingency-Prozess hat keine eigene Auftragsnummer (TSN).
– Contingency-Prozesse können ineinander geschachtelt werden.
– Ein Contingency-Prozess wird nicht durch SET-LOGON-PARAMETERS vom Benutzer gestartet. Sein Start wird durch das Eintreten eines vom Benutzer bestimmten Ereignisses ausgelöst.
– Ein Contingency-Prozess hat eine eigene Verarbeitungsebene (Priorität) und einen eigenen Process Control Block (PCB); damit einen eigenen Registersatz.

Der Benutzer kann durch einen Contingency-Prozess, dessen Start definitionsgemäß von einem Ereignis ausgelöst wird, Maßnahmen durchführen lassen, die auf dieses Ereignis genau abgestimmt sind. Als solches Ereignis könnte zum Beispiel gelten, dass ein anderer Prozess ein bestimmtes Verarbeitungsstadium erreicht hat, wie die Erstellung einer Datei oder der Start eines Programms.

Die Maßnahmen zur Behandlung mehrerer Ereignisse können durch die Vergabe entsprechender Prioritäten präzise koordiniert werden.

Die Register des Contingency-Prozesses enthalten bei seinem Start den Wert Null, und man muss ein Basisregister zuweisen und laden. Bei der Wahl des Basisregisters muss man berücksichtigen, dass die Register R1, R2, R3 und evtl. R4 für Informationsübergabe an den Contingency-Prozess vorgesehen sind.

Contingency-Prozesse können vom Benutzer durch den Aufruf folgender Makros verwendet werden:

ENACO	Eröffnen der Contingency-Definition
DISCO	Schließen der Contingency-Definition
RETCO	Beenden Contingency-Prozess
CONTXT	Zugriff auf Prozessdaten
LEVCO	Priorität ändern

Eröffnen und Schließen der Contingency-Definition

Die Definition eines Contingency-Prozesses wird durch den Aufruf des Makros **ENACO** vorgenommen. Dabei werden der Name, die Startadresse und die Priorität festgelegt. Der zum Zeitpunkt der Contingency-Definition eingeschaltete Adressierungsmodus (AMODE) wird durch das Betriebssystem auch bei Ablauf der Contingency-Routine eingeschaltet.

Der Makro **ENACO** kann sowohl in einem Prozess (Basisprozess) als auch in einem Contingency-Prozess aufgerufen werden. Der Basisprozess ist immer der zugehörige unabhängige Prozess, der als erster Prozess (mit SET-LOGON-PARAMETERS) gestartet wurde.

Sobald eine Routine nicht mehr zum Ablauf als Contingency-Prozess verwendet werden soll, kann der Benutzer die Contingency-Definition schließen, indem er den Makroaufruf **DISCO** mit dem Namen des Contingency-Prozesses gibt. Die Contingency-Definition wird dadurch gelöscht.

Nachfolgende **POSSIG**- und **SOLSIG**-Aufrufe, die sich auf eine gelöschte Contingency-Definition beziehen, werden zurückgewiesen. **POSSIG**- oder **SOLSIG**-Aufrufe, deren zugehöriger Eintrag zum Zeitpunkt der Löschung noch in den Warteschlangen der Ereigniskennung steht, sind von der Löschung der Contingency-Definition nicht betroffen. Der Contingency-Prozess wird also bei Eintreten der Bedingungen noch aktiviert.

Im Gegensatz zur Ereigniskennung ist der Geltungsbereich eines Contingency-Prozesses immer lokal, seine Verwendung ist also auf die definierende Task beschränkt. Eine Task kann maximal 400 Contingency-Prozesse gleichzeitig verwenden.

Das System stellt dem definierenden Prozess für den Namen eines Contingency-Prozesses unter einer Adresse eine Kurzkennung zur Verfügung. Die Verwendung dieser Kurzkennung beschleunigt die Verarbeitung und ist in den Makroaufrufen **POSSIG** und **SOLSIG** vorgeschrieben. Wird eine Contingency-Definition gelöscht und mit demselben Namen von neuem gegeben, so kann die Kurzkennung anders lauten als im vorhergehenden Fall.

Start und Ende des Contingency-Prozessablaufs

Der Benutzer ermöglicht den Start eines Contingency-Prozesses dadurch, dass er dessen Kurzkennung in einem **POSSIG**- oder **SOLSIG**-Makroaufruf (siehe Seite 116) als Operand angibt. Dadurch wird der Zusammenhang mit dem auslösenden Ereignis hergestellt.

In folgenden Fällen wird ein Contingency-Prozess aktiviert und - sofern seine Priorität das erlaubt - gestartet:

– Asynchroner Fall der ereignisgesteuerten Verarbeitung
  Der Contingency-Prozess wurde in dem Makroaufruf **SOLSIG** zur Anforderung eines Ereignissignals angegeben, und das Signal ist eingetroffen (Bild 17 auf Seite 102) bzw. die Wartezeit ist verstrichen.
  Der Contingency-Prozess wird auch gestartet, wenn der Basisprozess die Ereigniskennung vorher löscht (**DISEI**).

– Quittung für ein gegebenes Signal
  Der Contingency-Prozess wurde in dem Makroaufruf **POSSIG** zur Signalisierung eines Ereignisses angegeben, und das Signal wurde angefordert (Bild 19 auf Seite 114) bzw. die Wartezeit ist verstrichen. Der Contingency-Prozess wird auch gestartet, wenn der Basisprozess die Ereigniskennung vorher löscht (**DISEI**).

Der Start eines Contingency-Prozesses wird vom System unter folgenden Voraussetzungen durchgeführt:

– Kein Prozess mit höherer Priorität ist aktiviert oder gestartet.
– Kein Prozess mit gleicher Priorität ist vor ihm in der Warteschlange eingereiht.

Der Contingency-Prozess muss zu Beginn ein Basisregister definieren.

Der Contingency-Prozess wird mit dem Zugriffsrecht gestartet, das während des zugehörigen **SOLSIG**- bzw. **POSSIG**-Aufrufs gültig war. Die Contingency-Routine wird vom Betriebssystem in demselben Adressierungsmodus gestartet, der zum Zeitpunkt der Contingency-Definition (Makro **ENACO** oder **STXIT**) eingeschaltet war. Nach Ablauf des Contingency-Prozesses erhält der unterbrochene Prozess wieder das Zugriffsrecht, das er vor der Unterbrechung hatte.

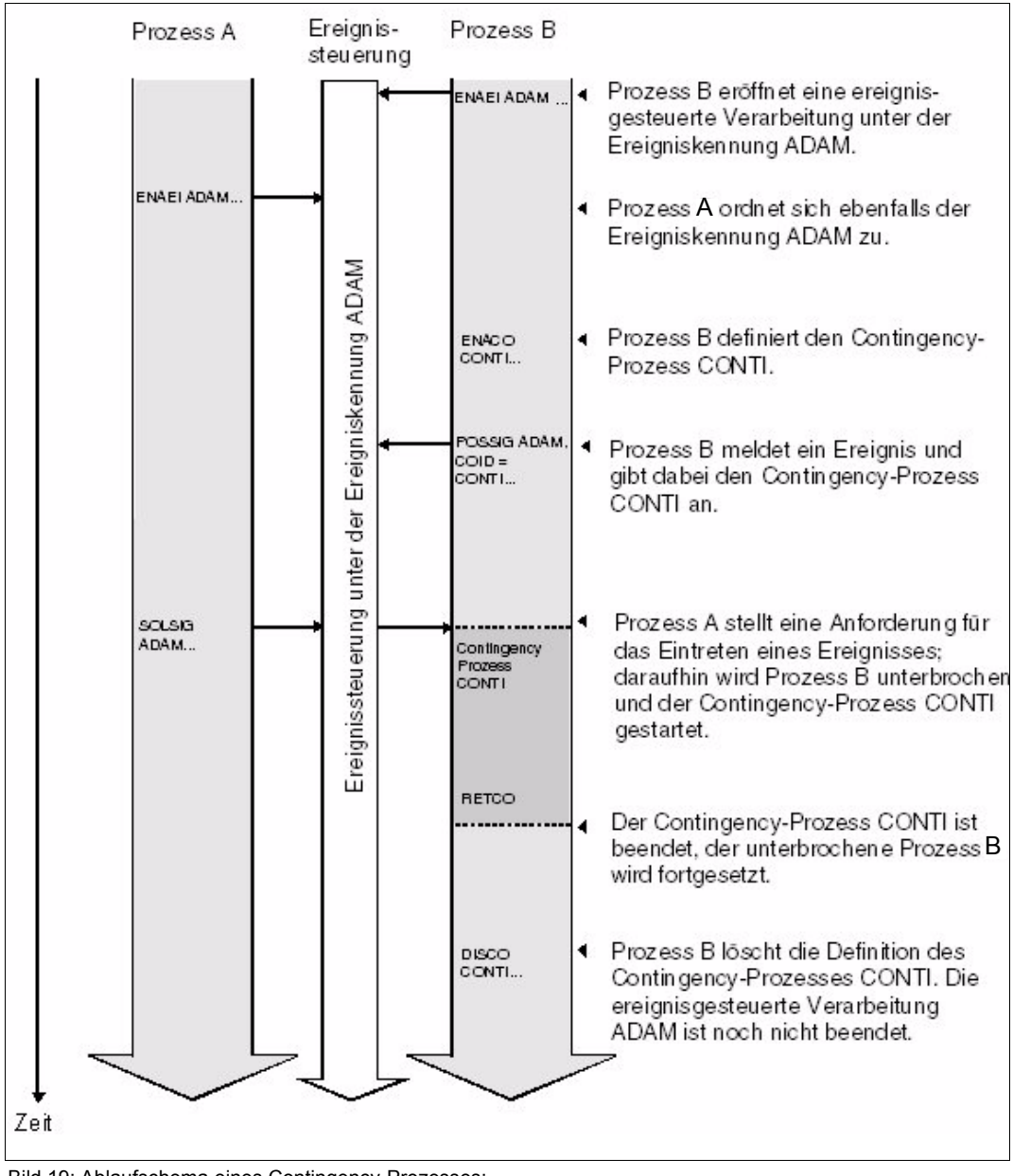


Bild 19: Ablaufschema eines Contingency-Prozesses:
Der Contingency-Prozess wird hier zur Quittierung eingesetzt

Der Ablauf eines Contingency-Prozesses wird dadurch beendet, dass von dem Contingency-Prozess der Makro **RETCO** aufgerufen wird. Die Steuerung wird dann, abhängig von den Prioritäten, dem Basisprozess oder einem anderen Contingency-Prozess übergeben.

Prozesspriorität bei der Contingency-Verarbeitung

Sind mehrere Contingency-Prozesse definiert, so kann durch Vergabe von Prioritäten bei der Contingency-Definition und durch Prioritätsänderung während des Ablaufs eine Koordinierung bei gleichzeitigem Eintreten von mehreren Ereignissen durchgeführt werden.

Zugelassener Prioritätsbereich:

Basisprozess	Priorität 0-127	Standardwert: 0
Contingency-Prozess	Priorität 1-127	Standardwert: 1

Ein Contingency-Prozess hat standardmäßig eine höhere Priorität als der Basisprozess und unterbricht diesen durch seinen Start. Der Basisprozess kommt erst wieder nach Beendigung des Contingency-Prozesses zum Ablauf. Auch ein Contingency-Prozess kann durch einen weiteren Contingency-Prozess mit höherer Priorität unterbrochen werden (Bild 20 auf Seite 117). Mehrere Contingency-Prozesse gleicher Priorität werden standardmäßig in eine Warteschlange eingereiht, die nach dem FIFO-Prinzip (first in - first out) abgearbeitet wird.

Sowohl ein Basisprozess als auch ein Contingency-Prozess kann seine eigene Priorität während des Ablaufs durch den Aufruf des Makros **LEVCO** ändern. Dabei kann zwischen dem FIFO- und dem LIFO-Prinzip gewählt werden. Der Prozess wird mit seiner geänderten Priorität unter Prozesse gleicher Priorität entsprechend eingereiht.

– FIFO-Prinzip (first in - first out):
  Der Prozess wird nach seiner Aktivierung oder nach Ausführung des Makros **LEVCO** am Ende der Warteschlange für Prozesse gleicher Priorität eingereiht. Alle vor ihm in der Warteschlange eingereihten Prozesse werden vor ihm gestartet. Setzt ein Prozess seine Priorität während seines Ablaufs herab und lässt das FIFO-Prinzip gelten, dann besteht die Möglichkeit, dass er auch durch Prozesse gleicher Priorität unterbrochen wird. Dies kann verhindert werden, indem im Aufruf des Makros **LEVCO** das LIFO-Prinzp gewünscht wird.

– LIFO-Prinzip (last in - first out):
  Der Prozess wird nach seiner Aktivierung oder nach Ausführung des Makros **LEVCO** an der Spitze der Warteschlange für Prozesse gleicher Priorität eingereiht. Vor ihm können also nur Prozesse höherer Priorität zum Ablauf kommen.

Zur Unterbrechbarkeit von Contingency-Prozessen siehe auch Makro **CONTXT**.

Es liegt in der Hand des Benutzers, die Prioritäten so zu vergeben, dass sich eine sinnvolle Prozess-Schachtelung ergibt. Bei Erhöhung der Basisprozess-Priorität sollte darauf geachtet werden, dass nachfolgend aktivierte Contingency-Prozesse nicht dadurch abgeblockt

werden, bis der Basisprozess beendet ist. Wird in einer derartigen Konstellation (der aktive Prozess hat eine höhere Priorität als bereits wartende Contingency-Prozesse in der Warteschlange) der aktive Prozess mit dem **SUSPEND**-Makro in den Wartezustand versetzt, führt das *nicht* zum Start der bereits wartenden Contingency-Prozesse.

Einschränkung

Ein Prozess darf seine Priorität nicht derartig herabsetzen, dass sie unter der eines Prozesses liegt, der bereits gestartet (und unterbrochen) war. Dieser zu irgendeinem früheren Zeitpunkt gestartete und unterbrochene Prozess darf erst nach Beendigung des nach ihm gestarteten Contingency-Prozesses wieder zum Ablauf kommen.

Zugriff zum unterbrochenen Prozess

Jeder Contingency-Prozess hat einen eigenen Registersatz.

Ein Contingency-Prozess hat Zugriff zu den Registern des Prozesses, den er unterbrochen hat, oder zu denen des Basisprozesses. Der Makro **CONTXT** ermöglicht das Lesen oder Verändern der Gleitpunktregister, des Befehlszählers und der allgemeinen Register des unterbrochenen oder des Basisprozesses.

Informationsübergabe an Contingency-Prozesse

Ein Contingency-Prozess kann bei seinem Start maximal drei Arten von Informationen erhalten.

Register R1	kann eine **Contingency-Mitteilung** (4 Byte) enthalten. Sie kann im Aufruf **ENACO** oder - zu einem späteren Zeitpunkt mit überschreibender Wirkung - in den Aufrufen **POSSIG** oder **SOLSIG** angegeben werden (Bild 18 auf Seite 103). Ihre Form und Bedeutung kann vom Anwender beliebig festgelegt werden.
Register R2	enthält immer den **Ereignis-Informationscode** (2 Byte), der angibt, welche Bedingungen zur Aktivierung des Contingency-Prozesses geführt haben. Seine Form und Bedeutung ist vorgegeben (siehe Tabelle 8 auf Seite 118).
Register R3 (+ R4)	kann einen **Post Code** (4 oder 8 Byte) enthalten. Umfasst der Post Code 8 Byte (2 Worte), wird das zweite Wort in Register R4 eingetragen. Der Post Code kann im Makroaufruf **POSSIG** angegeben werden (Bild 18 auf Seite 103). Seine Form und Bedeutung kann vom Benutzer innerhalb der geltenden Konventionen festgelegt werden (siehe Seite 97).

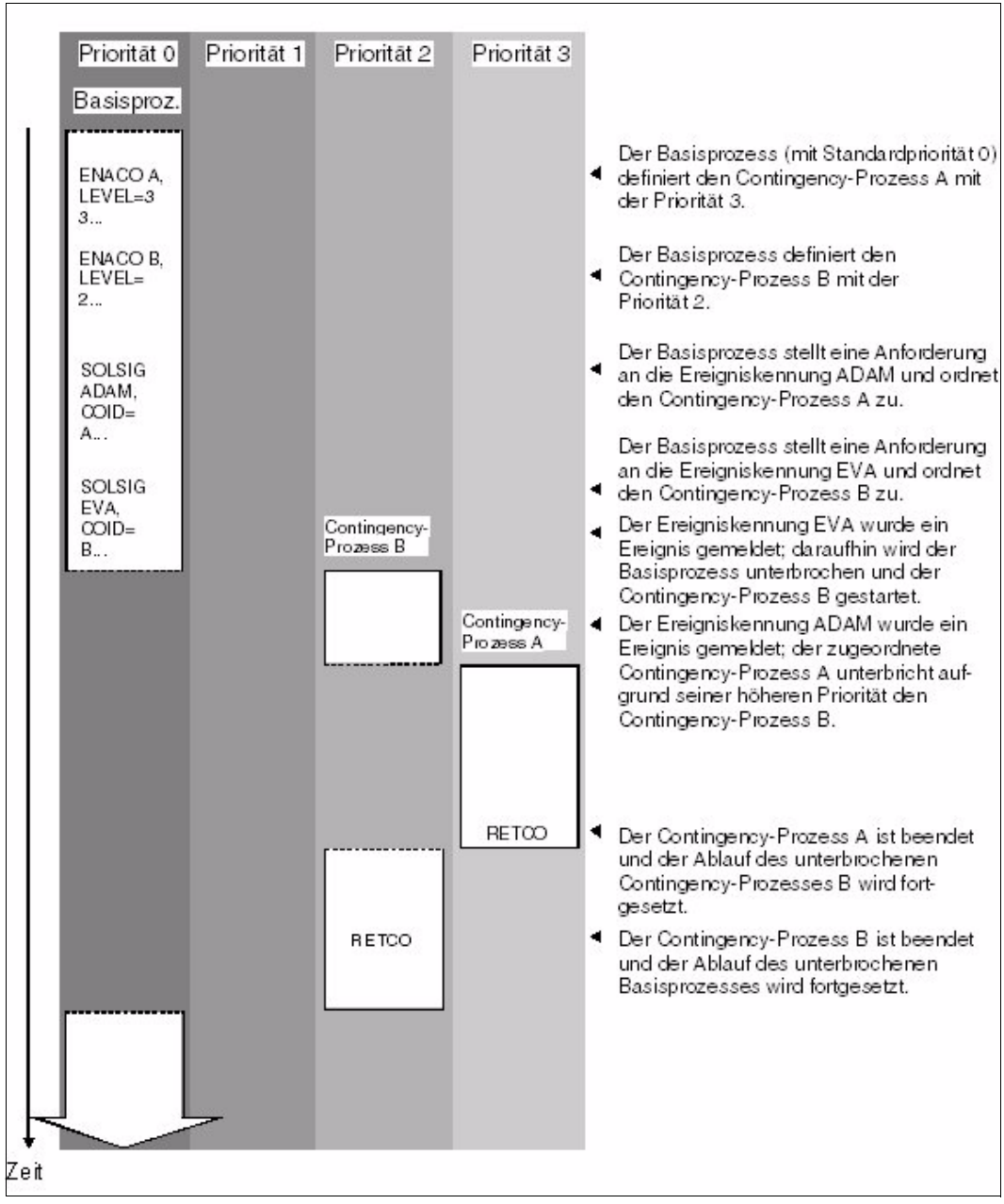


Bild 20: Contingency-Prozesse: Prioritätsbedingter Ablauf

Der **Ereignis-Informationscode** in Register R2 besteht aus dem Ereignisschalter ES im rechtsbündigen Byte und dem Informationsindikator II im linksbündigen Byte.

II	ES	Erläuterung
X'00'	X'00'	Das erwartete Ereignis ist eingetreten. Weder Post Code noch Contingency-Mitteilung sind vorhanden.
X'04'	X'00'	Das erwartete Ereignis ist eingetreten. Contingency-Mitteilung ist vorhanden; Post Code ist nicht vorhanden.
X'08'	X'00'	Das erwartete Ereignis ist eingetreten. Contingency-Mitteilung ist nicht vorhanden; Post Code (Länge = 4 Byte) wurde in Register R3 eingetragen.
X'0C'	X'00'	Das erwartete Ereignis ist eingetreten. Contingency-Mitteilung und Post Code (Länge = 4 Byte) sind vorhanden.
X'28'	X'00'	Das erwartete Ereignis ist eingetreten. Contingency-Mitteilung ist nicht vorhanden. Post Code (Länge = 8 Byte) wurde in die Register R3 + R4 eingetragen.
X'2C'	X'00'	Das erwartete Ereignis ist eingetreten. Contingency-Mitteilung und Post Code (Länge = 8 Byte) sind vorhanden.
X'00'	X'04'	Das erwartete Ereignis trat innerhalb der Zeitperiode nicht ein. Weder Contingency-Mitteilung noch Post Code sind vorhanden
X'04'	X'04'	Das erwartete Ereignis trat innerhalb der Zeitperiode nicht ein. Contingency-Mitteilung ist vorhanden; Post Code ist nicht vorhanden.
X'08'	X'04'	Das erwartete Ereignis trat innerhalb der Zeitperiode nicht ein. Contingency-Mitteilung ist nicht vorhanden Post Code (Länge = 4 Byte) wurde in Register R3 eingetragen.
X'0C'	X'04'	Das erwartete Ereignis trat innerhalb der Zeitperiode nicht ein. Contingency-Mitteilung und Post Code (Länge = 4 Byte) sind vorhanden.
X'10'	X'04'	Die Verwendung der Ereigniskennung wurde beendet (DISEI), bevor das erwartete Ereignis eintrat. Weder Contingency-Mitteilung noch Post Code sind vorhanden.
X'14'	X'04'	Die Verwendung der Ereigniskennung wurde beendet (DISEI), bevor das erwartete Ereignis eintrat. Contingency-Mitteilung ist vorhanden; Post Code ist nicht vorhanden.
X'18'	X'04'	Die Verwendung der Ereigniskennung wurde beendet (DISEI),bevor das Ereignis eintrat. Contingency-Mitteilung ist nicht vorhanden; Post Code (Länge = 4 Byte) wurde in Register R3 eingetragen.
X'1C'	X'04'	Die Verwendung der Ereigniskennung wurde beendet (DISEI), bevor das Ereignis eintrat. Contingency-Mitteilung und Post Code (Länge = 4 Byte) sind vorhanden.
X'28'	X'04'	Das erwartete Ereignis trat innerhalb der Zeitperiode nicht ein. Contingency-Mitteilung ist nicht vorhanden. Post Code (Länge = 8 Byte) wurde in die Register R3 + R4 eingetragen.
X'2C'	X'04'	Das erwartete Ereignis trat innerhalb der Zeitperiode nicht ein. Contingency-Mitteilung und Post Code (Länge = 8 Byte) sind vorhanden.

Tabelle 8: Ereignis-Informationscodes

II	ES	Erläuterung
X'38'	X'04'	Die Verwendung der Ereigniskennung wurde beendet (DISEI), bevor das Ereignis eintrat. Contingency-Mitteilung ist nicht vorhanden. Post Code (Länge = 8 Byte) wurde in Register R3 + R4 eingetragen.
X'3C'	X'04'	Die Verwendung der Ereigniskennung wurde beendet (DISEI), bevor das Ereignis eintrat. Contingency-Mitteilung und Post Code (Länge = 8 Byte) sind vorhanden.

Tabelle 8: Ereignis-Informationscodes

Beispiel: Asynchroner Fall

Teil 1: Dialogauftrag mit dem Programm PCOSOL1

Das Programm PCOSOL1 definiert zwei Ereigniskennungen (ADAM und EVE) und zwei Contingency-Prozesse (CONTA und CONTE). Das Programm fordert von jeder Ereigniskennung mit dem Makro **SOLSIG** ein Signal an und gibt jeweils einen Contingency-Prozess an (asynchroner Fall). Im Teil 1 des Beispiels wird kein **POSSIG**-Aufruf an die Ereigniskennungen gegeben. Der Contingency-Prozess CONTA wird gestartet, weil seine Wartezeit (60 Sekunden) verstrichen ist. CONTE wird gestartet, weil das Programm die Ereigniskennung EVE löscht.

Programm PCOSOL1

```
PCOSOL1  START
         PRINT NOGEN
         BALR  5,0
         USING *,5
         ENAEI EINAME=ADAM,SCOPE=GROUP,EIIDRET=ABBRADAM ————————————  (1)
         ENAEI EINAME=EVE,SCOPE=GROUP,EIIDRET=ABBREVE
         ENACO CONAME=A,COADAD=CONTAAD,COIDRET=ABBRA ————————————————  (2)
         ENACO CONAME=E,COADAD=CONTEAD,COIDRET=ABBRE
         GDATE TOD=TIMEBAS1
         SOLSIG EIID=ABBRADAM,COID=ABBRA,LIFETIM=60 —————————————————  (3)
         CL    15,NULL
         BNE   ERROR
         SOLSIG EIID=ABBREVE,COID=ABBRE ———————————————————————————  (4)
         CL    15,NULL
         BNE   ERROR
         WROUT MLDBAS1,ERROR
         VPASS 120 ———————————————————————————————————————————————  (5)
CONNECT  GDATE TOD=TIMEBAS2
         CHKEI EIID=ABBRADAM
         ST    1,WSADAM
         ST    15,WSARC
         CHKEI EIID=ABBREVE
         ST    1,WSEVE
         ST    15,WSERC
```

```
         WROUT MLDBAS2,ERROR
         DISCO COID=ABBRA ------------------------------------------------ (6)
         DISCO COID=ABBRE
         GDATE TOD=TIMEBAS3
         DISEI EIID=ABBRADAM ---------------------------------------------- (7)
         DISEI EIID=ABBREVE
         WROUT MLDBAS3,ERROR
DTH1     TERM
*
CONTA    BALR  6,0 ------------------------------------------------------- (8)
         USING *,6
         GDATE TOD=TIMEA
         ST    2,INFOADAM
         WROUT MLDCONA,ERROR
         CONTXT SAVE=LASTREG,PROCESS=LAST
         ST    15,RCCONTXT
         RETCO

*
         DS    0F
LASTREG  DS    CL68
INFOADAM DS    F

MLDCONA  DC    Y(ENDCA-MLDCONA)
         DS    L2
         DC    X'01'
         DC    'TIMEOUT CONTINGENCY A AT '
TIMEA    DS    CL8
ENDCA    EQU   *
*
CONTE    BALR  7,0 ------------------------------------------------------- (9)
         USING *,7
         GDATE TOD=TIMEE
         WROUT MLDCONE,ERROR
         ST    2,INFOEVE
         RETCO
*
INFOEVE  DS    F
MLDCONE  DC    Y(ENDCE-MLDCONE)
         DS    L2
         DC    X'01'
         DC    'TIMEOUT CONTINGENCY E AT '
TIME    DS    CL8
ENDCE    EQU   *
*
ERROR    CDUMP2 SCOPE=AREA
         TERM
```

```
CONTAAD  DC    A(CONTA)
CONTEAD  DC    A(CONTE)
NULL     DC    F'0'
RCCONTXT DS    F
*
ABBRADAM DS    F
ABBREVE  DS    F
ABBRA    DS    F
ABBRE    DS    F
*
MLDBAS1  DC    Y(ENDBAS1-MLDBAS1)
         DS    L2
         DC    X'01'
         DC    'BOTH SOLSIGS ISSUED AT '
TIMEBAS1 DS    CL8
ENDBAS1  EQU   *
MLDBAS2  DC    Y(ENDBAS2-MLDBAS2)
         DC    X'000001'
         DC    'QUEUES CHECKED AT '
TIMEBAS2 DS    CL8
ENDBAS2  EQU   *
MLDBAS3  DC    Y(ENDBAS3-MLDBAS3)
         DC    X'000001'
         DC    'EVENT ITEMS DISABLED AT '
TIMEBAS3 DS    CL8
ENDBAS3  EQU   *
*
WSADAM   DS    F
WSARC    DS    F
WSEVE    DS    F
WSERC    DS    F
ENDE     EQU   *
         END
```

Basisprozess

(1) Die Ereigniskennungen ADAM und EVE werden definiert. Die Adressen der Kurzkennungen lauten ABBRADAM und ABBREVE.

(2) Die Routine CONTA wird als Contingency-Prozess A definiert. Die Anfangsadresse (CONTA) ist unter der Adresse CONTAAD angegeben, die Adresse der Kurzkennung lautet ABBRA. Entsprechendes gilt für die folgende Definition der Routine CONTE als Contingency-Prozess E.

(3) Das Programm fordert mit einem **SOLSIG**-Aufruf von der Ereigniskennung ADAM ein Signal und gibt den Contingency-Prozess CONTA an. Wenn nach einer Wartezeit von 60 Sekunden das Signal nicht eingetroffen ist, soll die Ereignissteuerung den Contingency-Prozess CONTA starten. Das Programm läuft nach diesem **SOLSIG**-Aufruf weiter.

(4) Mit einem weiteren **SOLSIG**-Aufruf wird von EVE ein Signal angefordert. Für den Contingency-Prozess CONTE gilt die Standard-Wartezeit von 10 Minuten.

(5) Zur Demonstration wartet das Programm mit dem Makro **VPASS**. Dann wird es von dem Contingency-Prozess CONTA unterbrochen, dessen Wartezeit abgelaufen ist. Nachdem CONTA beendet ist, setzt das Programm mit der Prüfung der Ereigniswarteschlangen von ADAM und EVE fort.

(6) Das Programm löscht die Definitionen der Contingency-Prozesse CONTA und CONTE. Der **SOLSIG**-Aufruf an EVE mit dem Contingency-Prozess CONTE steht zu diesem Zeitpunkt noch in der Ereigniswarteschlange von EVE. CONTE kann noch gestartet werden, aber ein weiterer **SOLSIG**-Aufruf, in dem CONTE angegeben ist, würde zu diesem Zeitpunkt abgewiesen.

(7) Das Programm löscht die Ereigniskennung ADAM und dann EVE. Sobald EVE gelöscht wird, startet die Ereignissteuerung den Contingency-Prozess CONTE.

Contingency-Prozess CONTA

(8) Da ein Contingency-Prozess einen eigenen Registersatz hat, muss zu Beginn ein Basisregister definiert und geladen werden. Die Register R1, R2 und R3 sind nicht geeignet, weil sie eventuell von der Ereignissteuerung verwendet werden (siehe „Informationsübergabe an Contingency-Prozesse“ auf Seite 116). Der Ereignis-Informationscode, der dem Contingency-Prozess in Register R2 übergeben wurde, wird unter der Adresse INFOADAM gespeichert. Die Register des unterbrochenen Prozesses (hier ist es der Basisprozess) werden zur Demonstration mit dem **CONTXT**-Aufruf in den Bereich LASTREG übertragen. Der Contingency-Prozess beendet sich mit dem Aufruf **RETCO**.

Contingency-Prozess CONTE

(9) Ein Basisregister wird definiert und geladen. Der Ereignis-Informationscode wird unter INFOEVE abgespeichert und der Contingency-Prozess CONTE wird mit **RETCO** beendet.

Ablaufprotokoll des Dialogauftrags mit PCOSOL1

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,pcosol1), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,pcosol1)), -
//        test-support=*aid
%  ASS6011 ASSEMBLY TIME: 1238 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 171 MSEC
//end
%  ASS6012 END OF ASSEMBH
/load-executable-program library=macexmp.lib,element-or-symbol=pcosol1, -
/     test-options=*aid
%  BLS0523 ELEMENT 'PCOSOL1', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'PCOSOL1', VERSION ' ' OF '<date> <time>' LOADED
/%in dth1;%r
BOTH SOLSIGS ISSUED AT 16:06:33 ──────────────────────────────────────── (10)
TIMEOUT CONTINGENCY A AT 16:07:33 ────────────────────────────────────── (11)
QUEUES CHECKED AT 16:07:33 ───────────────────────────────────────────── (12)
TIMEOUT CONTINGENCY E AT 16:07:33 ────────────────────────────────────── (13)
EVENT ITEMS DISABLED AT 16:07:33
STOPPED AT LABEL: DTH1 , SRC_REF: 380, SOURCE: PCOSOL1 , PROC: PCOSOL1
/%d infoadam %x, %@(wsadam) -> %xl8 ──────────────────────────────────── (14)
*** TID: 00340180 *** TSN: 1PKB ***************************************
CURRENT PC: 00000162    CSECT: PCOSOL1  ***********************************
V'00000204' = INFOADAM + #'00000000'
00000204 (00000000) 00000004                                  ....
V'00000334' = PCOSOL1  + #'00000334'
00000334 (00000334) 000000D0 30000000                         ........
/%d infoeve %x, %@(wseve) -> %xl8 ────────────────────────────────────── (15)
V'0000025C' = INFOEVE  + #'00000000'
0000025C (00000000) 10000004                                  ....
V'0000033C' = PCOSOL1  + #'0000033C'
0000033C (0000033C) 00000001 28000000                         ........
/%r
```

(10) Das Programm fordert von jeder der beiden Ereigniskennungen ein Signal an.

(11) Die Wartezeit für den Contingency-Prozess CONTA ist verstrichen.

(12) Der Zustand der POSSIG-Warteschlange und der SOLSIG-Warteschlange jeder Ereigniskennung wird abgefragt.

(13) Der Contingency-Prozess CONTE läuft ab, weil die Ereigniskennung EVE von dem Basisprozess gelöscht wird.

(14) Ereigniskennung ADAM:
Der Ereignis-Informationscode X'04' bedeutet, dass die Wartezeit verstrichen ist, ohne dass das Ereignis eintrat.
Die Rücksprunginformation des Makros **CHKEI** (im Feld `WSARC`) X'30' bedeutet, dass die Warteschlangen leer sind: Der SOLSIG-Eintrag wurde also entfernt, weil der Contingency-Prozess CONTA abgelaufen ist.

(15) Ereigniskennung EVE:
Der Ereignisinformationscode X'10000004' bedeutet, dass die Ereigniskennung gelöscht wurde, bevor ein Ereignis eintrat.
Die Rücksprunginformation des Makros **CHKEI** (im Feld `WSERC`) X'28' bedeutet, dass die SOLSIG-Warteschlange nicht leer ist. Das Feld `WSEVE` X'01' gibt die Anzahl der bestehenden Einträge an, in diesem Fall ein Eintrag: Der SOLSIG-Eintrag besteht noch, weil der Contingency-Prozess CONTE noch nicht abgelaufen ist.

Teil 2: Dialogauftrag mit dem Programm PCOSOL2
ENTER-Auftrag ENTER.POSA mit dem Programm POSA
ENTER-Auftrag ENTER.POSE mit dem Programm POSE

Programm PCOSOL2

Das Programm PCOSOL2 unterscheidet sich von PCOSOL1 nur dadurch, dass es zwei ENTER-Aufträge startet, nachdem es die beiden **SOLSIG**-Aufrufe an die Ereigniskennungen ADAM und EVA gegeben hat. Von dem Quellprogramm PCOSOL2 wird hier nur der Teil gezeigt, der die Änderung enthält; der Rest ist identisch mit dem Programm PCOSOL1.

```
PCOSOL2  START
         PRINT NOGEN
         BALR  5,0
         USING *,5
         ENAEI EINAME=ADAM,SCOPE=GROUP,EIIDRET=ABBRADAM
         ENAEI EINAME=EVE,SCOPE=GROUP,EIIDRET=ABBREVE
         ENACO CONAME=A,COADAD=CONTAAD,COIDRET=ABBRA
         ENACO CONAME=E,COADAD=CONTEAD,COIDRET=ABBRE
         GDATE TOD=TIMEBAS1
         SOLSIG EIID=ABBRADAM,COID=ABBRA,LIFETIM=60
         CL    15,NULL
         BNE   ERROR
         SOLSIG EIID=ABBREVE,COID=ABBRE
         CL    15,NULL
         BNE   ERROR
         WROUT MLDBAS1,ERROR
         ENTER 'ENTER.POSE,JOB-CLASS=JCB00050'
         ENTER 'ENTER.POSA,JOB-CLASS=JCB00050'
         VPASS 90
CONNECT  GDATE TOD=TIMEBAS2
         ...
```

Programm POSA und ENTER-Datei ENTER.POSA

Der ENTER-Auftrag ENTER.POSA startet das Programm POSA. Das Programm POSA schließt sich der Ereignissteuerung unter der Ereigniskennung ADAM an und gibt einen **POSSIG**-Aufruf an ADAM. Bevor es die Ereigniskennung löscht, wartet es mit dem Makro **VPASS** 250 Sekunden, damit der Aufruf nicht aus der Ereigniswarteschlange entfernt wird, bevor das Programm PCOSOL2 den **SOLSIG**-Aufruf geben konnte.

```
POSA     START
         PRINT NOGEN
         BALR  5,0
         USING *,5
         ENAEI EINAME=ADAM,SCOPE=GROUP,EIIDRET=ABBRADAM
         GDATE TOD=TIMEA
         POSSIG EIID=ABBRADAM
         CL    15,=F'0'
         BE    OK
ERROR    EQU   *
*****  Error handling  *****
         TERM
*
OK       WROUT MLDPOSA,ERROR
         VPASS 250
         DISEI EIID=ABBRADAM
         TERM
ABBRADAM DS    F
MLDPOSA  DC    Y(ENDE-MLDPOSA)
         DC    X'000001'
         DC    'POSSIG FOR ADAM ISSUED AT '
TIMEA    DS    CL8
ENDE     EQU   *
         END
```

ENTER-Datei ENTER.POSA

```
/.POSA SET-LOGON-PARAMETERS
/ASSIGN-SYSDTA *SYSCMD
/START-ASSEMBH
//COMPILE SOURCE=*LIBRARY-ELEMENT(MACEXMP.LIB,POSA), -
//        COMPILER-ACTION=MODULE-GENERATION(MODULE-FORMAT=LLM), -
//        MODULE-LIBRARY=MACEXMP.LIB, -
//        LISTING=PARAMETERS(OUTPUT=*LIBRARY-ELEMENT(MACEXMP.LIB,POSA))
//END
/ASSIGN-SYSDTA *PRIMARY
/ASSIGN-SYSOUT PROT.POSA
/START-EXECUTABLE-PROGRAM LIBRARY=MACEXMP.LIB,ELEMENT-OR-SYMBOL=POSA
/EXIT-JOB
```

Programm POSE und ENTER-Datei ENTER.POSE

Der ENTER-Auftrag ENTER.POSE startet das Programm POSE. Das Programm schließt sich der Ereignissteuerung unter der Ereigniskennung EVE an und gibt einen **POSSIG**-Aufruf an EVE. Aus dem bei POSA genannten Grund wartet es, bevor es den Aufruf **DISEI** gibt.

```
POSE     START
         PRINT NOGEN
         BALR  5,0
         USING *,5
         ENAEI EINAME=EVE,SCOPE=GROUP,EIIDRET=ABBREVE
         GDATE TOD=TIMEE
         POSSIG EIID=ABBREVE
         CL    15,=F'0'
         BE    OK
ERROR    EQU   *
*****  Error handling  *****
         TERM
*
OK       WROUT MLDPOSE,ERROR
         VPASS 150
         DISEI EIID=ABBREVE
         TERM
ABBREVE  DS    F
MLDPOSE  DC    Y(END-MLDPOSE)
         DC    X'000001'
         DC    'POSSIG FOR EVE ISSUED AT '
TIMEE    DS    CL8
ENDE     EQU   *
         END
```

ENTER-Datei ENTER.POSE analog zu ENTER.POSA

Ablaufprotokoll des Dialogauftrags PCOSOL2 und der beiden ENTER-Aufträge

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,pcosol2), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,pcosol2)), -
//        test-support=*aid
%  ASS6011 ASSEMBLY TIME: 1217 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 166 MSEC//END
%  ASS6012 END OF ASSEMBH
```

```
/load-executable-program library=macexmp.lib,element-or-symbol=pcosol2, -
/     test-options=*aid
%  BLS0523 ELEMENT 'PCOSOL1', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'PCOSOL1', VERSION ' ' OF '<date> <time>' LOADED
/%in dth1;%r
BOTH SOLSIGS ISSUED AT 17:09:06 ------------------------------------------  (1)
%  JMS0066 JOB 'POSE' ACCEPTED ON 12-01-20 AT 17:09, TSN = 1PY0
%  JMS0066 JOB 'POSA' ACCEPTED ON 12-01-20 AT 17:09, TSN = 1PY1
```

Inzwischen geben die Enter-Aufträge POSE und POSA in ihr Ausgabeprotokoll folgende Nachrichten aus:

ENTER.POSE:

```
POSSIG FOR EVE ISSUED AT 17:09:21 ----------------------------------------  (2)
```

ENTER.POSA:

```
POSSIG FOR ADAM ISSUED AT 17:09:22 ---------------------------------------  (3)
```

Weiter im Ablauf des Dialogauftrags

```
TIMEOUT CONTINGENCY A AT 17:10:36 ----------------------------------------  (4)
TIMEOUT CONTINGENCY E AT 17:10:36
QUEUES CHECKED AT 17:10:36
EVENT ITEMS DISABLED AT 17:10:36
STOPPED AT LABEL: DTH1 , SRC_REF: 414, SOURCE: PCOSOL2 , PROC: PCOSOL2
/%d infoadam %x, %@(wsadam) -> %xl8 --------------------------------------  (5)
*** TID: 00340180 *** TSN: 1PKB ********************************************
CURRENT PC: 000001D2    CSECT: PCOSOL2  ***********************************
V'00000274' = INFOADAM + #'00000000'
00000274 (00000000) 00000000                                  ....
V'000003A4' = PCOSOL2  + #'000003A4'
000003A4 (000003A4) 00000140 30000000                         ... ....
/%d infoeve %x, %@(wseve) -> %xl8
V'000002CC' = INFOEVE  + #'00000000'
000002CC (00000000) 00000000                                  ....
V'000003AC' = PCOSOL2  + #'000003AC'
000003AC (000003AC) 00000158 30000000                         ........
/%r
```

(1) Das Programm PCOSOL2 definiert die beiden Ereigniskennungen ADAM und EVE und richtet an jede einen **SOLSIG**-Aufruf. Anschließend startet es zuerst den Enter-Auftrag ENTER.POSE und dann den ENTER-Auftrag ENTER.POSA.

(2) Der ENTER-Auftrag ENTER.POSE startet das Programm POSE, das die Ereigniskennung EVE definiert und einen **POSSIG**-Aufruf gibt.

(3) Der ENTER-Auftrag ENTER.POSA startet das Programm POSA, das die Ereigniskennung ADAM definiert und einen **POSSIG**-Aufruf gibt.

(4) Die Contingency-Prozesse CONTA und CONTE haben beide die Priorität 1. Welcher Contingency-Prozess zuerst gestartet wird, hängt davon ab, bei welcher Ereigniskennung zuerst ein **SOLSIG**- und ein **POSSIG**-Aufruf zusammentreffen. In diesem Fall liegt bei jeder Ereigniskennung schon ein **SOLSIG**-Aufruf vor, und bei der Ereigniskennung EVE trifft früher ein **POSSIG**-Aufruf ein als bei ADAM. Der Contingency-Prozess CONTE wird also zuerst gestartet, dann folgt der Contingency-Prozess CONTA.

(5) Nach Ablauf der beiden Contingency-Prozesse wird das Programm des Basisprozesses fortgesetzt. Es prüft die Warteschlangen bei der Ereigniskennungen: Sie sind leer. (Die Felder `WSERC` und `WSARC` enthalten nach dem Makroaufruf **CHKEI** X'30000000'.)
Die Ereignis-Informationscodes beider Contingency-Prozesse sind Null: Das erwartete Ereignis ist eingetroffen.

Teil 3: Dialogauftrag mit dem Programm PCOSOL3
ENTER-Aufträge ENTER.POSA und ENTER.POSE wie in Teil 2

Programm PCOSOL3

Das Programm PCOSOL3 unterscheidet sich von PCOSOL1 und PCOSOL2 dadurch, dass es die beiden ENTER-Aufträge zu Beginn startet, bevor es die beiden **SOLSIG**-Aufrufe gibt. Von dem Quellprogramm PCOSOL3 wird hier nur der Teil gezeigt, der die Änderung enthält; der Rest ist identisch mit dem Programm PCOSOL1.

```
PCOSOL3  START
         PRINT NOGEN
         BALR  4,0
         USING *,4
         ENTER 'ENTER.POSE,JOB-CLASS=JCB00050'
         ENTER 'ENTER.POSA,JOB-CLASS=JCB00050'
         ENAEI EINAME=ADAM,SCOPE=GROUP,EIIDRET=ABBRADAM
         ENAEI EINAME=EVE,SCOPE=GROUP,EIIDRET=ABBREVE
         ENACO CONAME=A,COADAD=CONTAAD,COIDRET=ABBRA
         ENACO CONAME=E,COADAD=CONTEAD,COIDRET=ABBRE
         VPASS 40
         GDATE TOD=TIMEBAS1
         SOLSIG EIID=ABBRADAM,COID=ABBRA,LIFETIM=60
         CL    15,NULL
         BNE   ERROR
         SOLSIG EIID=ABBREVE,COID=ABBRE
         CL    15,NULL
         BNE   ERROR
         WROUT MLDBAS1,ERROR
         VPASS 90
CONNECT  GDATE TOD=TIMEBAS2
         ...
```

ENTER-Auftrag ENTER.POSA mit dem Programm POSA: siehe Teil 2
ENTER-Auftrag ENTER.POSE mit dem Programm POSE: siehe Teil 2

Ablaufprotokoll des Dialogauftrags mit PCOSOL3 und der beiden ENTER-Aufträge

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,pcosol3), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,pcosol3)), -
//        test-support=*aid
%  ASS6011 ASSEMBLY TIME: 1260 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 168 MSEC
//end
%  ASS6012 END OF ASSEMBH
/load-executable-program library=macexmp.lib,element-or-symbol=pcosol3, -
/     test-options=*aid
%  BLS0523 ELEMENT 'PCOSOL3', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'PCOSOL3', VERSION ' ' OF '<date> <time>' LOADED
/%in dth1;%r
%  JMS0066 JOB 'POSE' ACCEPTED ON 12-01-20 AT 17:20, TSN = 1PY7 ————————  (1)
%  JMS0066 JOB 'POSA' ACCEPTED ON 12-01-20 AT 17:20, TSN = 1PY8
```

Inzwischen geben die Enter-Aufträge POSE und POSA in ihr Ausgabeprotokoll folgende Nachrichten aus:

ENTER.POSE:

```
POSSIG FOR EVE ISSUED AT 17:20:19 ————————————————————————————————————  (2)
```

ENTER.POSA:

```
POSSIG FOR ADAM ISSUED AT 17:20:20 ————————————————————————————————————  (3)
```

Weiter im Ablauf des Dialogauftrags

```
TIMEOUT CONTINGENCY A AT 17:20:52 ————————————————————————————————————  (4)
TIMEOUT CONTINGENCY E AT 17:20:52
BOTH SOLSIGS ISSUED AT 17:20:52 ————————————————————————————————————  (5)
QUEUES CHECKED AT 17:22:22
EVENT ITEMS DISABLED AT 17:22:22
STOPPED AT LABEL: DTH1 , SRC_REF: 417, SOURCE: PCOSOL3 , PROC: PCOSOL3
/%d infoadam %x, %@(wsadam) -> %xl8
*** TID: 00340180 *** TSN: 1PKB ****************************************
CURRENT PC: 000001DA    CSECT: PCOSOL3  ********************************
V'0000027C' = INFOADAM + #'00000000'
0000027C (00000000) 00000000                                ....
V'000003AC' = PCOSOL3  + #'000003AC'
```

```
000003AC (000003AC) 00000148 30000000                        ........
/%d infoeve %x, %@(wseve) -> %xl8
V'000002D4' = INFOEVE  + #'00000000'
000002D4 (00000000) 00000000                                 ....
V'000003B4' = PCOSOL3  + #'000003B4'
000003B4 (000003B4) 00000160 30000000                        ...-....
/%r
```

(1) Das Programm PCOSOL3 startet zuerst den ENTER-Auftrag ENTER.POSE und dann den ENTER-Auftrag ENTER.POSA. (Die Ereignissteuerung unter ADAM und EVE eröffnet das Programm erst, nachdem (zur Demonstration) einige Zeit verstrichen ist; die **SOLSIG**-Aufrufe für ADAM und EVE gibt das Programm noch später.)

(2) Das Programm PCOPOSE (gestartet im ENTER-Auftrag ENTER.POSE) definiert die Ereigniskennung EVE und gibt einen **POSSIG**-Aufruf für EVE.

(3) Das Programm PCOPOSA (gestartet im ENTER-Auftrag ENTER.POSA) definiert die Ereigniskennung ADAM und gibt einen **POSSIG**-Aufruf für ADAM.

(4) Sobald das Programm PCOSOL3 (im Dialogauftrag) den **SOLSIG**-Aufruf für die Ereigniskennung ADAM gegeben hat, wird es durch den Start des Contingency-Prozesses A unterbrochen, da bei ADAM schon ein **POSSIG**-Aufruf vorliegt. Nach dem Ablauf von A kann das Programm fortsetzen und den **SOLSIG**-Aufruf für EVE geben. Auch hier liegt schon ein **POSSIG**-Aufruf vor, und das Programm wird wieder unterbrochen, bis der Contingency-Prozess E abgelaufen ist.

(5) Das Programm prüft die Warteschlangen beider Ereigniskennungen:
Sie sind leer. (Die Felder `WSERC` und `WSARC` enthalten nach dem Makroaufruf **CHKEI** den Wert X'30000000'.)
Die Ereignis-Informationscodes beider Contingency-Prozesse sind Null: Das erwartete Ereignis ist eingetroffen.

Für ein weiteres **Beispiel** siehe Beschreibung des Makros **POSSIG** (Seite 760).

4.3.7 STXIT-Verfahren mit Contingency-Verarbeitung

Makro	Kurzbeschreibung
CONTXT	liest oder schreibt u.a. in den Registern des von ihm unterbrochenen Prozesses oder des Basisprozesses
EXIT	beendet einen STXIT-Prozess
LEVCO	ändert die Verarbeitungsebene (Priorität) des aufrufenden STXIT-Prozesses während des Ablaufs
SETIC	setzt Zeitgeberintervalle (CPU-Zeit/Realzeit)
STXIT	legt benutzereigene STXIT-Prozesse fest, mit denen das System die Verarbeitung fortsetzt, wenn eine Programmunterbrechung auftritt

Allgemeines

Während eines Programmlaufs auftretende Ereignisse, wie z.B.

– ungültiger SVC, ungültiger Operationscode
– Datenfehler, Überlauf
– Ende Programmlaufzeit, Break-/Escape-Unterbrechungen, INFORM-PROGRAM-Kommando
– Adressfehler

beeinflussen in der Regel den weiteren Programmablauf.
Das STXIT-Verfahren bietet dem Anwender die Möglichkeit, auf derartige Ereignisse mit eigenen (STXIT-) Routinen zu reagieren: bei Eintritt einer Programmunterbrechung wird die STXIT-Routine zum Ablauf gebracht und anschließend (im Normalfall) der unterbrochene Basisprozess fortgesetzt. Der Basisprozess ist das ablaufende Programm, das mit dem Kommando START-PROGRAM gestartet wurde. Die STXIT-Routinen sind Bestandteil des Programmes, laufen aber als selbstständige Prozesse ab.

Beispiele

- Ein Programmfehler (ungültiger SVC, ungültiger Operationscode, Datenfehler, Überlauf, ...) führt in der Regel zum Programmabbruch. Mit einer STXIT-Routine kann die Beendigung des Programmes abgefangen werden.
- Ein Break (K2-Taste) führt in der Regel zur Unterbrechung des Programmes (das Betriebssystem erwartet jetzt BS2000-Kommandos). Mit einer STXIT-Routine kann der Programmierer im Programm auf dieses Unterbrechungsereignis vorbeugend reagieren.

Beschreibung des STXIT-Verfahrens

- STXIT-Verwaltung

  Die mit dem STXIT-Verfahren zu behandelnden Unterbrechungsereignisse sind in STXIT-Ereignisklassen zusammengefasst (siehe Tabelle 9 auf Seite 139). Der Anwender kann mit einem Aufruf des Makros **STXIT** jeder Ereignisklasse eine STXIT-Routine zuordnen. Die angegebenen Zuordnungen werden systemintern in einen STXIT-Verwaltungsblock eingetragen. In weiteren **STXIT**-Aufrufen kann der Anwender unter Bezugnahme auf den angelegten STXIT-Verwaltungsblock die eingetragenen Zuordnungen modifizieren oder ergänzen - oder einen neuen STXIT-Verwaltungsblock mit weiteren Zuordnungen anlegen lassen. Die STXIT-Verwaltungsblöcke werden untereinander verkettet.

  Mit dem STXIT-Verfahren können in einem Programm(-system) bis zu 100 STXIT-Verwaltungsblöcke angelegt werden. Damit ist die Koexistenz mehrerer STXIT-Routinen für dieselbe Ereignisklasse möglich. Diese Erweiterung ist für die Anwendung in Programmsystemen vorteilhaft: bei Eintritt eines Ereignisses kann jedes Unterprogramm die für dieses Ereignis erforderliche Kontrolle erhalten.

- STXIT-Routine

  Eine STXIT-Routine ist ein Programmabschnitt in einem Programm. In dem Programmabschnitt reagiert der Programmierer auf das Unterbrechungsereignis. Die Routine endet mit dem Makro **EXIT**. Bei Eintritt des Unterbrechungsereignisses wird die STXIT-Routine zum Ablauf gebracht.
  Das konkrete Unterbrechungsereignis wird mit einem Ereigniscode beschrieben und dem Programm im Register R3 übergeben, siehe Tabelle 9 auf Seite 139.

  Wenn das Unterbrechungsereignis eintritt, läuft die STXIT-Routine in einem eigenen Prozess ab (STXIT-Prozess). Das Betriebssystem verwendet dabei die Register R1, R3 und R4 zur Verständigung mit der Routine.
  Die Register enthalten folgende Informationen:

R1: STXIT-Meldung, falls bei STXIT-Operand STXMSG angegeben wurde

R3: im rechten Byte: Ereigniscode des Unterbrechungsereignisses
(siehe Tabelle 9 auf Seite 139)

R4: im rechten Byte: SVC-Nummer bei Ereignisklasse SVC oder Funktionstastencode bei Ereignisklasse ESCPBRK. Funktionstastencodes können dem Handbuch „TIAM“ [16] oder der Tabelle „Tabelle der normierten Funktionstastencodes“ auf Seite 1193 entnommen werden. Diese Funktionalität wird nur zur Verfügung gestellt, wenn der TIAM-Partner ein Terminal ist und keine Applikation (z.B. OMNIS).

- STXIT-Prozess

– die Definition eines STXIT-Prozesses

  STXIT-Prozesse sind Contingency-Prozesse (das bedeutet: eigener Process Control Block (PCB) und damit eigener Registersatz; eigene Verarbeitungsebene).
  Die Startadresse des STXIT-Prozesses ist die Adresse der STXIT-Routine, die im **STXIT**-Aufruf angegeben wurde. Der STXIT-Prozess erhält standardmäßig die höchstmögliche Verarbeitungsebene 127.

– der Ablauf von STXIT-Prozessen

  Ein STXIT-Prozess wird im Normalfall sofort bei Eintritt des entsprechenden Unterbrechungsereignisses gestartet. Treten während des Ablaufs weitere Unterbrechungsereignisse ein, entsteht eine Prozesswarteschlange. Verarbeitungsebene und Einreihungsprinzip (LIFO/FIFO) eines STXIT-Prozesses bestimmen seinen Platz in der Prozesswarteschlange.

  LIFO (Last In First Out) bedeutet:
  Der Prozess wird nach seiner Aktivierung an der Spitze der Prozesse mit seiner Verarbeitungsebene eingereiht. Vor ihm können nur Prozesse mit höherer Verarbeitungsebene eingeordnet sein.
  STXIT-Prozesse für die Ereignisklassen mit zeitunabhängigen Ereignissen (wie Programmüberprüfung, nicht behebbarer Programmfehler, ..., Programmbeendigung) werden nach dem LIFO-Prinzip eingeordnet (Tabelle 9 auf Seite 139).

  FIFO (First In First Out) bedeutet:
  Der Prozess wird nach seiner Aktivierung am Ende der Prozesse mit seiner Verarbeitungsebene eingereiht. Vor ihm können aufsteigend früher aktivierte Prozesse seiner Verarbeitungsebene und Prozesse mit höherer Verarbeitungsebene eingeordnet sein.
  STXIT-Prozesse für die Ereignisklassen mit zeitabhängigen Ereignissen (Zeitgeber Realzeit, Zeitgeber CPU-Zeit, Ende Programmlaufzeit) werden nach dem FIFO-Prinzip eingeordnet (Tabelle 9 auf Seite 139).

  Ausgehend vom laufenden Basisprozess soll im Folgenden das Zusammenspiel mehrerer STXIT-Prozesse erläutert werden. Es sei vereinfachend angenommen:
  alle STXIT-Prozesse haben dieselbe Verarbeitungsebene und nach Beendigung der STXIT-Prozesse wird der Basisprozess fortgesetzt.

Für das Beispiel sei angenommen:

a) in dem Programm oder Programmsystem ist einer Ereignisklasse immer nur eine STXIT-Routine zugeordnet,

b) in dem Programm oder Programmsystem sind den Ereignisklassen jeweils mehrere STXIT-Routinen zugeordnet (STXIT-Parallelität).

zu a):

Bei Eintritt eines Unterbrechungsereignisses wird der Basisprozess (Verarbeitungsebene 0) unterbrochen und der entsprechende STXIT-Prozess (Verarbeitungsebene 127) gestartet. Nach Beendigung des STXIT-Prozesses wird der Basisprozess fortgesetzt, wenn zwischenzeitlich kein weiteres Unterbrechungsereignis aufgetreten ist. Tritt während des STXIT-Prozesses ein weiteres Unterbrechungsereignis ein, gilt entsprechend dem Einreihungsprinzip:

– der aktuelle STXIT-Prozess wird durch einen neuen STXIT-Prozess unterbrochen, wenn dieser nach dem Prinzip LIFO eingereiht wird (geschachtelter Ablauf). Nach Beendigung des neuen STXIT-Prozesses wird der unterbrochene STXIT-Prozess fortgesetzt und nach dessen Beendigung der Basisprozess, wenn während dieser Abläufe kein neues Unterbrechungsereignis aufgetreten ist. An der Schachtelung können auch mehr als 2 STXIT-Prozesse beteiligt sein. Das gilt auch, wenn das Unterbrechungsereignis immer wieder derselben Ereignisklasse angehört. Ein neuer STXIT-Prozess derselben STXIT-Routine wird dann gestartet. In diesem Fall ist aber die Anzahl der Schachtelungen begrenzt (Angabe im Makroaufruf **STXIT** und max. Schachtelungstiefe in der Tabelle 9 auf Seite 139).
– Der laufende STXIT-Prozess wird beendet und anschließend der neue STXIT-Prozess gestartet, wenn der neue STXIT-Prozess nach dem FIFO-Prinzip eingereiht wird. Die Anzahl der nacheinander ablaufenden Prozesse ist begrenzt, wenn das Unterbrechungsereignis immer wieder derselben Ereignisklasse angehört (max. Schachtelungstiefe in Tabelle 9 auf Seite 139). Der Basisprozess wird fortgesetzt, wenn während dieser Abläufe keine neuen Unterbrechungsereignisse aufgetreten sind.

zu b):

Bei Eintritt eines Unterbrechungsereignisses werden für alle STXIT-Routinen dieser Ereignisklasse STXIT-Prozesse erzeugt. Diese werden - beginnend beim zuerst angelegten STXIT-Verwaltungsblock mit dieser Ereignisklasse - entsprechend ihrem Einreihungsprinzip in die Prozesswarteschlange eingeordnet. Das bedeutet für den Ablauf:

– STXIT-Routinen einer Ereignisklasse mit dem LIFO-Prinzip:
der Basisprozess wird unterbrochen und die im zuletzt angelegten STXIT-Verwaltungsblock eingetragene STXIT-Routine kommt zum Ablauf, anschließend die im zuvor angelegten STXIT-Verwaltungsblock eingetragene STXIT-Routine, usw.

– STXIT-Routinen einer Ereignisklasse mit dem FIFO-Prinzip:
der Basisprozess wird unterbrochen und die im zuerst angelegten STXIT-Verwaltungsblock eingetragene STXIT-Routine kommt zum Ablauf, anschließend die im danach angelegten STXIT-Verwaltungsblock eingetragene STXIT-Routine, usw.

– In jeder STXIT-Routine kann im Makro **EXIT** angegeben werden, ob die Prozessfolge für dieselbe Ereignisklasse abgebrochen oder fortgesetzt werden soll.

Tritt während eines STXIT-Prozesses ein neues Unterbrechungsereignis ein, werden (wiederum) für alle STXIT-Routinen der betreffenden Ereignisklasse STXIT-Prozesse erzeugt und entsprechend ihrem Einreihungsprinzip in die Prozesswarteschlange eingereiht. Das ist auch so, wenn das Unterbrechungsereignis immer wieder derselben Ereignisklasse angehört. Für eine unterbrochene STXIT-Routine wird aber nur dann ein neuer STXIT-Prozess erzeugt, wenn die spezifizierte Schachtelungstiefe noch nicht erreicht ist. Dieser Ablauf ist für das LIFO-Prinzip im Bild unten dargestellt.

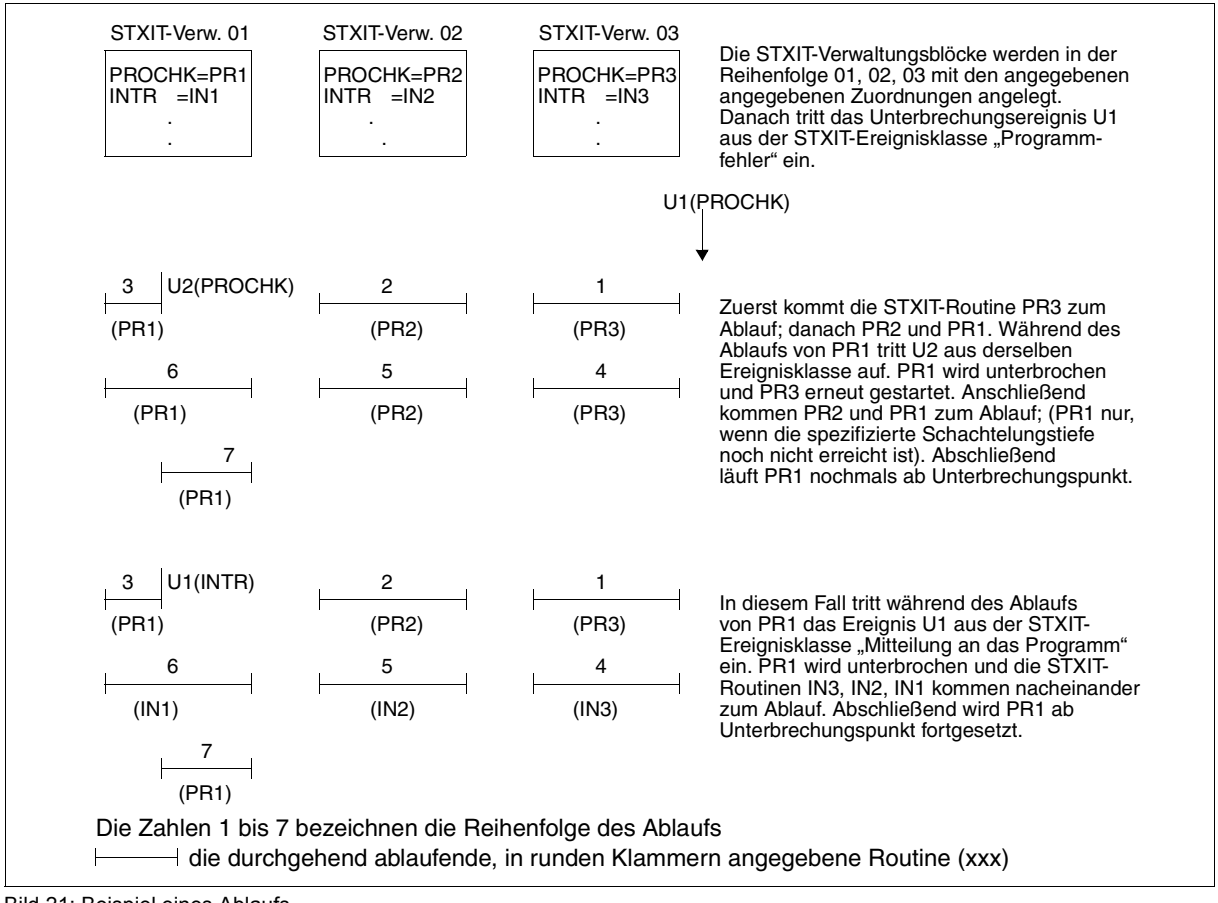


Bild 21: Beispiel eines Ablaufs

– Steuerung von STXIT-Prozessen

  Ändern der Verarbeitungsebene (Prozesspriorität):
  Wie angeführt, werden STXIT-Prozesse standardmäßig auf der Verarbeitungsebene 127 gestartet und bei gleicher Verarbeitungsebene nach einem vorgegebenen Einreihungsprinzip (LIFO/FIFO) in die Prozesswarteschlange eingereiht. Für den Wertebereich gilt:

Basisprozess:	Verarbeitungsebene 0 - 127 Standardwert 0
STXIT-Prozess:	Verarbeitungsebene 1 - 127 Standardwert 127

  Der Basisprozess (der mit dem Kommando START-PROGRAM gestartet wurde) hat die Verarbeitungsebene 0 und wird stets von einem STXIT-Prozess unterbrochen. Während des Ablaufs der STXIT-Routine kann mit dem Makro **LEVCO** die Verarbeitungsebene des STXIT-Prozesses im angegebenen Bereich geändert werden. Der Anwender kann somit die Reihenfolge der ablaufenden STXIT-Prozesse beeinflussen. Es ist aber zu beachten, dass ein laufender Prozess seine Verarbeitungsebene nicht unter die eines schon gestarteten und unterbrochenen Prozesses senken kann.

  Zugriff auf den PCB des unterbrochenen Prozesses:
  Mit dem Makro **CONTXT** kann auf den PCB des aktuell unterbrochenen Prozesses oder auf den PCB des Basisprozesses lesend und schreibend zugegriffen werden. Der Aufrufer hat Zugriff zu den allgemeinen Registern, den Gleitpunktregistern und zum Befehlszähler; außerdem wird ihm über den Returncode mitgeteilt, ob im aktuellen Unterbrechungszustand der PCB bereits durch einen **CONTXT**-Aufruf modifiziert wurde.

Einschränkungen

– Der Ereignisklasse „SVC“ kann in einem Programm-(system) nur eine STXIT-Routine zugeordnet werden. Die Zuordnung muss im ersten **STXIT**-Aufruf erfolgen oder sich auf den zuerst angelegten STXIT-Verwaltungsblock beziehen.

– Für die Ereignisklassen „Zeitgeber Realzeit“ und „Zeitgeber CPU-Zeit“ ist zu beachten, dass zu einem Zeitpunkt immer nur ein Zeitintervall für Realzeit und für CPU-Zeit gesetzt ist (Makro **SETIC**).

– Der Aufruf des Makros **TERM** in dem Programm(-system) führt zur Aktivierung aller der Ereignisklasse „TERM“ zugeordneten STXIT-Routinen - wenn die jeweils vorhergehende mit **EXIT** CONTINU=YES beendet wurde. Ein erneuter **TERM**-Aufruf (auch in einer STXIT-Routine) führt wunschgemäß zur sofortigen Programmbeendigung.

– Bei Überschreiten der spezifizierten Schachtelungstiefe wird das Programm mit einem Userdump beendet und das Unterbrechungsgewicht im PCB des Basisprozesses mit X'06' überschrieben. (X'06' ist kein Ereignis, auf das der Anwender mit einer STXIT-Routine reagieren kann).

Folgende Tabelle zeigt die STXIT-Ereignisklassen und die zugehörigen Unterbrechungsereignisse:

STXIT-Ereignisklasse	**Unterbrechungsereignis**	**Ereigniscode**	**Warteschlangen-Einreihung**	**max. Schachtelungstiefe**
Programmfehler	unzulässiger SVC unzulässiger Operationscode Datenfehler Exponentenüberlauf Divisionsfehler oder negative Quadratwurzel Mantisse = 0 Exponentenunterlauf Dezimalüberlauf Festpunktüberlauf	X'04' X'58' X'60' X'64' X'68' X'6C' X'70' X'74' X'78'	LIFO	127
Intervallzeitgeber für CPU-Zeit	„SETIC-Intervall“ abgelaufen für CPU-Zeit	X'20'	FIFO	127
Intervallzeitgeber für Realzeit	„SETIC-Intervall“ abgelaufen für Realzeit Sommer-/Winterzeitumstellung	X'A0' X'C0'	FIFO	127
Ende Programmlaufzeit	CPU-Zeitgrenze für die Task bzw. das Programm überschritten	X'80'	FIFO	0
nicht behebbarer Programmfehler	privilegierter SVC Zugriff auf eine nicht vorhandene Speicherseite privilegierte Operation Adressenfehler (z.B. Ausrichtungsfehler oder falsches Register) XA-Fehler bei SVC-Aufruf (im 31-Bit-Mode den 24-Bit-Datenbereich benutzt) Realtimer (Condition Error) Ausrichtungsfehler des Datenbereichs bei SVC-Aufruf Validierungsfehler Ungültige UNIT-Nr. im Standardheader	X'08' X'48' X'54' X'5C' X'9C' X'A4' X'AC' X'B0' X'C4'	LIFO	127

Tabelle 9: STXIT-Ereignisklassen und zugehörige Unterbrechungsereignisse

STXIT-Ereignis-klasse	Unterbrechungsereignis	Ereignis-code	Warteschlangen-Einreihung	max. Schach-telungstiefe
Mitteilung an das Programm	INFORM-PROGRAM-Kom-mando	X'44'	LIFO	127
ESCPBRK	BREAK/ESCAPE (über Tasten)	X'84'	LIFO	127
Programmbeendi-gung durch asyn-chrone Ereignisse	vom System erkannter Fehler, z.B. Fehler im System, Leitungverlust START-PROGRAM, LOAD-PROGRAM, ABEND, EXIT-JOB, CANCEL-JOB Adress-Übersetzungsfehler wegen Hardwarefehler Hardwarefehler(CPU) Erzwungenes Entladen eines Subsystems (Systemverw.) nicht behebbarer DMS-Fehler	X'88' X'8C' X'94' X'A8' X'B8' X'BC'	LIFO	0
Programmbeendi-gung durch syn-chrone Ereignisse	TERM-SVC aus TU-Programm Programmbeendigung durch CMD-/LGOFF-Makro	X'90' X'98'	LIFO	0
SVC-Unterbre-chung	SVC-Aufruf eines angegebe-nen SVCs	X'50'	LIFO	127
Hardwarefehler	Ein-/Ausgabefehler bei Data-In-Virtual-Technik	X'28'	LIFO	0
Live Migration	Live Migration	X'D0'	FIFO	127

Tabelle 9: STXIT-Ereignisklassen und zugehörige Unterbrechungsereignisse

Beispiel zur Programmstruktur mit STXIT-Routinen in einem Programmsystem.

Das Programmsystem besteht aus einem Hauptprogramm und mehreren Subprogrammen. Durch das Hauptprogramm wird ein STXIT-Verwaltungsblock angelegt und die der Ereignisklasse „Programmbeendigung“ zugeordnete STXIT-Routine „TERMR1“ eingetragen. Durch das Subprogramm A wird ein weiterer STXIT-Verwaltungsblock angelegt und ebenfalls für die Ereignisklasse „Programmbeendigung“ eine STXIT-Routine „TERMR2“ eingetragen. Bei Eintritt eines Ereignisses aus dieser Ereignisklasse laufen zu diesem Zeitpunkt die STXIT-Routinen „TERMR2“ und „TERMR1“ nacheinander ab.

Zu einem späteren Zeitpunkt wird durch das Subprogramm A der zuletzt angelegte STXIT-Verwaltungsblock (unter Bezugnahme auf seine ID) modifiziert; die Zuordnung für die Ereignisklasse „Programmbeendigung“ wird gelöscht und die der Ereignisklasse „ABEND“

zugeordnete STXIT-Routine „ABNDR“ eingetragen. Ab diesem Zeitpunkt wird bei Eintritt eines Ereignisses aus der Ereignisklasse „Programmbeendigung“ nur noch die STXIT-Routine „TERMR1“ aktiviert.

Programmstruktur

```
BEISP    START              * HAUPTPROGRAMM
         BALR  ...
         USING ...
***                         * 1. STXIT-Aufruf in dem Hauptprogramm
         STXIT STXDNEW=STXDIDF1,TERM=(TERMR1)
         :
TERMR1   EQU   *             *  STXIT-Routine für "Programmbeendigung"
         BALR  ...
         USING ...
         :
         EXIT
         :
STXIDF1  DC    F             * 4-Byte-Feld für die ID des STXIT-
         END                 * Verwaltungsblockes

         :                   * UNTERPROGRAMM A
         LA    6,STXDIDF2
***                         * 2. STXIT-Aufruf (Unterprogramm A)
         STXIT STXDNEW=(6),TERM=(TERMR2)
         :
***                         * Update des 2. STXIT-Verwaltungsblockes.
***                         * Der Unterbrechungsausgang für TERM wird
***                         * geschlossen und für die Ereignisklasse
***                         * ABEND wird eine STXIT-Routine spezifiziert.
         STXIT STXDID=STXIDF2,TERM=(CLOSE),ABEND=(ABNDR)
         :
TERMR2   EQU   *             * STXIT-Routine für "Programmbeendigung"
         BALR  ...           * (Unterprogramm A)
         USING ...
         :
         EXIT
         :
ABNDR    EQU   *             * STXIT-Routine für "ABEND" (Unterprogramm A)
         BALR  ...
         USING ...
         :
         EXIT
         :
STXDIDF2 DC    F             * 4-Byte-Feld für die ID des zweiten STXIT-
         END                 * Verwaltungsblockes
```

4.3.8 Distributed-Lock-Manager (DLM)

Makro	Kurzbeschreibung
LKCAN	löscht Lock-Anforderungen
LKCVT	konvertiert Lock-Anforderungen
LKDEQ	gibt Lock-Anforderungen frei
LKENQ	generiert einen Lock
LKEQU	generiert DLM-eigene Layouts
LKINF	gibt Informationen über Locks aus
LKLSB	generiert das Layout des Lock-Status-Blocks

Der Distributed-Lock-Manager (DLM) ermöglicht es Tasks, die auf verschiedenen Knoten eines XCS-Clusters (XCS = cross coupled system) ablaufen, ihre Zugriffe auf gemeinsam benutzbare Ressourcen (z.B. Dateien, Datensätze, Datenbank-Blöcke, Geräte, ...) zu serialisieren. Welche Ressource sich hinter einem Lock verbirgt, ist für den DLM nicht sichtbar: Die Absprache liegt in der Verantwortung der Benutzer.
Die DLM-Funktionalität ist vergleichbar mit der Serialisierung von Tasks auf einem einzelnen (lokalen) Knoten (Rechner), jedoch mit der zusätzlichen Wahlmöglichkeit zwischen knotenlokalen Locks, d.h. nur auf dem eigenen Rechner sichtbaren Locks, und Cluster-Locks, die auf allen Rechnern (=Knoten) im XCS-Verbund sichtbar sind. Der Aufbau eines XCS-Verbunds erfordert das kostenpflichtige Software-Produkt HIPLEX MSCF [26].
Darüber hinaus bietet DLM auch unterschiedliche Lock-Modi an, mit denen neben einer Serialisierung des (exklusiven) Zugriffs z.B. auch paralleles, gleichzeitiges Lesen mehrerer Anwender möglich ist.

Ein Lock wird aus DLM-Sicht identifiziert durch

– einen vom Benutzer frei wählbaren Lock-Namen
– den lokalen Geltungsbereich (Operand SCOPE)
– den globalen Geltungsbereich (Operand NAMRNGE)

Anhand dieser Angaben wird den unterschiedlichen Aufrufern (Tasks) der Lock zugeordnet, mit dem sie arbeiten wollen. Nur dann, wenn alle drei Angaben gleich sind, beziehen sich zwei Aufrufer auf den gleichen Lock.
Zur Vereinfachung erhält der Aufrufer dann eine Lock-Kurzkennung, die in allen nachfolgenden Aufrufen zur Identifikation des Locks verwendet wird.

Locks des DLM sind nicht mit Locks von irgendeinem anderen Lock-Manager verknüpft. Um sicherzustellen, dass eine Ressource richtig geschützt ist, müssen alle Benutzer dieser Ressource den gleichen Lock-Mechanismus verwenden.

4.3.8.1 Aufbau eines DLM-Locks

Ein Lock, der über den DLM verwaltet wird, besteht aus mindestens zwei Teilen:

– Den Benutzer-unabhängigen Teil, der die allgemeinen Informationen und die Verwaltungsdaten des Locks enthält. Er existiert nur einmal pro Lock und ist für alle Benutzer des Locks gültig.
– Pro Benutzer, der mit dem Lock arbeitet, einen Benutzer-abhängigen Teil, der die Informationen enthält, die spezifisch für den jeweiligen Benutzer des Locks sind. Ein Benutzer-abhängiger Teil des DLM-Locks wird nachfolgend auch Lock-Anforderung genannt.

Alle Benutzer-abhängigen Teile eines Locks, d.h. alle Lock-Anforderungen, sind mit dem Benutzer-unabhängigen Teil des Locks so verknüpft, dass sich daraus der Zustand der einzelnen Lock-Anforderung ergibt. Die möglichen Zustände einer Lock-Anforderung sind abhängig vom Modus, in dem sie angefordert werden.

- **Lock-Modus**

  Der DLM bietet sechs verschiedene Lock-Modi an:

  NU Null-Modus. Eine zugeteilte Lock-Anforderung mit diesem Modus ist mit allen anderen Lock-Anforderungen kompatibel. Sie darf aber nicht auf die Ressource zugreifen.

  CR Concurrent-Read-Modus. Dem Lock-Halter wird ein ungeschützter Lesezugriff auf die Ressource gewährt. D.h. es sind auch andere Lese- oder Schreibzugriffe auf diese Ressource zur gleichen Zeit erlaubt. Erlaubt sind andere Lock-Halter nur im NU-Modus, im CR-Modus, im CW-Modus, im PR-Modus und im PW-Modus.

  CW Concurrent-Write-Modus. Dem Lock-Halter wird ein ungeschützter Schreibzugriff auf die Ressource gewährt. D.h. es sind auch andere Lese- oder Schreibzugriffe auf diese Ressource zur gleichen Zeit erlaubt. Erlaubt sind andere Lock-Halter nur im NU-Modus, im CW-Modus oder im CR-Modus.

  PR Protected-Read-Modus. Dem Lock-Halter wird ein geschützter Lesezugriff auf die Ressource gewährt. D.h. es sind keine anderen Schreibzugriffe zur gleichen Zeit erlaubt. Erlaubt sind andere Lock-Halter nur im NU-Modus, im CR-Modus oder im PR-Modus.

  PW Protected-Write-Modus. Dem Lock-Halter wird ein geschützter Schreibzugriff auf die Ressource gewährt. Erlaubt sind andere Lock-Halter nur im CR-Modus und im NU-Modus.

  EX Exklusiv-Modus. Nur der Lock-Halter darf auf die Ressource zugreifen. Es sind keine anderen Lese- oder Schreibzugriffe erlaubt. Lock-Halter im NU-Modus sind zwar kompatibel zum EX-Modus, aber sie dürfen nicht auf die Ressource zugreifen.

- **Kompatibilität der Lock-Modi**

schwach
↑
Lock-Schutz
↓
stark

stärkster aktuell zugeteilter Lock-Modus	angeforderter Lock-Modus					
	NU 1	CR 2	CW 3	PR 4	PW 5	EX 6
NU 1	+	+	+	+	+	+
CR 2	+	+	+	+	+	-
CW 3	+	+	+	-	-	-
PR 4	+	+	-	+	-	-
PW 5	+	+	-	-	-	-
EX 6	+	-	-	-	-	-

\+ angeforderter Lock-Modus ist kompatibel zum zugeteiltem Lock-Modus
\- angeforderter Lock-Modus ist inkompatibel zum zugeteiltem Lock-Modus

- **Zustände einer Lock-Anforderung**

  Eine Lock-Anforderung kann sich in einem der folgenden Zustände befinden (vom Benutzer aus gesehen):

  GRANTED — Die Lock-Anforderung für einen Lock ist zugeteilt und ist mit allen anderen bereits zugeteilten Lock-Anforderungen kompatibel. Neue Lock-Anforderungen werden nur zugeteilt, wenn sie kompatibel sind oder die zugeteilte Lock-Anforderung freigegeben wird.

  CONVERTING — Die Lock-Anforderung ist bereits in einem Modus zugeteilt, der Modus soll jedoch geändert werden. Führt diese Konvertierung zu Inkompatibilitäten mit anderen zugeteilten Lock-Anforderungen, muss sie warten, bis der neue Lock-Modus kompatibel zugeteilt werden kann. Die zu konvertierende Lock-Anforderung bleibt in ihrem ursprünglichen Lock-Modus, bis sie konvertiert werden kann (oder bis sie das Zeitlimit überschreitet oder gelöscht wird).

  WAITING — Die Lock-Anforderung wartet auf ihre erstmalige Zuteilung. Die Zuteilung erfolgt, sobald sie zu allen bereits zugeteilten Lock-Anforderungen kompatibel ist.

  Lock-Anforderungen im Zustand CONVERTING werden vor allen neuen Lock-Anforderungen im Zustand WAITING behandelt.

  Existiert eine Lock-Anforderung im Zustand WAITING, die inkompatibel zu den aktuell zugeteilten Lock-Anforderungen ist, blockiert sie alle nachfolgenden Lock-Anforderungen für diesen Lock, auch wenn diese kompatibel wären (außer Locks im NU-Modus).

  Zur Kompatibilität der Lock-Modi siehe Tabelle oben.

- **Lock-Value-Block**

  Der Lock-Value-Block (LVB) ist ein kleiner Speicherbereich, der einem Lock direkt zugeordnet ist. Er kann gelesen oder geschrieben werden, je nachdem, in welchem Lock-Modus die Lock-Anforderung zugeteilt ist und ob die Lock-Anforderung den Lock erhält (stärkerer Lock-Modus) oder freigibt (schwächerer Lock-Modus). Über den LVB können die Benutzer des Locks untereinander Informationen austauschen.

  Der LVB besteht aus einem 16 Byte großen Bereich (Lock-Wert), der vom Benutzer mit beliebigen Informationen beschrieben werden kann. Der Lock-Wert, den der Benutzer schreiben will oder den er bei einer Leseanforderung geliefert bekommt, befindet sich im Lock-Status-Block.

  Der LVB existiert solange, wie irgendeine Lock-Anforderung für diesen Lock existiert. Wird die letzte (einzige) Lock-Anforderung dieses Locks freigegeben, wird der Lock selbst gelöscht und der LVB verworfen.

  Zugriffe auf den Lock-Value-Block erfolgen durch die Makros **LKENQ**, **LKCVT** und **LKDEQ**.

  Zugriffe auf den Lock-Value-Block

vom zugeteilten Lock-Modus	zum angeforderten Lock-Modus					
	NU	CR	CW	PR	PW	EX
NU	r	r	r	r	r	r
CR	-	r	r	r	r	r
CW	-	-	r	r	r	r
PR	-	-	-	r	r	r
PW	w	w	w	w	w	r
EX	w	w	w	w	w	w

  - Der Lock-Wert wird weder gelesen noch geschrieben.

  r (read) Der Lock-Wert wird gelesen.
  Der Lock-Wert kann vom Benutzer gelesen werden, wenn die Lock-Anforderung zugeteilt wurde. Wird der Lock-Wert eines Locks zum ersten Mal gelesen, enthält er den Initialisierungs-Wert (Binäre Nullen).

  w (write) Der Lock-Wert wird geschrieben.
  Der Lock-Wert kann vom Benutzer geschrieben werden, wenn der Lock von einem stärkeren Lock-Modus (z.B. PW- oder EX-Modus) in einen schwächeren oder gleichen Lock-Modus konvertiert wird. Oder während der Freigabe eines Locks, der sich im PW- oder EX-Modus befindet.

Wurde beim Makro **LKENQ** angegeben, dass der LVB gelesen werden soll, wird die Lock-Anforderung behandelt, als wenn sie vom NU-Modus in den angegebenen Lock-Modus konvertiert wird.

Den schematischen Aufbau eines DLM-Locks, der sich daraus ergibt, verdeutlicht das folgende Bild.

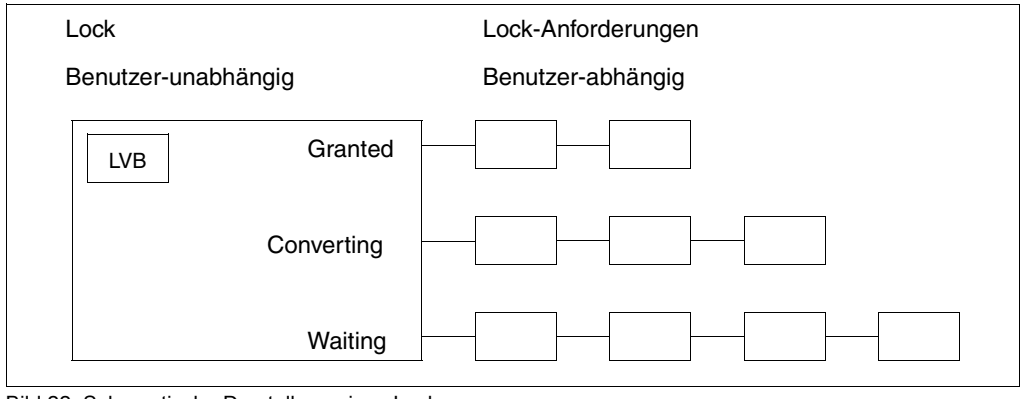


Bild 22: Schematische Darstellung eines Locks

4.3.8.2 Funktionen des DLM

- **Lock erstellen**

  Eine Task, die einen Lock verwenden will, muss ihn zuerst erstellen. Dies geschieht implizit beim Aufruf des Makros **LKENQ**. Der Lock wird durch den DLM erstellt.
  Bei der Erstellung eines noch nicht existierenden Locks werden beide Teile des Locks, der Benutzer-abhängige und der Benutzer-unabhängige Teil, erstellt.
  Existiert der zu erstellende Lock bereits, wird nur ein Benutzer-abhängiger Teil neu erstellt und mit dem Benutzer-unabhängigen Teil verkettet.

  Lock-Kurzkennung (LOCKID)

  Nach der Erstellung eines Locks erhält die Task eine (task-spezifische) Lock-Kurzkennung (Makro **LKENQ**, Operand LOCKID). Die Lock-Kurzkennung muss von der aufrufenden Task angegeben werden, wenn andere Aufrufe an den DLM erfolgen, die sich auf den gleichen Lock beziehen. Andere Tasks, die den gleichen Lock und damit den gleichen Lock-Namen verwenden, haben andere Lock-Kurzkennungen für diesen Lock.
  Die Lock-Kurzkennung wird ungültig, wenn das aufrufende Programm sie freigibt (Makro **LKDEQ**) oder das Programm beendet wird.
  Vom DLM wird niemals der Wert 0 für eine Lock-Kurzkennung geliefert.

  Um inkompatible Lock-Anforderungen zu vermeiden, sollte eine neue Lock-Anforderung immer mit LCKMODE=*NU erstellt werden (Makro **LKENQ**), da sie sofort zugeteilt wird (Zustand GRANTED). Sonst erhält die Lock-Anforderung den Zustand WAITING. Dort muss sie warten, bis alle Lock-Anforderungen im Zustand GRANTED und CONVERTING freigegeben wurden, bevor sie zugeteilt wird. Ist die Lock-Anforderung zugeteilt, kann der Lock-Modus mit dem Makro **LKCVT** konvertiert werden. Es erfolgen dann nur noch Wechsel zwischen den Zuständen GRANTED und CONVERTING (siehe Seite 144).

- **Lock-Anforderung konvertieren**

  Ein existierender und bereits zugeteilter Lock eines Benutzers kann von einem Lock-Modus in einen anderen Lock-Modus konvertiert werden (Makro **LKCVT**). Die **LKCVT**-Anforderungen werden ausgeführt, bevor eine neue Lock-Anforderung (Makro **LKENQ**) ausgeführt wird, außer Lock-Anforderungen im NU-Modus.

  Während der Konvertierung des Lock-Modus bleibt der Lock im ursprünglichen Lock-Modus zugeteilt. Die Lock-Anforderung erhält den Zustand CONVERTING. Kann die Lock-Anforderung zugeteilt werden, erhält sie den Zustand GRANTED. Der Lock-Modus wurde konvertiert.

Relation der Lock-Modi

Lock-Modus 1	Lock-Modus 2					
	NU	CR	CW	PR	PW	EX
NU	e	h	h	h	h	h
CR	l	e	h	h	h	h
CW	l	l	e	h	h	h
PR	l	l	h	e	h	h
PW	l	l	l	l	e	h
EX	l	l	l	l	l	e

e Lock-Modus 2 ist gleich Lock-Modus 1
l Lock-Modus 2 ist schwächer als Lock-Modus 1
h Lock-Modus 2 ist stärker als Lock-Modus 1

- **Lock freigeben**

  Die Freigabe eines Locks kann auf zwei Arten erfolgen:

  1. Der Lock kann von der aufrufenden Task freigegeben werden (Makro **LKDEQ**). Das bedeutet, dass die zugeteilte Lock-Anforderung des Aufrufers gelöscht wird, und die Lock-Kurzkennung (LOCKID) ungültig wird. Weiterer Gebrauch der ungültigen Lock-Kurzkennung führt zu Fehler-Codes, insbesondere wenn der Lock noch existiert. Der Benutzer-abhängige Teil des Locks und alle seine Benutzer-spezifischen Informationen, die zusammen mit dem Lock gespeichert wurden, werden aus der DLM-Datenstruktur gelöscht. Ist dieser Benutzer der letzte (einzige) Benutzer dieses Locks, wird der Benutzer-unabhängige Teil des Locks ebenfalls gelöscht. Dadurch wird der Lock selbst aus der DLM-Datenstruktur gelöscht.
  2. Die existierende Lock-Anforderung einer Task kann in einen schwächeren Lock-Modus konvertiert werden (siehe Tabelle auf Seite 144). Die Lock-Kurzkennung (LOCKID) bleibt gültig und kann weiter benutzt werden (Makro **LKCVT**).

- **Lock-Anforderung löschen**

  Lock-Anforderungen, die noch nicht vom DLM zugeteilt wurden, können mit dem Makro **LKCAN** gelöscht werden.

  Die Lock-Anforderungen, die vom Makro **LKENQ** erstellt wurden und sich im Zustand WAITING befinden, werden vollständig gelöscht. Befindet sich die Lock-Anforderung im Zustand GRANTED, d.h der Lock ist bereits zugeteilt, wird mit einem Fehler-Code abgebrochen. Diese Lock-Anforderung muss mit dem Makro **LKDEQ** freigegeben werden.

Die Lock-Anforderungen, die durch den Makro **LKCVT** konvertiert werden sollen und sich im Zustand CONVERTING befinden, werden nicht gelöscht. Es wird nur der Konvertierungs-Auftrag gelöscht.

Der Aufruf kann synchron oder asynchron erfolgen.

- **Informationen über Locks ausgeben**

  Der Makro **LKINF** informiert darüber, welche Locks bereits benutzt werden. Zur besseren Auswahl können einige Such-Filter aktiviert werden. Die Locks werden über ihre Lock-Namen identifiziert. Die Lock-Namen können voll- oder teilqualifizierte Namen sein. Bei Angabe von teilqualifizierten Lock-Namen kann der Zugriff ggf. sehr lange dauern, da die gesamte DLM-Datenbasis nach Treffern durchsucht werden muss. Dies hängt von der Größe der DLM-Datenbasis ab.

- **Erkennung der Zeitlimitüberschreitung**

  Der Benutzer kann eine Wartezeit und eine Haltezeit für die Lock-Anforderung festlegen, die vom DLM überprüft wird. Kann die Lock-Anforderung während der Wartezeit nicht zugeteilt werden, wird sie mit einem Fehler-Code abgebrochen oder es wird ein Wartezeit-Überschreitungs-Ereignis generiert.

  Überschreitet die Haltezeit das Zeitlimit, wird der Lock-Halter durch ein Freigabe-Ereignis darüber informiert.
  Wenn eine Haltezeit spezifiziert wird, muss auch ein Freigabe-Ereignis (RELEVTT) angegeben werden.

  Wurde der Wert 0 beim Zeitlimit für die Lock-Anforderung angegeben, wird diese Lock-Anforderung als Sofort-Anforderung bezeichnet. Sofort-Anforderungen, die nicht zugeteilt werden können, werden mit dem Returncode X'00828006' beendet.

- **Lock-Status-Block**

  Der Lock-Status-Block (LSB) ist Teil des Benutzer-Adressraums und hat zwei Hauptaufgaben.

  Erstens wird der LSB für alle asynchronen DLM-Aufrufe gebraucht. Der LSB ist der Kommunikations-Bereich zwischen dem DLM und dem aufrufenden Programm. Die asynchrone Zuteilung des Locks wird dem Benutzer durch das Setzen des entsprechenden Returncode im LSB gemeldet. Der Benutzer wird über die angegebene Ereignis-Methode informiert.

  Der LSB muss initialisiert werden, bevor asynchrone Funktionen vom DLM angefordert werden. Dies geschieht durch Aufruf des Makros **LKLSB** mit MF=L. Die Initialisierungs-Werte werden in den LSB-Bereich geschrieben. Der DLM kann nun entscheiden, ob die

übergebene Adresse auf einen gültigen LSB zeigt oder nicht. Wird der Lock zugeteilt, muss der LSB für den DLM verfügbar sein. Es werden sonst keine Daten übergeben. Der Fehler wird dem aufrufenden Programm über die Ereignis-Methode gemeldet.

Zweitens wird der LSB gebraucht, wenn der Lock-Value-Block gelesen oder geschrieben werden soll. Dies ist unabhängig von synchronen oder asynchronen Anforderungen.

Der Lock-Wert, den der Benutzer schreiben will oder den er bei einer Leseanforderung geliefert bekommt, befindet sich im Lock-Status-Block. Die Adresse des LSBs wird in der Operanden-Liste des aktuellen DLM-Aufrufs angegeben (Operand LSBADR in den Makros **LKENQ**, **LKCVT**, **LKDEQ** und **LKCAN**).

- **Status-Information**

  Der DLM behält Status-Informationen über die Freigabe des letzten Locks im PW- oder EX-Modus. Wurde die Freigabe normal ausgeführt, hat der Lock-Halter selbst den Lock freigegeben (Makro **LKDEQ**). Die Status-Information wird auf VALID gesetzt.

  Wurde die Freigabe abnormal beendet, wird die Status-Information auf INVALID gesetzt. Abnormal beendet wird die Freigabe, wenn das Programm, die Task oder der Knoten, auf dem sich der Lock-Halter befindet, beendet wurde, bevor der Lock freigegeben wurde.

  Die Status-Information liefert Informationen über die Gültigkeit des Lock-Value-Blocks, der auf dieselbe Weise behandelt wird. Die Behandlung der Status-Information ist nicht notwendigerweise an eine Behandlung des Lock-Value-Blocks geknüpft.

  Die Status-Information wird an jeden folgenden Lock-Halter übermittelt, bis die Status-Information wieder auf VALID gesetzt wurde. War der angeforderte Lock-Modus NU oder CR, ist die Information möglicherweise nicht mehr aktuell.

  Die Status-Information INVALID kann auf VALID zurückgesetzt werden, wenn ein weiterer Lock-Halter den Lock im PW- oder EX-Modus anfordert, zugeteilt bekommt und bei der Freigabe angibt, dass der Status zurückgesetzt werden soll. Dies kann unabhängig vom Lock-Value-Block geschehen. Es ist aber mit einem DLM-Aufruf möglich, den Lock-Value-Block zu verändern und den Status auf VALID zurückzusetzen.

- **Beendigungs-Sequenz der Lock-Anforderungen während der Beendigung des Lock-Halter-Prozesses**

  Während der abnormalen Beendigung eines Lock-Halter-Prozesses (Task oder Programm) werden die Lock-Anforderungen in einer definierten Sequenz freigegeben.

  Während der Erstellung einer Lock-Anforderung kann der Aufrufer die Freigabe-Zeit seines Locks in einer der drei Klassen angeben (Operand TERMNTE im Makro **LKENQ**):

  FIRST Diese Klasse enthält Locks, die bevor oder zur gleichen Zeit freigegeben werden, wie die Locks der folgenden Klasse SECOND freigegeben werden.

  SECOND Diese Klasse enthält Locks, die nachdem oder zur gleichen Zeit freigegeben werden, wie die Locks der vorherigen Klasse FIRST freigegeben werden. Und bevor oder zur gleichen Zeit, wie die Locks der folgenden Klasse THIRD freigegeben werden.

  THIRD Diese Klasse enthält Locks, die nachdem oder zur gleichen Zeit freigegeben werden, wie die Locks der vorherigen Klasse SECOND freigegeben werden.

  Um sicherzustellen, dass alle Locks von verschiedenen Prozessen, die zu derselben Anwendung gehören, auf dieselbe Weise behandelt werden, muss der Aufrufer sicherstellen, dass alle **LKENQ**-Anforderungen mit denselben Operanden aufgerufen werden.
  Während des **LKENQ**-Aufrufs wird dies vom DLM nicht überprüft.

4.3.8.3 Synchrone und asynchrone Lock-Anforderungen

Die DLM-Funktionen können auf zwei Arten ausgeführt werden:

Erstens kann der Aufruf synchron erfolgen. Das bedeutet, dass der Aufrufer erst wieder die Kontrolle erhält, wenn die Lock-Anforderung zugeteilt wurde oder eine Fehlerbedingung erkannt wurde. Die Information wird im Returncode zurückgeliefert.

Zweitens kann der Aufruf asynchron erfolgen. Das bedeutet, dass der Aufrufer wieder die Kontrolle erhält, wenn der Auftrag vom DLM akzeptiert wurde. Die Lock-Anforderung wird später zugeteilt. Der Aufrufer muss dazu eine Ereignis-Methode spezifizieren.

Es gibt zwei mögliche Methoden für die asynchrone Ereignis-Methode: der Contingency-Prozess oder die Ereignissteuerung.
Es ist möglich, dass für den gleichen Lock und zur gleichen Zeit, ein Benutzer die synchrone Methode und ein anderer Benutzer eine asynchrone Methode benutzt.

Die folgende Tabelle zeigt, welche Operanden beim **LKENQ**-Aufruf und beim **LKCVT**-Aufruf angegeben werden können, um die gewünschte Ereignis-Methode zu benutzen.

Operanden beim **LKENQ**-Aufruf und **LKCVT**-Aufruf

Ereignis-Methode		GRTEVTT	RELEVTT
synchrone Zuteilung		*SYNCH	*NO
		*SYNCH	*TUCONTI
		*SYNCH	*TUEVENT
asynchrone Zuteilung	Contingency	*TUCONTI	*TUCONTI
		*TUCONTI	*TUEVENT
	Eventing	*TUEVENT	*TUCONTI
		*TUEVENT	*TUEVENT

Unterstützungsübersicht:

Makro	**synchron**	**asynchron**
LKENQ	x	x
LKCVT	x	x
LKDEQ	x	x
LKCAN	x	x
LKINF	x	-

- **Ereignisse und Ereignis-Spezifikation**

  Der DLM bietet:

  – das Zuteilungs-Ereignis (GRANTID)
  – das Freigabe-Ereignis (RELID)

  Das Zuteilungs-Ereignis beendet eine asynchrone Lock-Anforderung entweder nachdem der Lock zugeteilt wurde oder wenn eine Fehlersituation erkannt wurde.
  Das Freigabe-Ereignis informiert den Lock-Halter, dass sein eigener zugeteilter Lock inkompatible Lock-Anforderungen von anderen Benutzern blockiert. Der Lock-Modus der blockierten Anforderung wird zusammen mit dem Ereignis übergeben. Der Lock-Halter kann seinen Lock-Schutz herabsetzen (Makro **LKCVT**) oder seinen Lock freigeben (Makro **LKDEQ**).
  Gibt ein Lock-Halter den Lock frei und der nächste, neue Lock-Halter blockiert andere Lock-Anforderungen, wird das Freigabe-Ereignis (das an eine blockierende Lock-Anforderung geschickt wird) auch für die neue blockierende Lock-Anforderung generiert. Deshalb muss der Benutzer Freigabe-Ereignisse berücksichtigen, die sofort nach der Zuteilung des Locks geschickt werden können. Es ist auch möglich, dass ein Lock-Halter, der andere Lock-Anforderungen blockiert, mehr als ein Freigabe-Ereignis erhält.

  Überschreitet ein asynchroner DLM-Aufruf das Zeitlimit, führt dies zu einem Wartezeit-Überschreitungs-Ereignis.
  Überschreitet ein Lock-Halter die angegebene Haltezeit des Locks, führt dies zu einem Haltezeit-Überschreitungs-Ereignis.

  – *asynchrone Ereignisse und Operand USERPAR*
    Die asynchronen Funktionen bieten die Angabe des Operanden USERPAR an. Der Wert, den der Benutzer beim Aufruf des Makros angegeben hat, wird beim abschließenden Ereignis dieser Lock-Anforderung wieder zurückgeliefert.
    Der USERPAR-Wert, der beim Makro **LKENQ** angegeben wurde, wird zusammen mit dem Ereignis bei einem Zuteilungs- oder Freigabe-Ereignis für diesen Lock (und diese Task) übergeben.
    Der bei einem nachfolgenden **LKCVT**-Aufruf übergebene USERPAR-Wert wird ab diesem Zeitpunkt vom DLM für die Ereignissteuerung benutzt. Jedes folgende Zuteilungs- oder Freigabe-Ereignis wird dann mit diesem USERPAR-Wert zusammen übergeben.
    Der USERPAR-Wert, der beim Makro **LKCAN** oder **LKDEQ** angegeben wurde, wird nur für das Bestätigen des Lösch-Ereignisses oder des Freigabe-Ereignisses benutzt.
    Zur einfacheren Benutzung sollte der Benutzer nur einen einzigen USERPAR-Wert für alle Aufrufe verwenden, die die gleiche Lock-Anforderung betreffen.

  – *synchron*
    Der Benutzer erhält erst wieder die Kontrolle, wenn die Lock-Anforderung zugeteilt wurde, die Wartezeit abgelaufen ist oder eine Fehlerbedingung erkannt wurde. Die Information wird im Returncode zurückgeliefert.

– *Contingency-Prozess*
  Eine Contingency-Kurzkennung kann für jedes der Ereignisse spezifiziert werden. Das Zuteilungs-Ereignis und das Freigabe-Ereignis können verschiedene Contingency-Prozesse spezifizieren. Tritt eins der beiden Ereignisse ein, wird der entsprechende Contingency-Prozess bereitgestellt. Bei dem Freigabe-Ereignis wird der blockierende Lock-Modus an den Contingency-Prozess übergeben.
  Der Contingency-Prozess muss mit dem Makro **ENACO** erstellt worden sein. Die zurückgelieferte Contingency-Kurzkennung muss bei jedem asynchronen Aufruf angegeben werden. Der Contingency-Prozess wird unter der Kontrolle der Task des Lock-Anforderers bereitgestellt.
  Die Contingency-Kurzkennung muss für den DLM verfügbar sein, sonst kann kein Ereignis zugestellt werden.
  Zum Zeitpunkt, an dem der Contingency-Prozess startet, wird ihm die Information über den asynchronen Aufruf mitgeteilt. Die Information besteht aus dem Ereignis, das zur Bereitstellung des Contingency-Prozesses führte. Sie wird durch Register an den Contingency-Prozess übergeben. Das Register 3 enthält die Ereignis-Spezifikation, das Register 4 enthält die Benutzer-definierten USERPAR-Werte aus dem Makro **LKENQ**, **LKCVT** oder **LKDEQ**. Der Status des asynchronen Aufrufs steht im Lock-Status-Block, der beim DLM-Aufruf spezifiziert wurde.
  Die Benutzung des Contingency-Prozesses benötigt die Spezifikation von zwei Contingency-Kurzkennungen. Eine für das Zuteilungs-Ereignis (GRANTID) und eins für das Freigabe-Ereignis (RELID). Es kann in beiden Fällen die gleiche Kurzkennung sein. Die Entscheidung, welches Ereignis gemeldet wird, kann mit der an den Contingency-Prozess übergebenen Information getroffen werden.

– *Ereignissteuerung (Eventing)*
  Es wird empfohlen, für beide Ereignisse die selbe Ereigniskennung zu spezifizieren. Die Ereigniskennung muss mit dem Makro **ENAEI** erstellt worden sein. Die zurückgelieferte EIID muss bei jedem asynchronen Aufruf angegeben werden. Die Ereigniskennung muss in dem Moment verfügbar sein, in dem die asynchrone Anforderung zugeteilt wird. Anderenfalls erfolgt keine Ereigniszustellung.

  Der Postcode, der zusammen mit dem Ereignis übergeben wird, enthält die Ereignis-Spezifikation (DLM-Ereignis). Im zweiten Wort des Postcodes wird der Benutzer-definierte Operand USERPAR mitübergeben. Es werden auch Informationen über die asynchrone Anforderung in den Lock-Status-Block übergeben, wenn verfügbar.

– *Ereignissteuerung (Eventing) und Contingency-Prozess*
  Eine Mischung der beiden Methoden ist möglich. Siehe dazu die Tabelle „Operanden beim LKENQ-Aufruf und LKCVT-Aufruf“ auf Seite 152.

4.3.8.4 Lock-Name

Der Lock-Name ist ein eindeutiger Bezeichner für den Lock. Vom DLM aus gesehen, besteht ein Lock-Name intern aus drei Teilen (Kennzeichen für lokalen und globalen Geltungsbereich und Zeichenkette des Lock-Namens). Vom Benutzer aus gesehen, beeinflussen die folgenden Operanden die Bildung des internen Lock-Namens (Makros **LKENQ** und **LKINF**).

1. Über den Operanden NAMRNGE wird der globale Geltungsbereich des Locks angegeben.
   - NAMRNGE=*OWNSYSTEM
     Der angegebene Lock-Name ist nur auf dem lokalen System gültig.
   - NAMRNGE=*CLUSTER
     Der angegebene Lock-Name ist clusterweit gültig.
2. Über den Operanden SCOPE wird der lokale Geltungsbereich des Lock angegeben.
   - SCOPE=*NAMESPACEID
     Der angegebene Lock-Name wird als interner Lock-Name verwendet. Der erste Teil (8 Bytes) des angegebenen Lock-Namens bildet dabei implizit den lokalen Geltungsbereich. Der lokale Geltungsbereich muss eine Zeichenfolge sein. Die gültigen Zeichen sind die Buchstaben „A..Z", „a..z"; die Ziffern „0..9" und die Sonderzeichen „@" und „#". Die maximale Länge des angegebenen Lock-Namens ist 48 Zeichen.
   - SCOPE=*USERID
     Die Benutzerkennung, zu der die Aufrufer-Task gehört, wird für die Bildung des internen Lock-Namens verwendet. Der DLM bestimmt die Benutzerkennung und setzt sie an den Anfang des Lock-Namens. Der erste Teil des angegebenen Lock-Namens wird nicht als lokaler Geltungsbereich betrachtet. Die maximale Länge des angegebenen Lock-Namens verringert sich auf 40 Zeichen.
     Durch die Angabe des Operanden SCOPE=*USERID können die Locks einer Anwendung auf einfache Weise gegen den Zugriff einer anderen Anwendung geschützt werden. Die Anwendungen müssen nur unter verschiedenen Benutzerkennungen gestartet werden.
   - SCOPE=*GROUPID
     Die Benutzergruppe, zu der die Aufrufer-Task gehört, wird für die Bildung des internen Lock-Namens verwendet. Der DLM bestimmt die Benutzergruppe und setzt sie an den Anfang des Lock-Namens. Der erste Teil des angegebenen Lock-Namens wird nicht als lokaler Geltungsbereich betrachtet. Die maximale Länge des angegebenen Lock-Namens verringert sich auf 40 Zeichen.
     Der Operand SCOPE=*GROUPID darf nur angegeben werden, wenn das kostenpflichtige Software-Produkt SECOS im Einsatz ist, sonst führt der **LKENQ**-Aufruf zu einem Fehler.

Durch die Angabe dieses Operanden können die Locks einer Anwendung auf einfache Weise gegen den Zugriff einer anderen Anwendung geschützt werden. Die Anwendungen müssen nur unter verschiedenen Benutzergruppen gestartet werden.

3. Der verbleibende Teil des Lock-Namens (bis zu 40 Zeichen) kann beliebige Zeichen enthalten.

Der lokale Geltungsbereich muss unter Beachtung der folgenden Regeln gebildet werden:

1. Benutzern ist es nicht erlaubt, einen lokalen Geltungsbereich zu bilden, der mit dem Sonderzeichen „$" anfängt.
2. Benutzer müssen die Präfixe „SYS" und „@" für den lokalen Geltungsbereich vermeiden. Diese Präfixe sind für interne TU-Applikationen reserviert.

4.3.8.5 Cluster-Systeme und Single-Systeme

Jeder Lock-Name gehört zu einem globalen Geltungsbereich. Ist der Bereich knotenlokal, dann ist der Name nur auf dem eigenen Knoten gültig. Er ist immer getrennt vom gleichen Namen mit dem Cluster als Geltungsbereich.

Zwei Anwendungen (die auf verschiedenen Knoten in einem Cluster ablaufen), die den gleichen Lock-Namen mit lokalem Geltungsbereich angegeben haben, werden nicht serialisiert, weil jeder der Lock-Namen ein lokaler / knoten-spezifischer Name ist.

Geben zwei Anwendungen den Geltungsbereich Cluster für identische Lock-Namen an, meinen sie den gleichen Lock und werden vom DLM serialisiert.

Geben zwei Anwendungen (die irgendwo auf dem Cluster ablaufen) den gleichen Lock-Namen, aber verschiedene Geltungsbereiche an, dann meinen sie verschiedene Locks.

Es besteht keine Möglichkeit einen Lock mit NAMRNGE=*OWNSYSTEM in einen Lock mit NAMRNGE=*CLUSTER zu überführen, weil diese beiden Locks komplett verschieden sind.

Um Benutzerprogrammen (die für Cluster-Systeme entwickelt wurden) zu ermöglichen auf Single-Systemen ohne Änderung abzulaufen, bietet der DLM folgendes Merkmal:

Ist ein System nicht Teil eines Clusters und wird in der aktuellen Session auch nicht Teil eines Clusters, akzeptiert der DLM **LKENQ**-Aufrufe, als wenn das System Teil eines Clusters wäre. Die Locks, die mit NAMRNGE=*CLUSTER in die Warteschlange eingereiht werden, sind nicht dieselben, wie die mit NAMRNGE=*OWNSYSTEM.

Kurzübersicht

beim Makroaufruf anzugeben	Makro	wird vom Makro zurückgeliefert	
MF=D	LKEQU		(1)
MF=L	LKLSB		(2)
	LKINF		(3)
	LKENQ	LOCKID	(4)
LOCKID	LKCVT		(5)
LOCKID	LKCAN		(6)
LOCKID	LKDEQ		(7)

(1) **LKEQU** generiert die DLM-eigenen Layouts

(2) **LKLSB** generiert das Layout des Lock-Status-Blocks

(3) **LKINF** gibt Informationen über Locks aus

(4) **LKENQ** erstellt einen Lock und liefert Lock-Kurzkennung

(5) **LKCVT** konvertiert Lock-Anforderungen

(6) **LKCAN** löscht Lock-Anforderungen

(7) **LKDEQ** gibt Lock-Anforderungen frei

4.4 Abfragen und Zugriff zu Listen und Tabellen

Makro	Kurzbeschreibung
AINF	ermittelt den Betriebsmittelverbrauch des Auftrags
CHKPRV	überprüft den laufenden Auftrag auf Privilegien
CTIME	rechnet mit Zeitstempeln (Darstellung von verschiedenen Formaten und arbeiten mit Zeitspannen)
CUPAB	generiert Adressierungshilfen (DSECTs) für die Operandentabellen der Makros RDATA, WROUT und WRTRD
DCSTA	generiert einen Bereich (CSECT) oder Adressierungshilfen (DSECTs) für die Informationen, die der Makro TSTAT übergibt
DJINF	generiert eine DSECT/Datenliste für den Ausgabebereich des Makros JINF
DTMODE	generiert eine DSECT/Datenliste für die 31-Bit-Schnittstelle des Makros TMODE
GCCSN	zeigt die aktuelle Codiertabelle für die Ein- und Ausgabe von Kommandos und Daten (SYSDTA/SYSCMD/SYSOUT/SYSLST) an
GEPRT	übergibt die verbrauchte und die noch verfügbare Laufzeit des Programmes bzw. des Auftrages
GTIME	übergibt das aktuelle Datum und die Uhrzeit sowie Informationen über die aktuelle Zeitzone
HSITYPE	informiert über das aktuelle HSI (Hardware-Software-Interface)
IOSID	informiert über Kennung und Version des Betriebssystems
JINF	übergibt eine Liste mit Jobdaten (Kommando SET-LOGON-PARAMETERS/ ENTER-JOB) des Jobs mit dem aufrufenden Programm
JMGDJP	generiert eine DSECT/Datenliste zum Datenbereich des Makros JMGJPAR
JMGJPAR	übergibt die im Kommando SET-LOGON-PARAMETERS/ENTER-JOB angegebenen Jobparameter
JOBINFO	übergibt eine Liste mit den Jobdaten (Kommando SET-LOGON-PARAMETERS oder ENTER-JOB) für einen ausgewählten Job
MINF	informiert über Speicherbelegung und Größe des Klasse-6-Speichers oder eines Memory Pools
NKDINF	übergibt Informationen über den Belegungs- und Verfügbarkeitszustand der (peripheren) Konfiguration
NKGTYPE	informiert über Namen, Gerätetypcode, Geräteeigenschaften, usw. eines Geräte- oder Volumetyps oder über Namen und Gerätetypcodes der Gerätetypen, die zu einer Gerätefamilie oder Geräteklasse gehören
NSIINF	übergibt Informationen über Zentraleinheit, Betriebssystem, HSI oder Speicher
NSIOPT	übergibt Informationen über Systemparameter
RDUID	übergibt dem Programm die Benutzerkennung der Task, unter der es abläuft

Makro	Kurzbeschreibung
SINF	übergibt Informationen über Zentraleinheit, Betriebssystem oder Systemparameter in einen Bereich des Benutzerprogramms
SRMUINF	überträgt Daten aus dem Benutzerkatalog in einen Bereich
STAMCE	liest Einträge aus dem MRSCAT
TMODE	übergibt Informationen über den laufenden Auftrag, z.B. TSN, Kennung, Abrechnungsnummer usw
TSPRIO	generiert symbolische Namen für Ober- und Untergrenzen von variablen und konstanten Prioritäten
TSTAT	übergibt Informationen über die Eigenschaften der Dialogstation (siehe auch DCSTA)
VMGINF	gibt Informationen im Zusammenhang mit dem VM2000-Betrieb
VTCSET	generiert symbolische Namen, mit denen logische Steuerzeichen in Line-Mode-Ausgabenachrichten eingefügt bzw. Line-Mode-Eingaben aufgefunden werden können

4.5 Ein-/Ausgabe

4.5.1 Systemdateien

Makro	Kurzbeschreibung
SYSFL	ändert die Zuordnung der Systemdateien SYSDTA, SYSLST, SYSLST01, SYSLST02, ..., SYSLST99 und SYSOUT sowie der TASKLIB
SYSTA	gibt die Zuweisung der Systemdateien und der TASKLIB aus

Die (Standard-)Dateinamen SYSDTA, SYSCMD, SYSLST, SYSLST01, SYSLST02, ..., SYSLST99 und SYSOUT bezeichnen vom Betriebssystem benutzte (System-) Dateien zur Daten- bzw. Kommandoeingabe an das Betriebssystem oder zur Datenausgabe durch das Betriebssystem. Diese Dateien werden jeweils durch die Anwendertask erzeugt und bezeichnen anfänglich (primär) vorgegebene Ein- bzw. Ausgabebereiche.
Der Anwender kann die primäre Zuordnung aufheben und den (Standard-) Dateinamen eigene (katalogisierte) Dateien zuweisen. Einige der Standardnamen können auch gleichgesetzt werden (siehe Makro **SYSFL**).

Die Systemdateien SYSIPT und SYSOPT werden nur aus Kompatibilitätsgründen bedient. Sie werden in diesem Handbuch nicht mehr beschrieben.

Systemdateien, die einer Task für die Eingabe zur Verfügung stehen

SYSCMD Von SYSCMD werden die Kommandos zur Steuerung des Jobs eingelesen.

SYSDTA dient zur Eingabe von Daten und Anweisungen für ein Programm. Sobald ein Programm abläuft, ist SYSDTA aktiv. Auf SYSDTA kann mit dem Makro **RDATA** zugegriffen werden.

Systemdateien, die einer Task für die Ausgabe zur Verfügung stehen

SYSOUT dient zur Aufnahme der protokollierenden Meldungen und Fehlermeldungen, die während des laufenden Auftrags anfallen. Auch Dienstprogramme und Übersetzer verwenden SYSOUT in dieser Weise. Auf SYSOUT kann mit dem Makro **WROUT** zugegriffen werden.

SYSLST dient zur Aufnahme von meist größeren Datenmengen, wie z.B. Speicherauszüge oder erzeugte Listen. Auf SYSLST kann mit dem Makro **WRLST** zugegriffen werden. Zusätzlich werden alle Datensätze nach SYSOUT auch in die Systemdatei SYSLST geschrieben, wenn bei SET-LOGON-PARAMETERS oder MODIFY-JOB-OPTIONS die entsprechenden Operanden angegeben wurden.

SYSLST01, SYSLST02,...,SYSLST99
haben, im Gegensatz zur Systemdatei SYSLST, keinen eigenen EAM-Bereich zur Speicherung von Ausgabedaten. Sie dienen der Zwischenspeicherung und sind nur wirksam, wenn ihnen katalogisierte Dateien zugewiesen werden. Auf die SYSLSTn-Dateien kann mit dem Makro **WRLST** zugegriffen werden.

Die Systemdateien für die Ausgabe werden vom Betriebssystem bei Bedarf unter der Kennung des Anwenders angelegt. Es sind SAM-Dateien mit den Dateinamen

S.OUT.tsn.yyyy-mm-dd.hhmmss (für SYSOUT)
S.LST.tsn.yyyy-mm-dd.hhmmss (für SYSLST)

Es bedeuten:

– tsn=TSN des Auftrags
– yyyy-mm-dd=Datumsangabe (yyyy=Jahr, mm=Monat, dd=Tag des Monats)
– hhmmss=Zeitangabe (hh=Stunde, mm=Minute, ss=Sekunde)

Der benutzte Speicherplatz zählt nicht zum zugeteilten Pubspace-Kontingent.

Die Dateien werden bei Auftragsende automatisch ausgedruckt und danach gelöscht. Der Anwender kann auf diese Dateien nicht zugreifen. Mit dem Kommando DELETE-FILE *SYSxyz (xyz=LST/OUT/OPT) wird der Inhalt der angegebenen Systemdatei (logisch) gelöscht, der Katalogeintrag bleibt erhalten. Eine leere Systemdatei wird nicht ausgedruckt.

Die Systemdateien für die Ausgabe können auch vorzeitig ausgegeben werden (Operand START-PROCESSING im Kommando PRINT-DOCUMENT).

In den Kommandos PRINT-DOCUMENT und DELETE-SYSTEM-FILE bzw. den Makros **ERASE** (siehe Handbuch „DVS Makros“ [7]) und **PRNT...** (siehe Handbuch „SPOOL & Print - Makros und Exits“ [23]) können die (Standard-) Dateinamen SYSOUT und SYSLST auch dann angegeben werden, wenn ihnen katalogisierte Dateien zugeordnet sind.

Primärzuweisung und Umadressierung von Systemdateien

Für die Systemdateien ist meist eine bestimmte Zuweisung vorgegeben. Diese Primärzuweisung kann mit Kommandos verändert werden. Die folgende Tabelle gibt darüber einen Überblick. Beispiele sind den Beschreibungen des jeweiligen Kommandos im Handbuch „Kommandos“ [19] zu entnehmen.

Systemdatei	**Zuweisungen für die Systemdateien Primärzuweisung**	**weitere Zuweisungen**	**Kommandos zur Veränderung der Zuweisung**
SYSCMD	im Dialog: Datenstation im Batch: Einspuldatei „S.INtsn“ (eingespult über Magnetbandgerät, oder ENTER-Datei)	katalogisierte Plattendatei (SAM/ISAM)	Kommando CALL-PROCEDURE: Zuweisung zu einer katalogisierten Datei Kommandos END-PROCEDURE: (nur in Prozedurdateien) und EXIT-PROCEDURE: Zurück zur letzten (mit (CALL-PROCEDURE verlassenen) Prozedurstufe
SYSDTA	wie SYSCMD-Primärzuweisung	katalogisierte Plattendatei (SAM/ISAM), S-Variable oder Element einer PLAM-Bibliothek	Kommando ASSIGN-SYSDTA: Zuweisung zur katalogisierten Datei, zu einer S-Variablen, zu einem Element einer PLAM-Bibliothek, zu SYSCMD oder zur Primärzuweisung zurück. Kommandos END-PROCEDURE (nur in Prozedurdateien) und EXIT-PROCEDURE: Zurück zur Zuordnung, die vor Aufruf der Prozedurebene gültig war

Tabelle 10: Zuweisung zu Systemdateien (Teil 1 von 2)

System-datei	**Zuweisungen für die Systemdateien Primärzuweisung**	**weitere Zuweisungen**	**Kommandos zur Veränderung der Zuweisung**
SYSOUT	im Dialog: Datenstation im Batch: temporäre (System-)Datei S.OUT.., die bei Auftragsende auf Drucker ausgegeben und anschließend gelöscht wird	im Dialog: katalogisierte Datei, S-Variable oder Element einer PLAM-Bibliothek im Batch: katalogisierte Datei, S-Variable oder Element einer PLAM-Bibliothek, die aber nicht automatisch auf Drucker ausgegeben werden; PRINT-FILE-Kommando erforderlich	wie bei SYSLST
SYSLST	temporäre (System-)Datei S.LST...., die bei Auftragsende auf Drucker ausgegeben und anschließend gelöscht wird; (wird erst bei Bedarf eingerichtet)	katalogisierte Datei, S-Variable, Element einer PLAM-Bibliothek, die aber nicht automatisch auf Drucker ausgegeben werden; PRINT-FILE-Kommando erforderlich	Kommando ASSIGN-SYSLST: Zuweisung zu einer katalogisierten Datei, zu einer S-Variablen, einem Element einer PLAM-Bibliotehk oder zur Primäranweisung zurück. Kommandos END-PROCEDURE (nur in Prozedurdateien) und EXIT-PROCEDURE: Zurück zur Zuordnung, die vor Aufruf der Prozedurebene gültig war
SYSLST01 . . . SYSLST99	Systemdateien Primäre Zuordnung = Zuordnung von SYSLST.	wie SYSLST; auch untereinander möglich	wie bei SYSLST

Tabelle 10: Zuweisung zu Systemdateien (Teil 2 von 2)

4.5.2 Dateien und Sätze

Makro	Kurzbeschreibung
CUPAB	generiert Adressierungshilfen (DSECTs) für die Operandentabellen der Makros RDATA, WROUT und WRTRD
GCCSN	zeigt die aktuelle Codiertabelle für die Ein- und Ausgabe von Kommandos und Daten (SYSDTA/SYSCMD/SYSOUT/SYSLST) an
RDATA	liest einen Satz von der Systemdatei SYSDTA, also von einer katalogisierten Datei, einer S-Variablen, einem Element einer PLAM-Bibliothek oder von der dialogführenden Datenstation (siehe auch CUPAB)
VTSUCB	erstellt VTSU-Parameter für die Ein-/Ausgabe
WRLST	überträgt einen Satz in die Systemdatei SYSLST und/oder SYSLST01, SYSLST02, ..., SYSLST99. Die jeweilige Systemdatei wird nach Auftragsende ausgedruckt, wenn sie der Benutzer nicht einer katalogisierten Datei zugewiesen hat
WROUT	überträgt einen Satz in die Systemdatei SYSOUT, also im Dialogbetrieb auf die Datenstation, eine katalogisierte Datei, eine S-Variable oder ein Element einer PLAM-Bibliothek. Im Batch-Betrieb wird die Systemdatei SYSOUT nach Auftragsende ausgedruckt (siehe auch CUPAB), wenn sie der Benutzer nicht einer katalogisierten Datei, einer S-Variablen oder einem Element einer PLAM-Bibliothek zugewiesen hat
WRTRD	überträgt im Dialogbetrieb eine Nachricht zur Datenstation und übernimmt anschließend von dort eine Nachricht (siehe auch CUPAB)

4.5.3 Verkehr mit Datenstationen

Makro	Kurzbeschreibung
CUPAB	generiert Adressierungshilfen (DSECTs) für die Operandentabellen der Makros RDATA, WROUT und WRTRD
DCSTA	generiert einen Bereich (CSECT) oder Adressierungshilfen (DSECT) für die Informationen, die der Makro TSTAT übergibt
RDATA	liest einen Satz von der Systemdatei SYSDTA, also von einer katalogisierten Datei, einer S-Variablen, einem Element einer PLAM-Bibliothek oder von der dialogführenden Datenstation (siehe auch CUPAB)
SETBF	ändert die Größe des systeminternen Ein-/Ausgabepuffers der Datenstation
TCHNG	legt fest, ob der Bildschirmüberlauf vom System oder vom Benutzerprogramm behandelt wird
TMODE	übergibt Informationen über den aufrufenden Prozess, z.B. Puffergröße, Zeilenlänge und logischen Typ der Datenstation
TSTAT	übergibt Informationen über die Eigenschaften der Dialogstation (siehe auch DCSTA)
TYPIO	gibt eine Nachricht an der Konsole aus und übernimmt eine Antwort des Operators
VTCSET	definiert logische Steuerzeichen
VTSUCB	erstellt VTSU-Parameter für die Ein-/Ausgabe
WROUT	überträgt einen Satz in die Systemdatei SYSOUT, also im Dialogbetrieb auf die Datenstation, eine katalogisierte Datei, eine S-Variable oder ein Element einer PLAM-Bibliothek. Im Batch-Betrieb wird die Systemdatei SYSOUT nach Auftragsende ausgedruckt (siehe auch CUPAB), wenn sie der Benutzer nicht einer katalogisierten Datei, einer S-Variablen oder einem Element einer PLAM-Bibliothek
WRTRD	überträgt im Dialogbetrieb eine Nachricht zur Datenstation und übernimmt anschließend von dort eine Nachricht (siehe auch CUPAB)

4.5.4 Meldungswesen

Makro	Kurzbeschreibung
MSG7	gibt eine Meldung (mit 7-stelligem Meldungsschlüssel) in die Systemdatei SYSOUT oder auf der Konsole aus und übernimmt eine Antwort
MSG7X	gibt eine Meldung (mit 7-stelligem Meldungsschlüssel) in die Systemdatei SYSOUT oder auf der Konsole aus und übernimmt eine Antwort
MSGSHOW	informiert über Geltungsbereich, Anzahl, Sprache, Namen und Zugriffsmethode der Meldungsdateien
MSGSINIT	Systemverwaltermakro. Der Makro modifiziert die globale Bereichszuordnungsliste, indem er die Meldungsdatei neu einträgt oder den Zugriff auf die Meldungsdatei sperrt
MSGSMOD	Systemverwalter- und Benutzermakro. Der Makro modifiziert die globale Bereichszuordnungsliste: Meldungsdateien neu eintragen und/oder Zugriff auf Meldungsdateien sperren. Der nichtprivilegierte Anwender kann taskspezifische Meldungsdateien in das Meldungssystem einbringen
MSGRC	gibt den Returncode nebst Erläuterung für die Makros des Meldungswesens aus
OPSGEN	MIP über S-Variablen-Generierung informieren
TYPIO	gibt eine Nachricht an der Konsole aus und übernimmt eine Antwort des Operators

4.5.5 Verschlüsselung

Makro	Kurzbeschreibung
CRYPT	verschlüsselt Wörter im Einweg-Verschlüsselungsverfahren (eine Entschlüsselung ist nicht möglich)

4.6 Testhilfe

Makro	Kurzbeschreibung
AUDIT	überwacht die Funktionszustände TU und TPR eines Programms
BKPT	übergibt die Steuerung an das System. Der Benutzer kann dann an seiner Datensichtstation Kommandos eingeben
CDUMP2	veranlasst einen Speicherabzug, ohne dass das Programm beendet wird
TERM	beendet Programm und Auftragsabschnitt und veranlasst ggf. einen Speicherabzug

4.7 Fixpunktschreiben

Makro	Kurzbeschreibung
WRCPT	setzt einen Fixpunkt. Das heißt: ein definierter System- und Programmzustand an einem bestimmten Punkt des Programmlaufs wird zwischengespeichert und steht für einen Wiederanlauf (Kommando RESTART-PROGRAM) an diesem Punkt zur Verfügung

4.8 Accounting

Makro	Kurzbeschreibung
ARDS	beschreibt die Struktur der Abrechnungssätze (Definitionsmakro)
AREC	schreibt Abrechnungssatz in die Accounting-Datei
ASPC	erfasst die Speicherplatzbelegung auf den öffentlichen Datenträgern. Nur für Systemverwalter

4.9 Kommunikation (Programm, Anwender, System)

Makro	Kurzbeschreibung
BKPT	übergibt die Steuerung an das System. Der Benutzer kann dann an seiner Datensichtstation Kommandos eingeben
CLCOM	schließt die Intertaskkommunikation für das aufrufende Programm (siehe Seite 76)
CMD	führt Kommandos aus, ohne dass der Programm-Modus verlassen wird
JINF	überträgt eine Liste der Jobdaten
OPCOM	eröffnet die Intertaskkommunikation für das aufrufende Programm und legt einen ITC-Namen fest (siehe Seite 76)
RELBF	löscht die erste Nachricht in der ITC-Empfangswarteschlange des aufrufenden Programms (siehe Seite 76)
REVNT	empfängt eine Nachricht für den ITC-Namen des aufrufenden Programms (siehe Seite 76)
SEVNT	sendet eine Nachricht an einen ITC-Teilnehmer (siehe Seite 76)
STXIT	legt benutzereigene Routinen zur Unterbrechungsbehandlung fest, mit denen das System die Verarbeitung fortsetzt, wenn eine Programmunterbrechung auftritt. Mit dem Kommando INFORM-PROGRAM kann man einer Unterbrechungroutine Daten übergeben (siehe Handbuch „Kommandos“ [19])
SWITCH	schaltet und übergibt die 32 Auftragsschalter des Auftrags oder die 32 Benutzerschalter der eigenen oder einer fremden Benutzerkennung
TMODE	übergibt Auftragsattribute, wie Merkmale der Datensichtstation (Typ, Puffer, Modus,...) und Runpriorität, TSN, CPU-Zeit, Userid
TYPIO	gibt eine Nachricht an der Konsole aus und übernimmt eine Antwort
WROUT	überträgt einen Satz in die Systemdatei SYSOUT, also im Dialogbetrieb auf die Datensichtstation. Im Batch-Betrieb wird die Systemdatei SYSOUT nach Auftragsende ausgedruckt (siehe auch CUPAB), wenn sie der Benutzer nicht einer katalogisierten Datei zugewiesen hat
WRTRD	überträgt im Dialogbetrieb eine Nachricht zur Datensichtstation und übernimmt anschließend von dort eine Nachricht (siehe auch CUPAB)

Der Makro **SWITCH** ersetzt die Makros **GETSW**, **GETUS**, **SETSW** und **SETUS**. Diese Makros werden noch aus Kompatibilitätsgründen unterstützt und sind im Anhang ab Seite 1147 beschrieben.

4.10 Mehrrechnersysteme

Makro	Kurzbeschreibung
MCSINFO	zeigt die aktuelle HIPLEX-MSCF-Konfiguration, zu der der Rechner gehört
MRSINF	gibt Informationen über das MSCF-Kommunikationsnetz aus
MRSSTA	gibt Zustand im MRS aus
STAMCE	übergibt MRSCAT-Einträge in einen Bereich

Der Makro **MCSINFO** ersetzt die beiden Makros **MRSINF** und **MRSSTA**. Diese werden nur noch aus Kompatibilitätsgründen unterstützt und sind im Anhang ab Seite 1152 beschrieben. Der Makro MCSINFO ist im Handbuch „HIPLEX MSCF“ [26] beschrieben.

Die Makroaufrufe **MCSINFO** und **MRSSTA** stehen nur dem Anwender des Mehrrechnersystems zur Verfügung, das im Handbuch „HIPLEX MSCF“ [26] ausführlich beschrieben ist.

4.11 XS-Programmierung

Makro	Kurzbeschreibung
AMODE31	informiert, ob der 31-Bit-Adressierungsmodus eingeschaltet ist
GPARMOD	bestimmt, ob für die nachfolgenden Makros die 24-Bit- oder die 31-Bit-Schnittstelle generiert wird
HSITYPE	informiert über die Größe des adressierbaren Klasse-6-Speichers und ob 24-Bit- oder 31-Bit-Adressierung ausgeführt werden kann

4.12 Jobscheduler

Makro	Kurzbeschreibung
DJSI	Definitionsmakro; erstellt Namensdefinitionen, DSECTs oder Datenbereiche für die 24-Bit-Schnittstelle der Jobscheduler-Makros
DJSIPL	Definitionsmakro; erstellt Namensdefinitionen, DSECTs oder Datenbereiche für die 31-Bit-Schnittstelle der Jobscheduler-Makros
JSATTCH	verbindet den Jobscheduler mit dem Job Management System
JSDETCH	löst die Verbindung des Jobschedulers zum Job Management System
JSEXPCT	fordert vom Job Management System das nächste vorliegende Ereignis für den Jobscheduler an
JSINFO	überträgt die STREAM-PARAMETER (S-PAR) der Stream-Definition in einen anzugebenden Bereich
JSRUNJB	fordert den Klassenscheduler auf, den angegebenen Job zu starten
JSWAKE	initiiert ein Timer Event für den Jobscheduler

Diese Makros ermöglichen den Anschluss eines selbst entwickelten Job Schedulers an das Job Management System (JMS).
Ein Job Scheduler läuft als Anwenderprogramm im TU-Funktionszustand und ist somit leicht austauschbar. Er kommuniziert mit JMS über eine privilegierte Schnittstelle (Job Scheduler Schnittstelle).
Der Anwender kann den Standard Scheduler durch einen selbst entwickelten - seinen eigenen speziellen Anforderungen entsprechenden - Job Scheduler ersetzen, ohne Eingriffe in das Betriebssystem vornehmen zu müssen. Der Funktionsumfang eines solchen Schedulers wird aber durch die funktionelle Breite der Job Scheduler Schnittstelle begrenzt.

4.13 Makros, die nur CSECTs oder DSECTs generieren

Makro	Kurzbeschreibung
ARDS	beschreibt die Struktur der Abrechnungssätze
CUPAB	generiert Adressierungshilfen für die Operandentabellen der Makros RDATA, WROUT und WRTRD
DCSTA	generiert einen Bereich (CSECT) oder Adressierungshilfen (DSECT) für die Informationen, die der Makro TSTAT übergibt
DJINF	generiert eine DSECT/Datenliste für den Ausgabebereich des Makros JINF
DJSI	generiert Namensdefinitionen, DSECTs oder Datenbereiche für die 24-Bit-Schnittstelle der Jobscheduler-Makros
DJSIPL	generiert Namensdefinitionen, DSECTs oder Datenbereiche für die 31-Bit-Schnittstelle der Jobscheduler-Makros
DTMODE	generiert eine DSECT/Datenliste für die 31-Bit-Schnittstelle des Makros TMODE
JMGDJP	generiert eine DSECT/Datenliste zum Datenbereich des Makros JMGJPAR
LKEQU	generiert eine DSECT/Datenliste für die Ereignis-Typ-Codes und die globalen Returncodes, die von den verschiedenen Makros des DLMs gesetzt werden
LKLSB	generiert das Layout des Lock-Status-Blocks
PBTABD	generiert eine DSECT/Datenliste zur Eingabetabelle des Makros TABLE (nur für 31-Bit-Schnittstelle). Siehe Seite 1178.
VTCSET	generiert symbolische Namen, mit denen logische Steuerzeichen in Line-Mode-Ausgabenachrichten eingefügt bzw. Line-Mode-Eingaben aufgefunden werden können

5 Beschreibung der Makroaufrufe

Dieses Kapitel enthält die ausführliche Beschreibung der Makros an den Ablaufteil in alphabetischer Reihenfolge. In der Regel folgt die Beschreibung folgender Gliederung:

– Makroname und Funktion
– Allgemeines: Anwendungsgebiet und Makrotyp
– Makrobeschreibung: Makrofunktion
– Makroaufrufformat und Operandenbeschreibung
– Rückinformation und Fehleranzeigen: Returncode und Erläuterung
– Beispiel und/oder Auflösung der DSECT

Die im Handbuch enthaltenen TIAM-Makros beschreiben den Funktionsumfang der aktuellen TIAM-Version V13.2A (siehe auch Handbuch „TIAM“ [16]):

CUPAB **RDATA** **TCHNG** **TSTAT**
WROUT **WRTRD**

Die im Handbuch enthaltenen VTSU-Makros beschreiben den Funktionsumfang der aktuellen VTSU-Version V13.3A (siehe auch Handbuch „VTSU“ [30]):

DCSTA **VTCSET** **VTSUCB**

Die im Handbuch enthaltenen DBL-Makros beschreiben den Funktionsumfang des aktuellen Bindeladers BLSSERV V2.8A (siehe auch Handbuch „BLSSERV“ [4]):

ASHARE **BIND** **DSHARE** **ETABIT** **ETABLE**
GETPRGV **ILEMGT** **ILEMIT** **LDSLICE** **PINF**
SELPRGV **UNBIND** **VSVI1**

AINF – Betriebsmittelverbrauch messen

Allgemeines

Anwendungsgebiet: Abfragen und Zugriff zu Listen und Tabellen; siehe Seite 158
Makrotyp: S-Typ, MF-Format **1**:
31-Bit-Schnittstelle: Standardform/E-/L-/D-Form; siehe Seite 29

Betriebsmittel, deren Verbrauch zu messen ist, sind in so genannten Informationspaketen definiert. Folgende Informationspakete können für eine Messung ausgewählt werden:

Bezeichnung	Betriebsmittel/Messwerte
Globale Werte	Ermittlung der CPU-Zeit, Gesamtzahl der Ein- und Ausgaben, Anzahl der transportierten Datenblöcke, Working-Set-Integral.
Zeitverbrauch	Ermittlung der CPU-Zeit, Zeitstempel bzw. Laufzeit.
Ein-/Ausgabe-Zähler	Anzahl der Ein- und Ausgaben.

Der Aufbau der Ausgabefelder für die einzelnen Informationspakete ist anschließend an die Formatbeschreibung dargestellt.

Working-Set-Integral: Summe der Produkte von (Hauptspeicherseiten in KB * Benutzungszeit in Sekunden).
Anzahl der transportierten Datenblöcke: Datentransport von und zur lokalen Peripherie in PAM-Seiten bei öffentlichen, systemprivaten und benutzerprivaten Platten bzw. in 2 KB-Blöcken bei Magnetbändern und Unit-Record-Geräten.

Makrobeschreibung

Der Makro **AINF** ermittelt den Betriebsmittelverbrauch eines Auftrages und übergibt die Werte in einen Bereich des Benutzerprogramms.
Zur Messung des Betriebsmittelverbrauchs stehen zwei Verfahren zur Verfügung:

– Verbrauchsstempel-Verfahren:
  Betriebsmittelverbrauch seit Beginn des Auftrags (Format 1)
– Messaufgaben-Verfahren:
  Betriebsmittelverbrauch einzelner Programmabschnitte (Formate 2 und 3)

Der Operand MF=D generiert eine DSECT für den Datenbereich und die Definition der möglichen Rückkehrinformationen. Zusätzlich kann der **AINF**-Makro die Definitionen der Ausgabestruktur einzelner Informationspakete ausgeben (Format 4).

Hinweis

Der Eigenbedarf an CPU-Zeit für die Systemaufruf-Bearbeitung verfälscht die Messung geringfügig.

Funktionsweise

Beim **Verbrauchsstempel-Verfahren** ermittelt der **AINF**-Makro den Betriebsmittelverbrauch seit Beginn des Auftrages und überträgt die Verbrauchswerte in einen Bereich des Benutzerprogramms. Der Benutzer wählt die zu messenden Betriebsmittel aus, indem er im Makro Informationspakete angibt (GLOBAL, TIME, IOCNT), denen verschiedene Betriebsmittel zugeordnet sind.

Beim **Messaufgaben-Verfahren** ermittelt der **AINF**-Makro den Betriebsmittelverbrauch einzelner Programmabschnitte. Eine Messung kann an beliebigen Programmpunkten gestartet, unterbrochen, fortgesetzt oder beendet werden. Mehrere Messungen können ineinander verschachtelt und beliebig überlappt werden. Damit unterschiedliche Messungen eindeutig identifiziert werden können, enthält jede Messung eine eigene Messkennung. Beim Start einer Messung (Operand READY) definiert der Benutzer eine Messkennung und wählt Informationspakete aus, die die zu messenden Betriebsmittel enthalten. Soll die Messung unterbrochen werden, muss der Benutzer einen weiteren **AINF**-Aufruf (Operand INTR) an der Stelle geben, an der die Unterbrechung gewünscht ist. Das System ermittelt die Verbrauchswerte der angegebenen Betriebsmittel seit Start der Messung und übergibt sie, falls gewünscht, in einen Bereich des Benutzerprogramms. Will der Benutzer die unterbrochene Messung fortsetzen, muss er an der gewünschten Stelle einen **AINF**-Aufruf (Operand READY) mit der zugehörigen Messkennung geben.
Auf diese Weise lässt sich eine Messung beliebig unterbrechen und wieder fortsetzen. Nach jeder Unterbrechung ermittelt das System die Summe der Verbrauchswerte aller bisherigen Messabschnitte.
Die Messung wird beendet, sobald ein **AINF**-Aufruf (Operand FINISH) für die entsprechende Messkennung gegeben wird oder das Programm beendet wird. Das System überträgt die Summe der Verbrauchswerte aller Messabschnitte (von Start bis Ende der Messung) in einen Bereich des Benutzerprogramms.

Aufbau des **AINF**-Datenbereichs:

Feldname	Distanz	Inhalt
IAMID	00	Makrokennzeichnung
IAIMFC	04	Funktionsauswahl
IAIMTARE	08	Operandentyp
IAIMAREA	0C	Ausgabebereich
IAIMCHAI	10	Kettungsadresse
IAIMMID	14	Messkennung

Makroaufrufformate und Operandenbeschreibungen

In den nachfolgenden Operandenbeschreibungen sind die Operanden alphabetisch geordnet.

Format 1: Verbrauchsstempel-Verfahren

AINF
AREA=adr / (r) ,GLOBAL=**N**O / **Y**ES ,TIME=**N**O / **Y**ES ,IOCNT=**N**O / **Y**ES / EXT / STD ,MF=S / (E,..) / L

Mindestens einer der Operanden GLOBAL, TIME oder IOCNT muss mit dem Wert YES angegeben werden, da sonst der Makroaufruf abgewiesen wird mit

– X'10' in Register R15, wenn nur Standardwerte (NO) explizit angegeben wurden.

– einer MNOTE-Meldung (im ASSEMBLER-Protokoll), wenn keiner der Operanden GLOBAL, TIME oder IOCNT explizit angegeben wurde.

AREA=
bezeichnet ein Feld, in das die Betriebsmittel-Verbrauchswerte eingetragen werden. Das Feld ist auf Wortgrenze auszurichten. Die Länge richtet sich nach den spezifizierten Informationspaketen (siehe „Aufbau der Ausgabestrukturen“ auf Seite 182).

adr
symbolische Adresse des Feldes

(r)
Register mit dem Adresswert adr

GLOBAL=
bezeichnet das Informationspaket „Globale Werte“.

NO
Das Informationspaket „Globale Werte“ wird nicht gewählt.

YES
Der Betriebsmittelverbrauch wird ausgegeben.

IOCNT=
bezeichnet das Informationspaket „Ein-/Ausgabe-Zähler“.

NO
Das Informationspaket „Ein-/Ausgabe-Zähler“ wird nicht gewählt.

YES
Die Summe der Ein-/Ausgaben auf öffentliche Datenträger und system-private Platten wird ausgegeben.

STD
Die Summe der Ein-/Ausgaben auf öffentliche Datenträger und system-private Platten wird ausgegeben.

EXT
Wie STD oder YES, jedoch inklusive der Ein-/Ausgaben auf Bänder, Benutzer-private Platten und sonstige Geräte.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

TIME=
bezeichnet das Informationspaket „Zeitverbrauch“.

NO
Das Informationspaket „Zeitverbrauch“ wird nicht gewählt.

YES
CPU-Zeit und Zeitstempel sind auszugeben. Der Zeitstempel beim Verbrauchsstempelverfahren ist der auf Sekunden und Nanosekunden umgerechnete Wert des TOD-Registers.

Format 2: Messaufgaben-Verfahren
Start oder Wiederaufnahme einer Messung

AINF
READY='messid' / adr / (r) ,GLOBAL=**N**O / **Y**ES ,TIME=**N**O / **Y**ES ,IOCNT=**N**O / **Y**ES / EXT / STD [,CHAIN=adr / (r)] ,MF=S / (E,..) / L

Mindestens einer der Operanden GLOBAL, TIME oder IOCNT muss mit dem Wert YES angegeben werden, da sonst der Makroaufruf abgewiesen wird mit

– X'10' in Register R15, wenn nur Standardwerte („NO“) explizit angegeben wurden.

– einer MNOTE-Meldung (im ASSEMBLER-Protokoll), wenn keiner der Operanden GLOBAL, TIME oder IOCNT explizit angegeben wurde.

CHAIN=
ermöglicht eine Verkettung des Makros **AINF** mit einem weiteren **AINF**-Makroaufruf durch Angabe einer Kettungsadresse. Diese zeigt auf den mit MF=L erzeugten Datenbereich des zweiten **AINF**-Makros. Siehe Hinweise zur Verkettung (Seite 185).

adr
Adresse des mit MF=L erzeugten Datenbereichs des zweiten Makros

(r)
r = Register mit dem Adresswert adr. Der Operand ist nur für die Standard- oder L-Form des Makroaufrufs erlaubt.

GLOBAL=
bezeichnet das Informationspaket „Globale Werte“.

NO
Das Informationspaket „Globale Werte“ wird nicht gewählt.

YES
Der Betriebsmittelverbrauch wird ausgegeben.

IOCNT=
bezeichnet das Informationspaket „Ein-/Ausgabe-Zähler“.

NO
Das Informationspaket „Ein-/Ausgabe-Zähler“ wird nicht gewählt.

YES
Die Summe der Ein-/Ausgaben auf öffentliche Datenträger und system-private Platten wird ausgegeben.

STD
Die Summe der Ein-/Ausgaben auf öffentliche Datenträger und system-private Platten wird ausgegeben.

EXT
Wie STD oder YES, jedoch inklusive der Ein-/Ausgaben auf Bänder, Benutzer-private Platten und sonstige Geräte.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

READY=
startet eine neue Messung und ordnet ihr die angegebene Messkennung zu.

'messid'
messid = Messkennung, die der Messung zugeordnet wird. Länge: maximal 8 Zeichen. Ist der Name 'messid' schon einer (unterbrochenen) Messung zugeordnet, so wird diese Messung wieder aufgenommen.

adr
symbolische Adresse des Feldes, das die Messkennung enthält.

(r)
r = Register mit dem Adresswert adr

TIME=
bezeichnet das Informationspaket „Zeitverbrauch“.

NO
Das Informationspaket „Zeitverbrauch“ wird nicht gewählt.

YES
CPU-Zeit und Zeitstempel sind auszugeben.

Hinweis

Die Auswahl der Informationspakete (GLOBAL, TIME, IOCNT) bei der Wiederaufnahme einer Messung muss mit den Angaben beim Start dieser Messung übereinstimmen. Bei abweichenden Angaben wird die Messung mit den beim Start vereinbarten Informationspaketen fortgesetzt und in Register R15 die Rückinformation X'24' gespeichert.

Format 3: Messaufgaben-Verfahren
Messunterbrechung oder Messbeendigung

```
AINF

{ INTR='messid' / adr / (r)[,AREA=adr / (r)] }
{ FINISH='messid' / adr / (r),AREA=adr / (r) }

[,CHAIN=adr / (r)]
,MF=S / (E,..) / L
```

AREA=
bezeichnet ein Feld, in das die Betriebsmittel-Verbrauchswerte eingetragen werden. Das Feld ist auf Wortgrenze auszurichten. Die Länge richtet sich nach den spezifizierten Informationspaketen (siehe „Aufbau der Ausgabestrukturen“ auf Seite 182).

adr
symbolische Adresse des Feldes.

(r)
r = Register mit dem Adresswert adr.

CHAIN=
ermöglicht eine Verkettung des Makros **AINF** mit einem weiteren **AINF**-Makroaufruf durch Angabe einer Kettungsadresse. Diese zeigt auf den mit MF=L erzeugten Datenbereich des zweiten **AINF**-Makros. Siehe Hinweise zur Verkettung (Seite 185).

adr
symbolische Adresse des mit MF=L erzeugten Datenbereichs des zweiten Makros.

(r)
r = Register mit dem Adresswert adr. Der Operand ist nur für die Standard- oder L-Form des Makroaufrufs erlaubt.

FINISH=
beendet die Messung mit der angegebenen Messkennung.

'messid'
messid = Messkennung, die der Messung zugeordnet wurde.

adr
symbolische Adresse des Feldes mit der Messkennung.

(r)
r = Register mit dem Adresswert adr.

INTR=
unterbricht die Messung mit der angegebenen Messkennung.

'messid'
messid = Messkennung, die der Messung zugeordnet wurde.

adr
symbolische Adresse des Feldes mit der Messkennung.

(r)
r = Register mit dem Adresswert adr.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Format 4: DSECTs für Datenbereich und Ausgabestruktur
(es muss mindestens 1 Operand angeben werden)

AINF
[MF=D] ,P=I / P / * [,GLOBAL=D] [,TIME=D] [,IOCNT=D]

GLOBAL=D
generiert eine DSECT für die Ausgabestruktur des Informationspakets „Globale Werte“.

IOCNT=D
generiert eine DSECT für die Ausgabestruktur des Informationspakets „Ein-/Ausgabe-Zähler“.

MF=D
generiert eine DSECT für den Datenbereich und Equates für den Returncode.

P=
bezeichnet ein Präfix für die symbolischen Namen der DSECT.

I
Die generierten Feldnamen beginnen mit dem Präfix `I`, die Längendefinitionen mit `LIxx`.

p
Präfix, der allen generierten Feldnamen der DSECT vorangestellt wird. Der Präfix wird außerdem in die Namen der Längendefinitionen übernommen: `L&p.xx`.
Länge des Präfixes = 1 Buchstabe.

*****
kein Buchstabe wird den generierten Feldnamen vorangestellt und in die Namen der Längendefinitionen übernommen.

TIME=D
generiert eine DSECT für die Ausgabestruktur des Informationspakets „Zeitverbrauch“.

```
          AINF  MF=D
1         #INTF INTNAME=AINF,REFTYPE=REQUEST,INTCOMP=001          GS 950
1         MFPRE DNAME=AIMPL,MF=D,PREFIX=I,MACID=AIM,DMACID=AIM    :*R200
2 IAIMPL  DSECT ,
2               *,##### PREFIX=I, MACID=AIM #####
1         DS    0F                      AINF PARAMETER LIST
1 IAIMID  DS    CL2                     IDENTIFICATION OF AINF MACRO
1 IAIMAINF EQU  C'AI'                   - AINF INFORMATION STAMP
1 IAIMPMAC EQU  C'PM'                   - PROG MEASUREMENT FUNCT.
1 IAIMVER DC    XL2'075'                 AINF MACRO VERSION
1 IAIMFC  DS    X                       FUNCTION CODE
1 IAIMFCCV EQU  X'01'                   - CURRENT USAGE VALUES
1 IAIMFCRE EQU  X'02'                   - MEASUREMENT READY
1 IAIMFCIN EQU  X'04'                   - MEASUREMENT INTR
1 IAIMFCFI EQU  X'08'                   - MEASUREMENT FINISH
1 IAIMNPAR DC   FL1'4'                     # OF PARAMETERS
1 *                                      + 8 = DIST OF FIRST PARAM ADDR
1 IAIMMSK1 DC   X'00'                   INFORMATION PACKAGE MASK1
1 IAIMGLOB EQU  X'80'                   - "GLOBAL"    REQUIRED
1 IAIMTIME EQU  X'40'                   - "TIME"      REQUIRED
1 IAIMIOCN EQU  X'20'                   - "IOCNT=STD" REQUIRED
1 IAIMIOCX EQU  X'10'                   - "IOCNT=EXT" REQUIRED
1 *        EQU  X'0F'                   - ALL OTHER BITS RESERVED
1 IAIMMSK2 DC   X'00'                   INFORMATION PACKAGE MASK2
1 *        EQU  X'FF'                   - ALL BITS RESERVED
1 IAIMTARE DS   X                       TYPE OF AREA PARAM
1 IAIMNONE EQU  X'00'                   - NOT SPECIFIED
1 IAIMADDR EQU  X'01'                   - GIVEN AS DIRECT ADDRESS
1 IAIMREG  EQU  X'02'                   - GIVEN IN A REGISTER
1 IAIMTCHA DS   X                       TYPE OF CHAIN PARAM
1 *                                     - EQUATES AS ABOVE
1 IAIMTMID DS   X                       TYPE OF MEASUREMENT ID
1 *                                     - EQUATES AS ABOVE, +
1 IAIMSTR  EQU  X'03'                   - GIVEN AS A STRING
1 IAIMTMIX DS   X                       TYPE OF MID CONTINUED
1 IAIMEXT  EQU  X'FF'                   - EXTENDED PARAMETER
1 IAIMAREA DS   A                       OUTPUT AREA ADDRESS
1 IAIMCHAI DS   A                       CHAIN ADDRESS
1 IAIMMID  DS   0A                      MEASUREMENT ID ADDR  OR
1 IAIMMIDS DS   CL8                     MEASUREMENT ID STRING
1 IAIMPLE  EQU  *                       END OF AINF PARAM LIST
1 LIAIMPL  EQU  *-IAIMPL               LENGTH OF AINF PARAM LIST
1          SPACE 2
```

Anschließend an die DSECT folgt die Definition der möglichen Rückinformationen. Dieser Teil der Auflösung ist bei den Rückinformationen aufgelistet.

Aufbau der Ausgabestrukturen

1. Informationspaket **Globale Werte**

Feldname	Distanz	Inhalt
IAIGTCPU	00	CPU-Zeit im folgenden Format: Distanz 00: volle Sekunden Distanz 04: Rest in Nanosekunden Derzeitige Messgenauigkeit: 1 Mikrosekunden
IAIG#IOS	08	Gesamtzahl der Ein-/Ausgaben
	0C	Anzahl der Datenblöcke
	10	Working-Set-Integral

```
           AINF  GLOBAL=D
1          #INTF INTNAME=AINF,REFTYPE=REQUEST,INTCOMP=001          GS 950
1          MFPRE DNAME=AIAREA,MF=D,PREFIX=I,MACID=AIA,DMACID=AIA   :*R200
2 IAIAREA  DSECT ,
2                *,##### PREFIX=I, MACID=AIA #####
1          DS    0F                       START OF OUTPUT AREA
1          MFPRE DNAME=AIGLOB,ALIGN=F,PREFIX=I,MACID=AIG,MF=S,     :*R200C
1                DMACID=AIG                                        :*R200
2          CNOP  0,4
2 IAIGLOB  DS    0F
1 *                                  "GLOBAL" INFO PACKAGE
1 IAIGTCPU DS    0FL8                CPU TIME
1 IAIGCPUS DS    F                   CPU TIME SECONDS
1 IAIGCPUN DS    F                   CPU TIME NANOSECONDS
1 IAIG#IOS DS    F                   TOTAL # IO'S
1 IAIG#BLK DS    F                   TOTAL # BLOCKS            GS 090
1 IAIGWSI  DS    FL8                 WORKING SET INTEGRAL      GS 090
1 IAIGLOBE EQU   *                   END OF "GLOBAL" INFO
1 LIAIGLOB EQU   *-IAIGLOB           LENGTH OF "GLOBAL" INFO
1          SPACE 2
1 *                                   END OF OUTPUT AREA
1          MFPRE DNAME=AIAEND,ALIGN=F,PREFIX=I,MACID=AIA,MF=S,     :*R200C
1                DMACID=AIA                                        :*R200
2          CNOP  0,4
2 IAIAEND  DS    0F
1 LIAIAREA EQU   *-IAIAREA           LENGTH OF OUTPUT AREA
1          SPACE 2
```

Die ermittelte CPU-Zeit setzt sich zusammen aus:
- CPU-Zeit im nicht-privilegierten Programm-Zustand.
- CPU-Zeit im privilegierten Programm-Zustand (Verarbeitung von SVC-Aufrufen und Programmfehlern)
- CPU-Zeit während der Kommandobearbeitung

2. Informationspaket **Zeitverbrauch**

Feldname	Distanz	Inhalt
IAITTCPU	00	CPU-Zeit im folgenden Format: Distanz 00: volle Sekunden Distanz 04: Rest in Nanosekunden Derzeitige Messgenauigkeit: 1 Mikrosekunden
IAITETIM	08	bei Verbrauchsstempel-Verfahren: Zeitstempelmessung bei Messaufgaben-Verfahren: Laufzeitmessung Distanz 08: volle Sekunden Distanz 0C: Rest in Nanosekunden Derzeitige Messgenauigkeit: 1 Mikrosekunde

```
           AINF  TIME=D
1          #INTF INTNAME=AINF,REFTYPE=REQUEST,INTCOMP=001          GS 950
1          MFPRE DNAME=AIAREA,MF=D,PREFIX=I,MACID=AIA,DMACID=AIA   :*R200
2 IAIAREA  DSECT ,
2                *,##### PREFIX=I, MACID=AIA #####
1          DS    0F                      START OF OUTPUT AREA
1          MFPRE DNAME=AITIME,ALIGN=F,PREFIX=I,MACID=AIT,MF=S,     :*R200C
1                DMACID=AIT                                        :*R200
2          CNOP  0,4
2 IAITIME  DS    0F
1 *                                      "TIME"   INFO PACKAGE
1 IAITTCPU DS    0FL8                    CPU TIME
1 IAITCPUS DS    F                       CPU TIME SECONDS
1 IAITCPUN DS    F                       CPU TIME NANOSECONDS
1 IAITETIM DS    0FL8                    ELAPSED TIME / TIME STAMP
1 IAITTIMS DS    F                       TIME IN SECONDS
1 IAITTIMN DS    F                       TIME NANOSECONDS
1 IAITIMEE EQU   *                       END OF "TIME"   INFO
1 LIAITIME EQU   *-IAITIME              LENGTH OF "TIME"   INFO
1          SPACE 2
1 *                                      END OF OUTPUT AREA
1          MFPRE DNAME=AIAEND,ALIGN=F,PREFIX=I,MACID=AIA,MF=S,     :*R200C
1                DMACID=AIA                                        :*R200
2          CNOP  0,4
2 IAIAEND  DS    0F
1 LIAIAREA EQU   *-IAIAREA              LENGTH OF OUTPUT AREA
1          SPACE 2
```

Wie sich die CPU-Zeit zusammensetzt, ist beim Informationspaket „Globale Werte“ beschrieben.
Der Zeitstempel beim Verbrauchsstempelverfahren ist der auf Sekunden und Nanosekunden umgerechnete Wert des TOD-Registers.

3. Informationspaket **Ein-/Ausgabe-Zähler**

Feldname	Distanz	Inhalt
IAII#IOS	00	Gesamtzahl der Ein-/Ausgaben
IAIIIOPD	04	Ein-/Ausgaben auf gemeinschaftlichen Datenträgern
IAIIIOSD	08	Ein-/Ausgaben auf mehrbenutzbare private Platten
IAIIIOUD	0C	Ein-/Ausgaben auf task-exclusive Platten
IAIIIOTP	10	Ein-/Ausgaben auf Magnetbänder
IAIIIOUR	14	Ein-/Ausgaben auf sonstige Geräte

```
           AINF  IOCNT=D
1          #INTF INTNAME=AINF,REFTYPE=REQUEST,INTCOMP=001          GS 950
1          MFPRE DNAME=AIAREA,MF=D,PREFIX=I,MACID=AIA,DMACID=AIA   :*R200
2 IAIAREA  DSECT ,
2                *,##### PREFIX=I, MACID=AIA #####
1          DS    0F                        START OF OUTPUT AREA
1          MFPRE DNAME=AIIOCN,ALIGN=F,PREFIX=I,MACID=AII,MF=S :*R200DMACIC
1                D=AII                                        :*R200
2          CNOP  0,4
2 IAIIOCN  DS    0F
1 *                                        "IOCNT"  INFO PACKAGE
1 IAII#IOS DS    F                         TOTAL # IO'S
1 IAIIIOPD DS    F                 # IO'S ON PUBLIC DEVICES
1 IAIIIOSD DS    F                 # IO'S ON SYSTEM PRIV. DISKS
1 IAIIIOUD DS    F                 # IO'S ON USER PRIVATE DISKS
1 IAIIIOTP DS    F                 # IO'S ON TAPE DEVICES
1 IAIIIOUR DS    F                 # IO'S ON UNIT RECORD DEVICES
1 IAIIOCNE EQU   *                         END OF "IOCNT"  INFO
1 LIAIIOCN EQU   *-IAIIOCN               LENGTH OF "IOCNT"  INFO
1          SPACE 2
1 *                                        END OF OUTPUT AREA
1          MFPRE DNAME=AIAEND,ALIGN=F,PREFIX=I,MACID=AIA,MF=S,     C
1                DMACID=AIA
2          CNOP  0,4
2 IAIAEND  DS    0F
1 LIAIAREA EQU   *-IAIAREA               LENGTH OF OUTPUT AREA
```

Bei den Ein-/Ausgaben werden nicht berücksichtigt:
- Terminal-Ein-/Ausgaben bei Dialog- und Transaktionsprozessen
- Paging-Ein-/Ausgaben

Zusätzlich gilt bei den Einzel-Ein-/Ausgaben auf Bänder, Benutzer-private Platten und sonstige Geräte (siehe Operand IOCNT=EXT):
Nicht berücksichtigt werden Ein-/Ausgaben
- von privilegierten Programmen
- auf Geräte, die vor Programmlauf freigegeben wurden (durch REMOVE-FILE-LINK oder SECURE-RESOURCE-ALLOCATION)

Da diese Einschränkungen nicht für die Gesamtzahl der Ein-/Ausgaben gelten, kann es sein, dass der errechnete Wert für die Gesamtzahl größer ist als die Summe der Einzelwerte.

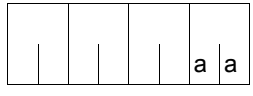
Hinweise zur Verkettung beim Messaufgaben-Verfahren (Operand CHAIN)

- Der durch CHAIN bezeichnete **AINF**-Datenbereich wird unmittelbar nach dem Datenbereich des aufrufenden **AINF**-Makros abgearbeitet. Dadurch können mehrere **AINF**-Aufrufe innerhalb einer SVC-Bearbeitung ausgeführt werden; z.B. Beenden oder Unterbrechen einer Messung und gleichzeitiger Start oder Wiederanlauf einer neuen Messung (unter einer anderen Messkennung); oder Unterbrechen einer Messung zur Ausgabe der aktuellen Messergebnisse und sofortiger Wiederanlauf der gleichen Messung.
- Die Übergabe der Adressen in Register erleichtert eine Reentrant-Programmierung, ist aber nur sinnvoll, solange nicht mehr als zwei Operandenlisten miteinander verknüpft werden.
- Nach Ablauf der SVC-Behandlung enthält Register R1 die Adresse des zuletzt bearbeiteten Datenbereichs.
- Tritt ein Fehler im **AINF**-Datenbereich einer Kette auf, so wird die Kette an dieser Stelle abgebrochen. Register R1 enthält dann die Adresse des fehlerhaften Datenbereichs.

Rückinformation und Fehleranzeigen

Register R1 enthält die Adresse des Datenbereichs; bei Makroverkettung die Adresse des zuletzt bearbeiteten Datenbereichs.

R15:



Über die Ausführung des Makros AINF wird ein Returncode übergeben (im rechtsbündigen Byte von Register R15; die restlichen drei Byte sind gelöscht).

Feldname	X'aa'	Bedeutung
IAIROK	X'00'	Aufruf wurde erfolgreich ausgeführt.
IAIRADER	X'04'	Aufruf wurde nicht ausgeführt, wegen ungültiger Adresse der Operandenliste.
IAIRFUER	X'08'	Aufruf wurde nicht ausgeführt, wegen ungültiger Makrokennung oder ungültigem Funktionscode (weder Verbrauchsstempelverfahren noch READY, INTR oder FINISH).
IAIRCHER	X'0C'	Aufruf wurde ausgeführt trotz ungültiger Kettungsadresse im zuletzt bearbeiteten Operandenblock (READY, INTR, FINISH).
IAIRPAER	X'10'	Aufruf wurde nicht ausgeführt, wegen – ungültigem Register oder ungültiger Adresse des Ausgabebereichs; – ungültigem Register oder ungültiger Adresse einer Messkennung; – fehlenden Operandenangaben.
IAIRMRDY	X'14'	Aufruf wurde nicht ausgeführt, weil unter der angegebenen Messkennung bereits eine Messung stattfindet (bei READY).
IAIRNCL5	X'18'	Aufruf wurde nicht ausgeführt, weil kein Klasse-5-Speicher mehr zur Anlage des Arbeitsbereichs verfügbar ist (bei READY).
IAIRMNTF	X'1C'	Aufruf wurde nicht ausgeführt, weil INTR oder FINISH ohne ein zugehöriges READY abgesetzt wurden; d.h. die Messung wurde noch nicht gestartet.
IAIRMINT	X'20'	Aufruf wurde ausgeführt, obwohl ein FINISH für eine mit INTR unterbrochene Messung abgesetzt wurde; d.h. die Messung wurde noch nicht fortgesetzt.
IAIRIRIN	X'24'	Aufruf wurde ausgeführt, obwohl der READY-Aufruf zur Wiederaufnahme einer Messung (nach INTR) nicht dieselben Informationen enthält wie der READY-Aufruf beim Start der Messung.

Bei Makroverkettung bezieht sich die Angabe „Aufruf ausgeführt / nicht ausgeführt“? auf den Datenbereich, der zuletzt bearbeitet wurde (=Datenbereich, dessen Adresse in Register R1 gespeichert ist). Bei Fehler in der Verkettung wird die Kette beim fehlerhaften Datenbereich abgebrochen. Dabei bedeutet,

„ausgeführt“ dass der letzte (fehlerhafte) Datenbereich noch bearbeitet wurde. Falls das Benutzerprogramm den Fehler korrigieren kann, müsste die Weiterverarbeitung mit dem nächsten Datenbereich angefordert werden.

„nicht ausgeführt“ dass der letzte (fehlerhafte) Datenbereich nicht mehr bearbeitet werden konnte. Falls das Benutzerprogramm den Fehler korrigieren kann, müsste die Verarbeitung mit derselben (korrigierten) Operandenliste fortgesetzt werden.

Beispiel 1: Verbrauchsstempel-Verfahren

```
AINF1    START
         PRINT NOGEN
         BALR  3,0
         USING *,3
         LA    5,AREAONE
         USING AINFDATA,5
         AINF  GLOBAL=Y,TIME=Y,IOCNT=EXT,AREA=AREAONE ----------------   (1)
         PRINT NOGEN
*
*        Display CPU time
*
         MVC   MESSTXT(L'MCPU),MCPU
         L     8,IAIGCPUS              CPU time in seconds
         BAL   7,PKD2ZND               Call conversion routine ------>
         MVC   MESSTXT+20(10),ASSIST2+6
         MVI   MESSTXT+30,C'.'
         L     8,IAIGCPUN              CPU time in nanoseconds
         BAL   7,PKD2ZND               Call conversion routine ------>
         MVC   MESSTXT+31(9),ASSIST2+7
         BAL   7,OUTPUT                Call output routine ---------->
*
*        Display elapsed time
*
         MVC   MESSTXT(L'MELA),MELA
         L     8,IAITTIMS              Elapsed time in seconds
         BAL   7,PKD2ZND               Call conversion routine ------>
         MVC   MESSTXT+20(10),ASSIST2+6
         MVI   MESSTXT+30,C'.'
         L     8,IAITTIMN              Elapsed time in nanoseconds
         BAL   7,PKD2ZND               Call conversion routine ------>
         MVC   MESSTXT+31(9),ASSIST2+7
         BAL   7,OUTPUT                Call output routine ---------->
*
*        Display # IO's
*
         MVC   MESSTXT(L'MIOS),MIOS    * Total # IO's ****************
         L     8,IAII#IOS
         BAL   7,PKD2ZND               Call conversion routine ------>
         MVC   MESSTXT+30(10),ASSIST2+6
         BAL   7,OUTPUT                Call output routine ---------->
*
         MVC   MESSTXT(L'MIOP),MIOP    * Public IO's *****************
         L     8,IAIIIOPD
         BAL   7,PKD2ZND               Call conversion routine ------>
         MVC   MESSTXT+30(10),ASSIST2+6
         BAL   7,OUTPUT                Call output routine ---------->
```

```
*
         MVC   MESSTXT(L'MIOSP),MIOSP  * System private IO's *********
         L     8,IAIIIOSD
         BAL   7,PKD2ZND               Call conversion routine ------>
         MVC   MESSTXT+30(10),ASSIST2+6
         BAL   7,OUTPUT                Call output routine ---------->
*
         MVC   MESSTXT(L'MIOSP),MIOSP  * User private IO's ***********
         L     8,IAIIIOUD

         BAL   7,PKD2ZND               Call conversion routine ------>
         MVC   MESSTXT+30(10),ASSIST2+6
         BAL   7,OUTPUT                Call output routine ---------->
*
         MVC   MESSTXT(L'MITP),MITP    * Tape devices ***************
         L     8,IAIIIOTP
         BAL   7,PKD2ZND               Call conversion routine ------>
         MVC   MESSTXT+30(10),ASSIST2+6
         BAL   7,OUTPUT                Call output routine ---------->
*
         MVC   MESSTXT(L'MIUR),MIUR    * Unit record devices *********
         L     8,IAIIIOUR
         BAL   7,PKD2ZND               Call conversion routine ------>
         MVC   MESSTXT+30(10),ASSIST2+6
         BAL   7,OUTPUT                Call output routine ---------->
END      TERM
*
*        Output routine
*
OUTPUT   WROUT MESSAGE,END,PARMOD=31
2                *,@DCEO      999    921011   53531004
         MVI   MESSTXT,C' '            Clear MESSTXT for next output
         MVC   MESSTXT+1(L'MESSTXT-1),MESSTXT
         BR    7                       Return ->
*
*        Conversion routine
*
PKD2ZND  CVD   8,ASSIST1               Convert register contents to
         UNPK  ASSIST2,ASSIST1         zoned decimal
         MVZ   ASSIST2+15(1),=X'F0'
         BR    7                       Return ->
*
*        Definitions
*
AREAONE  DS    0F
         DS    6F
         DS    4F
         DS    6F
```

```
  MESSAGE  DC    Y(ENDMESS-MESSAGE)      Record length
           DS    CL2                     Reserved
           DC    X'01'                   Print feed control character
  MESSTXT  DC    CL60' '                 Text
  ENDMESS  EQU   *
  ASSIST1  DS    D
  ASSIST2  DS    CL16
  MCPU     DC    C'CPU Time used:'
  MELA     DC    C'Elapsed Time used:'
  MIOS     DC    C'Total number of IOs:'
  MIOP     DC    C'IOs on public devices:'
  MIOSP    DC    C'IOs on system private disks:'
  MIOUP    DC    C'IOs on user private disks:'
  MITP     DC    C'IOs on tape devices:'
  MIUR     DC    C'IOs on unit record devices:'
           PRINT GEN
  AINFDATA AINF  GLOBAL=D,TIME=D,IOCNT=D ──────────────────────────────  (2)
1          #INTF INTNAME=AINF,REFTYPE=REQUEST,INTCOMP=001          GS 950
1 AINFDATA MFPRE DNAME=AIAREA,MF=D,PREFIX=I,MACID=AIA,DMACID=AIA   :*R200
2 AINFDATA DSECT ,
2                *,##### PREFIX=I, MACID=AIA #####
1 IAIAREA  DS    OF                      START OF OUTPUT AREA      LKH075
1          MFPRE DNAME=AIGLOB,ALIGN=F,PREFIX=I,MACID=AIG,MF=S,     :*R200C
1                DMACID=AIG                                        :*R200
2          CNOP  0,4
2 IAIGLOB  DS    OF
1 *                                    "GLOBAL" INFO PACKAGE     LKH075
1 IAIGTCPU DS    OFL8                  CPU TIME                  LKH075
1 IAIGCPUS DS    F                     CPU TIME SECONDS          LKH075
1 IAIGCPUN DS    F                     CPU TIME NANOSECONDS      LKH075
1 IAIG#IOS DS    F                     TOTAL # IO'S              LKH075
1 IAIG#BLK DS    F                     TOTAL # BLOCKS            GS 090
1 IAIGWSI  DS    FL8                   WORKING SET INTEGRAL      GS 090
1 IAIGLOBE EQU   *                     END OF "GLOBAL" INFO      LKH075
1 LIAIGLOB EQU   *-IAIGLOB            LENGTH OF "GLOBAL" INFO   LKH075
1          SPACE 2                                               LKH075
1          MFPRE DNAME=AITIME,ALIGN=F,PREFIX=I,MACID=AIT,MF=S,     :*R200C
1                DMACID=AIT                                        :*R200
2          CNOP  0,4
2 IAITIME  DS    OF
1 *                                    "TIME"   INFO PACKAGE     LKH075
1 IAITTCPU DS    OFL8                  CPU TIME                  LKH075
1 IAITCPUS DS    F                     CPU TIME SECONDS          LKH075
1 IAITCPUN DS    F                     CPU TIME NANOSECONDS      LKH075
1 IAITETIM DS    OFL8                  ELAPSED TIME / TIME STAMP LKH075
1 IAITTIMS DS    F                     TIME IN SECONDS           LKH075
1 IAITTIMN DS    F                     TIME NANOSECONDS          LKH075
1 IAITIMEE EQU   *                     END OF "TIME"   INFO      LKH075
```

```
1 LIAITIME EQU   *-IAITIME              LENGTH OF "TIME"   INFO   LKH075
1          SPACE 2                                                 LKH075
1          MFPRE DNAME=AIIOCN,ALIGN=F,PREFIX=I,MACID=AII,MF=S :*R200DMACIC
1                D=AII                                           :*R200
2          CNOP  0,4
2 IAIIOCN  DS    0F
1 *                                      "IOCNT"  INFO PACKAGE     LKH075
1 IAII#IOS DS    F                       TOTAL # IO'S              LKH075
1 IAIIIOPD DS    F                # IO'S ON PUBLIC DEVICES         LKH075
1 IAIIIOSD DS    F                # IO'S ON SYSTEM PRIV. DISKS     LKH075
1 IAIIIOUD DS    F                # IO'S ON USER PRIVATE DISKS     LKH075
1 IAIIIOTP DS    F                # IO'S ON TAPE DEVICES           LKH075
1 IAIIIOUR DS    F                # IO'S ON UNIT RECORD DEVICES    LKH075
1 IAIIOCNE EQU   *                       END OF "IOCNT"  INFO      LKH075
1 LIAIIOCN EQU   *-IAIIOCN              LENGTH OF "IOCNT"  INFO   LKH075
1          SPACE 2                                                 LKH075
1 *                                      END OF OUTPUT AREA        LKH075
1          MFPRE DNAME=AIAEND,ALIGN=F,PREFIX=I,MACID=AIA,MF=S,     :*R200C
1                DMACID=AIA                                        :*R200
2          CNOP  0,4
2 IAIAEND  DS    0F
1 LIAIAREA EQU   *-IAIAREA              LENGTH OF OUTPUT AREA     LKH075
1          SPACE 2                                                 LKH075
           END
                 =X'F0'
```

(1) Der Betriebsmittelverbrauch seit Beginn des Auftrags soll gemessen werden entsprechend den Informationspaketen „Globale Werte“, „Zeitverbrauch“ und „Ein-/Ausgabezähler“. Die Messergebnisse sollen in den Bereich `AREAONE` übertragen werden.

(2) Eine DSECT für die einzelnen Informationspakete wird generiert. Mit den dort definierten Feldnamen können die Messwerte in `AREAONE` symbolisch adressiert werden.

Ablaufprotokoll:

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,ainf1), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,ainf1))
%  ASS6011 ASSEMBLY TIME: 380 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 107 MSEC
//end
%  ASS6012 END OF ASSEMBH
/start-executable-program library=macexmp.lib,element-or-symbol=ainf1
%  BLS0523 ELEMENT 'AINF1', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'AINF1', VERSION ' ' OF '<date> <time>' LOADED
CPU Time used:      0000000058.526666000 ------------------------------  (3)
Elapsed Time used:  0000001333.942669000
Total number of IOs:          0000019462
IOs on public devices:        0000019462
IOs on system private disks:  0000000000
IOs on system private disks:  0000000000
IOs on tape devices:          0000000000
IOs on unit record devices:   0000000000
```

(3) Die ermittelten Messwerte werden ausgegeben.

Beispiel 2: Messaufgaben-Verfahren

Im Programm AINF2 werden zwei Messungen gestartet und diesen Messungen die Messkennungen MES1 und MES2 zugeordnet.
MES1 fordert Werte über den Betriebsmittelverbrauch an, die den Informationspaketen „Zeitverbrauch“ und „Ein-/Ausgabe-Zähler“ entsprechen. Beim Beenden der Messung sollen die Informationen in den Bereich AREA1 ausgegeben werden.
MES2 fordert Werte über den Betriebsmittelverbrauch an, die den Informationspaketen „Globale Werte“ und „Ein-/Ausgabe-Zähler“ entsprechen. Beim Unterbrechen der Messung, d.h. am Ende des ersten Messabschnittes sollen die Informationen in den Bereich AREA2A ausgegeben werden; beim Beenden der Messung in den Bereich AREA2B.
Zusätzlich werden für MES1 und MES2 die DSECTS für die Ausgabestrukturen der Informationspakete generiert und mit den entsprechenden Ausgabebereichen verknüpft.

```
AINF2    START
         PRINT NOGEN
         BALR  3,0
         USING *,3
MES1B    AINF  READY='MES1',TIME=Y,IOCNT=EXT    Start MES1
         LTR   15,15                   If Returncode not zero
         BNE   ERROR                   go to ERROR ->
         MVC   MESSTXT,STARTMSG
         BAL   7,OUTPUT                Call output routine ---------->
MES2B    AINF  READY='MES2',GLOBAL=Y,IOCNT=Y             Start MES2
         LTR   15,15                   If Returncode not zero
         BNE   ERROR                   go to ERROR ->
         MVI   STARTMSG+3,C'2'         Modify start message
         MVC   MESSTXT,STARTMSG
         BAL   7,OUTPUT                Call output routine ---------->
MES2I    AINF  INTR='MES2',AREA=AREA2A                   Interrupt MES2
MES2C    AINF  READY='MES2',GLOBAL=Y,IOCNT=Y             Continue MES2
         BKPT
MES2E    AINF  FINISH='MES2',AREA=AREA2B                 Terminate MES2
MES1E    AINF  FINISH='MES1',AREA=AREA1         Terminate MES1
*
*        Display Elapsed Time (MES1)
*
         USING MES1PAR,5
         LA    5,AREA1
         MVC   MESSTXT(L'MELA),MELA
         L     8,AAITTIMS              Elapsed time in seconds
         BAL   7,PKD2ZND               Call conversion routine ------>
         MVC   MESSTXT+20(10),ASSIST2+6
         MVI   MESSTXT+30,C'.'
         L     8,AAITTIMN              Elapsed time in nanoseconds
         BAL   7,PKD2ZND               Call conversion routine
         MVC   MESSTXT+31(9),ASSIST2+7
```

```
            BAL   7,OUTPUT                Call output routine ---------->
*
*           Display # IO's at interrupt (MES2)
*
            USING MES2PAR,6
            LA    6,AREA2A
            MVC   MESSTXT(L'MIOPI),MIOPI  * Public IO's *****************
            L     8,BAIIIOPD
            BAL   7,PKD2ZND               Call conversion routine ------>
            MVC   MESSTXT+30(10),ASSIST2+6
            BAL   7,OUTPUT                Call output routine ---------->
*
*           Display # IO's at end (MES2)
*
            LA    6,AREA2B
            MVC   MESSTXT(L'MIOPE),MIOPE  * Public IO's *****************
            L     8,BAIIIOPD
            BAL   7,PKD2ZND               Call conversion routine ------>
            MVC   MESSTXT+30(10),ASSIST2+6
            BAL   7,OUTPUT                Call output routine ---------->
END         TERM
ERROR       NOP   ERROR

*           :                 Error handling
            TERM
*
*           Output routine
*
OUTPUT      WROUT MESSAGE,END,PARMOD=31
2                 *,@DCEO      999    921011   53531004
            MVI   MESSTXT,C' '            Clear MESSTXT for next output
            MVC   MESSTXT+1(L'MESSTXT-1),MESSTXT
            BR    7                       Return ->
*
*           Conversion routine
*
PKD2ZND     CVD   8,ASSIST1               Convert register contents to
            UNPK  ASSIST2,ASSIST1         zoned decimal
            MVZ   ASSIST2+15(1),=X'F0'
            BR    7                       Return ->
AREA1       DS    10F
AREA2A      DS    12F
AREA2B      DS    12F
MESSAGE     DC    Y(ENDMESS-MESSAGE)      Record length
            DS    CL2                     Reserved
            DC    X'01'                   Print feed control character
MESSTXT     DC    CL60' '                 Text
ENDMESS     EQU   *
```

```
  STARTMSG DC    CL60'MES1 successfully started'
  ASSIST1  DS    D
  ASSIST2  DS    CL16
  MCPU     DC    C'CPU Time used:'
  MELA     DC    C'Elapsed Time used:'
  MIOPI    DC    C'IOs before BKPT:'
  MIOPE    DC    C'IOs after BKPT:'
           PRINT GEN
  MES1PAR  AINF  TIME=D,IOCNT=D,P=A
1          #INTF INTNAME=AINF,REFTYPE=REQUEST,INTCOMP=001          GS 950
1 MES1PAR  MFPRE DNAME=AIAREA,MF=D,PREFIX=A,MACID=AIA,DMACID=AIA   :*R200
2 MES1PAR  DSECT ,
2                *,##### PREFIX=A, MACID=AIA #####
1 AAIAREA  DS    OF                      START OF OUTPUT AREA      LKH075
1          MFPRE DNAME=AITIME,ALIGN=F,PREFIX=A,MACID=AIT,MF=S,     :*R200C
1                DMACID=AIT                                        :*R200
2          CNOP  0,4
2 AAITIME  DS    OF
1 *                                    "TIME"   INFO PACKAGE     LKH075
1 AAITTCPU DS    OFL8                  CPU TIME                    LKH075
1 AAITCPUS DS    F                     CPU TIME SECONDS            LKH075
1 AAITCPUN DS    F                     CPU TIME NANOSECONDS        LKH075
1 AAITETIM DS    OFL8                  ELAPSED TIME / TIME STAMP LKH075
1 AAITTIMS DS    F                     TIME IN SECONDS             LKH075
1 AAITTIMN DS    F                     TIME NANOSECONDS            LKH075
1 AAITIMEE EQU   *                     END OF "TIME"   INFO        LKH075
1 LAAITIME EQU   *-AAITIME            LENGTH OF "TIME"   INFO   LKH075
1          SPACE 2                                                 LKH075

1          MFPRE DNAME=AIIOCN,ALIGN=F,PREFIX=A,MACID=AII,MF=S :*R200DMACIC
1                D=AII                                           :*R200
2          CNOP  0,4
2 AAIIOCN  DS    OF
1 *                                    "IOCNT"  INFO PACKAGE     LKH075
1 AAII#IOS DS    F                     TOTAL # IO'S                LKH075
1 AAIIIOPD DS    F               # IO'S ON PUBLIC DEVICES         LKH075
1 AAIIIOSD DS    F               # IO'S ON SYSTEM PRIV. DISKS     LKH075
1 AAIIIOUD DS    F               # IO'S ON USER PRIVATE DISKS     LKH075
1 AAIIIOTP DS    F               # IO'S ON TAPE DEVICES           LKH075
1 AAIIIOUR DS    F               # IO'S ON UNIT RECORD DEVICES    LKH075
1 AAIIOCNE EQU   *                     END OF "IOCNT"  INFO      LKH075
1 LAAIIOCN EQU   *-AAIIOCN            LENGTH OF "IOCNT"  INFO   LKH075
1          SPACE 2                                                 LKH075
1 *                                    END OF OUTPUT AREA        LKH075
1          MFPRE DNAME=AIAEND,ALIGN=F,PREFIX=A,MACID=AIA,MF=S,     :*R200C
1                DMACID=AIA                                        :*R200
2          CNOP  0,4
2 AAIAEND  DS    OF
```

```
1 LAAIAREA EQU   *-AAIAREA              LENGTH OF OUTPUT AREA     LKH075
1          SPACE 2                                                LKH075
  MES2PAR  AINF  GLOBAL=D,IOCNT=D,P=B
1          #INTF INTNAME=AINF,REFTYPE=REQUEST,INTCOMP=001         GS 950
1 MES2PAR  MFPRE DNAME=AIAREA,MF=D,PREFIX=B,MACID=AIA,DMACID=AIA  :*R200
2 MES2PAR  DSECT ,
2                *,##### PREFIX=B, MACID=AIA #####
1 BAIAREA  DS    OF                     START OF OUTPUT AREA      LKH075
1          MFPRE DNAME=AIGLOB,ALIGN=F,PREFIX=B,MACID=AIG,MF=S,     :*R200C
1                DMACID=AIG                                       :*R200
2          CNOP  0,4
2 BAIGLOB  DS    OF
1 *                                     "GLOBAL" INFO PACKAGE     LKH075
1 BAIGTCPU DS    OFL8                   CPU TIME                  LKH075
1 BAIGCPUS DS    F                      CPU TIME SECONDS          LKH075
1 BAIGCPUN DS    F                      CPU TIME NANOSECONDS      LKH075
1 BAIG#IOS DS    F                      TOTAL # IO'S              LKH075
1 BAIG#BLK DS    F                      TOTAL # BLOCKS            GS 090
1 BAIGWSI  DS    FL8                    WORKING SET INTEGRAL      GS 090
1 BAIGLOBE EQU   *                      END OF "GLOBAL" INFO      LKH075
1 LBAIGLOB EQU   *-BAIGLOB             LENGTH OF "GLOBAL" INFO   LKH075
1          SPACE 2                                                LKH075
1          MFPRE DNAME=AIIOCN,ALIGN=F,PREFIX=B,MACID=AII,MF=S :*R200DMACIC
1                D=AII                                           :*R200
2          CNOP  0,4
2 BAIIOCN  DS    OF
1 *                                     "IOCNT"  INFO PACKAGE     LKH075
1 BAII#IOS DS    F                      TOTAL # IO'S              LKH075
1 BAIIIOPD DS    F                # IO'S ON PUBLIC DEVICES         LKH075
1 BAIIIOSD DS    F                # IO'S ON SYSTEM PRIV. DISKS     LKH075

1 BAIIIOUD DS    F                # IO'S ON USER PRIVATE DISKS     LKH075
1 BAIIIOTP DS    F                # IO'S ON TAPE DEVICES           LKH075
1 BAIIIOUR DS    F                # IO'S ON UNIT RECORD DEVICES    LKH075
1 BAIIOCNE EQU   *                       END OF "IOCNT"  INFO      LKH075
1 LBAIIOCN EQU   *-BAIIOCN              LENGTH OF "IOCNT"  INFO   LKH075
1          SPACE 2                                                LKH075
1 *                                      END OF OUTPUT AREA        LKH075
1          MFPRE DNAME=AIAEND,ALIGN=F,PREFIX=B,MACID=AIA,MF=S,     :*R200C
1                DMACID=AIA                                       :*R200
2          CNOP  0,4
2 BAIAEND  DS    OF
1 LBAIAREA EQU   *-BAIAREA              LENGTH OF OUTPUT AREA     LKH075
1          SPACE 2                                                LKH075
           END
                 =X'F0'
```

Ablaufprotokoll:

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,ainf2), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,ainf2))
%  ASS6011 ASSEMBLY TIME: 407 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 106 MSEC
//end
%  ASS6012 END OF ASSEMBH
/start-executable-program library=macexmp.lib,element-or-symbol=ainf2
%  BLS0523 ELEMENT 'AINF2', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'AINF2', VERSION ' ' OF '<date> <time>' LOADED
MES1 successfully started ──────────────────────────────────────────  (1)
MES2 successfully started ──────────────────────────────────────────  (2)
%  IDA0199 PROGRAM BREAK AT ADDRESS X'0000B8', AMODE=24 ────────────  (3)
/copy-file from-file=oldexp,to-file=newexp ──────────────────────────  (4)
/resume-program
Elapsed Time used:  0000000000.247344000 ──────────────────────────  (5)
IOs before BKPT:              0000000000
IOs after BKPT:               0000000011
```

(1) MES1 wurde gestartet; der Makro wurde ohne Fehler ausgeführt.

(2) MES2 wurde gestartet; der Makro wurde ohne Fehler ausgeführt.

(3) Der Programmablauf wurde mit **BKPT** unterbrochen.

Zuvor wurde MES2 unterbrochen, um die seit Start der Messung verbrauchte CPU-Zeit, Laufzeit und Anzahl der Ein-/Ausgaben im Bereich AREA2A abzuspeichern. Anschließend wurde MES2 wieder fortgesetzt.

(4) Zur Demonstration wird ein COPY-FILE-Kommando gegeben (Ein-/Ausgabe). Anschließend wird der Programmablauf fortgesetzt. MES2 und MES1 werden beendet. Der Bereich AREA2B enthält die seit Start der Messung MES2 verbrauchten „Globale Werte“ und die Anzahl der Ein-/Ausgaben. Im Bereich AREA1 werden die seit Start der Messung MES1 aufgelaufenen Werte von „Zeitverbrauch“ und „Ein-/Ausgabe-Zähler“ abgespeichert.

(5) Zur Demonstration werden einige der ermittelten Messwerte ausgegeben:

- Laufzeit in Sekunden (durch MES1 ermittelt)
- Anzahl der Ein-/Ausgaben zum Zeitpunkt der Unterbrechung der Messung MES2. Die durch das Kommando COPY-FILE verursachten Ein-/Ausgaben sind dabei noch nicht berücksichtigt.
- Anzahl der Ein-/Ausgaben am Ende der Messung MES2, d.h. inklusive der durch das Kommando COPY-FILE verursachten Ein-/ Ausgaben.

ALESRV – Task mit Datenraum verbinden oder lösen

Allgemeines

Anwendungsgebiet: Erweiterung durch Datenräume; siehe Seite 61
Makrotyp: S-Typ, MF-Format **3**: C-/D-/L-/M-/R-/E-Form; siehe Seite 29

Der Makro **ALESRV** kann auf allen BS2000-Servern verwendet werden (siehe Abschnitt „Erweiterung durch Datenräume“ auf Seite 61).

Makrobeschreibung

Der Makro **ALESRV** realisiert den Anschluss eines im AR-Modus laufenden Programms an einen mit dem Makro **DSPSRV** eingerichteten Datenraum. Das System stellt die Verbindung zu dem spezifizierten Datenraum her, indem es einen freien Eintrag (ALE) in der taskeigenen Zugriffsliste sucht, ihn belegt und den ALET zur Adressierung zurückliefert. Das System überprüft beim Anlegen eines ALE die Zugriffsberechtigung des Programms für diesen Datenraum. Hat das Programm einmal einen gültigen ALET in das Zugriffsregister geladen, erfolgt beim Zugriff keine Kontrolle mehr.

Mit dem Makro **ALESRV** kann ein Datenraum auch wieder freigegeben werden. Den Eintrag (ALE), der die Verbindung zwischen Programm und Datenraum realisiert, markiert das System in der taskeigenen Zugriffsliste als ungültig. Ein solcher markierter Eintrag steht für einen erneuten Anschluss (mit FCT=CONNECT) zur Verfügung. Wird bei der Adressumsetzung auf einen als ungültig markierten Eintrag zugegriffen, kommt es zu einer Unterbrechung (siehe Makro **STXIT**).

Mit dem Makro **ALESRV** kann sich der Anwender durch Angabe des ALET die zum Datenraum gehörende, sessionweit eindeutige Identifikation (SPID) ausgeben lassen.

Die Funktionen des **ALESRV**-Makros ermöglichen:
– eine Verbindung zwischen Programm und Datenraum herzustellen (FCT=CONNECT),
– diese Verbindung wieder zu lösen (FCT=DISCONN) und
– sich die SPID eines Datenraums ausgeben zu lassen (FCT=IDENTIFY).

Makroaufrufformat und Operandenbeschreibung

```
ALESRV

 FCT = { CONNECT, SPID=spid_adr  }
       { DISCONN, ALET=alet_adr  }
       { IDENTIFY, ALET=alet_adr }

,MF = C / D / L / M / R / E
[,PARAM = adr / (r)]
,PREFIX=N / p
,MACID = VDA / macid
```

FCT=
bestimmt die auszuführende Funktion des Makros **ALESRV**.

CONNECT
realisiert die Verbindung eines im AR-Modus laufenden Programms mit einem bestehenden Datenraum. Dabei wird ein freier Eintrag (ALE) in der Zugriffsliste belegt und der ALET ausgegeben.
Der Operand SPID muss angegeben werden.

DISCONN
löst die Verbindung des im AR-Modus laufenden Programms mit dem Datenraum. Dabei wird ein belegter Eintrag (ALE) in der Zugriffsliste freigegeben, d.h. als ungültig markiert.
Der Operand ALET muss angegeben werden.

IDENTIFY
identifiziert einen Datenraum anhand des Zeigers (ALET) auf einen Eintrag (ALE) in der Zugriffsliste. Es wird die SPID des Datenraums ausgegeben.
Der Operand ALET muss angegeben werden.

ALET=
ist der Inhalt eines Zugriffsregisters und Zeiger auf einen Eintrag in der Zugriffsliste. Dieser Eintrag verbindet das Programm mit dem Datenraum.
Dieser Operand kann sowohl Ein- als auch Ausgabeoperand sein.

alet_adr
symbolische Adresse (Name) eines Feldes (4 Byte), das den ALET eines bestimmten Datenraums enthält.

SPID=
kennzeichnet einen Datenraum eindeutig im Gesamtsystem.
Sie wird beim Anlegen eines Datenraumes vom System vergeben.
Der Operand kann sowohl Ein- als auch Ausgabeoperand sein.

spid_adr
symbolische Adresse (Name) eines Feldes (8 Byte), das die SPID des Datenraumes enthält.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Bei der C-Form, D-Form, R-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form, R-Form oder M-Form zusätzlich eine Macid MACID angegeben werden (siehe siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

Hinweise zum Makroaufruf

– Der Operand ALET kann sowohl Ein- als auch Ausgabeoperand sein, d.h. dass er durch FCT=CONNECT in dem erzeugten Datenbereich als Ausgabeoperand vorliegt und bei nochmaliger Verwendung dieses Datenbereichs auch als Eingabeoperand gültig ist.

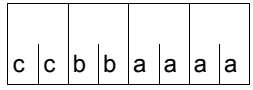
– Mehrfache Verbindungen zu ein und demselben Datenraum werden durch verschiedene ALETs realisiert. Jeder dieser ALETs muss durch einen entsprechenden FCT=DISCONNECT-Aufruf gelöscht werden, da das System bei der Freigabe eines Datenraums nicht auch automatisch alle korrespondierenden ALETs löscht.

– Die ALET-Wert-Vergabe erfolgt deterministisch, d.h. ist bei zwei Programmen innerhalb eines BS2000-Systemlaufs die Abfolge der erfolgreichen **ALESRV**-Aufrufe mit FCT=CONNECT und FCT=DISCONN identisch und jeweils gleich parametrisiert (bei CONNECT heißt dies: jeweils SPID desselben Data Space angegeben; bei DISCONN: jeweils gleicher ALET), so erhalten beide Programme denselben ALET-Output-Wert beim n-ten ALESRV-CONNECT.
Damit ist folgendes gemeint:
Verhalten sich zwei TU-Programme bzgl. Konnektierung an Data Spaces gleich, so bekommen sie auch dieselben ALETs zugeteilt.

– Die Freigabe eines Datenraums erfolgt mit dem Makro **DSPSRV** FCT=DESTROY oder bei Beendigung des Programms, das den Datenraum erzeugt hat.

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros ALESRV wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Funktion erfolgreich ausgeführt. Mit MF=R kann aus dem Datenbereich folgender Wert gelesen werden: – bei FCT=CONNECT: alet – bei FCT=IDENTIFY: spid
X'02'	X'00'	X'0001'	Warnung: Der Eintrag in der Zugriffsliste wurde gelöscht, aber der dazugehörende Datenraum war bereits freigegeben (bei FCT=DISCONN).
X'00'	X'01'	X'0003'	Fehler im Datenbereich.
X'01'	X'01'	X'0004'	Es fehlt der Operand SPID (bei FCT=CONNECT).
X'02'	X'01'	X'0004'	Es fehlt der Operand ALET (bei FCT=DISCONN oder FCT=IDENTIFY).
	X'20'	X'0005'	Interner Fehler.
X'00'	X'40'	X'0304'	Die Angabe des SPID-Operanden ist fehlerhaft: Der angegebene Datenraum existiert nicht oder der Aufrufer hat keine Berechtigung, auf diesen Datenraum zuzugreifen (bei FCT=CONNECT).
X'01'	X'40'	X'0304'	Die Angabe des SPID-Operanden ist fehlerhaft: Der angegebene Datenraum gehört zu einer anderen TU-Domain (bei FCT=CONNECT).
X'00'	X'40'	X'0404'	Die Angabe des ALET-Operanden ist fehlerhaft: Der angegebene ALET ist ungültig oder privilegiert (bei FCT=DISCONN oder FCT=IDENTIFY).
X'01'	X'40'	X'0404'	Die Angabe des ALET-Operanden ist fehlerhaft: Der angegebene ALET verweist auf einen Datenraum in einer anderen TU-Domain (bei FCT=DISCONN oder FCT=IDENTIFY).
X'00'	X'40'	X'0406'	Die Zugriffsliste ist voll belegt, d.h. es sind keine freien Einträge vorhanden, um eine neue Verbindung Programm zu Datenraum zu schaffen (bei FCT=CONNECT).
X'00'	X'40'	X'0604'	Der ALET verweist auf einen bereits gelöschten Datenraum (bei FCT=IDENTIFY).

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Beispiel siehe im Abschnitt „Erweiterung durch Datenräume“ auf Seite 68.

ALINF – Informationen über Zugriffslisten anfordern

Allgemeines

Anwendungsgebiet: Erweiterung durch Datenräume; siehe Seite 61
Makrotyp: S-Typ, MF-Format **3**: C-/D-/L-/M-/E-Form; siehe Seite 29

Der Makro **ALINF** kann auf allen BS2000-Servern verwendet werden (siehe Abschnitt „Erweiterung durch Datenräume“ auf Seite 61).

Makrobeschreibung

Der Makro **ALINF** informiert den Aufrufer darüber, welche ALETs (Access List Entry Tokens) in seiner taskeigenen Zugriffsliste mit einem bestimmten Datenraum korrespondieren. Dabei wird ein Datenraum eindeutig durch seine SPID identifiziert. Existieren ein oder mehrere ALETs für einen Datenraum, wird der erste ALET von ihnen im Feld `<PREFIX><MACID>ALET` zurückgegeben (mit RETURN=FIRST) bzw. bei weiteren Aufrufen des Makros der jeweils folgende ALET (mit RETURN=NEXT). Durch die Angabe des FROM-Operanden kann der Suchbeginn unabhängig von bereits gefundenen ALETs beliebig festgelegt werden.

Makroaufrufformat und Operandenbeschreibung

ALINF
SPID=spid_adr ,RETURN=FIRST / NEXT[,FROM=alet_adr / (r)] ,MF=C / D / L / M / E [,PARAM=adr / (r)] ,PREFIX=N / p ,MACID=VDI / macid

SPID=spid_adr
kennzeichnet einen Datenraum eindeutig im Gesamtsystem. Sie wird beim Anlegen eines Datenraumes vom System vergeben.
spid_adr: Symbolische Adresse (Name) eines Feldes (8 Byte), das die SPID des Datenraumes enthält.

RETURN=
legt fest, welcher der korrespondierenden ALETs im Feld <PREFIX><MACID>ALET zurückgegeben wird.

FIRST
Der erste gefundene ALET, der auf den mit SPID identifizierten Datenraum zeigt, wird im Ausgabefeld <PREFIX><MACID>ALET zurückgegeben.

NEXT
Der nächste gefundene ALET wird zurückgegeben. Ausgangspunkt der Suche ist der ALET, der im Ausgabefeld <PREFIX><MACID>ALET steht. Dieser zuletzt gefundene ALET (≙ Vorgänger) wird entweder durch einen **ALINF**-Aufruf mit RETURN=FIRST bestimmt oder durch direkte Angabe im FROM-Operanden. Ist das ALET-Ausgabefeld nicht durch einen früheren Aufruf mit einem Vorgänger belegt und auch der Operand FROM wird nicht angegeben, so beginnt die Suche nach dem nächsten korrespondierenden ALET an unbestimmter Stelle (an einem beliebigen Eintrag in der Zugriffsliste).

FROM=
gibt den ALET an, ab dem nach dem nächsten korrespondierenden ALET gesucht werden soll. Der Wert wird in das Ausgabefeld <PREFIX><MACID>ALET geschrieben und wird vom Makro **ALINF** als Eingabeoperand (≙Vorgänger) bei RETURN=NEXT gelesen. Angabe nur bei MF=M erlaubt.

alet_adr
symbolische Adresse des ALETs, der Vorgänger sein soll

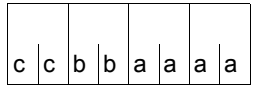
(r)
r = Register mit dem Adresswert von adr

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros ALINF wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Funktion erfolgreich ausgeführt. Aus dem Feld <pre> VDIALET des Datenbereichs kann der gefundene ALET ausgelesen werden.
X'00'	X'00'	X'0001'	Es wurde kein korrespondierender ALET gefunden.
X'00'	X'01'	X'0003'	Fehler im Datenbereich.
X'07'	X'01'	X'0003'	Ungültiger Operand RETURN.
X'08'	X'01'	X'0003'	Ungültiger Operand FROM: Der Operandenwert hat kein ALET-Format.
X'01'	X'01'	X'0004'	Es fehlt der Operand SPID.
	X'20'	X'0009'	Interner Fehler.
X'01'	X'40'	X'0005'	Die Angabe des SPID-Operanden ist fehlerhaft: – die angegebene Spid korrespondiert mit keinem Datenraum oder – der korrespondierende Datenraum ist privilegiert oder – der Datenraum liegt in einem fremden Geltungsbereich.
X'02'	X'40'	X'0005'	Die Angabe des SPID-Operanden ist fehlerhaft: Der korrespondierende Datenraum wurde von einer fremden Domain erzeugt.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Beispiel siehe Abschnitt „Erweiterung durch Datenräume“ auf Seite 68.

AMODE31 – Adressierungsmodus abfragen

Allgemeines

Anwendungsgebiet: XS-Programmierung; siehe Seite 168
Makrotyp: O-Typ; siehe Seite 28

Auf den /390-Servern kann ein 24-Bit-Adressierungsmodus oder ein 31-Bit-Adressierungsmodus eingeschaltet sein.
Oberhalb der 16MB-Grenze muss ein Programm im 31-Bit-Adressierungsmodus ablaufen.

Makrobeschreibung

Der Makro **AMODE31** informiert den Anwender über den eingestellten Adressierungsmodus. Die Information wird durch Setzen von Bit 2^{31} und 2^{0} im angegebenen Register übergeben. Die Bits 2^{30} bis 2^{1} werden überschrieben. Es bedeuten:

Bit 2^{31}	Bit 2^{0}	Adressierungsmodus
0	0	24-Bit-Adressierungsmodus (NXS)
1	0	31-Bit-Adressierungsmodus (XS)
1	1	32-Bit-Adressierungsmodus (x86-Server)

Ein Returncode über die Makroausführung wird nicht übergeben.

Makroaufrufformat und Operandenbeschreibung

AMODE31
reg

reg
Register, in das die Information eingetragen wird

ARDS – Accounting-Abrechnungssätze generieren

Allgemeines

Anwendungsgebiet: Accounting (Systemverwaltermakro); siehe Seite 166 und 170
Makrotyp: Definitionsmakro; siehe Seite 28

Makrobeschreibung

Der Makro **ARDS** generiert eine Dummy Section (DSECT) zur Beschreibung der Accounting-Abrechnungssätze, die vom Abrechnungssystem von BS2000 erstellt und in die Accounting-Datei geschrieben werden. Die Abrechnungssätze werden durch Angabe ihrer Satzkennung spezifiziert (UDAT, UACC, DSPC, ...) und sind im Handbuch „Abrechnungssätze“ [13] beschrieben.

Will der Anwender die Struktur eines Abrechnungssatzes erhalten, der von einem entkoppelten Subsystem erstellt wurde, kann er die Struktur dieses Satzes abhängig von der gerade laufenden Version des Subsystems anfordern.

Der Anwender kann angeben, ob die Struktur aller Accounting-Abrechnungssätze oder einer Auswahl davon oder nur eines Abrechnungssatzes beschrieben werden soll.

Makroaufrufformat und Operandenbeschreibung

[name] ARDS
ALL / id / (id,id,...) ,DRV=OLD / NEW ,DSSM=OLD / NEW ,FT=OLD / NEW ,SPOOL=OLD / NEW ,UTM=OLD / NEW ,VM=OLD / NEW ,HSMS=OLD / NEW

name
Name der generierten DSECT; Voreinstellung: name = ARDSECT.

ALL
Alle Accounting-Abrechnungssätze werden beschrieben.

id
Satzkennung des Abrechnungssatzes, der beschrieben werden soll

(id,id,...)
Liste mit Satzkennungen für die gewünschten Abrechnungssätze.
Die Liste muss in Klammern angegeben werden.

DRV=
bestimmt die versionsabhängige Struktur eines Abrechnungssatzes, der durch das DRV-Subsystem (Dual Recording by Volume) erstellt wurde.

OLD
Die Struktur wird durch die Version des Subsystems bestimmt, die zum Zeitpunkt der Übergabe der Abrechnungssätze dem Abrechnungssystem bekannt war.
Wenn seit der Abrechnungsübergabe keine andere Version des Subsystems übergeben und geladen wurde, entspricht diese Struktur der aktuellen.

NEW
Die Struktur wird durch die gerade geladene Version des Subsystems bestimmt und entspricht so immer dem aktuellen Stand.
Wurde nach der Übergabe der Abrechnungssätze eine andere Version des Subsystems übergeben und geladen, entspricht die Struktur der Abrechnungssätze der entladenen („alten“) Version und nicht der zum Zeitpunkt des Aufrufs von **ARDS** geladenen („neuen“) Version des Subsystems.
Deshalb muss bei Angabe von NEW durch den Aufrufer gewährleistet werden, dass der Assembler die aktuell gültige Makrobibliothek des Subsystems (.GCLIB.UR) kennt (Kommando SET-TASKLIB; siehe Handbuch „Kommandos“ [19] oder Anweisung COMPILE MACRO-LIBRARY=... im Übersetzungsprogramm; siehe Handbuch „ASSEMBH“ [2]).

DSSM=
bestimmt die versionsabhängige Struktur eines Abrechnungssatzes, der durch das DSSM-Subsystem (Dynamic Subsystem Management) erstellt wurde.

OLD
Operandenbeschreibung wie bei DRV-Operand (siehe dort).

NEW
Operandenbeschreibung wie bei DRV-Operand (siehe dort).

FT=
bestimmt die versionsabhängige Struktur eines Abrechnungssatzes, der durch das FT-Subsystem (File Transfer) erstellt wurde.

OLD
Operandenbeschreibung wie bei DRV-Operand (siehe dort).

NEW
Operandenbeschreibung wie bei DRV-Operand (siehe dort).

HSMS=
bestimmt die versionsabhängige Struktur eines Abrechnungssatzes, der durch das HSMS-Subsystem (Hierarchical Storage Management System) erstellt wurde.

OLD
Operandenbeschreibung wie bei DRV-Operand (siehe dort).

NEW
Operandenbeschreibung wie bei DRV-Operand (siehe dort).

SPOOL=
bestimmt die versionsabhängige Struktur eines Abrechnungssatzes, der durch das SPOOL-Subsystem (Simultaneous Peripheral Operation Online) erstellt wurde.

OLD
Operandenbeschreibung wie bei DRV-Operand (siehe dort).

NEW
Operandenbeschreibung wie bei DRV-Operand (siehe dort).

UTM=
bestimmt die versionsabhängige Struktur eines Abrechnungssatzes, der durch das UTM-Subsystem (Universal Transaction Monitor) erstellt wurde.

OLD
Operandenbeschreibung wie bei DRV-Operand (siehe dort).

NEW
Operandenbeschreibung wie bei DRV-Operand (siehe dort).

VM=
bestimmt die versionsabhängige Struktur eines Abrechnungssatzes, der durch das VM-Subsystem (Virtual Memory) erstellt wurde.

OLD
Operandenbeschreibung wie bei DRV-Operand (siehe dort).

NEW
Operandenbeschreibung wie bei DRV-Operand (siehe dort).

Hinweise zum Makroaufruf

- FT=NEW: Diese Angabe ist nur für ein FT-Subsystem ≥ V5.0 erlaubt.
- UTM=NEW: Diese Angabe ist nur für ein UTM-Subsystem ≥ V3.3 erlaubt.
- HSMS=NEW: Der Wert NEW muss angegeben werden, um die DSECT für HSMS zu erhalten.
- Die Struktur eines Abrechnungssatzes kann nur einmal pro Assemblermodul erzeugt werden, andernfalls kommt es zum Namenskonflikt.

AREC – Benutzer-Abrechnungssatz schreiben

Allgemeines

Anwendungsgebiet: Accounting; siehe Seite 166
Makrotyp: S-Typ, MF-Format **1**:
31-Bit-Schnittstelle: Standardform/L-/D-/E-Form; siehe Seite 30

Das Accounting-System ermittelt Abrechnungsdaten über die Nutzung der Betriebsmittel des Data Centers und schreibt diese Daten in Form von Abrechnungssätzen in die Accounting-Datei. Abrechnungsdaten sind z.B.:

– beanspruchte CPU-Zeit, E/A-Datenmenge, Working-Set-Integral, Geräte- und Datenträgerbelegungsdaten.
– beanspruchte Systemdienste (Benutzerdumps, etc.).
– Speicherplatzbelegung auf Public Volume Sets.

Die Auswertung der Accounting-Datei erfolgt mit einem speziellen Auswertungsprogramm. Grobstruktur eines Abrechnungssatzes:

`Satzbeschreibung`	Satzkennung, Zeitstempel, ...
`Benutzerbeschreibung`	Benutzerkennung, Abrechnungsnr., abgerechnete Benutzertask, ...
`Grundinformation`	Standarddaten
`variabler Satzteil`	Satzerweiterungen

Eine ausführliche Beschreibung ist im Handbuch „Abrechnungssätze“ [13] gegeben.

Makrobeschreibung

Der Makro **AREC** veranlasst das Schreiben eines Benutzer-Abrechnungssatzes in die Accounting-Datei.

Der Abrechnungssatz kann sein:

– ein UDAT-Abrechnungssatz mit einer Satzerweiterung
– ein UACC-Abrechnungssatz mit einer Satzkennung
– ein vom Anwender (frei) definierter Abrechnungssatz

Der Anwender kann sich die Grundstruktur eines Abrechnungssatzes generieren lassen. Es wird empfohlen, sich bei der Definition eigener Abrechnungssätze an diese Struktur anzulehnen.

Hinweise

– Zur Auswertung der Abrechnungssätze muss der Anwender entsprechende Programme verwenden (siehe Handbuch „Systembetreuung“ [10]).

– Der Systemverwalter kann die Anzahl der Benutzer-Abrechnungssätze je Task benutzerspezifisch begrenzen (Operand MAX-ACCOUNT-RECORDS im Benutzerkatalog). Die Begrenzung gilt für den gesamten Kommandomodus (außerhalb von Programmläufen) einer Task. Voreinstellung: MAX-ACCOUNT-RECORDS = 100; d.h. maximal 100 Abrechnungssätze im Kommandomodus einer Task.

– Das Schreiben eines frei definierten Abrechnungssatzes in die Accountingdatei erfordert die Berechtigung MAX-ACCOUNT-RECORDS=NL (No Limit).

Makroaufrufformat und Operandenbeschreibung

```
AREC

{ DSECT=RECORD      }
{ ,DATA=adr / (r)   }
{ ,ID=adr / (r)     }
{ ,RECORD=adr / (r) }
,MF=S / L / (E,...) / D
,P=I / p
```

DSECT=RECORD
eine Dummy Section (DSECT) wird generiert, die die Grundstruktur eines Abrechnungssatzes wiedergibt.
Es kann ein Präfix P (P = 1 Buchstabe) angegeben werden.
Der Operand MF darf nicht angegeben werden.

DATA=
beschreibt die Adresse der Satzerweiterung (Datenstring), die in den UDAT-Abrechnungssatz eingetragen wird. In Byte 0-1 des Datenstrings ist die Länge des nachfolgenden Textes anzugeben.
Textlänge ≤ 255 Byte.

adr
symbolische Adresse (Name) des Feldes mit dem Datenstring

(r)
r = Register mit dem Wert der Adresse adr

ID=
beschreibt die Adresse einer Zeichenfolge (Datenstring), die als Satzkennung in den UACC-Abrechnungssatz eingetragen wird.
Länge der Zeichenfolge ≤ 8 Byte.

adr
symbolische Adresse (Name) des Feldes mit dem Datenstring

(r)
r = Register mit dem Wert der Adresse adr

RECORD=
beschreibt die Adresse des vom Anwender (frei) definierten Abrechnungssatzes.
In dem Abrechnungssatz müssen auch die Satzkennung und die Benutzerbeschreibung selbst versorgt werden.
Satzlänge ≤ 496 Byte.

Der Operand ist nur für Anwender zugelassen, in deren Benutzerkatalog MAX-ACCOUNT-RECORDS=NL eingetragen ist.

adr
symbolische Adresse (Name) des Feldes mit dem Abrechnungssatz

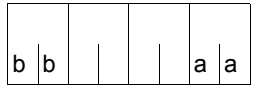
(r)
r = Register mit dem Wert der Adresse adr

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei MF=D kann ein Präfix P (P = 1 Buchstabe), wie im Aufrufformat dargestellt, angegeben werden.

Rückinformation und Fehlermeldung

R15:



Über die Ausführung des Makros AREC wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'bb'	**X'aa'**	**Erläuterung**
X'00'	X'00'	Funktion normal ausgeführt.
X'04'	X'00'	Satz wurde nicht geschrieben, weil Accounting ausgeschaltet war.
X'08'	X'00'	Satz wurde nicht geschrieben, weil der betreffende Satztyp ausgeschaltet war.
X'00'	X'04'	Adressfehler (Datenbereich).
X'00'	X'08'	Ungültiger Funktionscode (für SVC 99 oder für AREC).
X'00'	X'0C'	Funktion für diesen Benutzer nicht erlaubt.
X'00'	X'10'	Adressfehler (DATA-, ID- oder RECORD-Operand).
X'00'	X'14'	Ungültige Satzkennung.
X'00'	X'18'	Max. Satzlänge überschritten.
X'00'	X'1C'	Anzahl erlaubter Sätze überschritten.
X'00'	X'20'	Ungültiges Register.
X'00'	X'24'	System- oder Ressourcefehler. Mehr Information im sekundären Returncode.
X'04'	X'24'	Kein Speicher verfügbar.
X'08'	X'24'	Keine Jobinformation.
X'0C'	X'24'	CLTF-Fehler: Fehler beim Schreiben des Satzes.

Grundstruktur eines Abrechnungssatzes

```
          AREC  DSECT=RECORD
1         MFCHK DNAME=ARLDS,MF=D,PREFIX=I,MACID=ARL,DMACID=ARL
2 IARLDS  DSECT ,
2               *,##### PREFIX=I, MACID=ARL #####
1         DS    OF                      AREC PARAMETER LIST
1 ************************************************************
1 *             GENERAL LAYOUT OF ACCOUNT    RECORDS            *
1 ************************************************************
1 IARLRHDR DS   OF                      RECORD HEADER
1 IARLRL   DS   Y                       RECORD LENGTH
1          DS   XL2                     RESERVED
1 IARLID   DS   CL4                     RECORD ID
1          DS   FL8                     RESERVED (TIME STAMP)
1 IARLLUS  DS   Y                       LENGTH OF USER HEADER
1 IARLLBI  DS   Y                       LENGTH OF BASIC INFORMATION
1          DS   XL4                     RESERVED
1 *
1 *
1 IARLUSER DS   OF                      USER HEADER
1 IARLUSID DS   CL8                     USER IDENTIFICATION
1 IARLACNT DS   CL8                     ACCOUNT NUMBER
1 IARLTSN  DS   CL4                     TSN
1 *
1 *
1          ORG  IARLID+X'1000'
1 IARLBI   DS   OF                      BASIC INFORMATION
1 *
1 *
1          ORG  IARLID+X'2000'
1 IARLEXT  DS   OH                      VARIABLE EXTENSION PART
1 IARLEXTH DS   OH                      EXTENSION HEADER
1 IARL#EXT DS   H                       NUMBER EXTENSIONS
1 IARLDEXT DS   OH                      START LIST OF DISTANCES
1 *
1          ORG  IARLID+X'3000'
1 IARLSEXT DS   OH                      STRING EXTENSION
1 IARLSEXI DS   CL2                     EXTENSION ID
1 IARLLSEX DS   Y                       LENGTH OF STRING
1 IARLSEXB DS   OC                      BEGIN OF STRING
1 *
1          ORG  IARLID+X'3000'
1 IARLAEXT DS   OH                      ARRAY EXTENSION
1 IARLAEXI DS   CL2                     EXTENSION ID
1 IARLAEX# DS   X                       NUMBER OF ELEMENTS
1 IARLAEXL DS   X                       LENGTH OF ONE ELEMENT
1 IARLAEXB DS   OH                      BEGIN OF FIRST ELEMENT
```

ASHARE – Shared Code des Benutzers in Common Memory Pools laden

Allgemeines

Anwendungsgebiet: Binden und Laden; siehe Seite 47
Makrotyp: S-Typ, MF-Format **2**:
Standardform/C-/D-/L-/E-/M-Form; siehe Seite 30

Zum dynamischen Bindelader DBL siehe auch Handbuch „BLSSERV“ [4].

Makrobeschreibung

Der Makroaufruf **ASHARE** bindet und lädt Shared Code des Benutzers, der aus einer Menge von Modulen bestehen kann, in einen Common Memory Pool (siehe Abschnitt „Gemeinsamer Speicherbereich für mehrere Anwender (Memory Pool)“ auf Seite 55). Auf ein solches gemeinsam benutzbares Programm kann jeder Benutzer, der an den Common Memory Pool angeschlossen ist, über den Programmnamen bzw. über alle anderen nicht maskierten CSECTs oder ENTRYs zugreifen. Der Zugriff ist über die Kommandos LOAD-EXECUTABLE-PROGRAM und START-EXECUTABLE-PROGRAM (bzw. LOAD-PROGRAM und START-PROGRAM) bzw. über die Makros **BIND** und **VSVI1** möglich.

Beim indirekten Binden kann ASHARE zum Laden von Server-Modulen in einen Common Memory Pool verwendet werden.

Makroaufrufformat und Operandenbeschreibung

ASHARE

[,{ CONTEXT=name / CONTXT@=adr / (r) }]

[,{ LIBNAM=datei / * / LIBNAM@=adr / (r) / LIBLINK=name / LIBLNK@=adr / (r) }]

,ALTLIB=*DBLOPT / **N**O / **Y**ES

,INTVERS=BLSP2 / SRV001

,MAP=*DBLOPT / NO / BOTH / (BOTH,nn) / nn / SYSOUT

[,MPID=adr / (r)]

[,MSGCTRL=*DBLOPT / INFORMATION / WARNING / ERROR / NONE]

,{ PGMVERS=*STD / version / PGMVER@=adr / (r) }

[,{ PROGRAM=name / PROG@=adr / (r) }]

[,{ REPFILE=datei / REPFIL@=adr / (r) }]

,RESMP=**N**O / **Y**ES

,RESSYS=**N**O / **Y**ES

[,START@=adr / (r)]

,{ SYMBOL=name / SYMBOL@=adr / (r) }

,SYMTYP=**A**NY / **C**SECT / CSEN / **M**ODULE / **E**NTRY

[,{ VERSION=version / VERS@=adr / (r) }]

,MF=S / C / D / E / L / M

[,PARAM=adr / (r)]

,PREFIX=P / p

,MACID=BAS / macid

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

ALTLIB=
Legt fest, ob alternative Bibliotheken nach dem mit SYMBOL oder SYMBOL@ vereinbarten Objekt durchsucht werden. Alternative Bibliotheken werden mit dem Dateikettungsnamen BLSLIBnn (00≤nn≤99) bzw. $BLSLBnn (für alternative Systembibliotheken) zugewiesen. Sie werden auch für die Autolink-Funktion des DBL benutzt.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt ALTLIB=NO.

NO
Alternative Bibliotheken werden nicht durchsucht.

YES
Alternative Bibliotheken werden durchsucht.

Die Operanden ALTLIB=YES und LIBNAM/LIBNAM@/LIBLINK/LIBLNK@ dürfen zusammen angegeben werden. Ist jedoch die mit den Operanden LIBNAM/LIBNAM@/LIBLINK/LIBLNK@ angegebene Hauptbibliothek nicht vorhanden, bricht der DBL die Verarbeitung ab. Die Angabe ALTLIB=YES dient nicht als Ersatz für die fehlende oder ungültige Hauptbibliothek.

CONTEXT=
Gibt den Namen des Kontexts an, in den das Programm geladen wird. Existiert dieser Kontext bereits in einem für die Benutzertask zugreifbaren Memory Pool, so muss sich dieser Name auf den bei MPID angegebenen Memory Pool beziehen. In einem Memory Pool können maximal 15 Kontexte definiert werden.

name
Kontextname. Das erste Zeichen des Kontextnamens muss ein „#“ sein. Der Name darf maximal 32 Zeichen lang sein. Der Standardname ist ein „#“ gefolgt von den ersten 31 Zeichen des Namens des Memory Pools, der mit dem Operanden MPID angegeben wird.

CONTXT@=
Gibt die Adresse eines Feldes an, das den Kontextnamen enthält. Angabe nur mit MF=M.

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.

(r)
r = Register mit der gesuchten Feldadresse.

INTVERS=
Der Operand legt die Version der Makro-Schnittstelle ASHARE fest.

BLSP2
Default. Entspricht der Makro-Version 2.

SRV001
Entspricht der Makro-Version 3. Diese Version wird ab BLSSERV V2.4A unterstützt.

LIBLINK=name
Explizite Angabe des Dateikettungsnamens der Hauptbibliothek.
Er darf maximal 8 Zeichen lang sein.

LIBLNK@=
Gibt die Adresse eines Feldes an, das den Dateikettungsnamen der Hauptbibliothek enthält. Angabe nur mit MF=M.

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.

(r)
r = Register mit der gesuchten Feldadresse.

LIBNAM=
Legt die Hauptbibliothek fest, in der das mit SYMBOL oder SYMBOL@ vereinbarte Objekt gesucht wird. Die Hauptbibliothek kann durch explizite Angabe des Dateinamens der Bibliothek oder durch einen Dateikettungsnamen festgelegt werden. Die EAM-Bindemodulddatei wird durch den Dateinamen „*“ festgelegt. Ein Dateikettungsname ist für die EAM-Bindemoduldatei nicht möglich. Fehlen die Operanden LIBNAM, LIBNAM@, LIBLINK und LIBLNK@, wird die Bibliothek mit dem Dateikettungsnamen BLSLIB durchsucht.

Die Hauptbibliothek wird *vor* den alternativen Bibliotheken durchsucht (siehe Seite 270). Sie wird auch für die Autolink-Funktion des DBL benutzt. Ist der Operand LIBLINK oder LIBLNK@ angegeben, wird LIBNAM bzw. LIBNAM@ ignoriert.

datei
Explizite Angabe des Dateinamens der Hauptbibliothek. Der Dateiname darf maximal 54 Zeichen lang sein.

*****
Vereinbart als Hauptbibliothek die EAM-Bindemoduldatei.

LIBNAM@=
Gibt die Adresse eines Feldes an, das den Namen der Hauptbibliothek enthält. Angabe nur mit MF=M.

adr Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.

(r) r = Register mit der gesuchten Feldadresse.

MAP=
Angabe nur mit INTVERS=SRVxxx und xxx ≥ 001
Legt fest, ob eine DBL-Liste ausgegeben wird oder nicht und gibt das Ausgabeziel für die DBL-Liste an.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt MAP=NO.

NO
Es wird keine DBL-Liste ausgegeben.

BOTH
Das Ausgabeziel ist die Systemdatei SYSOUT und die Systemdatei SYSLST00.

(BOTH,nn)
Das Ausgabeziel ist die Systemdatei SYSOUT und eine Systemdatei SYSLSTnn (00≤nn≤99).

nn
Das Ausgabeziel ist eine Systemdatei aus der Menge SYSLST00 bis SYSLST99, deren Nummer hier anzugeben ist.
Die Nummer muss 2-stellig angegeben werden (00 für 0 usw.).

SYSOUT
Das Ausgabeziel ist die Systemdatei SYSOUT.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. angegebenen Operanden PREFIX, MACID und PARAM siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX (ein Buchstabe) und bei der C-Form oder M-Form zusätzlich eine Macid MACID (drei Buchstaben) angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

MPID=adr
Symbolische Adresse eines 4 Byte langen Feldes, das die Kurzkennung eines Memory Pools enthält, in den die Ladeeinheit geladen wird. Diese Kurzkennung wird dem Benutzer durch den Makroaufruf **ENAMP** zur Verfügung gestellt, der vor dem Makroaufruf **ASHARE** durchgeführt werden muss.

(r)
r = Register mit dem Adresswert adr. Angabe nur mit MF=M.

MSGCTRL=
Legt die niedrigste Meldungsklasse fest, ab der Meldungen des DBL ausgegeben werden. Als Default-Wert wird der Wert übernommen, der beim Ladeaufruf mit LOAD- oder START-EXECUTABLE-PROGRAM (bzw. LOAD- oder START-PROGRAM) angegeben wurde.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt MSGCTRL=INFORMATION.

INFORMATION
Die Meldungen aller Meldungsklassen werden ausgegeben.

WARNING
Nur Meldungen der Meldungsklasse WARNING und ERROR werden ausgegeben.

ERROR
Nur Meldungen der Meldungsklasse ERROR werden ausgegeben.

NONE
Es werden keine DBL-Meldungen ausgegeben.

PGMVERS=
Gibt die Programmversion an.

***STD**
Die aus dem Ladeaufruf resultierende Ladeeinheit erhält als Programmversion die Version des geladenen Bibliothekselements. Wenn das im Ladeaufruf angegebene Symbol bereits geladen ist, wird die Programmversion gesucht, die mit dem Kommando SELECT-PROGRAM-VERSION festgelegt wurde. Falls noch keine Programmversion festgelegt ist, verwendet DBL das zuerst gefundene Symbol.

version
Die Versionsangabe darf maximal 24 Zeichen lang sein. Wenn diese Version des Programms bereits im Common Memory Pool existiert, dann wird der Ladevorgang abgebrochen und der entsprechende Returncode übergeben.

PGMVER@=
Gibt die Adresse eines Feldes an, das die Programmversion enthält. Angabe nur mit MF=M.

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.

(r)
r = Register mit der gesuchten Feldadresse.

PROGRAM=
Identifiziert das Programm.

name
Der angegebende Name muss eindeutig und darf nicht länger als 32 Zeichen sein. Standardwert ist der bei SYMBOL oder SYMBOL@ angegebene Name. Existiert dieses Programm bereits in einem Common Memory Pool, auf den Benutzer zugreifen können, dann wird der Ladevorgang abgebrochen.

PROG@=
Gibt die Adresse eines Feldes an, das den Programmnamen enthält. Angabe nur mit MF=M.

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.

(r)
r = Register mit der gesuchten Feldadresse.

REPFILE=
Gibt den Namen der REP-Datei an, die REP-Sätze im BS2000-üblichen Format enthält (siehe Handbuch „Systembetreuung“ [10]). Tritt während der REP-Verarbeitung ein Fehler auf, so gibt DBL eine Fehlermeldung auf SYSOUT aus, ignoriert den fehlerhaften REP und setzt die Verarbeitung fort.

datei
Der Dateiname darf maximal 54 Zeichen lang sein.

REPFIL@=
Gibt die Adresse eines Feldes an, das den Namen der REP-Datei enthält. Angabe nur mit MF=M.

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.

(r)
r = Register mit der gesuchten Feldadresse.

RESMP=
Gibt den Geltungsbereich für die Befriedigung von Externverweisen im Shared Code im Memory Pool an.

NO
Nur der beim Operanden CONTEXT angegebene Kontext wird zur Befriedigung von Externverweisen genutzt.

YES
Alle Kontexte, die sich auf den Memory Pool beziehen, werden zur Befriedigung von Externverweisen genutzt.

RESSYS=
Gibt an, ob zur Befriedigung von Externverweisen auch der Shared Code im Systemadressraum (Klasse-3/4/5-Speicher) durchsucht wird. Dieser Shared Code wird mit DSSM in Form von nichtprivilegierten Subsystemen geladen.

NO
Keine Befriedigung von Externverweisen durch Shared Code im Systemadressraum.

YES
Shared Code im Systemadressraum wird zur Befriedigung von Externverweisen genutzt, wenn nach dem Durchsuchen des Shared Code in Memory Pools noch unbefriedigte Externverweise vorhanden sind.

START@=adr
Symbolische Adresse eines 4 Byte langen Feldes, in das der DBL die Startadresse der Ladeeinheit im Memory Pool übergibt. Das Feld muss auf Wortgrenze ausgerichtet sein. Der Benutzer muss die Startadresse mit START@ explizit anfordern, sie wird standardmäßig *nicht* zurückgegeben.

(r)
r = Register mit dem Adresswert adr. Angabe nur mit MF=M.

SYMBOL=name
Explizite Angabe eines Objektnamens. Der DBL nutzt diesen Namen, um das erste Modul in der Ladeeinheit zu bestimmen, die in den Memory Pool geladen werden soll. Der Name kann sich auf folgende Objekte beziehen:

– Programmabschnitt (CSECT),
– Einsprungstelle (ENTRY),
– Bindemodul (OM) (Elementname),
– Bindelademodul (LLM) (Elementname).

Der Name darf maximal 32 Zeichen lang sein. Der Typ des Objekts wird mit dem Operanden SYMTYP festgelegt.

SYMBOL@=
Gibt die Adresse eines Feldes an, das den Namen des Objektes enthält. Angabe nur mit MF=M.

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.

(r)
r = Register mit der gesuchten Feldadresse.

SYMTYP=
Gibt den Typ des Objekts an, das mit dem Namen SYMBOL oder SYMBOL@ vereinbart wurde und legt die Suchreihenfolge für das Objekt fest. Als Typ des Objekts kann ein Symbol (CSECT oder ENTRY) oder ein Modul (OM oder LLM) bestimmt werden.
Bei einem Symbol bezeichnet der Name SYMBOL oder SYMBOL@ einen Symbolnamen und kann sein:

– der Name eines nicht maskierten CSECT- oder ENTRY-Eintrags in einer Programmbibliothek (Typ R oder Typ L),
– der Name eines nicht maskierten CSECT- oder ENTRY-Eintrags in einer Bindemodulbibliothek (OML), oder in der EAM-Bindemoduldatei.

Bei einem Modul bezeichnet der Name SYMBOL oder SYMBOL@ einen Modulnamen und kann sein

– der Name eines Bibliothekselements (Typ R oder Typ L) in einer Programmbibliothek,
– der Name eines Bibliothekselement in einer Bindemodulbibliothek (OML).

ANY
Gesucht wird in folgender Reihenfolge:

1. LLMs mit dem Modulnamen SYMBOL oder SYMBOL@
2. OMs mit dem Modulnamen SYMBOL oder SYMBOL@
3. Symbole mit dem Symbolnamen SYMBOL oder SYMBOL@ in einem LLM. Zuerst werden CSECTs gesucht. Wird keine CSECT gefunden, werden ENTRYs gesucht.
4. Symbole mit dem Symbolnamen SYMBOL oder SYMBOL@ in einem OM. Zuerst werden CSECTs gesucht. Wird keine CSECT gefunden, werden ENTRYs gesucht.

CSECT
Nur Programmabschnitte (CSECTs) mit dem Symbolnamen SYMBOL@ oder SYMBOL werden gesucht.

ENTRY
Nur Einsprungstellen (ENTRYs) mit dem Symbolnamen SYMBOL oder SYMBOL@ werden gesucht.

CSEN
CSECTs *und* ENTRYs mit dem Symbolnamen SYMBOL oder SYMBOL@ werden gesucht. Zuerst werden CSECTs gesucht. Wird kein CSECT gefunden, werden ENTRYs gesucht.

MODULE
Nur Module mit dem Modulnamen SYMBOL oder SYMBOL@ werden gesucht.

VERSION=
Gibt die Elementversion des mit SYMBOL oder SYMBOL@ vereinbarten Bibliothekselements an.
Für SYMTYP=ANY wird der Operand VERSION nur berücksichtigt, wenn sich der Objektname (SYMBOL oder SYMBOL@) auf ein Modul in einer Programmbibliothek (Typ R oder Typ L) bezieht. Bei Angabe eines CSECT- oder ENTRY-Namens wird der Operand VERSION ignoriert. Fehlt der Operand, wird der Standardwert für die höchste Elementversion bei Programmbibliotheken übernommen (siehe Handbuch „LMS“ [29]).

version
Explizite Angabe der Elementversion. Die Versionsangabe darf maximal 24 Zeichen lang sein.

VERS@=
Gibt die Adresse eines Feldes an, das die Elementversion enthält. Angabe nur mit MF=M.

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.

(r)
r = Register mit der gesuchten Feldadresse.

Hinweise zum Makroaufruf

- Der Benutzer muss sich vor dem Makroaufruf **ASHARE** an den Common Memory Pool mit dem Makroaufruf **ENAMP** angeschlossen haben. Der Memory Pool muss mit dem **ENAMP**-Operanden FIXED=YES eingerichtet worden sein und SCOPE darf nicht LOCAL sein.
- Wurde der Memory Pool im Adressraum oberhalb 16 MB angelegt, müssen alle CSECTs der geladenen Module das Attribut RMODE=ANY haben.
- Externverweise, die nicht befriedigt werden können sowie Namenskonflikte sind nicht zulässig.
- LLMs mit benutzerdefinierten Slices können nicht mit dem **ASHARE**-Makro in Common Memory Pools geladen werden. Bei LLMs, deren Slices nach dem Attribut PUBLIC gebildet wurden, wird nur der PUBLIC-Teil mit **ASHARE** geladen. Das gilt auch für die Autolink-Funktion des DBL.
- Test- und Diagnoseinformationen (LSD) werden nicht berücksichtigt.
- Die Anzahl von Memory Pools, in denen der Benutzer Shared Code ablegen kann, ist für eine Benutzerkennung auf 16 pro Geltungsbereich (**ENAMP**-Operand SCOPE) begrenzt:

  1. maximal 16 Memory Pools mit dem Geltungsbereich SCOPE=GROUP,
  2. maximal 16 Memory Pools, die einer bestimmten Benutzergruppen-Nummer zugeordnet sind, mit dem Geltungsbereich SCOPE=USER-GROUP,
  3. maximal 16 Memory Pools mit dem Geltungsbereich SCOPE=GLOBAL.

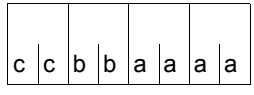
Eine Task kann demzufolge auf insgesamt 48 Memory Pools für Shared Code im Benutzeradressraum zugreifen.
Shared Code, der wie bei SCOPE=GLOBAL für alle Benutzer zugreifbar sein soll, kann mit DSSM auch als nichtprivilegiertes Subsystem in den Systemadressraum geladen werden.

– Wenn die angegebene Programmversion noch nicht geladen ist, aber ein Programm mit diesem Namen im Link-Kontext (siehe Operand CONTEXT) bereits vorhanden ist, wird das Laden wegen Namenskonflikt abgewiesen.
  Um solche Namenskonflikte zu vermeiden, müssen verschiedene Versionen eines Programmes in verschiedene Kontexte geladen werden.
– Für Binde- und Ladeinformationen der mit **ASHARE** geladenen Module steht nur eine begrenzte Anzahl von Speicherseiten im Systemadressraum zur Verfügung, die im Startup-Parameterservice mit dem Systemparameter BLSUSLIM festgelegt wird.
– Ein residenter Memory Pool kann bei **ASHARE** nur verwendet werden, wenn er in einem Lademodul erzeugt wird, das mit dem entsprechenden Wert für RESIDENT-PAGES (siehe Kommando START-EXECUTABLE-PROGRAM [19]) geladen wurde.
– Beim Suchen nach der Primäreingabe werden bei **ASHARE** Symbole des Typs ILE übergangen, da der Eintrittspunkt eines Server-Moduls für das indirekte Binden nie ein ILE-Symbol sein darf.

Rückinformation und Fehleranzeigen

Die Startadresse der Ladeeinheit wird in dem Feld übergeben, das mit dem Operanden START@ festgelegt wurde.

Standard-header:



Über die Ausführung des Makros ASHARE wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	**X'bb'**	**X'aaaa'**	**Erläuterung**
X'00'	X'00'	X'0000'	Die Funktion wurde normal ausgeführt.
X'00'	X'01'	X'0001'	Der Symbolname fehlt.
X'00'	X'01'	X'0002'	Die Kurzkennung des Memory Pools (Operand MPID) fehlt.
X'00'	X'01'	X'0003'	Die Kurzkennung bei MPID ist ungültig.
X'00'	X'01'	X'0004'	Der Kontextname ist ungültig.
X'00'	X'01'	X'0005'	Der Symboltyp ist ungültig.
X'00'	X'01'	X'0006'	Die Feldadresse für START@ ist ungültig.
X'00'	X'01'	X'0007'	Der Memory-Pool-Name ist ungültig.
X'00'	X'01'	X'0008'	Der Operand MAP ist ungültig.
X'00'	X'01'	X'0009'	Die SYSLST-Nummer beim Operanden MAP ist ungültig.
X'00'	X'01'	X'0011'	Die Task ist nicht an den Memory Pool angeschlossen.

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'01'	X'0012'	Der Memory Pool wurde mit falschen Attributen eingerichtet: FIXED=YES fehlt beim Makro ENAMP oder es wurde CLASS=5 oder SCOPE=LOCAL bei ENAMP angegeben.
X'00'	X'01'	X'0013'	Der Programmname bei PROGRAM ist nicht eindeutig; er existiert bereits in einem Memory Pool.
X' 00'	X'01'	X'0014'	Der bei CONTEXT angegebene Kontext existiert bereits in einem anderen Memory Pool.
X'00'	X'01'	X'0015'	Der Kontext kann nicht erzeugt werden, da die maximale Anzahl von 15 Kontexten in diesem Memory Pool erreicht ist.
X'00'	X'01'	X'0016'	Die maximal nutzbare Anzahl von 16 Memory Pools mit diesem Geltungsbereich ist für die Benutzerkennung erreicht. Die Nutzung weiterer Memory Pools mit diesem Geltungsbereich ist nicht zulässig.
X'00'	X'01'	X'0017'	Der Systemspeicherbereich für Binde- und Ladeinformation ist belegt (die mit dem Systemparameter BLSUSLIM festgelegte Anzahl von Speicherseiten ist erreicht).
X'00'	X'01'	X'0018'	Das angegebene Symbol ist im gegebenen Kontext bereits geladen.
X'00'	X'01'	X'0019'	Der Memory Pool wurde schon in einem BIND-Makroaufruf angegeben. Er kann deshalb nicht mehr für Shared Code verwendet werden.
X'00'	X'01'	X'0020'	Fehler während des Bindens/Ladens. Der Returncode dieses fehlerhaften Ladevorgangs wird in der Parameterliste übergeben. Die möglichen Werte sind beim BIND-Makro beschrieben.
X'00'	X'20'	X'0100'	Systemfehler
X'00'	X'20'	X'0101'	Interner DBL-Fehler
X'00'	X'20'	X'0103'	Fehler des DBL-Lock-Managers während der Abarbeitung des ASHARE-Makros.
X'00'	X'01'	X'FFFF'	Die Funktion wird nicht mehr oder noch nicht unterstützt.
X'00'	X'03'	X'FFFF'	Die Version der Schnittstelle wird nicht unterstützt.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

ASPC – Speicherplatzbelegung erfassen

Allgemeines

Anwendungsgebiet: Accounting-Systemverwaltermakro; siehe Seite 166
Makrotyp: S-Typ, MF-Format **1**:
31-Bit-Schnittstelle: Standardform/L-/D-/E-Form; siehe Seite 30

Der Makro **ASPC** kann nur unter der Kennung TSOS (Systemverwalter) aufgerufen werden. Unberechtigter Aufruf wird mit Returncode X'0C' zurückgewiesen.

Makrobeschreibung

Der Makro **ASPC** dient der Erfassung der momentanen Speicherplatzbelegung auf öffentlichen und privaten Datenträgern. Die PAM-Seitenbelegung wird in Form von Accounting-Abrechnungssätzen erfasst.

Der Makro **ASPC** schreibt für jeden lokal verfügbaren Public Volume Set (PVS) bzw. Privatplatte (PD) einen oder mehrere Abrechnungssätze (Speicherplatz-Bestandsaufnahmesätze: Satzkennung DSPC oder DSPP) in die Accounting-Datei. Jeder Satz enthält:

– die Katalogkennung (catid) des PVS.
– Zeitpunkt der Bestandsaufnahme (Datum, Uhrzeit).
– einen Indikator über die Vollständigkeit des Satzes:
  `'C'`: Fortsetzungszeichen („continue“)
  `'L'`: Aufzeichnung vollständig („last“)
  `'I'`: Aufzeichnung unvollständig („incomplete“; der PVS wurde während der Aufzeichnung exportiert)
– Benutzerkennungen des PVS (bis zu 26 Benutzerkennungen pro Satz).
– Anzahl der belegten PAM-Blöcke pro Benutzerkennung.

Ausführliche Beschreibung von DSPC- und DSPP-Sätzen siehe Handbuch „Abrechnungssätze“ [13]. Es werden mehrere Sätze für einen PVS geschrieben, wenn der PVS mehr als 26 Benutzerkennungen enthält. Die Sätze enthalten dann das Fortsetzungszeichen.
Der Makro **ASPC** besitzt keine funktionssteuernden Operanden.

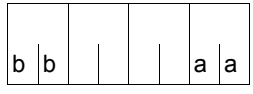
Makroaufrufformat und Operandenbeschreibung

ASPC
,MF=S / L / (E,..) / D ,P=I / p

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Bei MF=D kann ein Präfix P (p = 1 Buchstabe), wie im Aufrufformat dargestellt, angegeben werden.

Rückinformation und Fehleranzeigen

R15:



Über die Ausführung des Makros ASPC wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'bb'	X'aa'	Erläuterung
X'00'	X'00'	normale Ausführung
X'04'	X'00'	Satz wurde nicht geschrieben, weil Accounting oder der Satz DSPC ausgeschaltet war
X'00'	X'04'	Adressfehler (Datenbereich)
X'00'	X'08'	ungültiger Funktionscode (für SVC 99 im Datenbereich)
X'00'	X'0C'	Anwender hat keine Berechtigung zum Aufruf des Makros
X'00'	X'10'	System- oder Ressourcenfehler
X'04'	X'10'	keine Jobinformation
X'08'	X'10'	Fehler beim Zugriff auf den MRS-Katalog
X'0C'	X'10'	Fehler beim Zugriff auf eine interne USERTABLE
X'10'	X'10'	Fehler beim Schreiben des Satzes

In allen Fällen mit aa = X'10' erfolgt zusätzlich ein Eintrag in die SERSLOG-Datei, der den Datenbereich und den Returncode der gerufenen Systemfunktion enthält.

AUDIT – Sprungfolgemodus steuern

Allgemeines

Anwendungsgebiet: Testhilfe; siehe Seite 166
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/D-/E-Form; siehe Seite 30

Bei MF=L werden im Datenbereich keine symbolischen Namen generiert.

Die Funktion Hardware-AUDIT (siehe unten) ist nur auf /390-Servern verfügbar. Auf anderen BS2000-Servern liefert die Funktion zwar RC=0, wird aber nicht ausgeführt.

Makrobeschreibung

Der Makro **AUDIT** bietet dem Anwender programmüberwachende Funktionen.

Neben dem **Hardware-AUDIT** steht auch der **Linkage-AUDIT** zur Verfügung. Beide ermöglichen die Rückverfolgung des Programmlaufs durch Aufzeichnung der Adressen erfüllter Sprungbefehle (Hardware-AUDIT) bzw. der Zieladressen von Unterprogrammsprüngen (Linkage-AUDIT) und sind unabhängig voneinander.
Während der Hardware-AUDIT die Absprungadressen für jeden ausgeführten Sprungbefehl in eine Sprungfolgetabelle (Hardware-AUDIT-Tabelle) einträgt, schreibt der Linkage-AUDIT bei Ausführung der Befehle BASR, BALR, BASSM und BAKR die Sprungzieladressen in die Linkage-AUDIT-Tabelle. So können mit dem Linkage-AUDIT alle innerhalb des Programmlaufs aufgerufenen Unterprogramme ermittelt werden, sofern sie mit einem der oben genannten Befehle aufgerufen wurden.

Für Hardware- und Linkage-AUDIT werden getrennte AUDIT-Tabellen angelegt, die bei der Ausgabe in der Kopfzeile entsprechend gekennzeichnet werden. Eine AUDIT-Tabelle umfasst 64 Wort-Einträge beim Hardware-AUDIT bzw. 1024 Einträge beim Linkage-AUDIT und wird, wenn nicht anders vereinbart, zyklisch überschrieben. Die Anforderung kann für den gesamten Lauf einer Task oder aller Tasks oder für einen Prozess der eigenen Task (z.B. Contingency-Prozess) spezifiziert werden. Der Linkage-AUDIT kann auch prozessorlokal geschaltet werden.

Hardware-AUDIT und den Linkage-AUDIT können erlaubt oder verboten werden

– für eine ganze Session
  Die Steuerung erfolgt über den Systemparameter AUDALLOW=YES/NO.
  Ein bereits eingeschalteter prozessorlokaler Linkage-AUDIT wird wieder ausgeschaltet, wenn der Systemparameter AUDALLOW=NO gesetzt wurde.
  Hardware-AUDIT und Linkage-AUDIT können nur gemeinsam gesteuert werden.

– für eine Benutzerkennung
  Die Steuerung erfolgt mit den Operanden HARDWARE-AUDIT bzw. LINKAGE-AUDIT= *UNCHANGED / *NOT-ALLOWED / *ALLOWED der Kommandos MODIFY-USER-ATTRIBUTES bzw. ADD-USER.

– für eine Task
  Die Steuerung erfolgt mit den Operanden HARDWARE-AUDIT bzw. LINKAGE-AUDIT=*UNCHANGED / *NOT-ALLOWED / *ALLOWED des Kommandos MODIFY-TEST-OPTIONS.

Der Makro **AUDIT** wird mit dem Returncode X'81003C' abgewiesen, wenn Hardware- und Linkage-AUDIT nicht erlaubt sind.

Anmerkungen zum prozessorlokalen Linkage-AUDIT

Der Linkage-AUDIT ermöglicht das CPU-spezifische Einschalten des Linkage-AUDIT für alle aktiven CPUs bzw. alle logischen Maschinen einer Server-Konfiguration. Pro CPU wird eine Trace-Tabelle im privilegierten Klasse-3-Speicher angelegt, die während der gesamten Session erhalten bleibt. Der Linkage-AUDIT kann in der Startup-Phase über den Parameterservice eingeschaltet oder ausgeschaltet werden. Zum Ein- bzw. Ausschalten in der laufenden Session stehen dem Systemverwalter die Kommandos /START- bzw. /STOP-LINKAGE-AUDIT sowie der Makro **AUDIT** zur Verfügung.

Makroaufrufformat und Operandenbeschreibung

```
AUDIT

FCT=HWA / LNKA
,SCOPE=TASK / FUNCT / ALLTASK / SIHGLOB / SYSGLOB
,STATE=USER / SYS / PROC
,ACTION={ ON [,SAVE=n] | OFF | CONT | DISC | GET [,TABLE=adr] }
[,{ TID=tid | TSN=tsn }]
[,PARMOD=31]
,MF=S / L / (E,..) / D
,ID=AUD / pre
```

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

ACTION=
bezeichnet die gewünschte AUDIT-Aktion: Ein-/Ausschalten, Unterbrechen, Fortsetzen oder Ausgeben.

ON
Die 64 bzw. 1024 Worte (256 bzw. 4096 Byte) umfassende AUDIT-Tabelle wird, falls sie nicht schon vorhanden ist, angelegt und auf Anfangszustand gesetzt, indem alle Einträge mit binär Null überschrieben werden. Der laufende Zeiger wird auf den Anfang der Tabelle gesetzt und der AUDIT-Modus aktiviert.

Falls die AUDIT-Tabelle schon vorhanden ist, wird der laufende Zeiger hinter den letzten Eintrag, bei vollgeschriebener Tabelle wieder auf den Anfang der Tabelle (zyklische Überschreibung) gesetzt und der AUDIT-Modus aktiviert.

OFF
Hiermit wird der AUDIT-Modus abgeschaltet und die AUDIT-Tabelle, sowie eine eventuell vorhandene Sicherungstabelle, freigegeben. Wurde die AUDIT-Tabelle über den Operanden GET in einen vom Benutzer angegebenen Speicherbereich kopiert, so bleibt dieser Bereich bis zur Programmbeendigung erhalten. Der Operand SCOPE=FUNCT darf nicht angegeben werden, der Operand SAVE wird ignoriert.

CONT
bewirkt die Reaktivierung des AUDIT nach einem vorangegangenen ACTION=DISC-Aufruf. Die Protokollierung in die AUDIT-Tabelle wird dort fortgesetzt, wo sie durch den ACTION=DISC-Aufruf unterbrochen wurde. Eine eventuell vorhandene Sicherungstabelle wird weiterhin zur Sicherung der AUDIT-Tabellen benutzt.

Der Operand kann nur zusammen mit SCOPE=TASK angegeben werden. Ist zum Zeitpunkt des CONTINUE-Aufrufs kein AUDIT aktiv, so wird er taskweit für den angegebenen Funktionszustand (USER oder SYS) eingeschaltet. Die Operanden TID und TSN dürfen nicht angegeben werden. Der Operand SAVE wird ignoriert.

DISC
Hiermit wird der AUDIT-Modus abgeschaltet, aber die AUDIT-Tabelle und eine eventuell vorhandene Sicherungstabelle beibehalten. Die DISCONTINUE-Funktion ist nur anwendbar für den ganzen Lauf der eigenen Task (SCOPE=TASK), wobei die Operanden TID und TSN nicht angegeben werden dürfen, der Operand SAVE wird ignoriert.

GET
Nur bei SCOPE=TASK zulässig, der Operand SAVE wird ignoriert.

Für FCT=HWA:
Die 64 Worte umfassende Hardware-AUDIT-Tabelle (ohne Berücksichtigung einer Sicherungstabelle) wird in den Speicherbereich kopiert, den der Benutzer durch die im TABLE-Operanden angegebene virtuelle Adresse bezeichnet hat. Der Inhalt der AUDIT-Tabelle wird, im Gegensatz zum SHOW-Kommando, in unveränderter chronologischer Reihenfolge (First-in, First-out) übernommen.

Für FCT=LNKA:
Die gesamte Linkage-AUDIT-Traceinformation (AUDIT- und Sicherungstabelle) wird in den Speicherbereich kopiert, den der Benutzer im Linkage-AUDIT-Datenbereich mit der virtuellen Adresse `audTAB` und der Länge `audLBUF` bezeichnet hat.
Der Inhalt der AUDIT-Tabelle wird so ausgegeben, dass der letzte Eintrag der AUDIT-Tabelle an den Anfang der Ausgabe kommt (First-in, Last-out).

Beim Linkage-AUDIT müssen Adresse und Länge des Übergabepuffers in den Feldern `audTAB` bzw. `audLBUF` abgelegt sein. Dabei ist zu beachten, dass Adresse und Länge des Übergabepuffers nicht direkt beim Makroaufruf angegeben werden können, sondern dass sie mittels Befehlen in den Datenbereich eingetragen werden müssen. Neben der AUDIT-Tabelle wird im Feld `audTABE` die Adresse des nächsten freien Eintrages im Übergabepuffer zurückgeliefert, bzw. die Endadresse des Puffers, wenn dieser voll geschrieben ist.

Wird die Schnittstelle mit ACTION=GET und dem Wert X'00000000' im Feld `audTAB` aufgerufen, so wird im Feld `audLBUF` die benötigte Puffergröße für einen anschließenden GET-Aufruf zurückgeliefert.

Wird im Feld audTAB ein zu kleiner Übergabepuffer angegeben, so wird die Anforderung nur zum Teil erfüllt. Die Übertragung der Linkage-AUDIT-Tabelle erfolgt nur in der im Feld audLBUF angegebenen Länge. Zusätzlich wird der Returncode X'400020' zurückgeliefert.

FCT=
Dieser Operand gibt an, auf welche AUDIT-Funktion sich die folgenden Operanden beziehen.

HWA
Die AUDIT-Anforderung bezieht sich auf den **Hardware-AUDIT**.

LNKA
Die AUDIT-Anforderung bezieht sich auf den **Linkage-AUDIT**.

MF
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei MF=D oder MF=L kann ein Präfix ID (pre = 1..3 Buchstaben), wie im Aufrufformat dargestellt, angegeben werden.

PARMOD=
Dieser Parameter ist unnötig und wird nur noch aus Kompatibilitätsgründen unterstützt. Unabhängig von der Angabe des Operanden wird generell nur noch die 31-Bit-Schnittstelle generiert.

Hinweis: Bereits generiertes 24-Bit-Coding ist natürlich weiterhin ablauffähig.

SAVE=
legt für die AUDIT-Tabelle einen Sicherungsbereich (Sicherungstabelle) im privilegierten Klasse-5-Speicher (bei TU) bzw. im privilegierten Klasse-3-Speicher (bei TPR) derjenigen Task an, die mit **AUDIT** überwacht werden soll. Dieser Sicherungsbereich nimmt den Inhalt der AUDIT-Tabelle auf, bevor sie im Zyklus überschrieben wird. Er vergrößert damit die abrufbare AUDIT-Informationsmenge.
Der Operand wirkt nur in Verbindung mit ACTION=ON und nur, wenn noch keine AUDIT-Tabelle vorhanden ist (z.B. nach ACTION=OFF).

n
gibt die Anzahl der 4KB-Seiten für den Sicherungsbereich an.

Bei nichtprivilegierter Protokollierung des Funktionszustandes TU können für n ganzzahlige Werte mit $0 \leq n \leq 16$ angegeben werden. Maximal können also 64 KB für den Sicherungsbereich angefordert werden, dies entspricht der Größe von 256 Hardware-AUDIT-Tracetabellen bzw. von 16 Linkage-AUDIT-Tracetabellen.

Bei privilegierter Protokollierung des Funktionszustandes TPR mit dem Hardware-AUDIT sind für n nur die Werte 0 und 1 sinnvoll, größere Werte (n > 1) werden durch den Wert 1 ersetzt. Maximal können also im Hardware-AUDIT für TPR 4 KB für den Sicherungsbereich angefordert werden, dies entspricht der Größe von 16 Hardware-AUDIT-Tracetabellen.

Im Linkage-AUDIT ist bei privilegierter Protokollierung des Funktionszustandes TPR sowie bei SCOPE=SIHGLOB/SYSGLOB die Angabe dieses Operanden nicht erlaubt.

SCOPE=
bezeichnet den zu protokollierenden Programmbereich.

Hinweis
Ein AUDIT-Auftrag eines größeren Programmbereichs oder eines höheren PCBs ersetzt den eines kleineren Programmbereichs bzw. eines niedrigeren PCBs (z.B. TASK ersetzt FUNCT), jedoch nicht umgekehrt.

TASK
Die anfordernde Task bzw. die im Operanden TID oder TSN angegebene soll protokolliert werden.

FUNCT
Protokolliert werden soll diejenige PCB-spezifische Funktion, die sich auf den höchsten unterbrochenen PCB des im Operanden STATE angegebenen Funktionszustandes bezieht. FUNCT ist nur für die eigene Task und bei ACTION=ON erlaubt. Die Operanden TID und TSN dürfen nicht angegeben werden.

ALLTASK
Alle Sprungbefehlsadressen des Funktionszustandes TPR sämtlicher Tasks sollen protokolliert werden. Die gleichzeitige Angabe der Operanden TID oder TSN führt zu einer Fehlermeldung, die Angabe von SAVE=n wird bei FCT=HWA ignoriert. Bei FCT=LNKA ist die Angabe von SAVE=n nicht erlaubt (siehe Operand SAVE).
Ggf. schaltet ALLTASK in jeder Task einen aktiven taskweiten Hardware-AUDIT zunächst aus und anschließend ohne Sicherungstabelle wieder ein.
Ist für eine Task der taskweite TPR-Linkage-AUDIT schon eingeschaltet, so wird dieser durch die ALLTASK-Funktion nicht verändert.
Ein nicht-aktiver taskweiter AUDIT wird nicht ausgeschaltet; auch die Sicherungstabelle bleibt erhalten. Kommen neue Tasks hinzu, so wird für diese der AUDIT generell ohne Sicherungstabelle eingeschaltet.
Dieser Wert kann nur unter der Kennung der Systemverwaltung (TSOS) angegeben werden.

SIHGLOB
Alle Sprungbefehlsadressen des Funktionszustandes SIH werden in einer prozessorlokalen AUDIT-Tabelle protokolliert. Die Angabe ist nur in Verbindung mit FCT=LNKA, STATE=PROC und ACTION=ON/OFF zulässig. Die Operanden TID, TSN und SAVE dürfen nicht angegeben werden.
Dieser Wert kann nur unter der Kennung der Systemverwaltung (TSOS) angegeben werden.

SYSGLOB
Alle Sprungbefehlsadressen der Funktionszustände SIH und TPR werden in einer prozessorlokalen AUDIT-Tabelle protokolliert. Voraussetzung für die Anwendung von SCOPE=SYSGLOB ist, dass systemweit in keiner Task ein privilegierter Linkage-AUDIT eingeschaltet ist, bzw. dass sich kein TPR-Linkage-AUDIT im Zustand 'DISCONTINUE' befindet.
Die Angabe ist nur in Verbindung mit FCT=LNKA, STATE=PROC und ACTION= ON/OFF zulässig. Die Operanden TID, TSN und SAVE dürfen nicht angegeben werden.
Dieser Wert kann nur unter der Kennung der Systemverwaltung (TSOS) angegeben werden.

STATE=
Funktionszustand (TU, TPR, SIH), auf den sich der im Operanden SCOPE angegebene Wirkungsbereich bezieht. Sollen mehrere Funktionszustände gleichzeitig protokolliert werden, muss der Makro mehrmals aufgerufen werden.

USER
Der im Operanden SCOPE angegebene Wirkungsbereich bezieht sich auf den Funktionszustand TU (USER). Der Operandenwert P1 wird nur noch aus Kompatibilitätsgründen und nur für den Hardware-AUDIT unterstützt. Voreinstellung für SCOPE=FUNCT und SCOPE=TASK.

SYS
Der im Operanden SCOPE angegebene Wirkungsbereich bezieht sich auf den Funktionszustand TPR (SYS). Der Operandenwert P2 wird nur noch aus Kompatibilitätsgründen und nur für den Hardware-AUDIT unterstützt. Voreinstellung und zwingend für SCOPE=ALLTASK.
Dieser Wert kann nur unter der Kennung der Systemverwaltung (TSOS) angegeben werden.

PROC
Der im Operanden SCOPE angegebene Wirkungsbereich bezieht sich prozessorlokal auf den Funktionszustand SIH bzw. auf SIH und TPR. Die Angabe ist nur in Verbindung mit FCT=LNKA zulässig. Voreinstellung und zwingend für SCOPE=SIHGLOB/ SYSGLOB.
Dieser Wert kann nur unter der Kennung der Systemverwaltung (TSOS) angegeben werden.

TABLE=
bezeichnet die Adresse eines Bereichs, in den der Inhalt der Hardware-AUDIT-Tabelle übertragen werden soll. Die Angabe ist nur beim Hardware-AUDIT und in Verbindung mit ACTION=GET gültig und obligatorisch.

adr
virtuelle Speicheradresse. Die Adresse muss auf einen zugewiesenen Speicher zeigen. Schreibzugriff muss erlaubt sein. Die Adresse 0 darf nicht angegeben werden.
(adr kann mit 1-8 sedezimalen Ziffern bzw. mit der entsprechenden Anzahl von Dezimalziffern angegeben werden).

Es ist vorteilhaft, den Wert des TABLE-Operanden nur mit einem Dummy-Wert zu besetzen und dann dynamisch im Programm das Feld `audTAB` mit der entsprechenden Adresse zu überschreiben.

Beim Linkage-AUDIT wird die Übertragung der AUDIT-Trace-Information in einen Benutzerbereich ausschließlich mit dem Operanden GET gesteuert.

TID=
bezeichnet die Task, die mit AUDIT überwacht werden soll, durch ihre interne Tasknummer. Der Operand darf nur zusammen mit SCOPE=TASK angegeben werden (siehe auch Hinweis zum Operanden TSN).

tid
Interne Tasknummer. Sie kann in folgender Form angegeben werden:
h[hhhhhhh]: 1-8 Sedezimalziffern, die das System gegebenenfalls mit führenden Nullen auf 8 Stellen ergänzt.

TSN=
bezeichnet die Task, die mit AUDIT überwacht werden soll, durch ihre Auftragsnummer.

tsn
Auftragsnummer (TSN). Sie kann wie folgt angegeben werden:

n[nnn]: 1-4 Ziffern, die das System ggf. mit führenden Nullen auf 4 Stellen ergänzt,

a[aaa]: 1-4 alphanumerische Zeichen, die das System ggf. mit führenden Nullen auf 4 Stellen ergänzt,

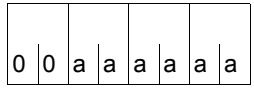
c'a[aaa]': 1-4 Zeichen, die das System ggf. mit führenden Nullen auf 4 Stellen ergänzt (vom Anwender eingegebene führende Blanks bleiben erhalten).

Hinweise

- Wird weder der Operand TID noch TSN angegeben, so gilt die AUDIT-Anforderung für die anfordernde Task selbst.
- Der TID- bzw. TSN-Operand darf nur zusammen mit SCOPE=TASK angegeben werden.
- Bei ACTION=DISC und ACTION=CONT ist die Angabe von TID oder TSN nicht erlaubt.
- Ein nicht-privilegierter Anwender kann die AUDIT-Funktion für eine fremde Task nur einschalten, wenn die fremde Task unter der gleichen Benutzerkennung abläuft wie die aufrufende Task.

Rückinformation und Fehleranzeigen

R15:



Über die Ausführung des Makros AUDIT wird ein Returncode im Register R15 übergeben.

Der aus Kompatibilitätsgründen im R15 übergebene Returncode wird zusätzlich im Standardheader des Makroaufrufs hinterlegt.

X'aaaaaa'	**Erläuterung**
X'000000'	a) die geforderte Aktion ist angenommen worden, b) die geforderte Aktion ist ignoriert worden, weil AUDIT bereits läuft (ACTION=ON), c) die geforderte Aktion ist ignoriert worden, weil AUDIT nicht läuft (ACTION=DISC), d) die geforderte Aktion ist ignoriert worden, weil keine AUDIT-Tabelle zugewiesen ist (ACTION=OFF), e) die geforderte Aktion mit SCOPE=FUNCT ist ignoriert worden weil der AUDIT bereits für SCOPE=TASK eingeschaltet ist. f) Die geforderte Aktion wurde ignoriert, weil der Hardware-AUDIT auf dieser Hardware nicht verfügbar ist. g) Ein SAVE- oder TABLE-Operand ist ggf. ignoriert worden.
X'000004'	Operanden-Fehler. Der generierte Datenbereich enthält unzulässige Operandenkombinationen oder der Benutzer bzw. das Programm ist für die geforderte Aktion nicht privilegiert.
X'000008'	Die Adresse des Datenbereichs ist ungültig oder der Datenbereich ist nicht auf Wortgrenze ausgerichtet oder es ist ein interner Fehler aufgetreten.
X'00000C'	Die Adresse im Operand TABLE (Tabellenübergabebereich) ist ungültig oder der Bereich ist nur lesbar (read only).
X'000010'	Es gibt keinen ausreichenden Speicher, entweder für Klasse-3- Originaltabellen oder für Klasse-4-Verwaltungsbereiche oder für Klasse-5-Großtabellen oder Klasse-5-Verwaltungsbereiche. Die AUDIT-Aktion konnte nicht durchgeführt werden.
X'000014'	Task mit bezeichneter TID bzw. TSN existiert nicht.
X'000018'	Der durch SCOPE=FUNCT angesprochene PCB existiert nicht.
X'01FFFF'	Die im Standardheader angegebene UNIT- bzw. FUNCTION-Nummer ist unzulässig. Verarbeitung wird abgebrochen.
X'03FFFF'	Ungültige Versionsnummer im Standardheader.
X'400020'	Der Tabellenübergabebereich für die Linkage-AUDIT-Tabelle (ACTION=GET) ist zu klein. Die Funktion wurde nicht vollständig ausgeführt.
X'400024'	Die Funktion konnte nicht ausgeführt werden, da a) ein prozessorlokaler Linkage-AUDIT läuft (STATE=SYS) oder b) ein Linkage-AUDIT im Funktionszustand TPR läuft (STATE=PROC) oder c) ein prozessorlokaler SIH-Linklage-AUDIT läuft (SCOPE=SYSGLOB) oder d) ein prozessorlokaler SIH- und TPR-Linkage-AUDIT läuft (SCOPE=SIHGLOB).

X'aaaaaa'	**Erläuterung**
X'400028'	Die TU-Sicherungstabelle ist wegen internem AUDIT-Fehler nicht im Übergabepuffer enthalten.
X'40002C'	Die mit TID oder TSN bezeichnete Task befindet sich in der Beendigung. Der Auftrag wurde abgelehnt.
X'810030'	Die Makro-Aufruf-Folge AUDIT FCT=HWA,ACTION=ON/-OFF/-ON für eine Fremdtask in TPR bringt beim 2. ACTION=ON diesen RC immer dann, wenn die Fremdtask zwischenzeitlich nicht gelaufen ist. Dies bedeutet, dass der zweite Makro-Aufruf mit ACTION=ON zu einem späteren Zeitpunkt wiederholt werden muss, weil der Makro-Aufruf mit ACTION=OFF sich noch in einer Warteschlange befindet.
X'810034'	Makroaufruf AUDIT FCT=HWA/LNKA,STATE=U Für die aktive Benutzerkennung wurde der Hardware- bzw. Linkage-AUDIT von der Systemverwaltung mit dem Kommando /MODIFY-USER-ATTRIBUTES gesperrt.
X'810038'	Makroaufruf AUDIT FCT=HWA/LNKA,STATE=U Hardware- bzw. Linkage-AUDIT wurde vom Benutzer mit dem Kommando /MODIFY-TEST-OPTIONS für die aktive Task gesperrt.
X'81003C'	Makroaufruf AUDIT FCT=HWA/LNKA Für die aktuelle Session wurden von der Systemverwaltung im STARTUP-Parameterservice mit dem Parameter AUDALLOW=NO alle AUDIT-Makroaufrufe verboten.
X'81FFFF'	Die zentrale Linkage-AUDIT-Verwaltung ist zurzeit gesperrt (bitte warten und den Auftrag nach einiger Zeit wiederholen).

BIND – Ladeeinheit binden und laden

Anwendungsgebiet: Binden und Laden; siehe Seite 47
Makrotyp: S-Typ, MF-Format **2**: Standardform/C-/D-/L-/E-/M-Form; siehe Seite 30

Zum dynamischen Bindelader DBL siehe auch Handbuch „BLSSERV“ [4].

Makrobeschreibung

Der Makro **BIND** bindet eine weitere Ladeeinheit in das ablaufende Programm ein.

Makroaufrufformat und Operandenbeschreibung

```
BIND
,ALTLIB=*DBLOPT / NO / YES / list-poss(2): *TASKLIB / *BLSLIB##
[,AMODE@=adr / (r) / label]
[,AMODCHK = *DBLOPT / STD / ADVANCED]
,AUTOLNK=*DBLOPT / YES / NO / ALTLIB
,BRANCH=NO / YES
,CLOSE=*DBLOPT / ALL / NONE / ALT
,ERREXIT=*DBLOPT / adr / (r) / label
,IGNATTR=*DBLOPT / NONE / READ
,INTVERS=BLSP2 / SRV001 / SRV002 / SRV003 / SRV004 / SRV005 / SRV006
[,LDINFO=*DBLOPT / DEF / MAP / NONE / REF ]
,{ LIBNAM@=adr / (r) / label
 { LIBNAM=*DBLOPT / datei / *
 { LIBLINK=name
,{ LNKCTX@=adr / (r) / label
 { LNKCTX=*DBLOPT / name
,LNKCTXS=*DBLOPT / ANY / OLD / NEW
,LOAD=YES / NO / ILESERVER
[,LOAD@=adr / (r) / label]
,MAP=*DBLOPT / NO / BOTH / (BOTH,nn) / nn / SYSOUT
[,MPID=adr / (r) / label]
[,MSG=*DBLOPT / INFORMATION / WARNING / ERROR / NONE]
[,NACOL=*DBLOPT / STD / ABORT]
```

BIND (Fortsetzung)

```
,OVERLAY=NO / YES
,{ PGMVER@=adr / (r) / label
 { PGMVERS=*DBLOPT / *STD / version }
,PROGMOD=*DBLOPT / ANY / 24
[,PURESOR= list-poss(3): USERSHARE / SYSSHARE / LNKCTX ]
[ ,PURESTY=*DBLOPT / STD / USER]
[,{ REFCTX@=adr / (r) / label
  { REFCTX=name / (name1,name2,...name200) }]
,REFCTX#=0 / n
,{ REPFIL@=adr / (r) / label
 { REPFILE=*DBLOPT / file }
,REPSCOP=*DBLOPT / CONTEXT / UNIT
[,RESORD= list-poss(4): LNKCTX / USERSHARE / SYSSHARE / REFCTX ]
,RESTYP=*DBLOPT / STD / USER
,SHARE=*DBLOPT / SYSTEM / NONE / USER / GROUP / USER_GROUP / GLOBAL / ALL
,{ SYMBOL@=adr / (r) / label
 { SYMBOL= name / *ALL }, SYMBLAD=adr / (r) / label
,SYMTYP=ANY / CSECT / ENTRY / CSEN / MODULE
[,TSTOPT=*DBLOPT / NONE / AID]
[,{ UNIT@=adr / (r) / label
  { UNIT= name }]
[,UNRES=*DBLOPT / STD / DELAY / DELAYWARN / ABORT]
,USRMAPI = NONE / STD / ALL
[,USRMAP@ = adr / (r) / <label> ]
[,USRMAPL = integer 1..21474836479]
[,USRUNR@ = adr / (r) / label]
[,USRUNRL = integer 1..21474836479]
,USRUNRI = STD / DELAY / BOTH
[,{ VERS@=adr / (r) / label
  { VERS=version }]
,XPAND=PARAM / XRC / USRMAP / USRUNR
[,XRC=adr / (r) / label]
,XRCL=28 / 36
,MF=S / C / D / E / L / M [,PARAM=adr / (r)] ,PREFIX=P / p [,LABEL=name]
```

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

ALTLIB=
Legt fest, ob alternative Bibliotheken oder Tasklibs nach dem mit SYMBOL@ oder SYMBOL vereinbarten Objekt durchsucht werden. Alternative Bibliotheken werden mit dem Dateikettungsnamen BLSLIBnn (00≤nn≤99) bzw. $BLSLBnn zugewiesen. Sie werden auch für die Autolink-Funktion des DBL benutzt.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt ALTLIB=NO.

NO
Alternative Bibliotheken und/oder Tasklibs werden nicht durchsucht.

YES
Alternative Bibliotheken werden durchsucht.

***TASKLIB**
Darf nur mit INTVERS=SRVxxx und xxx ≥ 002 angegeben werden.
Tasklibs werden in folgender Reihenfolge durchsucht:

1. Die Bibliothek, die mit dem Kommando SET-TASKLIB zugewiesen wurde
2. Die Bibliothek $userid.TASKLIB

   oder, falls diese nicht existiert:

   Die Bibliothek TASKLIB unter der System-Standardkennung (DEFLUID-Kennung)

***BLSLIB##**
Darf nur zusammen mit INTVERS=SRVxxx und xxx ≥ 002 angegeben werden.
Alternative Bibliotheken werden durchsucht.

Hinweise

– Die Operandenwerte *TASKLIB und *BLSLIB## können als Liste angegeben werden. Die Reihenfolge dieser Werte in der Liste legt die Reihenfolge fest, in der die entsprechenden Bibliotheken durchsucht werden.

– Die Angaben ALTLIB=YES und ALTLIB=BLSLIB## haben dieselbe Bedeutung.

– Die Operanden ALTLIB=YES und LIBNAM@/LIBNAM/LIBLINK dürfen zusammen angegeben werden. Ist jedoch die mit den Operanden LIBNAM@/LIBNAM/LIBLINK angegebene Hauptbibliothek nicht vorhanden, bricht der DBL die Verarbeitung ab. Die Angabe ALTLIB=YES dient nicht als Ersatz für die fehlende oder ungültige Hauptbibliothek.

AMODCHK=
Bestimmt, ob während des Ladens zusätzliche Prüfungen des Adressierungsmodus stattfinden (nur zusammen mit INTVERS=SRVxxx und xxx ≥ 005). Fehlt der Operand AMODCHK, wird als Standardwert der eingestellte Wert aus dem Ladeaufruf START-EXECUTABLE-PROGRAM bzw. LOAD-EXECUTABLE-PROGRAM (oder START-PROGRAM bzw. LOAD-PROGRAM) übernommen.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt AMODCHK=STD.

STD
Es werden nur die zu BLSSERV < V2.5 kompatiblen Prüfungen durchgeführt.

ADVANCED
Es werden die Prüfungen wie bei AMODE-CHECK = *STD durchgeführt.
Zusätzlich wird während des Ladens geprüft, ob sich durch den Adressierungsmodus der Ladeeinheit Inkonsistenzen beim Auflösen der Externverweise ergeben können.

AMODE@=
Gibt die Adresse eines 1 Byte langen Feldes an, in das der DBL den Adressierungsmodus für den Aufruf der Ladeeinheit einträgt. Das Feld muss auf Wortgrenze ausgerichtet sein. Mögliche Werte für den Adressierungsmodus sind:
2 für AMODE 24
1 für AMODE 31
4 für AMODE 32

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.
Angabe nur mit MF=M.

(r)
r = Register mit der gesuchten Feldadresse. Angabe nur mit MF=M.

label
Symbolische Adresse des Feldes. Angabe nur mit MF=S oder MF=L.

AUTOLNK=
Legt fest, ob die Autolink-Funktion des DBL eingeschaltet oder ausgeschaltet wird.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt AUTOLNK=YES.

YES
Die Autolink-Funktion wird eingeschaltet.

NO
Die Autolink-Funktion wird ausgeschaltet. Externverweise werden nur durch bereits geladene Programme (private und gemeinsam benutzbare) befriedigt.

ALTLIB
Die Autolink-Funktion greift nur auf alternative Bibliotheken zu, die mit dem Dateikettungsnamen BLSLIBnn (00≤nn≤99) oder $BLSLBnn zugewiesen sind.

BRANCH=
Legt fest, ob unmittelbar nach dem Laden der Ladeeinheit das aufrufende Programm weiter ausgeführt wird oder ob die geladene Ladeeinheit verarbeitet wird. Der Adressierungsmodus wird vom DBL eingestellt.

NO
Nach dem Laden der Ladeeinheit wird im aufrufenden Programm der dem Makroaufruf **BIND** folgende Befehl ausgeführt.

YES
Nach dem Laden wird zuerst die geladene Ladeeinheit verarbeitet. Die Adresse bestimmt der DBL aus dem mit SYMBOL@ oder SYMBOL angegebenen Symbolnamen bzw. Modulnamen.

CLOSE=
Legt fest, ob vom DBL benutzte Bibliotheken am Ende der Bearbeitung des DBL-Aufrufs geschlossen werden oder eröffnet bleiben. Dies sind alle Bibliotheken, die vom DBL nach Modulen durchsucht werden. Der Operand kann benutzt werden, um die Verarbeitung zu beschleunigen, wenn der DBL mehrmals mit derselben Bibliothek aufgerufen wird.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt CLOSE=ALL.

ALL
Alle benutzten Bibliotheken werden geschlossen.

NONE
Alle benutzten Bibliotheken bleiben eröffnet. Sie können für einen weiteren Aufruf des DBL verwendet werden.

ALT
Alle benutzten alternativen Bibliotheken werden geschlossen. Nur die Hauptbibliothek, die mit dem Operanden LIBNAM@, LIBNAM oder LIBLINK angegeben wurde, bleibt eröffnet.

ERREXIT=
Legt die Adresse eines 4 Byte langen Feldes fest.
In diesem Feld ist die Adresse einzutragen, die unbefriedigte Externverweise erhalten sollen, falls der Operand UNRES mit dem Wert STD/DELAY/DELAYWARN angegeben ist.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, erhalten unbefriedigte Externverweise die Adresse X'FFFFFFFF'.

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.
Angabe nur mit MF=M.

(r)
r = Register mit der gesuchten Feldadresse. Angabe nur mit MF=M.

label
Symbolische Adresse des Feldes. Angabe nur mit MF=S oder L.

IGNATTR=
Gibt an, welche CSECT-Attribute beim Laden ignoriert werden.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt IGNATTR=NONE.

NONE
Alle CSECT-Attribute werden beim Laden beachtet.

READ
Das CSECT-Attribut READ-ONLY wird beim Laden ignoriert. Die CSECT wird in eine lese-/schreibbare Hauptspeicherseite geladen. Dadurch wird z.B. das Setzen von Haltepunkten beim Testen mit AID ermöglicht.

INTVERS=
Der Operand legt die Version der Makro-Schnittstelle BIND fest.

BLSP2
Default. Entspricht der Makro-Version 5.

SRV001
Entspricht der Makro-Version 6. Diese Version wird ab BLSSERV V2.2 unterstützt.

SRV002
Entspricht der Makro-Version 7. Diese Version wird ab BLSSERV V2.3A unterstützt.

SRV003
Entspricht der Makro-Version 8. Diese Version wird ab BLSSERV V2.3B unterstützt.

SRV004
Entspricht der Makro-Version 9. Diese Version wird ab BLSSERV V2.4A unterstützt.

SRV005
Entspricht der Makro-Version 10. Diese Version wird ab BLSSERV V2.5A unterstützt.

SRV006
Entspricht der Makro-Version 11. Diese Version wird ab BLSSERV V2.6A unterstützt.

LABEL=name
Angabe nur mit MF=M
Name der Struktur, d.h. der DSECT, die die Operanden des **BIND**-Makros beschreibt. Der Operand muss angegeben werden, wenn keine gültige USING-Anweisung für die Definition des Basisadressregisters für die DSECT des Datenbereichs angegeben ist.
Der Operand LABEL muss zusammen mit dem Operand PARAM angegeben werden. Beide Operanden werden benutzt, um eine gültige USING-Anweisung zu gewinnen.
Als Name kann angegeben werden:

– Der Name der im Namensfeld eines vorhergehenden Makroaufrufs `name BIND MF=D` angegeben wurde.
– Der Name „xPBBNDS“, wenn bisher noch kein Name „name“ angegeben wurde. Dabei ist „x“ der Wert des Operanden PREFIX eines vorhergehenden Makroaufrufs `BIND MF=D, PREFIX=x`. Der Standardwert von „x“ ist „P“.
– Der Name der längeren DSECT, die den Datenbereich des **BIND**-Makros enthält, wenn zuvor der Makroaufruf `BIND MF=C` angegeben wurde.

LDINFO=
Legt die Ladeinformation der Ladeeinheit fest.
Fehlt der Operand LDINFO, wird als Standardwert der eingestellte Wert aus dem Ladeaufruf START-EXECUTABLE-PROGRAM bzw. LOAD-EXECUTABLE-PROGRAM (oder START-PROGRAM bzw. LOAD-PROGRAM) übernommen

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt der Wert, der in der Makro-Syntaxbeschreibung auf *DBLOPT folgt.

DEF
Ein Externadressbuch, das die Programmdefinitionen aller Module der Ladeeinheit enthält, wird geladen.
Programmdefinitionen sind Programmabschnitte (CSECTs), Einsprungstellen (ENTRYs), COMMON-Bereiche, Pseudoabschnitte (DSECTs), Externe Pseudoabschnitte (XDSECs-D) und Modulnamen.

MAP
Nur ein Externadressbuch, das für den Aufbau der DBL-Liste notwendig ist, wird *temporär* geladen. Das Externadressbuch wird entladen, sobald die DBL-Liste aufgebaut ist.

NONE
Es wird kein Externadressbuch geladen.

REF
Ein Externadressbuch, das zusätzlich zu den Programmdefinitionen die befriedigten Referenzen aller Module der Ladeeinheit enthält, wird geladen. Referenzen sind Externverweise (EXTRNs), V-Konstanten, bedingte Externverweise (WXTRNs) und Externe Pseudoabschnitte (XDSECs-R).

LIBLINK=name
Dateikettungsname der Hauptbibliothek. Der Name darf maximal 8 Zeichen lang sein.

LIBNAM@=
Legt die Hauptbibliothek fest, in der das mit SYMBOL@ oder SYMBOL vereinbarte Objekt gesucht wird. Die Hauptbibliothek wird festgelegt durch die Adresse eines Feldes, das den Dateinamen der Bibliothek enthält. Die EAM-Bindemoduldatei wird durch den Dateinamen „*“ festgelegt. Ein Dateikettungsname ist für die EAM-Bindemoduldatei nicht möglich.
Die Hauptbibliothek wird *vor* den alternativen Bibliotheken durchsucht (siehe Seite 270). Sie wird auch für die Autolink-Funktion des DBL benutzt. Ist der Operand LIBLINK angegeben, wird LIBNAM@ ignoriert.

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.
Angabe nur mit MF=M.

(r)
r = Register mit der gesuchten Feldadresse. Angabe nur mit MF=M.

label
Symbolische Adresse des Feldes. Angabe nur mit MF=S oder MF=L.

LIBNAM=
Dateiname der Hauptbibliothek. Angabe nur mit MF=S oder MF=L.
Ist der Operand LIBLINK angegeben, wird LIBNAM ignoriert.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, wird die Bibliothek mit dem Dateikettungsnamen BLSLIB durchsucht.

datei
Explizite Angabe des Dateinamens der Hauptbibliothek. Der Dateiname darf maximal 54 Zeichen lang sein.

*****
Vereinbart als Hauptbibliothek die EAM-Bindemoduldatei.

LNKCTX@=
Gibt die Adresse eines Feldes an, das den Namen des Link-Kontext enthält. Der Name muss mit einem Buchstaben beginnen.

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.
Angabe nur mit MF=M.

(r)
r = Register mit der gesuchten Feldadresse. Angabe nur mit MF=M.

label
Symbolische Adresse des Feldes. Angabe nur mit MF=S oder MF=L.

LNKCTX=
Angabe nur mit MF=S oder MF=L.

name
Explizite Angabe des Namens für den Link-Kontext.
Der Name darf maximal 32 Zeichen lang sein und muss mit einem Buchstaben beginnen.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, wird der Link-Kontext „LOCAL#DEFAULT“ verwendet.

LNKCTXS=
Legt fest, ob der neue über LNKCTX@ oder LNKCTX definierte Link-Kontext unter den bereits vorhandenen Benutzerkontexten existieren darf oder nicht.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt LNKCTXS=ANY.

ANY
Falls ein Benutzerkontext mit dem beim Operanden LNKCTX@ oder LNKCTX angegebenen Namen vorhanden ist, wird dieser benutzt.
Wenn kein gleichnamiger Kontext vorhanden ist, wird ein neuer Kontext angelegt.

OLD
Der angegebene Link-Kontext muss bereits existieren. Es wird kein neuer Kontext angelegt.

NEW
Der angegebene Link-Kontext darf noch nicht existieren. Ein neuer Kontext wird erzeugt.

LOAD=
Legt fest, ob ein Symbol mit dem Symbolnamen SYMBOL@ oder SYMBOL geladen werden soll oder ob das Laden eines ILE-Servermoduls gefordert ist. Davon ist abhängig, ob der DBL beim Suchen nach der Primäreingabe Symbole des Typs ILE berücksichtigt.

YES
Das Symbol wird geladen, falls es noch nicht geladen ist. ILE-Symbole werden einbezogen.

NO
Das Symbol wird nicht geladen. Falls es bereits geladen ist, wird die Ladeadresse vom DBL übergeben. ILE-Symbole werden einbezogen.
Der Aufruf **BIND** ...,LOAD=NO sollte aus Performance-Gründen durch den Aufruf **VSVI1** ...,SELECT=BYNAME ersetzt werden.

ILESERVER
Das zu ladende Symbol ist der Eintrittspunkt eines ILE-Servermoduls. ILE-Symbole werden deshalb übergangen.

LOAD@=
Gibt die Adresse eines Bereichs unterhalb 16 MB im Klasse-6-Speicher an, ab der das erste Modul geladen werden soll. Falls noch weitere Module vorhanden sind, werden sie in einem freien Bereich hinter dem ersten Modul geladen.
Wenn die Adresse oberhalb 16 MB im Klassse-6-Speicher liegt, wird der Operand LOAD@ übergangen und der Operand OVERLAY auf NO gesetzt. Das Laden wird nicht durchgeführt wenn:
– die Adresse nicht auf Doppelwortgrenze ausgerichtet ist oder eine unzulässige Adresse im Klasse-6-Speicher angegeben wurde,
– der Benutzer mehrere Kontexte im Klasse-6-Speicher definiert hat,
– der Bereich im Klasse-6-Speicher, in dem das erste Modul geladen werden soll, bereits belegt ist und der Operand OVERLAY=NO angegeben wurde,
– ein LLM entweder als erstes Modul oder durch Autolink geladen wurde,
– gleichzeitig der Operand MPID angegeben wurde.

Falls der Operand LOAD@ nicht angegeben wird, bestimmt der DBL die Ladeadresse. Wurde beim Speichern eines LLM eine Ladeadresse angegeben, wird das LLM ab dieser Adresse geladen, falls dies möglich ist.

adr
Adresse eines Feldes, das die Bereichsadresse enthält.
Angabe nur mit MF=M.

(r)
r = Register mit der Bereichsadresse. Angabe nur mit MF=M.

label
Bereichsadresse. Sie kann als symbolische Adresse oder als Konstante (X'...') angegeben werden. Angabe nur mit MF=S oder MF=L.

MAP=
Legt fest, ob eine DBL-Liste ausgegeben wird oder nicht und gibt das Ausgabeziel für die DBL-Liste an.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt MAP=NO.

NO
Es wird keine DBL-Liste ausgegeben.

BOTH
Das Ausgabeziel ist die Systemdatei SYSOUT und die Systemdatei SYSLST00.

(BOTH,nn)
Das Ausgabeziel ist die Systemdatei SYSOUT und eine Systemdatei SYSLSTnn (00≤nn≤99).

nn
Das Ausgabeziel ist eine Systemdatei aus der Menge SYSLST00 bis SYSLST99, deren Nummer hier anzugeben ist.
Die Nummer muss 2-stellig angegeben werden (00 für 0 usw.).

SYSOUT
Das Ausgabeziel ist die Systemdatei SYSOUT.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. angegebenen Operanden PARAM und PREFIX siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Bei der C-Form oder D-Form des Makroaufrufs kann ein Präfix PREFIX angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

MPID=
Adresse eines 4 Byte langen Feldes, das die Kurzkennung des Memory Pools enthält, in den die Ladeeinheit geladen wird.
Diese Kurzkennung wird dem Benutzer durch den Makroaufruf **ENAMP** zur Verfügung gestellt.
Der Operand darf nicht angegeben werden:
– wenn die Operanden LOAD@ und OVERLAY angegeben sind,
– wenn ein LLM geladen wird, der benutzerdefinierte Slices enthält.
Für die Verwaltung des Memory Pools ist der Benutzer verantwortlich. Der DBL übergibt Informationen über die geladene Ladeeinheit im Memory Pool nur an den Benutzer, der den Makro **BIND** aufgerufen hat.
Innerhalb eines Benutzerkontexts können mehrere Ladeeinheiten in einen Memory Pool geladen werden. Zum Laden von Shared Code in Memory Pools, der dort als gemeinsam benutzbar zur Verfügung gestellt werden soll, muss der Makro **ASHARE** aufgerufen werden.

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.
Angabe nur mit MF=M.

(r)
r = Register mit der gesuchten Feldadresse. Angabe nur mit MF=M.

label
Symbolische Adresse des Feldes. Angabe nur mit MF=S oder MF=L.

MSG=
Legt die niedrigste Meldungsklasse fest, ab der Meldungen ausgegeben werden. Fehlt der Operand MSG, wird als Standardwert der eingestellte Wert aus dem Ladeaufruf START-EXECUTABLE-PROGRAM bzw. LOAD-EXECUTABLE-PROGRAM (oder START-PROGRAM bzw. LOAD-PROGRAM) übernommen.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt der Wert, der in der Makro-Syntaxbeschreibung auf *DBLOPT folgt.

INFORMATION
Die Meldungen aller Meldungsklassen werden ausgegeben.

WARNING
Nur Meldungen der Meldungsklasse WARNING und ERROR werden ausgegeben. Nicht ausgegeben werden Meldungen der Meldungsklasse INFORMATION.

ERROR
Nur Meldungen der Meldungsklasse ERROR werden ausgegeben.

NONE
Keine Meldungen werden ausgegeben.

NACOL=
Legt fest, wie Namenskonflikte bei Symbolen mit gleichem Namen behandelt werden. Namenskonflikte werden nur entdeckt, wenn die Symbole *nicht* maskiert sind.
Fehlt der Operand NACOL, wird als Standardwert der eingestellte Wert aus dem Ladeaufruf START-PROGRAM bzw. LOAD-PROGRAM übernommen.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt NACOL=STD.

STD
Namenskonflikte zwischen nicht maskierten Symbolen werden durch Warnungsmeldungen angezeigt. Das Modul, das das Symbol mit dem gleichen Namen enthält, wird geladen. Die neue Ausprägung des Symbols wird maskiert, d.h. es wird nicht mehr benutzt, um Externverweise zu befriedigen.

ABORT
Das Laden der aktuellen Ladeeinheit wird abgebrochen, wenn eine Namenskollision zwischen nicht maskierten Symbolen entdeckt wird.

OVERLAY=
Legt fest, ob die Module der Ladeeinheit das Modul überlagern dürfen, das an der mit LOAD@ angegebenen Adresse steht. Der Operand erfordert die Angabe des Operanden LOAD@ und ist unvereinbar mit dem Operanden MPID.

NO
Die Module der Ladeeinheit dürfen das Modul nicht überlagern, das an der mit LOAD@ angegebenen Adresse steht.

YES
Die Module der Ladeeinheit dürfen das Modul überlagern. Das erste Modul der Ladeeinheit wird ab der mit LOAD@ angegebenen Adresse geladen. Falls noch weitere Module vorhanden sind, werden sie lückenlos anschließend geladen. Ausgenommen sind Bereiche, die durch CSECT-Attribute READ-ONLY und PAGEABLE gesperrt sind. Der Benutzer muss dafür sorgen, dass der überlagerte Bereich später nicht mehr gebraucht wird.
Alle Module, die von den Modulen der Ladeeinheit überlagert werden, werden vom DBL entladen. Der benützte Speicherinhalt wird jedoch vor dem erneuten Laden nicht gelöscht.
COMMON-Bereiche werden unabhängig von der angegebenen Ladeadresse geladen.

PGMVER@=
Gibt die Adresse eines Feldes an, das die Programmversion enthält. Wenn diese Version des Programms bereits geladen ist, wird eine Verbindung dazu aufgebaut. Ist diese Programmversion noch nicht geladen, dann erhält die neue Ladeeinheit die angegebene Version.

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.
Angabe nur mit MF=M.

(r)
r = Register mit der gesuchten Feldadresse. Angabe nur mit MF=M.

label
Symbolische Adresse des Feldes. Angabe nur mit MF=S oder MF=L.

PGMVERS=
Gibt die Programmversion an.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt PGMVERS=*STD.

***STD**
Die aus dem Ladeaufruf resultierende Ladeeinheit erhält als Programmversion die Version des geladenen Bibliothekselements. Wenn das im Ladeaufruf angegebene Symbol bereits geladen ist, wird die Programmversion gesucht, die mit dem Kommando SELECT-PROGRAM-VERSION festgelegt wurde. Falls noch keine Programmversion festgelegt ist, verwendet DBL das zuerst gefundene Symbol.

version
Die Versionsangabe darf maximal 24 Zeichen lang sein.

PROGMOD=
Legt fest, in welchen Teil des Adressraums (oberhalb oder unterhalb 16 MByte) die Module der Ladeeinheit geladen werden.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt PROGMOD=ANY.

ANY
Die Module der Ladeeinheit können ober- oder unterhalb 16 MByte geladen werden.

24
Die gesamte Ladeeinheit wird unterhalb 16 MB geladen.
Die Programmausführung erfolgt im 24-Bit-Adressierungsmodus.
Externverweise werden als 24-Bit-Adressen interpretiert.

PURESOR=
Legt die benutzerdefinierte Suchreihenfolge bei der Befriedigung von Externverweisen in PUBLIC-Teilen von LLMs fest, falls PURESTY=USER angegeben wurde (nur zusammen mit INTVERS=SRVxxx und xxx ≥ 001).

list-poss(3): USERSHARE / SYSSHARE / LNKCTX
Die Suchreihenfolge wird durch die Reihenfolge der Schlüsselwörter in der Liste festgelegt. Die Schlüsselwörter haben folgende Bedeutung:

USERSHARE	für Shared Code des Benutzers
SYSSHARE	für Shared Code des Systems
LNKCTX	für Link-Kontext

Wenn beispielsweise PURESOR=(SYSSHARE, LNKCTX, USERSHARE) angegeben ist, wird bei der Befriedigung von Externverweisen in PUBLIC-Teilen in folgender Reihenfolge gesucht:

1. im Shared Code des Systems
2. im Link-Kontext
3. und schließlich im Shared Code des Benutzers.

Hinweise

– Jedes Schlüsselwort darf nur ein einziges Mal in der Liste vorkommen.

– Schlüsselwörter, die in der Liste nicht angegeben sind, werden intern vom DBL entsprechend der vordefinierten Reihenfolge (USERSHARE, SYSSHARE, LNKCTX) ans Ende der Liste angehängt. So wird z. B. die Angabe PURESOR=(SYSSHARE) wie PURESOR=(SYSSHARE, USERSHARE, LNKCTX) behandelt.

– Die Operanden PURESTY und PURESOR haben keinen Einfluß darauf, **welche** Kontexte durchsucht werden. Sie bestimmen ausschließlich die Reihenfolge. **Ob** Shared Code durchsucht wird, muss mit dem Operanden SHARE festgelegt werden. Der DBL führt auch keine Konsistenzprüfung dieser Operanden durch. Wenn zum Beispiel PURESOR=(LNKCTX, SYSSHARE), SHARE=NONE angegeben ist, wird der Shared Code des Systems nicht zur Befriedigung von Externverweisen herangezogen, obwohl das entsprechende Schlüsselwort in der Liste der Suchreihenfolge angeben ist.

PURESTY=
Legt die Suchstrategie für die Befriedigung von Externverweisen in PUBLIC-Teilen von LLMs fest (nur zusammen mit INTVERS=SRVxxx und xxx ≥ 001).

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt PURESTY=STD.

STD
Es gilt die von DBL vordefinierte Suchreihenfolge:

1. Shared Code des Benutzers
2. Shared Code des Systems
3. Link-Kontext

USER
Die Suchreihenfolge wird vom Benutzer mit dem Operanden PURESOR festgelegt.

REFCTX@=
Gibt die Adresse eines Feldes an, das eine Namensliste von Referenz-Kontexten enthält, die zum Befriedigen von Externverweisen durchsucht werden.
In die Liste können maximal 200 Namen von Referenz-Kontexten eingetragen werden. Ein Name darf nicht mit dem Zeichen „$“ beginnen.
Die in der Liste angegebenen Kontexte müssen vorhanden sein. Die Referenz-Kontexte werden in der Reihenfolge durchsucht, wie sie in der Liste angegeben sind. Mit dem Operanden REFCTX# kann die Anzahl der Referenz-Kontexte festgelegt werden. Ein mit LNKCTX@ oder LNKCTX definierter Link-Kontext kann nicht als Referenz-Kontext benutzt werden.

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.
Angabe nur mit MF=M.

(r)
r = Register mit der gesuchten Feldadresse. Angabe nur mit MF=M.

label
Symbolische Adresse des Feldes. Angabe nur mit MF=S oder MF=L.

REFCTX=name / (name1,...,name200)
Angabe nur mit MF=S oder MF=L.
Explizte Angabe einer Liste für die Namen von Referenz-Kontexten.
Jeder Name darf maximal 32 Zeichen lang sein. Bis zu 200 Namen dürfen in die Liste eingetragen werden. Die Namen müssen mit einem Buchstaben beginnen.

REFCTX#=<u>0</u> / n
Legt die Anzahl der Referenz-Kontexte in der mit REFCTX@ oder REFCTX definierten Namensliste fest.
$0 \leq n \leq 200$; Standardwert: 0

REPFIL@=
Gibt die Adresse eines Feldes an, das den Dateinamen einer REP-Datei enthält.
Der Benutzer hat damit die Möglichkeit, REP-Sätze auf die Module einer Ladeeinheit anzuwenden. Die REP-Sätze müssen das Format haben, das vom Dienstprogramm RMS verarbeitet wird (siehe Handbuch „Dienstprogramme“ [27]). Tritt ein Fehler bei der Verarbeitung von REP-Sätzen auf, wird eine Warnungsmeldung ausgegeben und der fehlerhafte REP-Satz wird übergangen. Anschließend wird die REP-Verarbeitung fortgesetzt.

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.
Angabe nur mit MF=M.

(r)
r = Register mit der gesuchten Feldadresse. Angabe nur mit MF=M.

label
Symbolische Adresse des Feldes. Angabe nur mit MF=S oder MF=L.

REPFILE=
Angabe nur mit MF=S oder MF=L.
Legt den Dateinamen der REP-Datei fest.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, wird keine REP-Datei verwendet.

datei
Explizite Angabe des Dateinamens der REP-Datei. Der Name darf maximal 54 Zeichen lang sein.

REPSCOP=
Legt fest, ob die REP-Verarbeitung für alle Module im Kontext oder nur für die Module der aktuellen Ladeeinheit durchgeführt wird.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt REPSCOP=CONTXT.

CONTEXT
Die REP-Verarbeitung wird für alle Module im Kontext durchgeführt.

UNIT
Die REP-Verarbeitung wird nur für die Module der aktuellen Ladeeinheit durchgeführt. Alle übrigen Module im Kontext werden übergangen.

RESORD=
Angabe der benutzerdefinierten Suchreihenfolge bei der Befriedigung von Externverweisen, falls RESTYP=USER angegeben wurde (nur zusammen mit INTVERS=SRVxxx und xxx $\geq$ 001).

list-poss(4): LNKCTX / USERSHARE / SYSSHARE / REFCTX
Die Suchreihenfolge wird durch die Reihenfolge der Schlüsselwörter in der Liste festgelegt. Die Schlüsselwörter haben folgende Bedeutung:

LNKCTX	für Link-Kontext
USERSHARE	für Shared Code des Benutzers
SYSSHARE	für Shared Code des Systems
REFCTX	für Referenz-Kontext

Wenn beispielsweise RESORD=(REFCTX, USERSHARE, SYSSHARE, LNKCTX) angegeben ist, wird bei der Befriedigung von Externverweisen in folgender Reihenfolge gesucht:

1. im Referenz-Kontext
2. im Shared Code des Benutzers
3. im Shared Code des Systems
4. und schließlich im Link-Kontext.

Hinweise

– Jedes Schlüsselwort darf nur ein einziges Mal in der Liste vorkommen.

– Schlüsselwörter, die in der Liste nicht angegeben sind, werden intern vom DBL entsprechend der vordefinierten Reihenfolge (LNKCTX, USERSHARE, SYSSHARE, REFCTX) ans Ende der Liste angehängt. So wird z. B. die Angabe RESORD=(REFCTX, USERSHARE) wie RESORD=(REFCTX, USERSHARE, LNKCTX, SYSSHARE) behandelt.

– Die Operanden RESTYP und RESORD haben keinen Einfluß darauf, **welche** Kontexte durchsucht werden. Sie bestimmen ausschließlich die Reihenfolge. **Ob** Shared Code oder Referenz-Kontext durchsucht wird, muss mit dem Operanden SHARE bzw. REFCTX (oder REFCTX@) festgelegt werden. Der DBL führt auch keine Konsistenzprüfung dieser Operanden durch. Wenn zum Beispiel RESORD=(LNKCTX, SYSSHARE), SHARE=NONE angegeben ist, wird der Shared Code des Systems nicht zur Befriedigung von Externverweisen herangezogen, obwohl das entsprechende Schlüsselwort in der Liste der Suchreihenfolge angegeben ist.

RESTYP=
Legt die Suchstrategie für die Befriedigung von Externverweisen fest (nur zusammen mit INTVERS=SRVxxx und xxx ≥ 001).

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt RESTYP=STD.

STD Es gilt die von DBL vordefinierte Suchreihenfolge:

1. Link-Kontext
2. Shared Code des Benutzers
3. Shared Code des Systems
4. Referenz-Kontext(e)

USER
Die Suchreihenfolge wird vom Benutzer mit dem Operanden RESORD festgelegt.

SHARE=
Legt fest, welcher Teil des Shared Code in die Suche nach dem mit SYMBOL@ bzw. SYMBOL angegebenen Symbol und beim Befriedigen von Externverweisen einbezogen wird. Dies gilt auch für die Autolink-Funktion des DBL, falls AUTOLNK=YES angegeben wird. Befindet sich das gesuchte Symbol in einem Common Memory Pool, gibt der DBL die Ladeadresse zurück, verbindet die Benutzertask mit dem Common Memory Pool und beendet den Ladevorgang. Befindet sich das gesuchte Symbol in einem nichtprivilegierten Subsystem (siehe Handbuch „Verwaltung von Subsystemen“ [12]), gibt der DBL die Ladeadresse ebenfalls zurück, baut eine Verbindung zum Subsystem auf und beendet den Ladevorgang. Bei BRANCH=YES wird danach zur Adresse des gefundenen Symbols verzweigt.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt SHARE=SYSTEM.

SYSTEM
Nur der Shared Code des Systems (im Klasse-3/4/5-Speicher) wird beim Suchen berücksichtigt.

NONE
Kein Shared Code wird beim Suchen berücksichtigt. Der DBL veranlasst das Laden einer privaten Kopie des Programmes.

USER
Nur Shared Code des Benutzers in Common Memory Pools wird beim Suchen berücksichtigt. Dabei spielt es keine Rolle, welchen Geltungsbereich der Common Memory Pool hat.

GROUP
Nur der Shared Code in Common Memory Pools mit dem Geltungsbereich GROUP wird beim Suchen berücksichtigt.

USER_GROUP
Nur der Shared Code in Common Memory Pools mit dem Geltungsbereich USER_GROUP wird beim Suchen berücksichtigt.

GLOBAL
Nur der Shared Code in Common Memory Pools mit dem Geltungsbereich GLOBAL wird beim Suchen berücksichtigt.

ALL
Sowohl der Shared Code des Systems als auch der Shared Code des Benutzers wird beim Suchen berücksichtigt.

SYMBLAD=
Gibt die Adresse eines auf Wortgrenze ausgerichteten, 4 Byte langen Feldes an. In dieses Feld trägt DBL die Adresse ein, an der der Programmlauf fortgesetzt werden muss, wenn die Ladeeinheit referenziert wird.

Diese Adresse hängt vom Operanden SYMTYP ab:

– Wenn SYMTYP=MODULE oder SYMTYP=ANY angegeben ist und ein Modul mit dem Namen nachgeladen wird, der mit dem Operanden SYMBOL (bzw. SYMBOL@) festgelegt wurde, dann ist die Adresse die Startadresse des Moduls (LLM oder OM). Näheres zur Berechnung der Startadresse von LLMs siehe Handbuch "BINDER" [5]. Wenn die Startadresse eines LLMs ein externer Name ist, wird die Adresse dieser CSECT bzw. des ENTRYs eingetragen.

– In allen anderen Fällen (falls SYMTYP=CSECT, ENTRY oder CSEN angegeben ist bzw. wenn SYMTYP=MODULE oder SYMTYP=ANY angegeben ist und ein Modul mit einem anderen Namen nachgeladen wird, als mit dem Operanden SYMBOL (bzw. SYMBOL@) festgelegt wurde) ist die eingetragene Adresse die Adresse der CSECT oder des ENTRYs mit dem Namen, der mit SYMBOL (bzw. SYMBOL@) festgelegt wurde.

Der Adressierungsmodus, der für den Aufruf der Ladeeinheit zu verwenden ist, wird vom DBL folgendermaßen angezeigt:

– auf /390-Servern im höchstwertigen Bit des mit SYMBLAD festgelegten Feldes,

– auf anderen BS2000-Servern im Feld AMODE@, falls angegeben.

In den folgenden Fällen wird das Laden abgebrochen:

– SYMBLAD wurde nicht angegeben,

– das Feld ist nicht auf Wortgrenze ausgerichtet,

– der entsprechende Speicherbereich hat keinen Schreibzugriff oder ist nicht allokiert.

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.
Angabe nur mit MF=M.

(r)
r = Register mit der gesuchten Feldadresse. Angabe nur mit MF=M.

label
Adresse des Feldes. Sie kann als symbolische Adresse oder als Konstante (X'...') angegeben werden. Angabe nur mit MF=S oder MF=L.

SYMBOL@=
Gibt die Adresse eines Feldes an, das den Namen eines Objektes enthält. Der DBL benutzt diesen Namen, um das erste Modul in der Ladeeinheit zu bestimmen, das geladen werden soll. Der Name kann sich auf folgende Objekte beziehen:
– Programmabschnitt (CSECT),
– Einsprungstelle (ENTRY),
– Bindemodul (OM) (Elementname),
– Bindelademodul (LLM) (Elementname).
Der Typ des Objekts wird mit dem Operanden SYMTYP festgelegt.

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.
Angabe nur mit MF=M.

(r)
r = Register mit der gesuchten Feldadresse. Angabe nur mit MF=M.

label
Symbolische Adresse des Feldes. Angabe nur mit MF=S oder MF=L.

Das Feld, das den Namen enthält, muss 32 Zeichen lang sein. Ist der Name kürzer, muss er mit Leerzeichen aufgefüllt werden.

SYMBOL=
Angabe nur mit MF=S oder MF=L.
Gibt den Namen eines Programmabschnitts (CSECT), einer Einsprungstelle (ENTRY), eines Bindemoduls (OM) oder eines Bindelademoduls (LLM) an. Der DBL benutzt diesen Namen, um das erste Modul in der Ladeeinheit zu bestimmen, das geladen werden soll.

name
Explizite Angabe des Objektnamens. Der Name darf maximal 32 Zeichen lang sein.

Als letztes Zeichen des Names darf ein Stern (*) angegeben werden. Dieser repräsentiert eine beliebige Zeichenfolge. In diesem Fall werden aus der Hauptbibliothek (Operand LIBNAM/LIBNAM@/LIBLINK) alle Bibliothekselemente, deren Name dem angegebenen Muster entspricht, in **eine** List-Name-Unit geladen.

Sind im Namen mehrere Sterne (*) angegeben, werden alle Zeichen nach dem ersten Stern ignoriert.

***ALL**
Alle Bibliothekselemente der Hauptbibliothek (Operand LIBNAM/LIBNAM@/LIBLINK) werden in **eine** List-Name-Unit geladen.

SYMTYP=
Gibt den Typ des Objekts an, das mit dem Namen SYMBOL@ oder SYMBOL vereinbart wurde, und legt die Suchreihenfolge für das Objekt fest. Als Typ des Objekts kann ein Symbol (CSECT oder ENTRY) oder ein Modul (OM oder LLM) bestimmt werden.
Bei einem Symbol bezeichnet der Name SYMBOL@ oder SYMBOL einen Symbolnamen und kann sein:
– der Name eines nicht maskierten CSECT- oder ENTRY-Eintrags in einer Programmbibliothek (Typ R oder Typ L),
– der Name eines nicht maskierten CSECT- oder ENTRY-Eintrags in einer Bindemodulbibliothek (OML), oder in der EAM-Bindemoduldatei.

Bei einem Modul bezeichnet der Name SYMBOL@ oder SYMBOL einen Modulnamen und kann sein
– der Name eines Bibliothekselements (Typ R oder Typ L) in einer Programmbibliothek,
– der Name eines Bibliothekselement in einer Bindemodulbibliothek (OML).

ANY
Alle Symboltabellen werden in die Suche einbezogen. Die Suchreihenfolge ist vom DBL wie folgt festgelegt:

1. LLMs mit dem Modulnamen SYMBOL@ oder SYMBOL
2. OMs mit dem Modulnamen SYMBOL@ oder SYMBOL
3. Symbole mit dem Symbolnamen SYMBOL@ oder SYMBOL in einem LLM. Zuerst werden CSECTs gesucht. Wird keine CSECT gefunden, werden ENTRYs gesucht.
4. Symbole mit dem Symbolnamen SYMBOL@ oder SYMBOL in einem OM. Zuerst werden CSECTs gesucht. Wird keine CSECT gefunden, werden ENTRYs gesucht.

CSECT
Nur Programmabschnitte (CSECTs) mit dem Symbolnamen SYMBOL@ oder SYMBOL werden gesucht.

ENTRY
Nur Einsprungstellen (ENTRYs) mit dem Symbolnamen SYMBOL@ oder SYMBOL werden gesucht.

CSEN
CSECTs *und* ENTRYs mit dem Symbolnamen SYMBOL@ oder SYMBOL werden gesucht. Zuerst werden CSECTs gesucht. Wird keine CSECT gefunden, werden ENTRYs gesucht.

MODULE
Nur Module mit dem Modulnamen SYMBOL@ oder SYMBOL werden gesucht.

Eine Suche nach evtl. bereits geladenen CSECT- bzw. ENTRY-Namen findet nicht statt. Auf diese Weise kann durch wiederholte BIND-Aufrufe ein Mehrfachladen verursacht werden, was zu BLS0339-Meldungen führen kann.

TSTOPT=
Gibt an, ob symbolische Adressen im Quellprogramm beim Testen mit AID verwendet werden dürfen. Mit symbolischen Adressen können nur Programme getestet werden, für die beim Übersetzen Test-und Diagnoseinformation (LSD) erzeugt wurde. Dazu müssen beim Übersetzen des Quellprogramms bestimmte Compileroptionen gesetzt werden (siehe Benutzerhandbücher der Sprachübersetzer).
Wird der Operand TSTOPT nicht angegeben, wird als Standardwert der eingestellte Wert aus dem Ladeaufruf START-EXECUTABLE-PROGRAM bzw. LOAD-EXECUTABLE-PROGRAM (oder START-PROGRAM bzw. LOAD-PROGRAM) übernommen.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt TSTOPT=NONE.

NONE
Test- und Diagnoseinformation (LSD) wird nicht berücksichtigt.

AID
Erlaubt die Verwendung von symbolischen Adressen des Quellprogramms beim Testen des Programms mit AID (siehe Handbuch „AID“ [3]). Diese Angabe ist nur zulässig, wenn gleichzeitig LDINFO=DEF oder LDINFO=REF angegeben wird.

UNIT@=
Gibt die Adresse eines Feldes an, das den Namen der Ladeeinheit enthält. Der Name kann in nachfolgenden Makroaufrufen **UNBIND** benutzt werden.
Fehlt der Operand, wird der mit SYMBOL@ oder SYMBOL vereinbarte Name übernommen.

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.
Angabe nur mit MF=M.

(r)
r = Register mit der gesuchten Feldadresse. Angabe nur mit MF=M.

label
Symbolische Adresse des Feldes. Angabe nur mit MF=S oder MF=L.

UNIT=name
Angabe nur mit MF=S oder MF=L.
Explizite Angabe des Namens der Ladeeinheit. Der Name darf maximal 32 Zeichen lang sein.

UNRES=
Legt fest, wie nicht befriedigte Externverweise behandelt werden. Alle nicht befriedigten Externverweise werden in die Systemdatei SYSOUT ausgegeben, wobei externe Pseudoabschnitte (XDSECs-R) getrennt aufgelistet werden.

Fehlt der Operand UNRES, wird als Standardwert der eingestellte Wert aus dem Ladeaufruf START-EXECUTABLE-PROGRAM bzw. LOAD-EXECUTABLE-PROGRAM (oder START-PROGRAM bzw. LOAD-PROGRAM) übernommen.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt der Wert, der in der Makro-Syntaxbeschreibung auf *DBLOPT folgt.

STD
Nicht befriedigte Externverweise (außer externe Pseudoabschnitte (XDSECs-R)) erhalten eine Adresse, die im Operanden ERREXIT angegeben ist.

DELAY
Nicht befriedigte Externverweise werden zu einem späteren Zeitpunkt befriedigt. Der Operand ist nur zulässig bei LDINFO=REF.
Der DBL speichert die nicht befriedigten Externverweise im Link-Kontext. Wird die nächste Ladeeinheit im Kontext geladen, versucht der DBL am Ende des Ladens, die gespeicherten Externverweise mit CSECTs und ENTRYs dieser Ladeeinheit zu befriedigen. Dieser Vorgang wiederholt sich beim Laden weiterer Ladeeinheiten, so lange der Kontext besteht.
Externe Pseudoabschnitte (XDSECs-R) können nicht gespeichert werden.
Beim Speichern im Kontext erhalten die nicht befriedigten Externverweise eine (vorläufige) Adresse, die im Operand ERREXIT angegeben ist.

DELAYWARN
Angabe nur zusammen mit INTVERS=SRVxxx, wobei xxx $\geq$ 004.
Das Verhalten ist wie bei UNRES=DELAY. Zusätzlich wird beim Auftreten nicht befriedigter Externverweise ein entsprechender Returncode zurückgeliefert.

Außerdem ist UNRES=DELAYWARN Voraussetzung für USRUNRI=DELAY.

ABORT
Nicht befriedigte Externverweise sind unzulässig. Das Laden der aktuellen Ladeeinheit wird abgebrochen.

USRMAP@=
Adresse eines benutzerdefinierten Datenbereichs, in den Informationen der DBL-Liste ausgegeben werden sollen, die mit dem Operanden USRMAPI festgelegt werden (nur zusammen mit INTVERS=SRVxxx und xxx ≥ 005). Der Datenbereich muss auf Wortgrenze ausgerichtet sein. Die Länge des Bereichs muss mit dem Operanden USRMAPL übergeben werden. Das Layout der Information erhalten Sie mit
`BIND MF=D,XPAND=USRMAP,INTVERS=SRV005.`

Die Angabe wird ignoriert, wenn USRMAPI=NONE.

adr
Adresse eines Hilfsfeldes, das die gesuchte Adresse des Datenbereichs enthält. Angabe nur mit MF=M.

(r)
r = Register mit der gesuchten Adresse des Datenbereichs. Angabe nur mit MF=M.

label
Symbolische Adresse des Datenbereichs. Angabe nur mit MF=S oder MF=L.

USRMAPI=
Bestimmt, welcher Teil der DBL-Liste in einen Datenbereich ausgegeben werden soll, dessen Adresse der Benutzer mit dem Operanden USRMAP@ übergibt (nur zusammen mit INTVERS=SRVxxx und xxx ≥ 005).

NONE
Die DBL-Liste wird nicht in einen benutzerdefinierten Datenbereich ausgegeben.

STD
Kopf der DBL-Liste und die Informationen über Ladeeinheit und Module werden in einen benutzerdefinierten Datenbereich ausgegeben, jedoch keine Informationen über CSECTs und ENTRYs.

ALL
Die gesamte DBL-Liste (wie auf SYSOUT) wird in einen benutzerdefinierten Datenbereich ausgegeben.

USRMAPL=integer 1..21474836479
Länge des benutzerdefinierten Datenbereichs, dessen Adresse mit dem Operanden USRMAP@ übergeben wird (nur zusammen mit INTVERS=SRVxxx und xxx ≥ 005). Die Länge ist in Byte anzugeben. Sie muss (für die Mindest-Information) wenigstens 248 Byte für INTVERS=SRV005 und 336 Byte für INTVERS=SRV006 betragen.

Die Angabe wird ignoriert, wenn USRMAPI=NONE.

USRUNR@=
Adresse eines benutzerdefinierten Datenbereichs, in den eine Liste der nicht befriedigt verbleibenden Externverweise ausgegeben werden soll (nur zusammen mit INTVERS=SRVxxx und xxx ≥ 005). Der Datenbereich muss auf Wortgrenze ausgerichtet sein. Die Länge des Bereichs muss mit dem Operanden USRUNRL übergeben werden.

Die ausgegebene Information besteht im Wesentlichen aus einem Listenkopf, der die Länge der ausgebenen Information und die Anzahl der nicht befriedigt verbleibenden Externverweise enthält, und einer Liste von Sätzen, die jeweils einen dieser Externverweise beschreiben.
Das Layout des Bereichs kann mit `BIND MF=D,XPAND=USRUNR,INTVERS=SRVxxx` mit xxx ≥ 005 generiert werden. Das Layout ist versionsabhängig (Operand `INTVERS`).

adr
Adresse eines Hilfsfeldes, das die gesuchte Adresse des Datenbereichs enthält. Angabe nur mit MF=M.

(r)
r = Register mit der gesuchten Adresse des Datenbereichs. Angabe nur mit MF=M.

label
Symbolische Adresse des Datenbereichs. Angabe nur mit MF=S oder MF=L.

USRUNRI=
Bestimmt, welche nicht befriedigt verbleibenden Externverweise in den Datenbereich ausgegeben werden soll, dessen Adresse der Benutzer mit dem Operanden USRUNR@ übergibt (nur zusammen mit INTVERS=SRVxxx und xxx ≥ 006).

Die Angabe wird ignoriert, wenn USRUNR@ nicht angegeben ist.

STD
Die nicht befriedigt verbleibenden Externverweise aus dem aktuellen Ladevorgang werden in den benutzerdefinierten Datenbereich ausgegeben.

DELAY
Die aus früheren Ladevorgängen (mit UNRES=DELAY) nicht befriedigt verbliebenen Externverweise werden in den benutzerdefinierten Datenbereich ausgegeben.

Bei USRUNRI=DELAY muss auch UNRES=DELAYWARN angegeben werden.

BOTH
Beide Arten von nicht befriedigten Externverweisen werden in den benutzerdefinierten Datenbereich ausgegeben.

Bei USRUNRI=BOTH muss auch UNRES=DELAYWARN angegeben werden.

USRUNRL=integer 1..21474836479
Länge des benutzerdefinierten Datenbereichs, dessen Adresse mit dem Operanden USRUNR@ übergeben wird (nur zusammen mit INTVERS=SRVxxx und xxx ≥ 005). Die Länge ist in Byte anzugeben. Sie muss (für den Ausgabekopf) wenigstens 16 Byte für INTVERS=SRV005 und 20 Byte für INTVERS=SRV006 betragen.

Die Angabe wird ignoriert, wenn USRUNR@ nicht angegeben ist.

VERS@=
Gibt die Adresse eines Feldes an, das die Elementversion des mit SYMBOL@ oder SYMBOL vereinbarten Elements enthält.
Für SYMTYP=ANY wird der Operand VERS@ nur berücksichtigt, wenn sich der Objektname (SYMBOL@ oder SYMBOL) auf ein Modul in einer Programmbibliothek (Typ R oder Typ L) bezieht. Bei Angabe eines CSECT- oder ENTRY-Namens wird der Operand VERS@ ignoriert. Fehlt der Operand, wird der Standardwert für die höchste Elementversion bei Programmbibliotheken übernommen (siehe Handbuch „LMS“ [29]).

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.
Angabe nur mit MF=M.

(r)
r = Register mit der gesuchten Feldadresse. Angabe nur mit MF=M.

label
Symbolische Adresse des Feldes. Angabe nur mit MF=S oder MF=L.

VERS=version
Angabe nur mit MF=S oder MF=L
Explizite Angabe der Elementversion. Der Name darf maximal 24 Zeichen lang sein.

XPAND=
Angabe nur mit MF=D.
Bestimmt, das Layout des zu generierenden Datenbereichs.

PARAM
Generiert das Layout der Parameterliste für den Aufruf des BIND-Makros.

XRC
Generiert das Layout für den erweiterten Returncode.

USRMAP
Angabe nur zusammen mit INTVERS=SRVxxx und xxx ≥ 005.

Generiert das Layout für den Datenbereich, in den die DBL-Liste ausgegeben werden kann (siehe Operanden USRMAPI, USRMAP@ und USRMAPL).

USRUNR
Angabe nur zusammen mit INTVERS=SRVxxx und xxx ≥ 005.

Generiert das Layout für den Datenbereich, in den eine Liste der nicht befriedigten Externverweise ausgegeben werden kann (siehe Operanden USRUNR@, USRUNRI und USRUNRL).

XRC=
Bestimmt die Adresse eines Feldes, das den **erweiterten Returncode** enthält.
Die Adresse muss auf Wortgrenze ausgerichtet sein. Das Feld hat folgenden Aufbau:

Byte	Länge	Inhalt
0- 6	7	Meldungsschlüssel der letzten ausgegebenen Meldung während der Abarbeitung des BIND-Makros
7	1	Leerzeichen
8-11	4	DVS-Fehlerschlüssel
12-23	12	PLAM-Fehlerschlüssel
24-27	4	Fehlerschlüssel für sonstige Fehler beim Suchen in Bibliotheken
28	1	ILE-Flag und 7 reservierte Bits
29-31	3	reserviert
32	4	Adresse des ILE-Servermoduls

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.
Angabe nur mit MF=M.

(r)
r = Register mit der gesuchten Feldadresse. Angabe nur mit MF=M.

label
Symbolische Adresse des Feldes. Angabe nur mit MF=S oder MF=L.

XRCL=
Bestimmt die Länge des bei XRC angegebenen Feldes. In Abhängigkeit von dieser Länge übergibt DBL alle oder nur einen Teil der möglichen Informationen.

28
Das XRC-Feld ist 28 Byte lang. Informationen über ILEs werden nicht übergeben.

36
Das XRC-Feld ist 36 Byte lang. Alle Informationen einschließlich ILE-Informationen werden übergeben.

Hinweise zum Makroaufruf

- Der DBL lädt Module oder Programmabschnitte (CSECTs), die bereits im selben Link-Kontext nachgeladen wurden, nicht noch einmal. Bei BRANCH=YES verzweigt der DBL an die Adresse des Symbols (CSECT oder ENTRY), das geladen wurde. Bei BRANCH=NO wird im aufrufenden Programm fortgesetzt.
- CSECTs bzw. COMMON-Bereiche, deren Namen aus Leerzeichen bestehen, erhalten als Standardnamen „%CSECT“ bzw. „%COM“.
- Kann ein LLM, der keine Relativierungsinformation enthält, nicht an eine passende Adresse geladen werden, tritt ein Fehler auf.
- Beim Laden der Ladeeinheit versucht der DBL, nicht befriedigte Externverweise einer vorhergehenden Ladeeinheit, die im Link-Kontext gespeichert wurden (Operand UNRES=DELAY) für die aktuelle Ladeeinheit zu befriedigen.
- Als OPEN-Modus für die Programmbibliotheken und Elemente wird der Wert angenommen, der im Systemparameter BLSOPENX festgelegt ist
- Die Operanden ALTLIB=YES und LIBNAM@/LIBNAM/LIBLINK dürfen zusammen angegeben werden. Ist jedoch die mit den Operanden LIBNAM@/LIBNAM/LIBLINK angegebene Hauptbibliothek nicht vorhanden, bricht der DBL die Verarbeitung ab. Die Angabe ALTLIB=YES dient nicht als Ersatz für die fehlende oder ungültige Hauptbibliothek.
- Wenn die angegebene Programmversion noch nicht geladen ist, aber ein Programm mit diesem Namen im Link-Kontext (siehe Operand CONTEXT) bereits vorhanden ist, wird das Laden wegen Namenskonflikt abgewiesen.
- Bei Angabe einer REP-Datei (REPFILE, REPFIL@) kann DBL auch eine zugehörige NOREF-Datei verarbeiten, wenn die Namenskonvention für BS2000-REP-Dateien eingehalten werden: Entweder enthält der NOREF-Datei-Name den Namensteil „SYSNRF“ (an Stelle von SYSREP) oder die NOREF-Datei hat den gleichen Namen wie die Hauptbibliothek mit dem Suffix „.NOREF“. NOREF-Dateien müssen in der Benutzerumgebung das gleiche Format wie in der Systemumgebung haben.
- Bei Angabe von `BIND MF=D,XPAND=USRMAP` muss dasselbe Präfix angegeben werden wie bei `BIND MF=D,XPAND=PARAM`.
- Wenn der benutzerdefinierte Ausgabebereich für die DBL-Liste (Operanden USRMAPI, USRMAP@ und USRMAPL) für die Aufnahme der vollständigen Information zu klein ist, wird ein Returncode (X’08010129’) in den Kopf der Liste eingetragen. Die Ausgabe wird abgebrochen, aber der Ladevorgang wird fortgesetzt.
- Die Liste nicht befriedigt verbleibender Externverweise oder aus früheren Ladevorgängen (mit UNRES=DELAY) nicht befriedigt verbliebenen Externverweisen (sofern vorhanden) wird in den mit USRUNR@, USRUNRI und USRUNRL festgelegten Bereich ausgegeben. Diese Ausgabe erfolgt unabhängig vom Operanden UNRES. Wenn der

Bereich für die Aufnahme der vollständigen Information zu klein ist, wird ein Returncode (X'0801012D') in den Kopf der Liste eingetragen. Die Ausgabe wird abgebrochen, aber der Ladevorgang wird fortgesetzt. Wenn alle Externverweise befriedigt wurden, wird in den Kopf der Liste als Anzahl 0 eingetragen.

– Bei Angabe von `SYMBOL=*ALL` oder `SYMBOL=name`, wobei das letzte Zeichen von `name` das Wildcard-Symbol „*“ ist, entsteht beim Laden eine so genannte List-Name-Unit. Nähere Informationen dazu finden Sie im Handbuch „BLSSERV“ [4].

Aufbau der Namensstruktur im Datenbereich des BIND-Makros

Die einzelnen Namen, die im Makroaufruf **BIND** definiert wurden, verwaltet der DBL mit Hilfe des Datenbereichs des **BIND**-Makros. Der Datenbereich enthält für jeden Namen einen Zeiger, der auf die Adresse des zugehörigen Namenfeldes verweist (siehe folgendes Bild). Außer dem Namen SYMBOL@ sind alle Namen wahlfrei. Wird ein Name im Makroaufruf nicht definiert, setzt der Makro den zugehörigen Zeiger auf den Adresswert X'FFFFFFFF'.

Ein Name kann auch explizit im Makroaufruf angegeben werden. In diesem Fall werden die Namen in den Datenbereich eingetragen.

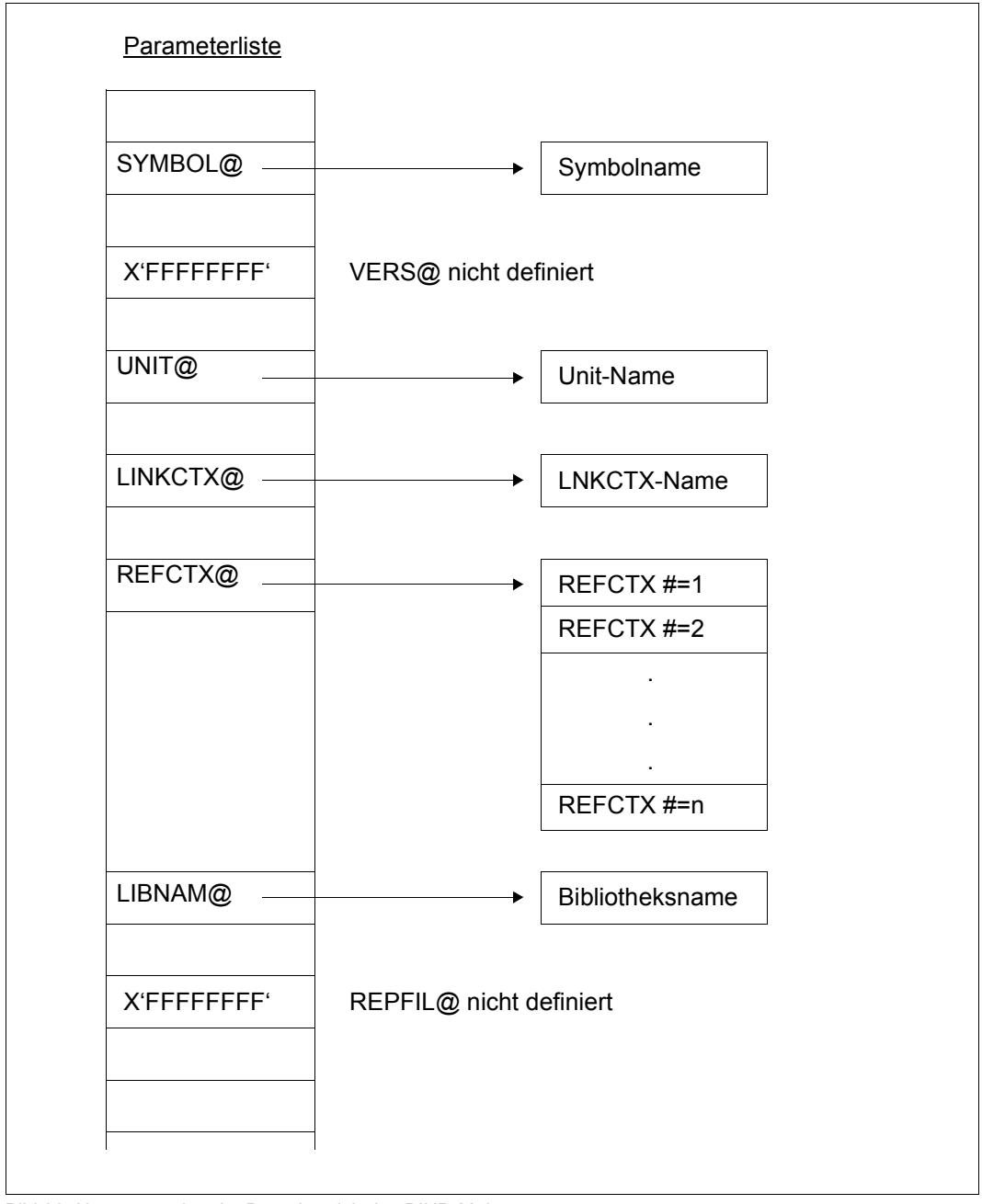


Bild 23: Namensstruktur im Datenbereich des BIND-Makros

Suchstrategie

Der DBL sucht in verschiedenen „Behältern“ nach dem mit SYMBOL@ und SYMTYP vereinbarten Objekt. Wird ein passendes Objekt in einem Behälter gefunden, wird es in die Ladeeinheit eingefügt, die Ladeeinheit wird geladen und die Startadresse an den Benutzer im Feld SYMBLAD übergeben.
Der Suchvorgang verläuft in folgenden Stufen:

1. Suchen im Link-Kontext. Der Referenz-Kontext wird nicht durchsucht.
2. Suchen im Shared Code des Benutzers, der mit dem **ASHARE**-Makro des DBL in einen Common Memory Pool geladen wurde. Die Suche kann auf Memory Pools mit einem definierten Geltungsbereich eingeschränkt oder ganz unterdrückt werden (Operand SHARE-SCOPE).
3. Suchen im Shared Code im Systemadressraum, in den die nichtprivilegierten Subsysteme geladen sind (siehe Handbuch „Verwaltung von Subsystemen“ [12]). Das Durchsuchen der nichtprivilegierten Subsysteme kann der Benutzer im Ladeaufruf unterdrücken, wenn er für SHARE-SCOPE ≠ SYSTEM angibt.
4. Durchsuchen von Bibliotheken, die der Benutzer im Ladeaufruf mit dem Operanden LIBNAM@ oder LIBNAM oder LIBLINK angegeben hat.
5. Durchsuchen von alternativen (System-)Bibliotheken, die mit dem Dateikettungsnamen BLSLIBnn (00≤nn≤99) oder $BLSLBnn zugewiesen wurden. Die Bibliotheken werden nach aufsteigenden Nummern „nn“ durchsucht. Zuerst werden die alternativen Systembibliotheken $BLSLB00..49 durchsucht, danach die alternativen Bibliotheken BLSLIB00..99 und zuletzt die restlichen alternativen Systembibliotheken $BLSLB50..99.

   Abhängig vom Operanden ALTLIB können auch die System- und/oder Benutzer-Tasklib durchsucht werden.

   Das Suchen in alternativen Bibliotheken kann vom Benutzer im Ladeaufruf mit dem Operanden ALTLIB=NO unterdrückt werden.

Hinweis

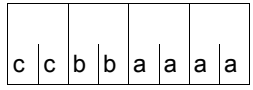
Bei List-Name-Units findet keine Suche nach Namen im Link-Kontext oder Shared Code statt, da es sich bei den Namen um Elementnamen und nicht um Symbolnamen handelt.

Nähere Informationen dazu finden Sie im Handbuch „BLSSERV“ [4].

Rückinformation und Fehleranzeigen

Die Startadresse der Ladeeinheit wird in dem Feld übergeben, das mit dem Operanden SYMBLAD festgelegt wurde. Das höchstwertige Bit der übergebenen Startadresse zeigt an, welcher Adressierungsmodus eingestellt werden muss
(Bit = 1 für 31-Bit-Adressierung, Bit = 0 für 24-Bit-Adressierung).

Standard-header:



Über die Ausführung des Makros BIND wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Der Makro wurde normal ausgeführt.
X'0C'	X'01'	X'0018'	Ein reserviertes Feld des Datenbereichs ist nicht mit Nullen vorbelegt.
X'0C'	X'01'	X'0100'	Unzulässige Kombination von Parametern in der Parameterliste. Unzulässige Kombinationen können z.B. sein: – OVERLAY=YES wurde ohne LOAD@ angegeben. – LDINFO=NONE bei gleichzeitiger Angabe von TEST-OPTIONS im Kommando LOAD-EXECUTABLE-PROGRAM oder LOAD-PROGRAM – RESTYP=USER wurde ohne RESORD angegeben – USRMAPI=STD/ALL wurde ohne USRMAP@ oder USRMAPL angegeben
X'0C'	X'01'	X'0104'	Der Aufruf ist ungültig, da er unter einer Benutzerkennung gegeben wurde, die nur für die Wartung erlaubt ist.
X'0C'	X'40'	X'0114'	LNKCTXS=OLD wurde gegeben und der angegebene Kontext ist nicht vorhanden.
X'0C'	X'40'	X'0118'	LNKCTXS=NEW wurde gegeben und der angegebene Kontext ist vorhanden.
X'0C'	X'40'	X'011C'	Ein Kontext, der mit REFCTX@ angegeben wurde, ist nicht vorhanden.
X'0C'	X'01'	X'0120'	– Ein Ausgabebereich hat nur Lesezugriff, ist nicht zugewiesen oder nicht auf Wortgrenze ausgerichtet, – MPID ist nicht zugewiesen oder nicht auf Wortgrenze ausgerichtet.
X'0C'	X'01'	X'0121'	Ungültige Parameterliste: Einer der Eingabeparameter ist falsch
X'0C'	X'01'	X'0124'	Ein Bereich, der einen Namen enthalten soll, ist nicht zugewiesen. Der Name kann sein: – der Objektname (SYMBOL) – der Name der Ladeeinheit (UNIT) – der Versionsname (VERS) – der Bibliotheksname (LIBNAM) – der Dateiname der REP-Datei (REPFILE) – der Link-Kontext-Name (LNKCTX) – ein Name in der Liste der Referenz-Kontext-Namen (REFCTX).

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'0C'	X'01'	X'0125'	Die Anzahl der Referenz-Kontext-Namen ist zu groß. Maximal 200 Namen sind erlaubt.
X'0C'	X'01'	X'0127'	Der obligatorische Operand SYMBLAD wurde nicht angegeben.
X'08'	X'01'	X'0129'	Die mit USRMAPL angegebene Länge ist zu klein. Die Ausgabe der DBL-Liste in einen benutzerdefinierten Datenbereich wird ohne End-Satz abgebrochen. Das Laden wird fortgesetzt. Im Kopf des Datenbereichs wird dieser Returncode ausgegeben
X'0C'	X'01'	X'012A'	Die mit USRMAPL angegebene Länge ist zu klein.
X'08'	X'01'	X'012D'	Die mit USRUNRL angegebene Länge ist zu klein. Die Ausgabe der Liste nicht befriedigt verbleibender Externverweise oder aus früheren Ladevorgängen (mit UNRES=DELAY) nicht befriedigt verbliebenen Externverweisenin einen benutzerdefinierten Datenbereich wird abgebrochen. Das Laden wird fortgesetzt. Im Kopf des Datenbereichs wird dieser Returncode ausgegeben
X'0C'	X'01'	X'012E'	Die mit USRUNRL angegebene Länge ist zu klein.
X'0C'	X'01'	X'0134'	Unzulässiger Operand LNKCTXS.
X'0C'	X'01'	X'0144'	Ungültiger Name für einen Link-Kontext.
X'0C'	X'01'	X'0146'	Ein Referenz-Kontext hat den gleichen Namen wie der Link-Kontext.
X'0C'	X'01'	X'0148'	Ein Link- oder Referenz-Kontext kann nicht benutzt werden.
X'0C'	X'01'	X'014C'	Ein ungültiger Symbolname aus Leerzeichen wurde angegeben.
X'0C'	X'01'	X'014D'	Es wurde kein Symbolname angegeben.
X'0C'	X'01'	X'0150'	Unzulässige Angaben im Operanden SYMTYP. Ein Objekttyp kann sein: CSECT (1) / ENTRY (2) / CSEN (3) / MODULE (4) / ANY (5). Eine von diesen Objekttypen abweichende Angabe wurde gemacht.
X'0C'	X'20'	X'0158'	Maximale Anzahl von 16 Benutzerkontexten ist erreicht. Ein neuer Kontext kann nicht mehr erzeugt werden.
X'0C'	X'40'	X'016C'	Interner Fehler der AID-Testroutine (Systemfehler).
X'0C'	X'40'	X'0184'	Ungültige Kurzkennung bei MPID weil: – Memory Pool (MP) war nicht eingerichtet oder der Benutzer nicht an den MP angeschlossen – der MP befindet sich nicht im Klasse-6-Speicher – der MP wird bereits für Shared Code des Benutzers verwendet.
X'0C'	X'01'	X'0188'	Ungültige Ladeadresse bei LOAD@.
X'0C'	X'40'	X'018C'	Es wurde versucht, einen Kontext zu benutzen, der durch einen vorhergehenden Fehler verfälscht wurde.

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'0C'	X'40'	X'0190'	Ein unzulässiger Residenz-Modus (RMODE) wurde für die Ladeeinheit vom DBL festgelegt. Dies kann beim Laden von Modulen oder COMMON-Bereichen auftreten. Mögliche Ursache beim Laden eines Moduls: – Der festgelegte Pseudo-RMODE des Moduls ist 24 und die Ladeeinheit wird in einen Memory Pool (MP) oberhalb 16 MByte geladen. Mögliche Ursachen beim Laden eines COMMON-Bereiches: 1. Der COMMON-Bereich soll unterhalb 16 MByte geladen werden und die Ladeeinheit wird in einen MP oberhalb 16 MByte geladen. Dies ist der Fall, wenn: – der COMMON-Bereich RMODE 24 hat oder – die Ladeeinheit den PROGMOD 24 oder – mindestens ein Modul der Ladeeinheit unterhalb 16 MByte geladen wird. 2. Der COMMON-Bereich wurde im selben Kontext in einer vorhergehenden Ladeeinheit oberhalb 16 MByte geladen und soll in der aktuellen Ladeeinheit unterhalb 16 MByte geladen werden
X'0C'	X'01'	X'0194'	Ein Klasse-6-Speicherbereich, den der DBL angefordert hat, wurde vom Benutzer freigegeben oder die Attribute von Speicherseiten wurden durch den Benutzer verändert. Bei der Gültigkeitsprüfung des freigegebenen Bereichs trat ein Fehler auf.
X'0C'	X'20'	X'0196'	Verbindungsfehler bei einem Common Memory Pool. Die Task kann nicht an den Memory Pool angeschlossen werden, weil z.B. ein Teil des Speicherbereiches für den Memory Pool bereits von der aktuellen Task angefordert wurde.
X'0C'	X'20'	X'0198'	Für das Laden der vom Benutzer angegebenen Objekte steht nicht genügend Speicherplatz zur Verfügung.
X'0C'	X'20'	X'01A0'	Auf ein Betriebsmittel kann nicht zugegriffen werden. Es ist gesperrt.
X'0C'	X'40'	X'0200'	Fehler während DBL-Ablauf (Systemfehler).
X'0C'	X'40'	X'0204'	Inkonsistenzen in den Memory Management Tabellen der DBL (Systemfehler).
X'0C'	X'40'	X'0208'	Inkonsistenzen in den Data Management Tabellen der DBL (Systemfehler).
X'0C'	X'40'	X'020C'	Inkonsistenzen in den symb. Informationstabellen (Systemfehler).
X'0C'	X'20'	X'0300'	Fehler bei $REQM, $RELM, $CSTAT, RDTFT (Systemfehler).
X'0C'	X'40'	X'0400'	Fehlerhafte Objektsätze bei der Verarbeitung eines Moduls (OM oder LLM) entdeckt.
X'0C'	X'40'	X'0404'	Ungültiger Typcode im ESD/ESV-Satz.
X'0C'	X'40'	X'0408'	Inkonsistenzen im Eingabemodul (z.B. Reihenfolge der ESD/ESV-Sätze).

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'0C'	X'40'	X'040C'	Das angegebene Modul ist nicht in der aktuellen Ladeeinheit. Fehler im RLD/LRLD-Satz.
X'0C'	X'40'	X'0410'	Fehler im TXT-Satz.
X'0C'	X'40'	X'0414'	Fehler bei der Verarbeitung von symbolischen Informationen.
X'04'	X'40'	X'0418'	Fehlerhafter REP-Satz. Die Verarbeitung wird fortgesetzt (auf die Meldung BLS0230 wurde im Dialogmodus mit YES geantwortet).
X'08'	X'40'	X'0418'	Fehlerhafter REP-Satz. Die Verarbeitung wird fortgesetzt (Batch-Modus).
X'0C'	X'40'	X'0418'	Fehlerhafter REP-Satz. Die Verarbeitung wurde abgebrochen (auf die Meldung BLS0230 wurde im Dialogmodus mit NO geantwortet).
X'04'	X'40'	X'041C'	Das angegebene Modul ist nicht in der aktuellen Ladeeinheit. Der REP-Satz wird ignoriert. Die Verarbeitung wird fortgesetzt. (Die Meldung BLS0234 wurde mit YES beantwortet.)
X'0C'	X'40'	X'041C'	Das angegebene Modul ist nicht in der aktuellen Ladeeinheit. Der REP-Satz kann nicht verarbeitet werden. Die Verarbeitung wurde abgebrochen. (Die Meldung BLS0234 wurde mit NO beantwortet.)
X'04'	X'40'	X'0420'	Fehler in der INCLUDE-Anweisung. Die Verarbeitung wird fortgesetzt (die Meldung BLS0230 wurde mit YES beantwortet).
X'0C'	X'40'	X'0420'	Fehler in der INCLUDE-Anweisung. Die Verarbeitung wurde abgebrochen.
X'0C'	X'40'	X'0430'	Das LLM kann nicht geladen werden. Mögliche Gründe: – Es ist mit der aktuellen BLSSERV-Version nicht ladbar. – Es kann nur ein LLM mit benutzerdefinierten Slices in einem Kontext geladen werden, und es ist bereits eines vorhanden. – LLM kann nur durch BINDER und LLMAM verarbeitet werden. – LLM mit CSECT AMODE=31 kann nicht mit PROGMOD=24 geladen werden. – Ein LLM mit benutzerdefinierten Slices kann nicht in eine List-Name-Unit geladen werden.
X'0C'	X'40'	X'0432'	Eine Adresse, die aus einer Relativierung hervorgeht, wurde abgeschnitten. Die Verarbeitung wird fortgesetzt.
X'0C'	X'40'	X'0433'	Eine Adresse, die aus einem Befehl resultiert, liegt außerhalb des aktuellen Segments. Die Verarbeitung wird abgebrochen.
X'0C'	X'40'	X'0434'	Dieses LLM ist nicht mit der „PRE-LOADING“-Option vereinbar und Vorladen wurde angefordert. Die Verarbeitung wird abgebrochen.
X'0C'	X'40'	X'0435'	An DSSM übergebene Länge des vorladbaren Teils stimmt nicht mit der tatsächlichen Länge des vorladbaren LLM-Teils überein.

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'04'	X'01'	X'0600'	Das in einer INCLUDE-Anweisung angegebene Modul kann nicht gefunden werden. Die Verarbeitung wird fortgesetzt. (Die Meldung BLS0232 wurde mit YES beantwortet.)
X'0C'	X'01'	X'0600'	Das in einer INCLUDE-Anweisung angegebene Modul kann nicht gefunden werden. Die Verarbeitung wurde abgebrochen. (Die Meldung BLS0232 wurde mit NO beantwortet.) oder Das mit SYMBOL@ oder SYMBOL festgelegte Modul kann in den angegebenen Bibliotheken nicht gefunden werden.
X'04'	X'01'	X'0604'	Ein Namenskonflikt ist aufgetreten und wurde akzeptiert.
X'0C'	X'01'	X'0604'	Ein Namenskonflikt ist aufgetreten. Die Verarbeitung wurde abgebrochen.
X'04'	X'01'	X'0608'	Externverweise können nicht befriedigt werden. Die Verarbeitung wird fortgesetzt. (Die Meldung BLS0350 wurde mit YES beantwortet.)
X'08'	X'01'	X'0608'	Nicht befredigte Externverweise verbleiben im Kontext und werden nach Möglichkeit später verarbeitet. Die Verarbeitung wird fortgesetzt. (Returncode tritt nur bei `UNRES=DELAYWARN` auf.)
X'0C'	X'01'	X'0608'	Externverweise können nicht befriedigt werden. Die Verarbeitung wurde abgebrochen. (Die Meldung BLS0350 wurde mit NO beantwortet.)
X'04'	X'01'	X'0609'	XDSECs-R können nicht befriedigt werden. Die Verarbeitung wird fortgesetzt.
X'0C'	X'01'	X'0609'	XDSECs-R können nicht befriedigt werden. Die Verarbeitung wurde abgebrochen.
X'04'	X'40'	X'060C'	Es wurde LOAD=NO angegeben, aber das Modul ist nicht geladen.
X'0C'	X'40'	X'060D'	Das Symbol ist im LNKCTX bereits geladen. Es wird nicht noch einmal geladen, um Namenskonflikte zu vermeiden.
X'0C'	X'01'	X'0610'	Die angegebene Datei ist keine Programmbibliothek oder Bindemodulbibliothek (OML).
X'0C'	X'01'	X'0614'	Die angegebene Datei ist keine gültige Bindemodulbibliothek (OML).
X'0C'	X'01'	X'0618'	Die Bibliotheksroutine PLAM ist nicht verfügbar
X'0C'	X'01'	X'061C'	PAM-Lesefehler in der Bibliothek.
X'0C'	X'01'	X'0620'	Fehler im OML-Directory.
X'0C'	X'40'	X'0624'	Die Ladeeinheit enthält eine CSECT mit AMODE 31 und: – die Hardware erlaubt keine 31-Bit-Adressierung – PROGMOD=24 wurde angegeben oder – das Modul mit dem angegebenen Symbol wurde bereits oberhalb 16 MByte geladen und PROGMOD=24 wurde angegeben

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'04'	X'40'	X'0628'	Der aktuelle Adressierungsmodus entspricht nicht dem Adressierungsmodus, der für die Einsprungstelle genutzt werden muss (z.B. wenn das aufgerufene Programm mit AMODE=31 abläuft und die Einsprungstelle AMODE=24 hat). Der Adressierungsmodus muss umgeschaltet werden. Der DBL übergibt im Feld SYMBLAD dennoch die richtige Startadresse.
X'0C'	X'40'	X'062C'	Es wurde versucht, ein Modul mit RMODE=24 in einen Memory Pool oberhalb 16 MByte zu laden.
X'0C'	X'01'	X'0630'	Ein LLM mit Slices, die ober- und unterhalb 16 MByte geladen werden, kann nicht in einen Memory Pool geladen werden.
X'0C'	X'01'	X'0634'	Ein LLM mit Slices, die vom Benutzer definiert wurden, kann nicht in einen Memory Pool geladen werden.
X'0C'	X'01'	X'0636'	Die HSI-Codes von ILE und Servermodul sind inkompatibel. Das Servermodul wurde nicht geladen.
X'00'	X'01'	X'FFFF'	Die Funktion wird nicht mehr oder noch nicht unterstützt.
X'00'	X'03'	X'FFFF'	Die Version der Schnittstelle wird nicht unterstützt.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Beispiel

Während des Programmlaufs von BIND1 wird mit dem Makro **BIND** ein zweiter Programmabschnitt BIND2 nachgeladen. BIND2 steht als LLM in der Bibliothek MACEXMP.LIB. Beide Programmabschnitte sollen im 31-Bit-Adressierungsmodus ablaufen. BIND1 soll unterhalb und BIND2 oberhalb der 16MB-Grenze geladen werden. Nach Aufruf des Makros **BIND** soll zuerst BIND2 ablaufen.

Programm BIND1

```
BIND1    START
         PRINT NOGEN
BIND1    AMODE 31 ------------------------------------------------------  (1)
BIND1    RMODE 24
         BALR  3,0
         USING *,3
         USING BINDSECT,6 ---------------------------------------------  (2)
         ST    3,AREA11
         UNPK  AREAH,AREA1
         MVC   AREAA(8),AREAH
WROUT1   WROUT OUT,ERROR,PARMOD=31 ----------------------------------  (3)
BACK     LA    12,MVC
BIND     BIND  MF=E,PARAM=BINDPAR -------------------------------------  (4)
         LA    6,BINDPAR
         CLC   XBINRET,=X'00000000' -----------------------------------  (5)
         BE    MVC
         MVC   OUT+5(26),='BIND error!               '
         WROUT OUT,ERROR,PARMOD=31
         B     ERROR
MVC      MVC   OUT+5(26),='Back in BIND1             '
         WROUT OUT,ERROR,PARMOD=31 ------------------------------------  (6)
ERROR    TERM
OUT      DC    Y(OUTE-OUT)
         DS    CL3
         DC    C'BIND1: BASEREG.= '
AREAA    DS    CL8
OUTE     EQU   *
AREA     DS    0F
AREA1    DS    0CL5
AREA11   DS    CL4
AREA12   DC    C'0'
         DS    0F
AREAH    DS    CL9
BINDPAR  BIND  MF=L,SYMBOL=BIND2,SYMBLAD=BIND2@,BRANCH=YES,PROGMOD=ANY,*
               LIBLINK=PLAMLIB,MAP=SYSOUT
BIND2@   DS    A
BINDSECT BIND  MF=D,PREFIX=X ---------------------------------------  (7)
         END
```

Programm BIND2

```
BIND2    CSECT ------------------------------------------------------------ (8)
         PRINT NOGEN
BIND2    AMODE ANY  ------------------------------------------------------- (9)
BIND2    RMODE ANY
         BALR  4,0
         USING *,4
         ST    4,AREA11
         UNPK  AREAH,AREA1
         MVC   AREAA(8),AREAH
         WROUT OUT,ERROR,PARMOD=31 ---------------------------------------- (10)
         BR    12
ERROR    TERM
OUT      DC    Y(OUTE-OUT)
         DS    CL3
         DC    C'BIND2: BASEREG.= '
AREAA    DS    CL8
OUTE     EQU   *
AREA     DS    0F
AREA1    DS    0CL5
AREA11   DS    CL4
AREA12   DC    C'0'
AREAH    DS    CL9
         END
```

(1) Für Programmabschnitt BIND1 wird das Attribut AMODE=31 vereinbart. Mit RMODE=24 wird BIND1 immer unterhalb der 16MB-Grenze geladen.

(2) Register R6 wird dem Assembler als Basisadressregister zur Adressierung der DSECT für den Datenbereich des **BIND**-Makros zugewiesen, die an der symbolischen Adresse BINDSECT durch einen **BIND**-Aufruf mit MF=D erzeugt wird.

(3) Der Inhalt des Basisregisters von BIND1 wird zur Darstellung des Adressierungsmodus und der Ladeadresse ausgegeben.

(4) An der symbolischen Adresse BIND wird der Makro **BIND** in seiner E-Form aufgerufen. An dieser Stelle im Programm wird daher nur der Befehlsteil erzeugt. Der zughörige Datenbereich wird an der symbolischen Adresse BINDPAR durch einen **BIND**-Aufruf mit MF=L angelegt. Die dort angegebenen Operandenwerte veranlassen den **BIND**-Makro, bei der Programmausführung
- die CSECT BIND2 (SYMBOL=BIND2,SYMTYP=CSECT) aus der mit dem Linknamen PLAMLIB zugewiesenen Bibliothek (LIBLINK=PLAMLIB) nachzuladen,
- die Startadresse von BIND2 im Feld BIND2@ zu hinterlegen (SYMBLAD=BIND2@),
- für BIND2 den 31-Bit-Adressierungsmodus einzustellen (PROGMOD=ANY),
- ein DBL-Protokoll nach SYSOUT auszugeben (MAP=SYSOUT) und

– nach dem Laden von BIND2 den Programmlauf in BIND2 fortzusetzen (BRANCH=YES).

(5) Nach der Ausführung des **BIND**-Makros wird geprüft, ob das Feld XBINRET des Standardheaders den Returncode X'00000000' enthält, der eine fehlerfreie Makroausführung anzeigt. Der Name XBINRET stammt aus der DSECT, die unter der symbolischen Adresse BINDSECT durch einen **BIND**-Aufruf mit MF=D und PREFIX=X erzeugt wurde (siehe (7)). Diese DSECT beschreibt den Aufbau des Datenbereichs des **BIND**-Makros. Die symbolischen Namen der DSECT können zur Adressierung innerhalb des Datenbereichs verwendet werden, nachdem das zugeordnete Basisadressregister (hier Register R6) mit der Anfangsadresse des Datenbereichs (hier BINDPAR) geladen worden ist.

(6) Eine Meldung nach SYSOUT informiert darüber, dass nach dem Ablauf von BIND2 die Programmausführung in BIND1 fortgesetzt wird.

(7) Der Makroaufruf **BIND** mit MF=D erzeugt eine DSECT, die den Aufbau des Datenbereichs des **BIND**-Makros beschreibt. Der Operand PREFIX=X bewirkt, dass alle symbolischen Namen in dieser DSECT (Feldnamen und Equates) mit dem Buchstaben X beginnen.

(8) Die CSECT-Anweisung definiert den Programmabschnitt BIND2.

(9) AMODE=ANY zeigt dem Betriebssystem an, dass BIND2 sowohl im 24-Bit- als auch im 31-Bit-Adressierungsmodus ablaufen kann.

(10) Der Inhalt des Basisregisters von BIND2 wird zur Darstellung des Adressierungsmodus und der Ladeadresse ausgegeben.

Ablaufprotokoll

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,bind1), - ————————————— (11)
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,bind1))
%  ASS6011 ASSEMBLY TIME: 557 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 84 MSEC
//compile source=*library-element(macexmp.lib,bind2), - ————————————— (12)
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,bind2))
%  ASS6011 ASSEMBLY TIME: 270 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
```

```
%  ASS6006 LISTING GENERATOR TIME: 82 MSEC
//end
%  ASS6012 END OF ASSEMBH
/add-file-link link-name=plamlib,file-name=macexmp.lib ——————————— (13)
/start-executable-program library=macexmp.lib, - ————————————————— (14)
/      element-or-symbol=bind1,prog-mode=*any
%  BLS0523 ELEMENT 'BIND1', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'BIND1', VERSION ' ' OF '<date> <time>' LOADED
BIND1: BASEREG.= 80000002 ———————————————————————————————————— (15)
———————————————————————————————————————————————————————————— (16)
############################## A D B L   M A P ##############################
#                                                                            #
# LOAD UNIT: BIND2. . . . . . . . . . . . . LOAD INFO  =DEF                  #
# VERSION  : ~                              LOAD TIME  =<date> <time>        #
# CONTEXT  : LOCAL#DEFAULT. . . . . . . . . TEST OPTION=NONE                 #
#                                                                            #
#  LLM     : BIND2. . . . . . . . . . . . . BY_ATTR  / EXPLICIT  PLAMLIB    #
#   OM     : BIND2. . . . . . . . . . . . . STANDARD                        #
#    CSECT : BIND2. . . . . . . . . . . . . @= 1000000 L=      7E           #
#                                                                            #
# LOAD UNIT STARTING POINT. . . . . . . . . @= 1000000 AMODE=31 HSI=/7500 #
#                                                                            #
######################## E N D   O F   A D B L   M A P ########################
BIND2: BASEREG.= 81000002 ———————————————————————————————————— (17)
Back in BIND1 ————————————————————————————————————————————— (18)
```

(11) Das Programm BIND1 wird übersetzt.

(12) Das Programm BIND2 wird übersetzt.

(13) Der im BIND-Aufruf (4) verwendete Dateikettungsname wird zugewiesen.

(14) Der DBL wird aufgerufen, um das Programm zu binden, zu laden und zu starten.

(15) Der Inhalt des Basisregisters von BIND1 wird ausgegeben. 31-Bit-Adressierung ist eingestellt (Bit 2^{31} = 1); die Ladeadresse liegt unterhalb der 16Mbyte-Grenze.

(16) Der DBL hat die CSECT BIND2 nachgeladen. Er gibt ein Protokoll nach SYSOUT aus.

(17) Der Inhalt des Basisregisters von BIND2 wird ausgegeben. 31-Bit-Adressierung ist eingestellt (Bit 2^{31} = 1); die Ladeadresse liegt oberhalb der 16Mbyte-Grenze.

(18) Nach der Rückkehr aus BIND2 wird der Programmlauf in BIND1 fortgesetzt.

BKPT – Programmlauf unterbrechen

Allgemeines

Anwendungsgebiet: Testhilfe; siehe Seite 166
Unterbrechen des Programmablaufs; siehe Seite 72
Kommunikation; siehe Seite 167
Makrotyp: O-Typ; siehe Seite 28

Makrobeschreibung

Mit dem Makro **BKPT** kann man den Programmablauf unterbrechen, um Kommandos einzugeben.
Im Dialogbetrieb kann man an der Datenstation System-Kommandos oder Testhilfe-Kommandos eingeben. Das Programm wird mit dem Kommando RESUME-PROGRAM oder %RESUME bzw. %TRACE fortgesetzt. Bei Batch- und Dialogprozeduren werden Kommandos bis zum nächsten RESUME-PROGRAM- oder %RESUME- bzw. %TRACE-Kommando verarbeitet.

Makroaufrufformat und Operandenbeschreibung

BKPT

Beispiel

```
BKPT     START
         PRINT NOGEN
         BALR  3,0
         USING *,3
         WROUT MESS1,ERROR ———————————————————————————————————  (1)
         BKPT ————————————————————————————————————————————————  (2)
         WROUT MESS1,ERROR ———————————————————————————————————  (3)
ERROR    TERM
***  Definitions   ****
MESS1    DC    Y(END-MESS1)
         DS    L2
         DC    X'01'
         DC    C'Here is BKPT'
END      EQU   *
         END
```

Ablaufprotokoll:

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,bkpt), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,bkpt))
%  ASS6011 ASSEMBLY TIME: 314 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 80 MSEC
//end
%  ASS6012 END OF ASSEMBH
/start-executable-program library=macexmp.lib,element-or-symbol=bkpt
%  BLS0523 ELEMENT 'BKPT', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'BKPT', VERSION ' ' OF '<date> <time>' LOADED
Here is BKPT ------------------------------------------------------------  (1)
%  IDA0199 PROGRAM BREAK AT ADDRESS X'000014', AMODE=24 ----------------  (2)
/show-user-status inf=*prog
NAME      TSN TYPE        SIZE CURR-CMD
MACTEST  2QSE 3 DIALOG1      1 SHOW-USER-STATUS
              PROG::2OSG:$QM212.MACEXMP.LIB(BKPT,@,L)
/resume-program
Here is BKPT ------------------------------------------------------------  (3)
```

(1) Es wird eine Nachricht nach SYSOUT geschrieben.

(2) Der Makro **BKPT** unterbricht den Programmablauf. Die Meldung `IDA0199` wird ausgegeben.

Nun können an der Datenstation Kommandos eingegeben werden. Im Beispiel wird das Kommando SHOW-USER-STATUS INF=*PROG eingegeben. Mit dem Kommando RESUME-PROGRAM wird das unterbrochene Programm an der dem Makro **BKPT** folgenden Zeile fortgesetzt.

(3) Es wird eine Nachricht nach SYSOUT geschrieben.

CALL – Segmente laden

Allgemeines

Anwendungsgebiet: Binden und Laden; siehe Seite 47
Makrotyp: O-Typ; siehe Seite 28

Makrobeschreibung

Mit dem Makroaufruf **CALL** ist es möglich, über ein Symbol ein Segment zu laden, sofern es noch nicht im Speicher steht. Weitere Segmente innerhalb desselben Astes der Überlagerungsstruktur werden automatisch nachgeladen.

Nach dem Ladevorgang wird an dem angegebenen Symbol im nachgeladenen Segment fortgesetzt.

Makroaufrufformat und Operandenbeschreibung

CALL
symbol

symbol
symbolische Adresse (Einsprungpunkt) innerhalb des zu ladenden Segmentes. Eine 4 Byte lange V-Konstante wird für dieses Symbol erzeugt. Mit diesem Operanden wird implizit das Segment angegeben, welches geladen werden soll, und gleichzeitig die Adresse, bei welcher nach der Makroausführung fortgesetzt wird.

Funktionsweise

Mit dem Makro **CALL** (und auch mit dem Makro **SEGLD**) wird das automatische Laden von Segmenten realisiert (das nicht automatische Laden wird mit dem Makro **LPOV** durchgeführt).

Durch eine Anweisung im Makro **CALL** wird der Assembler veranlasst, aus der angegebenen symbolischen Adresse eine V-Konstante zu erzeugen, durch die sowohl das zu ladende Segment als auch die Fortsetzadresse nach dem Ladevorgang gekennzeichnet ist. Auf Grund dieser V-Konstanten im Makro **CALL** und des Operanden CONTROL=YES in der Steueranweisung PROGRAM (siehe Handbuch „Dienstprogramme“ [27]) erzeugt der Binder ein für automatisches Laden notwendiges Überlagerungssteuermodul, das die V-Konstante mit einer Adresse befriedigt. Der Makro **CALL** enthält einen Sprung zu der V-Konstanten: Zur Ausführungszeit von **CALL** wird also die Steuerung dem Überlagerungssteuermodul übertragen, der prüft, ob das verlangte Segment schon im Speicher ist. Ist das Segment schon geladen, wird der Programmablauf mit dem Befehl fortgesetzt, der durch den Operanden „symbol“ bezeichnet ist. Ist das angegebene Segment noch nicht geladen, so wird es zusammen mit allen im gleichen Ast enthaltenen Segmenten in den Speicher gebracht. Nach dem Ladevorgang wird das Programm - ebenso wie bei nicht erfolgtem Laden - bei der angegebenen Adresse in dem zu ladenden Segment fortgesetzt.

Hinweise

– Beim Entwurf der Überlagerungstruktur eines Programms, in dem mit automatischem Laden von Segmenten gearbeitet wird, sollte der zusätzliche Speicherbedarf für das Überlagerungssteuermodul, für ENTAB und für SEGTAB berücksichtigt werden (siehe Handbuch „Dienstprogramme“ [27]).

– Der Makroaufruf **CALL** muss in einem Programmbereich liegen, der durch eine USING-Anweisung abgedeckt wird. Da das Register R15 von dem Makro **CALL** zum Laden der Segmente verwendet wird (es enthält die Einsprungadresse), darf es nicht als Basisregister für den Programmteil verwendet werden, in dem der Makroaufruf **CALL** liegt.

CDUMP2 – User-, System- oder Areadump ausgeben

Die drei von **CDUMP2** unterstützten Dumpformen unterscheiden sich im Umfang der angebotenen Informationen und im Namen und der Kennung für die ausgegebene Dump-Datei. Die folgenden Abschnitte stellen die wichtigsten Eigenschaften des Area-, User- und Systemdumps zusammen:

Areadump

Umfang des Areadumps:

– Die im **CDUMP2**-Aufruf angegebenen Bereiche des Benutzeradressraumes aus dem Klasse-6-Speicher und Klasse-5-Speicher, sofern die Ausgabe nicht durch den Systemparameter DUMPCL5P verboten ist.
– Besonders geschützte Seiten („Secret Pages“, Makroaufruf **CSTAT** PROTECT=YES) werden nur in dem Umfang ausgegeben, der durch den Systemparameter DUMPSEPA festgelegt ist.
– Zusätzliche Systembereiche: Modul AIDSYSD, der Bereich mit der COMAREA, die Tabellen TCB und TU-PCB (siehe Userdump); darüberhinaus - falls mit dem Operanden MODE=*EXP angefordert - alle weiteren Systembereiche des Userdumps.
– Datenräume (siehe Abschnitt „Erweiterung durch Datenräume“ auf Seite 61), an die der **CDUMP2**-Aufrufer konnektiert ist. Der Aufrufer bestimmt über den Datenbereich, welche Bereiche aus welchen Datenräumen gesichert werden sollen.
– Mit /MODIFY-TEST-OPTIONS USERDUMP-OPTIONS=*PARAMETERS(...) kann die Ausgabe gesteuert werden, siehe „Kommandos“ [19].

Kennung für die Areadump-Datei:

– Benutzerkennung der Task, die **CDUMP2** aufgerufen hat, falls der Benutzer berechtigt ist, alle im Dump ausgegebenen Daten zu lesen.
– Systemkennung SYSUSER, wenn der Dump lesegeschützte Daten enthält, auf die der Benutzer nicht zugreifen darf (z.B. Programme, die mit einem Lese-Kennwort geschützt sind, das der Benutzer nicht in die Passwort-Tabelle der Task eingetragen hat).

Name der Areadump-Datei:

– `$userid.SYS.ADUMP[.jobname].tsn.i`, falls die Datei unter der Kennung des **CDUMP2**-Aufrufers angelegt wird
– `$SYSUSER.SYS.ADUMP[.jobname].tsn.i.userid`, falls die Datei unter der Systemkennung SYSUSER angelegt wird.

Dabei bedeuten:

`userid`	Benutzerkennung des **CDUMP2**-Aufrufers
`jobname`	JOB-NAME des Auftrags, der **CDUMP2** aufgerufen hat (aus dem SET-LOGON-PARAMETERS-Kommando)
`tsn`	TSN des Auftrags, der **CDUMP2** aufgerufen hat
`i`	laufende Nummer des Area-/ Userdumps innerhalb der Task

Userdump

Umfang des Userdumps:

– Gesamter Klasse-6- und Klasse-5-Speicher, sofern nicht durch den Systemparameter DUMPCL5P verboten.
– Besonders geschützte Seiten („Secret Pages“, Makroaufruf **CSTAT** PROTECT=YES) werden nur in dem Umfang ausgegeben, der durch den Systemparameter DUMPSEPA festgelegt ist.
– Zusätzliche Systembereiche: Modul AIDSYSD, Trace Dump List; taskspezifische Systemtabellen wie TCB (Task Control Block), TFT (Task File Table), TU-PCB (Process Control Block), TU-AUDIT-TABLE (AUDIT-Tabelle, siehe Makro **AUDIT**).
– Die System-Trace-Table wird zum Zeitpunkt des **CDUMP2**-Aufrufs zwischengespeichert und bei der Dump-Erstellung in den Dump eingearbeitet (an der Stelle, wo sie auch im System steht).
– Zeigt der Fehlertask-Befehlszähler (PC) in den Systemadressraum, werden zusätzlich alle über Mehrzweckregister und PCs referenzierte Seiten (plus 5 Seiten davor und 5 Seiten danach) ausgegeben. Ist mindestens eine Seite dabei, die privilegiert, aber nicht „common readable“ ist, wird der Userdump auf die Kennung SYSUSER ausgegeben.
– Datenräume (siehe Abschnitt „Erweiterung durch Datenräume“ auf Seite 61), an die der **CDUMP2**-Aufrufer konnektiert ist.
  Über den Makro-Operanden DS bestimmt der Aufrufer, ob bestimmte, alle oder keine Datenräume gesichert werden sollen. Wird der Operand DS nicht angegeben, so bestimmt der bei /MODIFY-TEST-OPTIONS Operand DATA-SPACES=*YES/*NO eingestellte Wert die Ausgabe der Datenräume.
– Mit /MODIFY-TEST-OPTIONS USERDUMP-OPTIONS=*PARAMETERS(...) kann die Ausgabe gesteuert werden, siehe „Kommandos“ [19].

Kennung für die Userdump-Datei:

– Benutzerkennung der Task, die **CDUMP2** aufgerufen hat, wenn der Benutzer berechtigt ist, alle im Dump ausgegebenen Daten zu lesen und der Dump keine referenzierten Seiten enthält, die im privilegierten Adressraum liegen und nicht „common readable“ sind.
– Systemkennung SYSUSER, wenn der Dump lesegeschützte Daten enthält, auf die der Benutzer nicht zugreifen darf (z.B. Programme, die mit einem Lese-Kennwort geschützt sind, das der Benutzer nicht in die Passwort-Tabelle der Task eingetragen hat).

Name der Userdump-Datei:

– `$userid.DUMP[.jobname].tsn.i`, falls die Datei unter der Benutzerkennung des **CDUMP2**-Aufrufers eingerichtet wird.
– `$SYSUSER.DUMP[.jobname].tsn.i.userid`, falls die Datei unter der Benutzerkennung SYSUSER angelegt wird.

  Bedeutung siehe Areadump.

Systemdump

Einen Systemdump kann ein Anwender nur dann anfordern, wenn er mit /MODIFY-TEST-OPTIONS PRIVILEGE=*PARAMETERS(READ=m,WRITE=1) seine Leseprivilegierung auf einen Wert m≥3 eingestellt hat. Dazu ist er nur berechtigt, wenn ihm diese Privilegierungsmöglichkeit im Benutzerkatalog eingeräumt wurde.

Ein User- oder Areadump kann bei vorliegender Testprivilegierung (siehe oben) in einen Systemdump umgewandelt werden

– mit /MODIFY-TEST-OPTIONS Operand DUMP=*SYSTEM.
  Die Meldung `IDA0N45` wird dann unterdrückt.

– durch die Antwort `Y,SYSTEM` auf die Meldung `IDAN045`.

Der Operator kann steuern, ob der Systemdump auf Platte oder auf Magnetband bzw. Magnetbandkassette ausgegeben wird.
Beim Systemdump wird, abhängig vom Wert des Systemparameters DUMPCTRL, bei einem abnormalen Dumpabbruch die Meldung `IDA0N99` ausgegeben.

Umfang des Systemdumps:

– Gesamter Klasse-6-Speicher - abhängig vom Wert des Systemparameters DUMPSREF - mit Ausnahme der Seiten, die zu „Secret Pages“ erklärt wurden.
– Gesamter Klasse-5, Klasse-3- und Klasse-1-Speicher mit Ausnahme der Seiten, die zu „Secret Pages“ erklärt wurden.
– Alle Datenbereiche aus dem Klasse-4-Speicher sowie jene Coding-Seiten aus dem Klasse-4- und Klasse-2-Speicher, die bis zu einem Abstand von 5 Seiten vor und nach einer Adresse liegen, auf die die Befehlszähler (PC) aus den PCBs und der Trace Table sowie die Mehrzweckregister der PCBs und die Börsenregister des TCB verweisen. Ausgenommen davon sind „Secret Pages“.
– Besonders geschützte Seiten („Secret Pages“, Makroaufruf **CSTAT** PROTECT=YES) werden nur in dem Umfang ausgegeben, der durch den Systemparameter DUMPSEPA festgelegt ist.
– Mit SNAP wird der Klasse-1-, Klasse-3- und der residente Klasse-4-Speicher vor der eigentlichen Dump-Erstellung zwischengespeichert und anschließend in den Dump eingearbeitet.
– Die System-Trace-Table wird zum Zeitpunkt des **CDUMP2**-Aufrufs zwischengespeichert und bei der Dump-Erstellung in den Dump eingearbeitet (an der Stelle, wo sie auch im System steht).
– Zusätzliche Systembereiche: Die Module AIDSYSD, EOLDTAB, DMCHD, NSISINF, CLASS2OP; Bereiche mit TU-AUDIT, Trace Dump List und die Systemdateien REPLOG und SERSLOG; Systemtabellen wie beim Userdump.
– Datenräume (siehe Abschnitt „Erweiterung durch Datenräume“ auf Seite 61), die das Betriebssystem über eine privilegierte Schnittstelle ($DMPDEF(I)) beim Dumperzeuger CDUMP angemeldet hat.

Kennung für die Systemdump-Datei: SYSDUMP

Name der Systemdump-Datei:

```
                    ⎧ABSOLU ⎫
:catid:$SYSDUMP.    ⎨       ⎬ .pc.ec.tsn.datum.uhrzeit
                    ⎩modul  ⎭
```

Dabei bedeuten:

`modul`	Name des Moduls, aus dem **CDUMP2** aufgerufen wurde, max. 8 Zeichen
`ABSOLU`	Wird eingesetzt, wenn kein Modulname existiert
`pc`	Adresse im Befehlszähler (relativ zum Modulanfang bzw. absolut)
`ec`	Ereigniscode (sedezimal)
`tsn`	TSN der Task, die **CDUMP2** aufgerufen hat
`datum`	Datum in der Form `Dyymmdd` (D ist Aufmerker für Datumsbeginn)
`uhrzeit`	Uhrzeit in der Form `hhmmss`

Dumpausgabe

- Der Anwender kann einen Dump anfordern, wenn vorher /MODIFY-TEST-OPTIONS USERDUMP-OPTIONS=*PARAMETERS(DUMP=*YES) angegeben wurde bzw. DUMP=*STD voreingestellt ist. Bei DUMP=*YES wird der Dump automatisch erstellt, sonst erscheint bei der Dumpanforderung folgende Anfrage:

  ```
  IDAON45 DUMP DESIRED? REPLY (Y=USER-/AREADUMP TO DISK;
                        Y,<VOLUMETYPE>=USER-/AREADUMP TO TAPE;
                        Y,SYSTEM=SYSTEMDUMP;
                        N=NO)
  ```

  Der Anwender kann wahlweise einen User- oder Areadump (mit der Antwort: Y) oder einen Systemdump (mit der Antwort: Y,SYSTEM) auf Platte anfordern. Für die Anforderung eines Systemdumps ist eine entsprechende Testprivilegierung erforderlich. Bei Angabe von `<VOLUMETYPE>` wird der Dump auf Band ausgegeben.

- Wenn /MODIFY-TEST-OPTIONS USERDUMP-OPTIONS= *PARAMETERS(DUMP=*STD) voreingestellt ist, wird in Prozeduren oder im Batch-Betrieb kein Speicherauszug ausgegeben. Es erscheint folgende Meldung:

  ```
  IDAON48 TASK/SYSTEM SETTINGS PROHIBIT DUMP
  ```

- Ist /MODIFY-TEST-OPTIONS USERDUMP-OPTIONS=*PARAMETERS(DUMP=*NO) gesetzt, wird die Dumpanforderung abgewiesen. Es erscheint folgende Meldung:

  ```
  IDAON47 DUMP PROHIBITED BY /MODIFY-TEST-OPTIONS COMMAND
  ```

- Hat der Anwender einen Systemdump angefordert, erscheint an der Konsole folgende Meldung:

  ```
  IDAON52 SYSTEM DUMP DESIRED? REPLY (EOT=OUTPUT TO DISK;
                               <VOLUMETYP>=OUTPUT TO TAPE;
                               N=NO)
  ```

Die Konsolemeldung IDAON52 kann mit den Systemparametern DUMPCTRL und DUMPSD# unterdrückt werden. Generell wird die Meldung unterdrückt, wenn DUMPCTRL auf „operatorlosen Betrieb“ oder „Fehler in Systemtask“ gesetzt ist. Weiter kann mit DUMPSD# die Anzahl der Meldungen IDAON52 angegeben werden, die pro Session nicht an der Konsole erscheinen sollen.

Wird IDAN052 unterdrückt, so wird der Dump automatisch erstellt. Dadurch kann eine bessere Konsistenz der Daten im Klasse-3- und KLasse-4-Speicher erreicht werden.

- Ist für die Ausgabe eines User- oder Areadumps zu wenig Speicherplatz vorhanden, so wird mit folgender Meldung abgebrochen:
  IDAON57 INSUFFICIENT DISK SPACE. NO DUMP OUTPUT
- Sind die **CDUMP2**-Operanden nicht ordnungsgemäß angegeben, wird die Speicherausgabeanforderung mit folgender Meldung abgewiesen:
  IDAON46 CDUMP OPERAND LIST INCORRECT OR NOT AVAILABLE .
  Tritt bei der Ausgabe des Speicherabzugs ein Fehler im System auf, so werden (abhängig vom Fehler) folgende Meldungen ausgegeben:
  - IDAON63 DMS xxxx ERROR OCCURRED. DUMP PROCESSING CONTINUED
    Die Dump-Verarbeitung wird fortgesetzt.
  - IDAON61 DUMP PROCESSING ABORTED DUE TO aaa ERROR AT PC=bbb. RC=ccc
    Diese Meldung wird bei User- oder Systemdump auch auf Konsole ausgegeben. Gleichzeitig wird bei Ausgabe der Meldung ein SERSLOG-Eintrag geschrieben. Beim Systemdump wird die SERSLOG-Datei in den Dump aufgenommen. Die Dump-Verarbeitung wird abgebrochen.
- Die Dump-Ausgabe wird zeitlich überwacht. Kann der Dump nicht innerhalb von 36 Minuten ausgegeben werden, wird die Dump-Ausgabe abgebrochen.
- Einspielen von Dumpdateien vom Band
  Wurde der Dump auf Magnetbandkassette gezogen, so kann er mittels COPY-FILE-Kommando auf Platte kopiert werden. Das Dienstprogramm PERCON kann in diesem Fall nicht benutzt werden.

  Beispiel

```
/IMPORT-FILE SUPPORT=*TAPE(VOLUME=<volume>,DEVICE=<device>,-
       FILE-NAME=<originaldateiname>)

/ADD-FILE-LINK LINK=DMCOPY11,FILE-NAME=<originaldateiname>,-
       ACCESS-METHOD=*UPAM,BUF-LEN=*STD(2)

/ADD-FILE-LINK LINK=DMCOPY22,FILE-NAME=<ausgabedateiname>,-
       ACCESS-METHOD=*UPAM,BUF-LEN=*STD(2)

/COPY-FILE FROM-FILE=<originaldateiname>,TO-FILE=<ausgabedateiname>
```

  Siehe dazu auch COPY-FILE-Kommando im Handbuch „Kommandos“ [19].

Allgemeines

Anwendungsgebiet: Testhilfe; siehe Seite 166
Makrotyp: S-Typ, MF-Format **1**: C-/D-/L-/M-/E-Form; siehe Seite 29

CDUMP2 generiert immer eine 31-Bit-Schnittstelle. Zur Generierung einer 24-Bit-Schnittstelle, muss der alte Makro CDUMP (Anhang) verwendet werden.

Makrobeschreibung

Der Makro **CDUMP2** erstellt (in einer eigenen Dumptask) einen Speicherabzug für die Task, die **CDUMP2** aufgerufen hat. Durch Angabe des Operanden SCOPE kann der Anwender bestimmen, ob ein Area-, User- oder Systemdump ausgegeben werden soll.

Der Dump wird unaufbereitet als PAM-Datei auf Platte geschrieben (wahlweise auch auf Magnetband/Magnetbandkassette). Es ist nicht möglich, den Dump auf mehrere Bänder zu verteilen. Zur Auswertung und Druckausgabe kann der Dump mit den Software-Produkten DAMP und AID aufbereitet werden (siehe „Diagnosehandbuch“ [9] und „AID“ [3]).

Makroaufrufformat und Operandenbeschreibung

```
CDUMP2

MF=C / D / L / M / E
,SCOPE=*USER / *SYSTEM / *AREA
[,XPAND=PARAM / DSCB / AREA]
,PC=*STD / <var: pointer> / (<reg: pointer> )
,EC=*STD / <var: pointer> / (<reg: pointer> )
,CODE=<var: pointer> / (<reg: pointer> )
,INSERT=<var: pointer> / (<reg: pointer> )
,TITL=*STD / <var: pointer> / (<reg: pointer> )
,SNAP=*STD / *YES / *NO
,ELSN=*NONE / <var: pointer> / (<reg: pointer> )
,DIAG=*NO / *YES
,DIV=*STD / *YES / *NO
,DS=*STD / *YES / *NO / <var: pointer> / (<reg: pointer>)
,MMAP=*STD / *YES / *NO
,NUM=<integer 1..2048>
,MODE=*STD / *EXP
,DSCTRL=<var: pointer> / (<reg: pointer> )
```

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

CODE=
bezeichnet eine Zeichenfolge zur Kennzeichnung des Dumps; Länge = 7 Byte. Die Zeichenfolge wird bei der Meldung IDA0N51 ausgegeben. Der Operand darf nur zusammen mit SCOPE=*USER oder SCOPE=*SYSTEM angegeben werden.

<var: pointer>
Name des Feldes mit der Adresse der Zeichenfolge; nur in Verbindung mit MF=M zulässig.

(<reg: pointer>)
Register mit der Adresse der Zeichenfolge; nur in Verbindung mit MF=M zulässig.

DIAG=
steuert, ob die Meldung IDA0N50 an den Operator gesendet wird, die die Adresse des CDUMP2-SVCs enthält. Der Operand darf nur in Verbindung mit SCOPE=*SYSTEM angegeben werden.

***NO**
Die Meldung IDA0N50 wird nicht ausgegeben.

***YES**
Die Meldung IDA0N50 wird ausgegeben.

DIV=
gibt an, ob DIV-Fenster im Userdump enthalten sein sollen. Der Operand darf nur zusammen mit SCOPE=*USER angegeben werden, andernfalls wird DIV=*YES angenommen.
Siehe auch Abschnitt „Erweiterung durch Datenräume“ auf Seite 61.

***STD**
Der mit dem Operanden DATA-IN-VIRTUAL in /MODIFY-TEST-OPTIONS eingestellte Wert bestimmt, ob DIV-Fenster im Userdump enthalten sein sollen (*YES) oder nicht (*NO).

***YES**
Es sollen alle DIV-Fenster im Userdump enthalten sein.

***NO**
Es sollen keine DIV-Fenster im Userdump enthalten sein.

DS=
bestimmt, welche Speicherbereiche aus Datenräumen (DS) in den Userdump aufgenommen werden sollen. Angabe nur zusammen mit SCOPE=*USER.
Im Areadump sind genau diejenigen Bereiche aus Datenräumen enthalten, die der Benutzer beim DSCTRL-Operanden angegeben hat. Bei Systemdumps werden nur die Bereiche aus Datenräumen aufgenommen, die das Betriebsystem über eine priviligierte Schnittstelle ($DMPDEF(I)) beim Dumperzeuger CDUMP angemeldet hat.
Siehe auch Abschnitt „Erweiterung durch Datenräume“ auf Seite 61.

***STD**
Der mit dem Operanden DATA-SPACES in /MODIFY-TEST-OPTIONS eingestellte Wert bestimmt, ob Datenräume in den Userdump aufgenommen werden sollen (*YES) oder nicht (*NO).

***YES**
Es werden alle, aber max. 100, vom Aufrufer genutzte Datenräume in den Userdump aufgenommen.

***NO**
Es werden keine Datenräume in den Userdump aufgenommen.

<var: pointer>
Name des Feldes mit der Adresse der Liste der Datenräume; nur in Verbindung mit MF=M zulässig.

(<reg: pointer>)
Register mit der Adresse der Liste der Datenräume; nur in Verbindung mit MF=M zulässig.
Diese Liste enthält die SPIDs (8 Byte pro Eintrag). Sie muss mit einem Null-Eintrag (D(0)) abgeschlossen werden.

DSCTRL=
ist ein Zeiger auf einen Data Space Control Block (DSCB). Im DSCB können ein Datenraum und die zugehörigen Bereiche definiert werden. DSCTRL muss auf den ersten DSCB zeigen. Der Operand darf nur zusammen mit SCOPE=*AREA angegeben werden.

Datenstruktur eines DSCB:

```
CDDDSCB    DSECT ,        DATA SPACE CTRL BLOCK
CDDDS@     DS    A                   Zeiger auf nächsten DSCB
CDDSPID    DS    XL8                 SPID des DS
CDDNUM     DS    H                   Anzahl der Bereiche im DS
CDDSTRT    DS    A                   Beginn des ersten Bereiches
CDDEND     DS    A                   Ende des ersten Bereiches
```

Maximale Anzahl der Bereiche im DSCB ist 2048. Der Anwender kann DSCB-Blöcke verketten, um Bereiche mehrerer Datenräume seiner Task angeben zu können. Standardmäßig sind keine Bereiche aus einem Datenraum angegeben. Siehe auch Abschnitt „Erweiterung durch Datenräume“ auf Seite 61.

<var: pointer>
Name des Feldes mit der Adresse des DSCB; nur in Verbindung mit MF=M zulässig.

(<reg: pointer>)
Register mit der Adresse des DSCB; nur in Verbindung mit MF=M zulässig.

Angabe, um die DSECT eines DSCB zu erhalten: CDUMP2 MF=D,XPAND=DSCB.
Angabe, um die DSECT eines Bereichs zu erhalten: CDUMP2 MF=D,XPAND=AREA.

EC=
bestimmt, woher der Ereigniscode (Eventcode, Unterbrechungsgewicht) geholt werden soll. Der Ereigniscode wird in der Meldung `IDA0N51` ausgegeben. Der Operand darf nur zusammen mit SCOPE=*USER oder SCOPE=*SYSTEM angegeben werden, bei SCOPE=*AREA wird der Ereigniscode stets aus dem Aufrufer-Stack geholt.

<u>*STD</u>
Der Ereigniscode soll aus dem Aufrufer-Stack geholt werden.

<var: pointer>
Name des Feldes mit der Adresse des Ereigniscodes; nur in Verbindung mit MF=M zulässig.

(<reg: pointer>)
Register mit der Adresse des Ereigniscodes; nur in Verbindung mit MF=M zulässig.

ELSN=
bezeichnet die Adresse eines Feldes mit der Nummer eines Error-Log-Sequence-Blocks, in den der Aufrufer spezielle Daten in die Error-Log-Datei abgelegt hat. Die Nummer wird in der Meldung `IDA0N51` ausgegeben.
Feldlänge = 4 Byte; Ausrichtung auf Wortgrenze erforderlich. Die Nummer muss als Binärzahl eingegeben werden. Der Operand darf nur in Verbindung mit SCOPE=*SYSTEM angegeben werden.

***NONE**
Ein ELSN wird nicht ausgegeben.

<var: pointer>
Name des Feldes mit der Adresse des ELSN; nur in Verbindung mit MF=M zulässig.

(<reg: pointer>)
Register mit der Adresse des ELSN; nur in Verbindung mit MF=M zulässig.

INSERT=
legt einen Text fest, der mit der Meldung `IDA0N51` ausgegeben wird. In ihm können z.B. nähere Angaben zur Ursache des Speicherabzuges gemacht werden.
Der Anwender muss diesen Text, der bis zu 60 Zeichen lang sein darf, in einem Datenbereich hinterlegen. Aufbau des Datenbereichs:

Byte 1: Länge (sedezimal) des auszugebenden Textes (in Byte). Enthält Byte 1 den Wert 0, wird kein INSERT-Text ausgegeben.
Byte 2 bis n (n≤61): Auszugebender Text.

Der Operand darf nur zusammen mit SCOPE=*USER oder SCOPE=*SYSTEM angegeben werden.

<var: pointer>
Name des Feldes mit der Adresse des Textes; nur in Verbindung mit MF=M zulässig.

(<reg: pointer>)
Register mit der Adresse des Textes; nur in Verbindung mit MF=M zulässig.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.b. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Für Areadumps sind folgende Angaben Pflicht: Bei MF=C/D/E muss der Operand SCOPE, bei MF=C/D zusätzlich der Operand NUM angegeben werden.
Für Systemdumps ist folgende Angabe Pflicht: Bei MF=C/D/E muss der Operand SCOPE angegeben werden.

MMAP=
legt fest, ob Memory-Map-Seiten (Datenbereiche von POSIX-Seiten) im Dump gesichert werden sollen. Dieser Operand wird nur zusammen mit SCOPE=*USER ausgewertet. Für SCOPE=*SYSTEM wird MMAP=*YES angenommen.

***STD**
Der mit dem Operanden MEMORY-MAP in /MODIFY-TEST-OPTIONS eingestellte Wert bestimmt, ob Memory-Map-Seiten im Userdump enthalten sein sollen.

***YES**
Es sollen alle zur Task gehörenden Memory-Map-Seiten im Userdump enthalten sein.

***NO**
Es sollen keine Memory-Map-Seiten im Userdump enthalten sein.

MODE=
legt bei einem Areadump den Umfang der auszugebenden Diagnosedaten fest. Der Operand darf nur zusammen mit SCOPE=*AREA angegeben werden.

***STD**
Bewirkt, dass zusätzlich zu den angegebenen Bereichen im Klasse-6- und Klasse-5-Speicher nur das Modul AIDSYSD und die Bereiche mit COMAREA, TU-PCB und TCB ausgegeben werden (siehe Abschnitt „Areadump“ auf Seite 285).

***EXP**
veranlasst, dass zusätzlich zu den angegebenen Bereichen in Klasse-6- und Klasse-5-Speicher auch der Bereich mit COMAREA sowie alle weiteren Systembereiche wie beim Userdump ausgegeben werden (siehe Abschnitt „Userdump“ auf Seite 286).

NUM=
bezeichnet die Anzahl der auszugebenden Bereiche. NUM darf nur in Verbindung mit MF=L/C/D und nur in Verbindung mit SCOPE=*AREA angegeben werden. Die Anfangs- und Endadressen der auszugebenden Bereiche müssen (dynamisch) im erzeugten Datenbereich eingetragen werden. Die DSECT des Bereichs wird mit `CDUMP2 MF=D, XPAND=AREA` erzeugt. In Verbindung mit MF=C/D ist NUM Pflichteingabe.

<integer 1..2048>
Anzahl der auszugebenden Bereiche als Direktangabe; Zahlenbereich: 1..2048.

PC=
bezeichnet ein Register oder Feld, das den zu protokollierenden Befehlszähler enthält.

***STD**
Der Befehlszähler soll aus dem Aufrufer-Stack geholt werden.

<var: pointer>
Name des Feldes mit der Adresse des Befehlszählers; nur in Verbindung mit MF=M zulässig.

(<reg: pointer>)
Register mit der Adresse des Befehlszählers; nur in Verbindung mit MF=M zulässig.

SCOPE=
gibt an, ob ein User-, System- oder Areadump ausgegeben werden soll.

***USER**
Ein Userdump wird ausgegeben.

***SYSTEM**
Ein Systemdump wird ausgegeben. Angabe nur erlaubt für Anwender mit Lese-Testprivilegierung ≥ 3 bzw. für TPR-Programme.

***AREA**
Ein Areadump wird ausgegeben.
Die Adressen für die Bereiche müssen direkt nach der Parameterliste angegeben werden. In Abhängigkeit von der Anzahl der Bereiche müssen die Adressen in der Abfolge Anfangsadresse, Endadresse, Anfangsadresse, Endadresse, ..., Anfangsadresse, Endadresse angegeben werden.

SNAP=
steuert die Verwendung des SNAP zur Erstellung eines hochkonsistenten Systemdumps. Der Operand darf nur zusammen mit SCOPE=*SYSTEM angegeben werden.

***STD**
bedeutet, dass zur Erstellung des Systemdumps ein zugehöriger Snapdump verwendet wird, falls ein solcher vorhanden ist (wenn der Systemdump auf Grund eines TPR-Programmfehlers entstanden ist, so wurde in der Regel vom System dazu ein Snapdump erzeugt).

***YES**
Bedeutet, dass CDUMP2 einen Snapdump erzeugt (falls nicht schon ein zugehöriger vorhanden ist) und diesen zur Erstellung des Systemdumps verwendet.

***NO**
bedeutet, dass CDUMP2 den Systemdump ohne Verwendung eines (eventuell trotzdem vorhandenen) zugehörigen Snapdump erstellt.

TITL=
bezeichnet eine zusätzliche Titelzeile für den Speicherabzug. Länge = 132 Zeichen.

***STD**
CDUMP2 erzeugt eine Standard Titelzeile.

<var: pointer>
Name des Feldes mit der Adresse der Titelzeile; nur in Verbindung mit MF=M zulässig.

(<reg: pointer>)
Register mit der Adresse der Titelzeile; nur in Verbindung mit MF=M zulässig.

XPAND=
gibt an, ob eine DSECT für die CDUMP2-Parameterliste, den DSCB oder einen Bereich erzeugt werden soll. Die Angabe ist nur in Verbindung mit MF=D möglich.

PARAM
Es wird eine DSECT für die CDUMP2-Parameterliste erzeugt.

DSCB
Es wird eine DSECT für den Data-Space-Control-Block erzeugt.

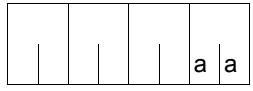
AREA
Es wird eine DSECT für einen Bereich erzeugt.

Angabe, um die DSECT eines DSCB zu erhalten: CDUMP2 MF=D,XPAND=DSCB.
Angabe, um die DSECT eines Bereichs zu erhalten: CDUMP2 MF=D,XPAND=AREA.

Rückinformation und Fehleranzeigen

Die Returncodes sind kompatibel zu denen des alten Makro CDUMP

Standard-header:



Über die Ausführung des Makros CDUMP2 wird im Standardheader folgender Returncode übergeben (aa=Maincode):

X'aa'	**Erläuterung**
X'00'	Dump fehlerfrei abgeschlossen.
X'04'	Dump unter Verwendung von Standardwerten abgeschlossen.
X'08'	Dump wegen /OPTION DUMP-Operand unterdrückt.
X'0C'	Dump wegen Systemkonventionen unterdrückt.
X'10'	Dump wegen schwerer CDUMP2-Operandenfehler unterdrückt.
X'14'	Dump wegen unzureichender Testprivilegierung unterdrückt.
X'18'	Dump wegen Fehler in DMS-Routinen unterdrückt.
X'1C'	Dump wegen Systemfehler unterdrückt.
X'20'	Dump unterdrückt, weil CDUMP2 zuvor unterbrochen wurde.
X'28'	Dump wegen SHUTDOWN-Bearbeitung unterdrückt.
X'2C'	Dump vom Aufrufer unterdrückt.
X'30'	Alle Bereichsangaben sind ungültig. Keine Dumpausgabe.
X'34'	Falsche Angabe bei NUM=... . Keine Dumpausgabe.
X'38'	Einige Dumpbereiche liegen nicht im Adressraum des Aufrufers oder die Angaben sind inkonsistent. Mindestens ein Speicherbereich wurde aber ausgegeben.
X'3C'	Dump unterdrückt, da DMS-Ready noch nicht erreicht ist.

Beispiel

Das folgende Programm legt im Klasse-6-Speicher einen Memory Pool an und erzeugt einen User- sowie einen Areadump.

```
CDUMP2   START
         PRINT NOGEN
         BALR  3,0
         USING *,3
         ENAMP MPNAME=MEMP,SCOPE=GLOBAL,MPIDRET=PID,BSIZE=48 ————————  (1)
REQMP    REQMP MPID=PID,BSIZE=5 ——————————————————————————————————  (2)
         LA    4,4095(1)
         LA    4,1(4)
         ST    4,AR1ANF
         MVC   0(TEXT1LEN,4),TEXT1 ——————————————————————————————  (3)
         LA    4,4095(4)
         ST    4,AR1END
         LA    4,1(4)
         ST    4,AR2ANF
         MVC   0(TEXT2LEN,4),TEXT2 ——————————————————————————————  (4)
         LA    4,4095(4)
         ST    4,AR2END
CDUMPUS  CDUMP2 MF=E,PARAM=PARAM ——————————————————————————————  (5)
CDUMPAR  CDUMP2 MF=E,PARAM=AREA ——————————————————————————————  (6)
TERM     TERM
****  Definitions  ****
         DS    0F
PID      DS    F
TEXT1    DC    C'AREADUMP: '
         DC    C'PAGE 2 OF MEMORY POOL '
         DC    100C'A'
         DC    100C'B'
TEXT1LEN EQU   *-TEXT1
TEXT2    DC    C'AREADUMP: '
         DC    C'PAGE 3 OF MEMORY POOL '
         DC    100C'C'
         DC    100C'D'
TEXT2LEN EQU   *-TEXT2
PARAM    CDUMP2 MF=L,SCOPE=*USER ——————————————————————————————  (7)
AREA     CDUMP2 MF=L,SCOPE=*AREA,NUM=2,MODE=*EXP ——————————————————  (8)
AR1ANF   DS    A ——————————————————————————————————————————————  (9)
AR1END   DS    A
AR2ANF   DS    A
AR2END   DS    A
         END
```

(1) Oberhalb der 16MB-Grenze wird ein 48 Hauptspeicherseiten großer Memory Pool eingerichtet. Die Anfangsadresse des Memory Pools wird in Register R1 abgelegt.

(2) Ab der Anfangsadresse des Memory Pools werden fünf Hauptspeicherseiten angefordert.

(3) Der Text `AREADUMP: PAGE 2 OF MEMORY POOL` und 100 mal A und B wird in den Memory Pool eingetragen.

(4) Der Text `AREADUMP: PAGE 3 OF MEMORY POOL` und 100 mal C und D wird in den Memory Pool eingetragen.

(5) An der symbolischen Adresse CDUMPUS des Programms wird der Makro **CDUMP2** in seiner E-Form aufgerufen. Der zugehörige Datenbereich wird an der symbolischen Adresse PARAM durch einen **CDUMP2**-Aufruf mit MF=L erzeugt (siehe (7)).

Die dort angegebenen Operanden veranlassen den Makro bei seiner Ausführung einen Userdump auszugeben (SCOPE=*USER).

(6) An der symbolischen Adresse CDUMPAR des Programms wird der Makro **CDUMP2** erneut in seiner E-Form aufgerufen. Der zugehörige Datenbereich wird in diesem Fall an der symbolischen Adresse AREA durch einen **CDUMP2**-Aufruf mit MF=L erzeugt (siehe (8)).

Die dort angegebenen Operanden veranlassen den Makro, bei seiner Ausführung einen Areadump für zwei Bereiche (NUM=2) in seiner erweiterten Form auszugeben.

(7) Datenbereich für den **CDUMP2**-Aufruf von Punkt (5).

(8) Datenbereich für den **CDUMP2**-Aufruf von Punkt (6).

(9) Die Adressen für die Bereiche müssen direkt nach der Parameterliste angegeben werden. In Abhängigkeit von der Anzahl der Bereiche müssen die Adressen in der Abfolge Anfangsadresse, Endadresse, Anfangsadresse, Endadresse, ..., Anfangsadresse, Endadresse angegeben werden.

Ablaufprotokoll:

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,cdump2), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,cdump2))
%  ASS6011 ASSEMBLY TIME: 492 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
```

```
%  ASS6006 LISTING GENERATOR TIME: 82 MSEC
//end
%  ASS6012 END OF ASSEMBH
/start-executable-program library=macexmp.lib,element-or-symbol=cdump2
%  BLS0523 ELEMENT 'CDUMP2', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'CDUMP2', VERSION ' ' OF '<date> <time>' LOADED
%  IDAON51 PROGRAM INTERRUPT AT LOCATION '0000006A (CDUMP2), (CDUMP)'
%  IDAON45 DUMP DESIRED? REPLY (Y=USER-/AREADUMP TO DISK;
   Y,<VOLUMETYPE>= USER-/AREADUMP TO TAPE; Y,SYSTEM=SYSTEMDUMP TO DISK;
   Y,SYSTEM,<VOLUMETYPE>=SYSTEMDUMP TO TAPE; N=NO)? y ——————————— (10)
%  IDAON53 DUMP BEING PROCESSED. PLEASE HOLD ON
%  IDAON54 'USERDUMP' WRITTEN TO FILE '$QM212.DUMP.V.2RCU.00001' ——— (11)
%  IDAON55 TITLE: 'TSN-2RCU  UID-QM212  AC#-89002  USERDUMP  PC-0000006A
   EC-50  VERS-150  DUMP-TIME  13:47:16  12-01-20'
%  IDAON51 PROGRAM INTERRUPT AT LOCATION '00000070 (CDUMP2), (ADUMP)'
%  IDAON45 DUMP DESIRED? REPLY (Y=USER-/AREADUMP TO DISK;
   Y,<VOLUMETYPE>=USER-/AREADUMP TO TAPE; Y,SYSTEM=SYSTEMDUMP TO DISK;
   Y,SYSTEM,<VOLUMETYPE>=SYSTEMDUMP TO TAPE; N=NO)? y ——————————— (12)
%  IDAON53 DUMP BEING PROCESSED. PLEASE HOLD ON
%  IDAON54 'AREADUMP' WRITTEN TO FILE '$QM212.SYS.ADUMP.V.2RCU.00002' — (13)
%  IDAON55 TITLE: 'TSN-2RCU  UID-QM212  AC#-89002  AREADUMP  PC-00000070
   EC-50  VERS-150  DUMP-TIME  13:47:22  12-01-20'
```

(10) Ein User- oder Areadump wird auf Platte ausgegeben, wenn die Abfrage in der Meldung IDAON45 mit Y[ES] beantwortet wird. Wurde zuvor /MODIFY-TEST-OPTIONS USERDUMP-OPTIONS=*PARAMETERS(DUMP=*YES) ausgeführt, wird die Meldung IDAON45 nicht ausgegeben.

(11) Die Datei mit dem Userdump wird unter der Benutzerkennung des Aufrufers katalogisiert.

(12) siehe (10)

(13) Die Datei mit dem Areadump wird unter der Benutzerkennung des Aufrufers katalogisiert.

Die Dumps können mit dem Dienstprogramm DAMP (siehe „Diagnosehandbuch“ [9]) für die Dialogauswertung oder Druckausgabe aufbereitet werden.

CHKEI – Ereigniskennung prüfen

Allgemeines

Anwendungsgebiet: Ereignissteuerung; siehe Seite 95
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

Makrobeschreibung

Der Makroaufruf **CHKEI** übergibt dem aufrufenden Programm Informationen über die Belegung der Warteschlangen, die zu der angegebenen Ereigniskennung gehören.

Makroaufrufformat und Operandenbeschreibung

```
CHKEI

{ { EINAME=name                                  }                                                  }
{ { EINAMAD=adr / (r) [,EINAMLN=länge] },SCOPE=LOCAL / GROUP / USER_GROUP / GLOBAL                  }
{ EIID=adr / (r)                                                                                    }

[,PARMOD=24 / 31]
[,MF=L / (E,..)]
```

EINAME=name
gibt den Namen der Ereigniskennung an. Die ereignisgesteuerte Verarbeitung muss bereits unter einer Ereigniskennung eröffnet sein, bevor der Aufruf **CHKEI** gegeben werden darf. Zur eindeutigen Bezeichnung der Ereigniskennung ist die Angabe des Operanden SCOPE nötig.

EINAMAD=
bezeichnet die Ereigniskennung. Die Ereigniskennung wird erst durch die zusätzliche Angabe von SCOPE eindeutig bezeichnet.

adr
symbolische Adresse des Feldes, das den Namen der Ereigniskennung enthält

(r)
r = Register, das die Adresse des Feldes enthält

EINAMLN=
gibt die Länge des Namens der Ereigniskennung in Byte an. Die Länge muss mindestens 1 Byte sein und darf 54 Byte nicht überschreiten. Fehlt der Operand, so wird das Längenattribut des EINAMAD-Operanden angenommen, wenn EINAMAD=adr angegeben ist; bei EINAMAD=(r) wird die maximale Länge von 54 Byte angenommen.

länge
Länge des Namens der Ereigniskennung.

SCOPE=
beschreibt den Geltungsbereich (Teilnehmerkreis) für die Ereigniskennung.

LOCAL
Die Ereigniskennung wird nur von der Task des Aufrufers benutzt.

GROUP
Teilnehmer sind alle Tasks unter der Benutzerkennung des Aufrufers.

USER_GROUP
Teilnehmer können alle Tasks sein, deren Benutzerkennungen der gleichen Benutzergruppe angehören wie die Benutzerkennung des einrichtenden Teilnehmers.

Der Operandenwert setzt die Existenz von Benutzergruppen voraus und kann daher nur angegeben werden, wenn die Funktionseinheit SRPM des Software-Produkts SECOS im System vorhanden ist. Vor einem Makroaufruf mit SCOPE=USER_GROUP muss deshalb mit dem Makro GETUGR (siehe Handbuch „SECOS“ [14]) geprüft werden, ob SRPM zur Verfügung steht; abhängig vom Ergebnis (Returncode) ist im Programm zu reagieren.

GLOBAL
Teilnehmer sind alle Tasks im System.

EIID=
bezeichnet die Kurzkennung der Ereigniskennung. Diese Kurzkennung wird dem Benutzer durch den **ENAEI**-Makroaufruf zur Verfügung gestellt. Die Verwendung der Kurzkennung an Stelle des Namens zur Identifizierung der Ereigniskennung beschleunigt die Verarbeitung. Durch die Kurzkennung wird die Ereigniskennung eindeutig bezeichnet.

adr
symbolische Adresse des Feldes, das die Kurzkennung enthält

(r)
r = Register, das die Adresse des Feldes enthält

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

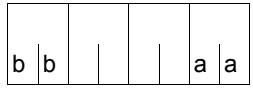
24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB).

Rückinformation und Fehleranzeigen

Register R1 enthält bei der Ausführung des Makros die Adresse der Operandenliste, nach der Ausführung die Anzahl der anstehenden **SOLSIG**- bzw. **POSSIG**-Aufrufe

R15:



Über die Ausführung des Makros CHKEI wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'bb'	X'aa'	Erläuterung
X'28'	X'00'	Funktion ausgeführt: Ein oder mehrere SOLSIG-Anforderungen stehen in der Warteschlange an. R1 enthält die Anzahl.
X'2C'	X'00'	Funktion ausgeführt: Ein oder mehrere POSSIG-Anforderungen stehen in der Warteschlange an. R1 enthält die Anzahl.
X'30'	X'00'	Funktion ausgeführt: Die Warteschlangen enthalten keine Anforderungen.
X'0C'	X'04'	Keine Aktion: Die vom System erstellte Ereigniskennung ist der Task nicht zugeordnet.
X'10'	X'04'	Keine Aktion: Es wurden ungültige Operanden angegeben.
X'14'	X'04'	Keine Aktion: Ungültiger Name bzw. ungültige Kurzkennung. Es existiert keine Ereigniskennung mit spezifizierter Identifikation.

Beispiele enthalten der Abschnitt „Ereignisgesteuerte Verarbeitung (Eventing)“ (Seite 107), der Abschnitt „Contingency-Prozesse“ (Seite 119) sowie die Beschreibung des Makros **SOLSIG** (Seite 850).

CHKPRV – Systemprivilegien abfragen

Allgemeines

Anwendungsgebiet: Abfragen und Zugriff zu Listen und Tabellen; siehe Seite 158
Makrotyp: S-Typ, MF-Format **2**: Standardform/E-/L-/M-/C-/D-Form; siehe Seite 29

Makrobeschreibung

Um privilegierte Systemdienste besser vor unbefugten Zugriffen zu schützen, ermöglicht das Software-Produkt SECOS eine Aufteilung der Privilegien, mit denen in früheren Versionen der Inhaber der Kennung TSOS (Systemverwaltung) ausgestattet war, auf verschiedene Verwaltungsinstanzen, die jeweils für einen Teilbereich der Systemverwaltung zuständig sind. Jede dieser Instanzen besitzt nur die Systemprivilegien, die sie für ihre Aufgaben benötigt. Einzelheiten dieser Privilegienverteilung können dem Handbuch „SECOS“ [14] entnommen werden.

Mit dem Makro **CHKPRV** kann der Benutzer in seinem Programm prüfen, ob der Auftrag, unter dem es läuft, eines oder mehrere dieser Systemprivilegien besitzt. Dabei können 31 Systemprivilegien abgefragt werden. Das Ergebnis dieser Überprüfung wird im Standardheader des Datenbereichs als Returncode übergeben (siehe Returncodetabelle im Anschluss an die Operandenbeschreibung).

Makroaufrufformat und Operandenbeschreibung

CHKPRV
PRIV=(priv[,priv]...) ,MF=S / E / L / C / D / M [,PARAM=adr / (r)] ,PREFIX=S / p ,MACID=RMC / macid

PRIV=
gibt das Systemprivileg an, das überprüft werden soll.

(priv[,priv]...)
Bezeichnungen der zu prüfenden Systemprivilegien. Bei Angabe mehrerer Privilegien wird der Returncode „Kein Fehler“ übergeben, wenn der Auftrag mindestens eines dieser Privilegien besitzt.

Die folgende Übersicht stellt die möglichen Werte für priv den zugehörigen Systemprivilegien gegenüber:

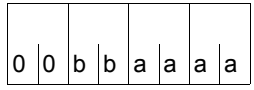
ACSADM	Alias-Catalog-Service-Verwaltung
CUPRV001 : CUPRV008	} von der Systemverwaltung flexibel zu belegen (damit können einzelnen Benutzern ganz individuelle Rechte zugewiesen werden)
FTACADM	FTAC-Verwaltung
FTADM	File-Transfer-Verwaltung
GUAADM	Systemglobale Guard-Administration
HSMSADM	HSMS-Verwaltung
HWMAINT	Hardware-Online-Wartung
NETADM	Netzverwaltung
NOTIFADM	Notification-Service-Administration
OPERATING	BS2000-Systembedienung
POSIXADM	POSIX-Benutzerverwaltung
PROPADM	Programmierung von Administrationsprozeduren
PRSRVADM	SPOOL-Verwaltung
SATFEVAL	Auswertung der SAT-Dateien
SATFMGMT	Verwaltung der SAT-Dateien
SECADM	Sicherheitsverwaltung
STDPROC	Verwenden von Benutzerkommandos
SUBSMGMT	Subsystemverwaltung
SWMONADM	Software-Monitor-Verwaltung
TAPEADM	Bandverwaltung
TAPEKEYADM	Encryption Key Verwaltung für Bänder
TSOS	TSOS-Privilegien, die keiner der anderen hier angegebenen Verwaltungsinstanzen zugeordnet sind
USERADM	Benutzerverwaltung
VMPRIV	Verwaltung einer virtuellen Maschine
VM2ADM	Verwaltung von VM2000

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros CHKPRY wird im Standardheader folgender Returncode übergeben (bb=Subcode1, aaaa=Maincode):

X'bb'	X'aaaa'	Erläuterung
X'00'	X'0000'	Funktion erfolgreich ausgeführt. Die Task besitzt mindestens eines der angegebenen Privilegien.
X'00'	X'0002'	Die Task besitzt keines der angegebenen Privilegien.
X'01'	X'0003'	Operandenfehler: Unzulässige Angabe für Privilegien.
X'20'	X'00FF'	Systemfehler

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Das aufrufende Programm wird beendet, wenn folgende Fehler auftreten:
– Der Datenbereich ist dem Aufrufer nicht zugewiesen.
– Der Datenbereich ist nicht auf Wortgrenze ausgerichtet.
– Der Datenbereich ist gegen Schreibzugriff geschützt.

CHKSI – Serialisierungskennung prüfen

Allgemeines

Anwendungsgebiet: (Task-)Serialisierung; siehe Seite 92
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

CHKSI generiert je nach Angabe die 24-Bit- oder die 31-Bit-Schnittstelle. Bei Makrokettung müssen alle Makros der Kette dieselbe Schnittstelle (24-Bit oder 31-Bit-Schnittstelle) benutzen.

Makrobeschreibung

Dieser Makroaufruf wird benutzt, um den Zustand einer Serialisierungskennung zu prüfen. Das Ergebnis dieser Überprüfung wird im Register R15 zur Verfügung gestellt (siehe unten).

Zur Ausführungszeit des **CHKSI**-Makroaufrufs muss die angesprochene Serialisierungskennung bereits vorhanden sein; es erfolgt kein implizites Einrichten.

Mit dem CONTINU-Operand können bis zu 255 **CHKSI**-Aufrufe gekettet werden.

Makroaufrufformat und Operandenbeschreibung

```
CHKSI

{ { SINAME=name                                  }                                                  }
{ { SINAMAD=adr / (r) [,SINAMLN=länge] },SCOPE=LOCAL / GROUP / USER_GROUP / GLOBAL                  }
{ SIID=adr / (r)                                                                                    }

,CONTINU=NO / YES
[,PARMOD=24 / 31]
[,MF=L / (E,..)]
```

SINAME=name
gibt den Namen der Serialisierungskennung an. Zur eindeutigen Bezeichnung der Serialisierungskennung ist die zusätzliche Angabe von SCOPE nötig.

SINAMAD=
bezeichnet den Namen der Serialisierungskennung. Nur durch die zusätzliche Angabe des Operanden SCOPE ist die Serialisierungskennung eindeutig bezeichnet.

adr
symbolische Adresse des Feldes, das den Namen enthält

(r)
r = Register, das die Adresse enthält.

SINAMLN=
gibt die Länge des Namens der Serialisierungskennung in Byte an. Die Länge muss mindestens 1 Byte sein und darf 54 Byte nicht überschreiten.
Fehlt der Operand, so wird das Längenattribut des SINAMAD-Operanden angenommen, wenn SINAMAD=adr angegeben ist; bei SINAMAD=(r) wird die maximale Länge (54) angenommen.

länge
Länge des Namens der Serialisierungskennung in Byte.

SCOPE=
beschreibt den Geltungsbereich (Teilnehmerkreis) für die Serialisierungskennung.

LOCAL
Die Serialisierungskennung wird nur von der aufrufenden Task benutzt.

GROUP
Teilnehmer sind alle Tasks unter der Benutzerkennung des Aufrufers.

USER_GROUP
Teilnehmer können alle Tasks sein, deren Benutzerkennungen der gleichen Benutzergruppe angehören wie die Benutzerkennung des einrichtenden Teilnehmers.

Der Operandenwert setzt die Existenz von Benutzergruppen voraus und kann daher nur angegeben werden, wenn die Funktionseinheit SRPM des Software-Produkts SECOS im System vorhanden ist. Vor einem Makroaufruf mit SCOPE=USER_GROUP muss deshalb mit dem Makro GETUGR (siehe Handbuch „SECOS“ [14]) geprüft werden, ob SRPM zur Verfügung steht; abhängig vom Ergebnis (Returncode) ist im Programm zu reagieren.

GLOBAL
Teilnehmer sind alle Tasks im System.

SIID=
bezeichnet die Kurzkennung der Serialisierungskennung. Diese Kurzkennung wird dem Benutzer durch den **ENASI**-Makroaufruf zur Verfügung gestellt. Die Verwendung der Kurzkennung an Stelle des Namens zur Identifizierung einer Serialisierungskennung beschleunigt die Verarbeitung. Diese Angabe ist eindeutig.

adr
symbolische Adresse eines 4 Byte langen Feldes, das die Kurzkennung enthält

(r)
r = Register, das die Adresse enthält

CONTINU=
bestimmt, ob eine Kettung von bis zu 255 **CHKSI**-Aufrufen erfolgen soll.

NO
Dieser Aufruf ist der letzte (bzw. einzige) Aufruf einer Folge.

YES
bedeutet, dass unmittelbar nach diesem **CHKSI**-Aufruf ein weiterer folgt.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

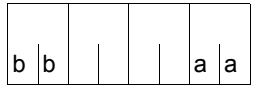
31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB).

Hinweis zum Makroaufruf in Listenform (Operand MF=L)

Zur Ausführung muss nur ein Makroaufruf mit MF=E gegeben werden, unabhängig davon, ob dieser Aufruf für eine einzelne Anforderung oder für eine Folge von Anforderungen gilt. Der Datenbereich bei einer Folge von Anforderungen wird durch Kettung der Makroaufrufe (MF=L) durch den CONTINU-Operanden erzeugt.

Rückinformation und Fehleranzeigen

R15:



Über die Ausführung des Makros CHKSI wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'bb'	**X'aa'**	**Erläuterung**
X'28'	X'00'	Alle Serialisierungskennungen wurden geprüft. Alle angegebenen Serialisierungskennungen sind verfügbar, d.h. keine der Serialisierungskennungen ist belegt.
X'2C'	X'00'	Alle Serialisierungskennungen wurden geprüft. Alle angegebenen Serialisierungskennungen sind von der Task des Aufrufers belegt, d.h. die Zugriffsanforderung für diese Serialisierungskennungen wurde erfüllt.
X'30'	X'00'	Alle Serialisierungskennungen wurden geprüft. Eine oder mehrere der angegebenen Serialisierungskennungen sind von der Task des Aufrufers belegt, der Rest ist verfügbar.
X'34'	X'00'	Alle Serialisierungskennungen wurden geprüft. Der Task des Aufrufers belegt keine der angegebenen Serialisierungskennungen, eine oder mehrere sind jedoch von anderen Tasks belegt.
X'38'	X'00'	Alle Serialisierungskennungen wurden geprüft. Eine oder mehrere der angegebenen Serialisierungskennung sind von der Task des Aufrufers belegt; der Rest ist von anderen Tasks belegt oder verfügbar.
X'10'	X'04'	Nicht alle Serialisierungskennungen wurden geprüft. Es wurden ungültige Operanden angegeben: – ungültige Adresse; z.B. Adresse innerhalb einer DSECT – ungültige Länge – ungültiger Name – nichtdefinierter Geltungsbereich oder CONTINU-Wert
X'14'	X'04'	Nicht alle Serialisierungskennungen wurden geprüft. Es wurde eine ungültige Kennung angegeben.
X'20'	X'04'	Nicht alle Serialisierungskennungen wurden geprüft. Für mindestens eine Serialisierungskennung wurde von dem aufrufenden Programm noch keine Enable-Funktion (siehe ENASI-Makroaufruf) ausgeführt.

CLCOM – Teilnahme an der Intertaskkommunikation beenden

Allgemeines

Anwendungsgebiet: Intertaskkommunikation; siehe Seite 76
Kommunikation; siehe Seite 167
Makrotyp: R-Typ; siehe Seite 28

Makrobeschreibung

Mit **CLCOM** beendet das aufrufende Programm die Teilnahme an der Intertaskkommunikation (sein ITC-Name wird von der Teilnehmerliste gestrichen).

Makroaufrufformat und Operandenbeschreibung

CLCOM
NOKEEP / KEEP / (1)

NOKEEP
legt fest, dass die Empfangswarteschlange aufgelöst wird. Die eventuell noch vorhandenen Nachrichten können nicht mehr angefordert werden.

KEEP
legt fest, dass die Empfangswarteschlange bestehen bleibt. Bereits eingetroffene Nachrichten können nach dem **CLCOM**-Aufruf noch angefordert werden.

(1)
Register R1 enthält die Anfangsadresse des Operandenfeldes. Das Operandenfeld muss 4 Byte lang sein, an einer Wortgrenze beginnen und folgenden Inhalt haben:

X'04000000' entspricht NOKEEP oder
C'KEEP' entspricht KEEP

Funktionsweise

Mit **CLCOM** beendet die Task des Aufrufers ihre Teilnahme an der Intertaskkommunikation (ITC). Sobald ein Programm **CLCOM** NOKEEP aufgerufen hat, kann es keine Nachrichten mehr abschicken, und auch keine mehr empfangen. Die Nachrichten, die noch in seiner Empfangswarteschlange stehen, werden gelöscht.

Nach dem Aufruf **CLCOM** KEEP wird die Empfangswarteschlange nicht aufgelöst. Das Programm kann noch Nachrichten absenden und diejenigen anfordern, die bereits eingetroffen sind. Neue Nachrichten werden nicht mehr übermittelt. War beim Aufruf **CLCOM** KEEP die Empfangswarteschlange leer, so wird wie bei **CLCOM** NOKEEP verfahren.

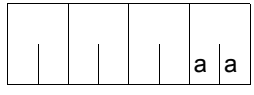
Der Aufruf **CLCOM** NOKEEP veranlasst das System, den ITC-Namen des Aufrufers aus der Teilnehmerliste zu streichen. Danach steht der ITC-Name wieder zur Verfügung. Der Aufrufer kann sich - wenn gewünscht - der ITC wieder anschließen (mit einem beliebigen noch nicht vergebenen ITC-Namen). Wenn der letzte ITC-Teilnehmer den Aufruf **CLCOM** NOKEEP gibt, wird die ITC geschlossen (das System löst die Kommunikationstabelle auf).

Hinweis

Wird der Operand NOKEEP weggelassen (Standardwert soll gelten), aber in diese Zeile eine Bemerkung geschrieben, dann führt das System die Funktion **CLCOM** KEEP aus.

Rückinformation und Fehleranzeigen

R15:



Über die Ausführung des Makros CLCOM wird im rechtsbündigen Byte des Registers R15 ein Returncode übergeben.

X'aa'	**Erläuterung**
X'00'	Die ITC-Teilnahme ist beendet und die Empfangswarteschlange ist aufgelöst.
X'04'	Fehler in der Operandenangabe. Die ITC-Teilnahme ist nicht beendet.
X'08'	Die Task des Aufrufers ist kein ITC-Teilnehmer.
X'0C'	Die Empfangswarteschlange enthält noch Nachrichten. Die ITC-Teilnahme muss noch beendet werden (nur bei KEEP).

CMD – Kommando aufrufen

Allgemeines

Anwendungsgebiet: Kommando aufrufen; siehe Seite 45
Kommunikation; siehe Seite 167
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/D-/C-/E-Form; siehe Seite 29

Makrobeschreibung

Der Makro **CMD** ermöglicht den Aufruf eines Kommandos bzw. einer Liste von Kommandos im Programmodus. Der Makro **CMD** ruft den MCLP (Macro Command Language Processor) auf und übergibt an diesen den Kommandonamen und die angegebenen Kommandooperanden bzw. eine Liste von Kommandonamen und die zugehörigen Kommandooperanden. Nach Ausführung des (letzten) Kommandos wird das Programm fortgesetzt.
Der Anwender kann das Protokoll der Kommandobearbeitung (z.B. Systemmeldungen) auf SYSOUT und/oder in einen Bereich seines Programms oder in eine S-Variable übertragen lassen. Der Kommando-Returncode kann ebenfalls in einen Bereich des Programms übertragen werden.

In Tabelle 12 auf Seite 322 sind die Kommandos aufgeführt, die über den Makro **CMD** nicht aufgerufen werden können.

Makroaufrufformat und Operandenbeschreibung

```
CMD

'kommandoname' [{ ,'oplist1'[, adr / (r)] }]
               { , , adr / (r)            }

[,OPART2='oplist2'] [,OPART3='oplist3'] [,OPART4='oplist4']
[,OPART5='oplist5'] [,OPART6='oplist6'] [,OPART7='oplist7']
,LIST=NO / YES
[,CMDRC=adr]
,DIALOG=NO / YES
,SYSOUT=YES / NO
,SUBST=NO / JV / ALL
,ORIGIN=CMD / CURRENT
[,DTAVAR@=adr]
```

CMD (Fortsetzung)
[,DTAVARL=länge]
,DTAEXT=NO / YES
[,MSGVAR@=adr]
[,MSGVARL=länge]
,MSGEXT=NO / YES
,VER=1 / 2 / 3 / 4
,BUFMOD=SHORT / LONG
[,PARMOD=24 / 31]
[,MF=L / (E,..) / D / C]
,PREFIX=M / p

Nach der Beschreibung der Stellungsoperanden werden die Schlüsselwortoperanden in alphabetischer Reihenfolge aufgeführt.

'kommandoname'
kommandoname = Name des Kommandos, das über **CMD** aufgerufen werden soll.
Wird der Operand LIST=YES angegeben, kann die Zeichenkette 'kommandoname' eine Liste mehrerer Kommandos enthalten, die durch Semikolon voneinander getrennt werden. Die Kommandos werden durch ein Leerzeichen von ihrer Kommandooperandenliste getrennt. Tritt bei der Abarbeitung eines Kommandos dieser Liste ein Fehler auf, wird der **CMD** an dieser Stelle abgebrochen.
Besteht der Operand 'kommandoname' aus einer Liste von Kommandos, entfällt die Angabe der Operanden 'oplist1' und OPARTx.

'oplist1'
oplist1 = op1,op2,op3,..... = Liste der Kommandooperanden.
Länge der Liste ≤ 248 Zeichen. Die Liste kann an beliebiger Stelle bei den Operanden OPART2, ..., OPART7 fortgesetzt werden. Die nächstfolgende Liste wird immer als Fortsetzung der vorhergehenden betrachtet (ohne Beachtung der Ziffern 2...7).
Der Operand darf nicht zusammen mit LIST=YES angegeben werden.

adr
Adresse des Empfangsfeldes für das SYSOUT-Protokoll. Wenn adr nicht angegeben ist, geht die Ausgabe nur nach SYSOUT. Das ist auch der Fall, wenn die Länge des Empfangsfeldes Null ist. Das Empfangsfeld muss auf Wortgrenze ausgerichtet sein. Aufbau und maximale Länge des Empfangsfeldes hängen vom Wert des Operanden BUFMOD ab. Jeder Satz des SYSOUT-Protokolls, der in das Empfangfeld übertragen wird, enthält in den ersten vier Byte sein Satzlängenfeld (Byte 0-1 enthält die Satzlänge, Byte 2-3 sind der reservierte Teil). Der eigentliche Ausgabetext beginnt ab Byte 4 jedes Satzes. Die Ausgabesätze werden fortlaufend in den Bereich geschrieben, bis die Bereichsgrenze erreicht ist.

Weitere Ausgabesätze, die in dem Bereich nicht mehr Platz haben, werden nur noch nach SYSOUT ausgegeben (wenn SYSOUT=YES angegeben ist). Bei Überschreiten der Bereichsgrenze kann ein Satz abgeschnitten werden (Returncode X'0C').

Hinweis

Das SYSOUT-Ausgabeformat für ein Kommando im Dialogbetrieb kann verschieden sein von dem im Batch-Betrieb. Das muss bei der Definition des Empfangsfeldes beachtet werden.

(r)
r = Register, das die Adresse des Empfangsfeldes enthält.

BUFMOD=
legt den Aufbau und die maximale Größe des Empfangsfeldes für das SYSOUT-Protokoll fest.

SHORT
Das Empfangsfeld kann bis zu 32 KB lang sein und hat folgenden Aufbau:

Byte 0-1: Länge l (sedezimal) des Empfangsfeldes in Byte ($l \leq 2^{15}-1$), die vom Anwender anzugeben ist.

Byte 2-3: reserviert, kein Eintrag

Byte 4-n: SYSOUT-Protokoll

LONG
darf nur angegeben werden, wenn die 31-Bit-Schnittstelle des Makros generiert wird (PARMOD=31). Dieser Wert darf nur in Verbindung mit VER=2/3/4 angegeben werden. Das Empfangsfeld kann bis zu 2 GB lang sein und hat folgenden Aufbau:

Byte 0-3: Länge l (sedezimal) des Empfangsfeldes in Byte ($l \leq 2^{31} - 1$), die vom Anwender anzugeben ist. Folgende Werte für l sind zu beachten:

`l=0:`	Das Empfangsfeld wird ignoriert
$1 \leq l \leq 16$:	Die Makroausführung wird mit dem Returncode X'08' abgebrochen

Byte 4-7: Länge (sedezimal) der Nutzinformation im Empfangsfeld in Byte (einschließlich des 16 Byte langen Vorspanns) wird bei der Makroausführung vom System eingetragen. Voreinstellung: Länge=16

Byte 8-11: reserviert; sie müssen jedoch beim Aufruf des CMD-Makros gelöscht sein (binär Null oder -1), sonst Abbruch mit Fehlercode X'08'

Byte 12-15: reserviert; wird vom System verwendet

Byte 16-n: SYSOUT-Protokoll

Hinweise

Ist das Empfangsfeld nicht vollständig zugewiesen, bricht die Makrobearbeitung mit dem Returncode X'08' ab.
Wird die angegebene Länge `l` überschritten, wird die Ausgabe wie bei BUFMOD=SHORT abgeschnitten. **CMD** liefert den Returncode X'0C' zurück.

CMDRC=
gibt die symbolische Adresse eines 9 Byte langen Feldes mit folgendem Aufbau an, in das der Kommando-Returncode des vom **CMD**-Makro bearbeiteten Kommandos geschrieben wird.

Byte 0: Subcode2 im Assembler-Format X'nn'

Byte 1: Subcode1 im Assembler-Format X'nn'

Byte 2-8: Maincode im Assembler-Format CL7

Wird im Operanden 'kommandoname' eine Liste mehrerer Kommandos übergeben (ist nur zusammen mit LIST=YES möglich), steht im angegeben Feld der Returncode des zuletzt an den MCLP übergebenen Kommandos.
Der Operand darf nur in Verbindung mit VER=3/4 angegeben werden.

adr
symbolische Adresse des Feldes, in das der Returncode geschrieben werden soll. Voreinstellung: 0, es wird kein Returncode übergeben.

DIALOG=
beschreibt, ob ein Fehler- oder Hilfedialog beim Erkennen von Syntaxfehlern geführt werden soll.

NO
Es wird kein Fehlerdialog geführt.

YES
Es soll ein Fehlerdialog geführt werden, wenn der Typ der Datensichtstation dies ermöglicht.

DTAEXT=
bestimmt, ob die bei DTAVAR@ angegebene S-Variable um den Inhalt der von OPS generierten Variablen erweitert werden soll oder nicht.
Die Angabe dieses Operanden ist nur in Verbindung mit VER=4 und den Angaben für DTAVAR@ und DTAVARL erlaubt.

NO
Die S-Variable soll nicht erweitert werden.
Der (alte) Inhalt der S-Variablen wird durch den Inhalt der OPS-Variablen ersetzt.

YES
Die S-Variable wird erweitert, indem der Inhalt der OPS-Variablen angehängt wird.
Kann die S-Variable nicht erweitert werden, wird der (alte) Inhalt vor dem Abspeichern gelöscht.

DTAVAR@=
gibt die symbolische Adresse eines Bereiches an, in dem der Name einer zusammengesetzten S-Variable steht. In diese S-Variable sollen die Inhalte aller von OPS generierten Variablen gespeichert werden.
Wurde die S-Variable vor dem Makroaufruf nicht deklariert, wird die Abarbeitung mit dem Returncode X'10' abgebrochen.
Der Operand darf nur in Verbindung mit VER=4 angegeben werden.

adr
symbolische Adresse des Bereichs mit dem Namen der S-Variablen. Sie kann mit der beim Operanden MSGVAR@ angegebenen Adresse identisch sein, um alle von MIP oder OPS generierten Variablen in einem einzigen Bereich zu sichern.

DTAVARL=
gibt die Länge des bei DTAVAR@ adressierten Bereiches an.
Der Operand darf nur in Verbindung mit VER=4 angegeben werden.

länge
anzugebende Länge des Feldes in Byte

LIST=
ermöglicht für den Operanden 'kommandoname' die Eingabe einer Liste mehrerer Kommandos und ihrer Operanden.

NO
Der Operand 'kommandoname' besteht aus einem einzigen Kommando.

YES
Der Operand 'kommandoname' enthält eine Liste mehrerer Kommandos, die mit Semikolon voneinander getrennt werden.
Dieser Wert darf nur in Verbindung mit VER=3/4 angegeben werden.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) sieh Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Bei MF=C oder MF=D kann ein Präfix PREFIX (p = 1 Buchstabe), wie im Aufrufformat dargestellt, angegeben werden.

MSGEXT=
bestimmt, ob die bei MSGVAR@ angegebene S-Variable um den Inhalt der von MIP generierten Variablen erweitert werden soll oder nicht.
Die Angabe dieses Operanden ist nur in Verbindung mit VER=4 und den Angaben für MSGVAR@ und MSGVARL erlaubt.

NO
Die S-Variable soll nicht erweitert werden.
Der (alte) Inhalt der S-Variablen wird durch den Inhalt der MIP-Variablen ersetzt.

YES
Die S-Variable wird erweitert, indem der Inhalt der MIP-Variablen angehängt wird. Kann die S-Variable nicht erweitert werden, wird der (alte) Inhalt vor dem Abspeichern gelöscht.

MSGVAR@=
gibt die symbolische Adresse eines Bereiches an, in dem der Name einer zusammengesetzten S-Variable steht. In diese S-Variable sollen die Inhalte aller von MIP generierten Variablen gespeichert werden.
Wurde die S-Variable vor dem Makroaufruf nicht deklariert, wird die Abarbeitung mit dem Returncode X'10' abgebrochen.
Der Operand darf nur in Verbindung mit VER=4 angegeben werden.

adr
symbolische Adresse des Bereichs mit dem Namen der S-Variablen. Sie kann mit der beim Operanden DTAVAR@ angegebenen Adresse identisch sein, um alle von MIP oder OPS generierten Variablen in einem einzigen Bereich zu sichern.

MSGVARL=
gibt die Länge des bei MSGVAR@ adressierten Bereiches an. Der Operand darf nur in Verbindung mit VER=4 angegeben werden.

länge
anzugebende Länge des Feldes in Byte.

OPARTx=
ermöglicht die Fortsetzung der Operandenliste.
x = Element aus der Menge (2, 3, .., 7).

'oplistx'
oplistx = op_i,op_j,op_k,.... = (Fortsetzungs-)Liste der Kommandooperanden; x = Element aus der Menge (2,3,...,7).
Länge: oplistx = 1..248 Zeichen.
Der Operand darf nicht zusammen mit LIST=YES angegeben werden.

ORIGIN=
gibt den Ursprung des Kommandos an, der benutzt werden muss, wenn überprüft wird, ob das Kommando erlaubt ist.
Der Operand darf nur in Verbindung mit VER=4 und MF=D oder MF=C angegeben werden.

CMD
Die Kommandoeingabe muss mit dem CMD-ALLOWED-Attribut aus der Syntax-Datei überprüft werden.

CURRENT
Zusätzlich zum CMD-ALLOWED-Attribut, überprüft SDF auch, ob das Kommando im aktuellen Modus erlaubt ist. Das ist der Modus, in dem das Programm gestartet wurde. Der Modus kann sein: Dialog, Dialog-Prozedur, Batch oder Batch-Prozedur.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).
Dieser Wert darf nur in Verbindung mit VER=1 angegeben werden.

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

SUBST=
bestimmt, welche Substitutionen in der Kommandoeingabe erfolgen sollen.

NO
In der Kommandoeingabe wird keine Substitution durchgeführt.

JV
Alle Jobvariablen der Kommandoeingabe werden durch ihre Inhalte ersetzt.
Dieser Wert darf nur in Verbindung mit VER=4 angegeben werden.

ALL
Die Kommandoeingabe wird, abhängig von der Programmumgebung, vollständig ersetzt, d.h. alle S-Variablen, Jobvariablen und SYSFILE-Prozedurparamter in der Kommandoeingabe werden in der angegebenen Reihenfolge ersetzt. Die Ersetzung von SYSFILE-Prozedurparametern ist nur in Nicht-S-Prozeduren sinnvoll.
S-Variablen werden nur dann ersetzt, wenn sie in der aktuellen Programmumgebung sichtbar (bekannt) sind.
Dieser Wert darf nur in Verbindung mit VER=4 angegeben werden.

SYSOUT=
beschreibt, ob das Protokoll auch nach SYSOUT ausgegeben werden soll.

YES
Die Ausgabe erfolgt auch auf SYSOUT.

NO
Es erfolgt keine Ausgabe nach SYSOUT. In diesem Fall muss das Empfangsfeld adr spezifiziert sein.

VER=1 / 2 / 3 / 4
hat nur Bedeutung für MF ungleich E und wird andernfalls ignoriert. Der Operand legt fest, welche Version des Datenbereichs erzeugt werden soll.
Zur Verträglichkeit mit anderen Operanden siehe folgende Tabelle.

		VER=1	VER=2	VER=3	VER=4
PARMOD	= 24	x			
	=31	x	x	x	x
BUFMOD	= SHORT	x	x	x	x
	= LONG		x	x	x
LIST	= NO	x	x	x	x
	= YES			x	x
CMDRC				x	x
SUBST	= NO	x	x	x	x
	= JV				x
	=ALL				x
ORIGIN					x
DTA...					x
MSG...					x
andere Operanden		x	x	x	x

Tabelle 11: Verträglichkeit des Operanden VER mit anderen Operanden (CMD)

x bedeutet, dass die betreffenden Operanden zusammen angegeben werden können.

Hinweise zum Makroaufruf

- Die Angabe DIALOG=YES wird im Expertenmodus (MODIFY-SDF-OPTIONS GUIDANCE=*EXPERT, siehe Handbuch „Kommandos“ [19]) ignoriert. Es wird der Returncode X'10' (statt X'14') zurückgegeben.
- Wird DIALOG=NO angegeben, liefert SDF die folgenden Returncodes zurück, wenn die Hilfe-Anforderung im CMD-Input benutzt wird:
  - „?“: Es wird der Returncode X'10' zurückgegeben.
  - „<operation?>“: (? als Teil von <operation>) : Es wird der Returncode X'14' zurückgegeben.
  - „<operation> ?“ : Ist <operation> gültig, dann wird der Returncode X'10' zurückgegeben. Ist <operation> ungültig, dann wird der Returncode X'14' zurückgegeben.

  In beiden Modi (Dialog oder Batch) wird ein unklares Kommando für alle Werte des Operanden DIALOG mit dem Returncode X'14' zurückgewiesen.
- Die Abarbeitung der Kommandoliste (bei LIST=YES) wird beendet, wenn eines der gegebenen Kommandos einen Fehler verursacht (Syntaxfehler, Speicherfehler,..). Ein solcher Fehler führt nicht zum SPIN-OFF.
- Der mit MF=C/D generierte Datenbereich muss dem mit MF=L genutzten Datenbereich entsprechen.
- Wenn der **CMD**-Datenbereich mit der Standard- oder MF=L-Form initialisiert wurde, können im Feld `<PREFIX>CLPLNTH` folgende Informationen gelesen werden:
  - die Länge des gesamten, vorgegebenen Datenbereichs des **CMD**-Aufrufs im ersten Halbwort (d.h. ab Byte 0 in der Länge von 2 Byte).
  - die Länge der abgearbeiteten Kommandoeingabe im zweiten Halbwort (d.h. ab Byte 2 in der Länge von 2 Byte). Wurde die Abarbeitung des **CMD**-Makros wegen eines Fehlers vorzeitig beendet, zeigt dieses Halbwort die Abbruchstelle an.
- Strukturierte Ausgaben des **CMD**-Makros können nicht (wie beim Kommando EXECUTE-CMD) auf den S-Variablenstrom SYSINF gelenkt werden.
- Wenn der **CMD**-Makro seine Ausgabe in eine von OPS generierte S-Variable lenken soll,
  - kann das Kommando START-PROGRAM nicht vom **CMD**-Makro abgesetzt werden
  - und ist die Verkettung mehrerer Kommandos in einem Eingabesatz (Operanden 'kommandoname' und LIST=YES) nicht möglich.
- Einige BS2000-Kommandos können nicht über den Makro **CMD** aufgerufen werden, siehe folgende Tabelle:

Kommando	Funktion	Handbuch
ADD-CJC-ACTION	CJC-Kommandofolge einleiten (Jobvariablen)	[19]
BEGIN-BLOCK	leitet einen Kommandoblock ein	[21]
BEGIN-PARAMETER-DECLARATION	leitet die Deklaration der Prozedurparameter im Prozedurkopf ein	[19],[21]
BEGIN-PROCEDURE	Prozedurdateimerkmale festlegen	[19]
BREAK	Kommando-Modus anfordern	[33]
CANCEL-PROCEDURE	Prozedur(ablauf) abbrechen	[19]
CANCEL-PROGRAM	Programmlauf abbrechen	[19]
CHANGE-ACCOUNTING-FILE	Systemabrechnungsdatei wechseln	[19]
COPY-SYSTEM-FILE	kopiert Systemdateien	[19]
CYCLE	bricht einen Schleifendurchlauf ab	[21]
DELON	ON-Kommando löschen	[33]
ENDON	beendet eine ON-Anweisungsfolge	[33]
ELSE	leitet den ELSE-Zweig im IF-Block ein	[19],[21]
ELSE-IF	leitet im IF-Block einen Alternativzweig ein	[21]
END-BLOCK	schließt einen Kommandoblock ab	[21]
END-CJC-ACTION	CJC-Kommandofolge abschließen (Jobvariablen)	[19]
END-FOR	schließt einen FOR-Block ab	[21]
END-IF	schließt einen IF-Block ab	[19],[21]
END-PARAMETER-DECLARATION	schließt eine Prozedurparameterdeklaration ab	[19],[21]
END-PROCEDURE	beendet eine Prozedurdatei	[19]
END-WHILE	schließt einen WHILE-Block ab	[21]
ENDP	Prozedurdatei beenden	[33]
EOF	Dateiende für SYSDTA kennzeichnen	[19]
ESCAPE	Prozedurablauf unterbrechen	[33]
EXIT-BLOCK	bricht die Verarbeitung eines Kommandoblockes ab	[21]
FOR	leitet einen FOR-Block ein	[21]
GOTO	springt zu einer Marke	[19],[21]
HOLD-PROCEDURE	unterbricht Prozedurablauf und ermöglicht die Kommandoeingabe vom Datensichtgerät	[19]
HOLD-PROGRAM	unterbricht Programmablauf und ermöglicht die Kommandoeingabe vom Datensichtgerät	[19]

Tabelle 12: Kommandos, die nicht über den Makro CMD aufgerufen werden können (Teil 1 von 2)

Kommando	Funktion	Handbuch
IF	leitet einen Block ein	[19],[21]
IF-BLOCK-ERROR	leitet eine Block-Fehlerbehandlung ein	[19],[21]
IF-CMD-ERROR	leitet eine Kommando-Fehlerbehandlung ein	[21]
LOGON	Auftrag (Job) einleiten	[33]
MODIFY-ACCOUNTING-PARAMETERS	legt Abrechnungssätze und Satzerweiterungen für die Accountingdatei fest	[19]
MODIFY-JV-CONDITIONALLY	ändert den Wert einer Jobvariablen und verzweigt zum Sprungziel	[19]
ON	bedingte Ausführung einer Kommandofolge	[33]
PROCEDURE	Prozedurdateimerkmale festlegen	[33]
REMOVE-CJC-ACTION	Wirksamkeit von ADD-CJC-ACTION-Kommandos aufheben (Jobvariablen)	[19]
REPEAT	leitet einen REPEAT-Block ein	[21]
RESTART-PROGRAM	startet ein Programm an seinem Fixpunkt	[19]
RESUME-PROCEDURE	unterbrochenen Prozedurablauf fortsetzen	[19]
SELECT-PRODUCT-VERSION	wählt eine Produktversion aus	[19]
SET-JOB-STEP	beendet SPIN-OFF	[19]
SET-LOGON-PARAMETERS	leitet einen Dialog- oder Batch-Auftrag ein	[19]
SHOW-ACCOUNTING-STATUS	informiert über das Abrechnungssystem	[19]
SKIP-COMMANDS	bedingt oder unbedingt springen	[19]
SKIPJV	springen in Abhängigkeit von Jobvariablen	[33]
SKIPUS	springen in Abhängigkeit von Benutzerschaltern	[33]
START-ACCOUNTING	Abrechnungssystem einschalten	[19]
STEP	beendet SPIN-OFF	[33]
STOP-ACCOUNTING	Abrechnungssystem ausschalten	[19]
UNTIL	schließt einen REPEAT-Block ab	[21]
WAIT-EVENT	bedingte Wartezeit (Batch-Auftrag) angeben	[19]
WHEN	Batch-Auftrag bedingt (Benutzerschalter) anhalten	[33]
WHILE	leitet einen WHILE-Block ein	[21]

Tabelle 12: Kommandos, die nicht über den Makro CMD aufgerufen werden können (Teil 2 von 2)

– Beim Aufruf folgender Kommandos durch den Makro **CMD** wird das aufrufende Programm entladen:

Kommando	**Funktion**	**Handbuch**
ABEND	laufenden Auftrag abbrechen	[33]
CALL	Prozedur aufrufen	[33]
CALL-PROCEDURE	Prozedur aufrufen	[19]
DO	Prozedur aufrufen	[33]
EXECUTE	Modul laden und starten	[33]
EXIT-JOB	Auftrag (Job) beenden	[19]
HELP-SDF	Anleitung zum Aufruf von SDF-Kommandos	[19]
LOAD	Modul laden	[33]
LOAD-EXECUTABLE-PROGRAM	Modul laden	[19]
LOAD-PROGRAM	Modul laden	[19]
LOGOFF	Auftrag (Job) beenden	[33]
START-EXECUTABLE-PROGRAM	Modul binden, laden und starten	[19]
START-PROGRAM	Modul binden, laden und starten	[19]

Tabelle 13: Kommandos, bei deren Aufruf das aufrufende Programm entladen wird

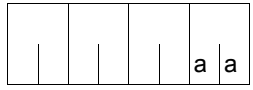
Das aufrufende Programm wird ebenfalls entladen, wenn das aufgerufene Kommando durch eine Kommandoprozedur realisiert ist. Dies ist z.B. für alle EDIT-Kommandos der Fall (siehe Handbuch „Kommandos“ [19]). Es kann aber auch mittels SDF-A selbstdefinierte Kommandos dieser Art geben (siehe Handbuch „SDF-A“ [20]).

Um das Entladen zu verhindern, können Kommandos, die über eine Kommandoprozedur realisiert sind, sowie die Kommandos CALL-PROCEDURE, CALL und DO indirekt über das SDF-P-Kommando INCLUDE-CMD aufgerufen werden (siehe Handbuch „SDF-P“ [21]). Das eigentlich auszuführende Kommando wird dabei als Operand von INCLUDE-CMD angegeben.
INCLUDE-CMD unterbricht das aufrufende Programm, führt das als Operand angegebene Kommando (und damit die zugehörige Kommandoprozedur) aus und kehrt dann wieder zum Programm zurück. Die aufgerufene Kommandoprozedur darf aber ihrerseits kein Kommando ausführen, das zum Entladen des Programms führt.

Das SYSOUT-Protokoll wird bei Kommandos, die das aufrufende Programm beenden, demzufolge nicht in das Empfangsfeld adr übertragen. Eine ggf. definierte STXIT-Routine für die Ereignisklasse „Programmbeendigung“ wird aktiviert.
Eine den Auftrag überwachende Jobvariable wird auf „$T“ gesetzt.

Rückinformation und Fehleranzeigen

R15:

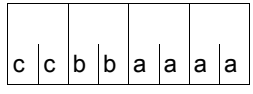


Über die Ausführung des Makros CMD wird im Register R15 rechtsbündig ein Returncode übergeben.

X'aa'	Erläuterung
X'00'	Die Funktion wurde erfolgreich ausgeführt.
X'04'	Die Funktion wurde abgebrochen: Es ist nicht genügend Speicherbereich vorhanden.
X'08'	Die Funktion wurde abgebrochen: Fehler im Datenbereich (Adressbereich).
X'0C'	Die Funktion wurde abgebrochen: Das Empfangsfeld ist zu kurz. Der letze Ausgabesatz, der in das Empfangsfeld gebracht wurde, ist abgeschnitten.
X'10'	Die Funktion wurde abgebrochen: Makroaufruf-/Kommando-Fehler (das Kommando gab dem MCLP einen Fehler zurück).
X'14'	Die Funktion wurde abgebrochen: Ungültiges Kommando in der Operandenliste.
X'24'	Die Funktion wurde abgebrochen: Fehler bei der Substitution der Kommandoeingabe.

zusätzlich bei PARMOD=31:

Standard-header:



Über die Ausführung des Makros CMD wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

cc	bb	aaaa	Erläuterung
00	00	0000	Die Funktion wurde erfolgreich ausgeführt.
00	01	0008	Die Funktion wurde abgebrochen: Operandenfehler.
00	20	0004	Die Funktion wurde abgebrochen: Interner Fehler.
00	40	0010	Die Funktion wurde abgebrochen: Makroaufruf-/Kommandofehler.
00	40	0014	Die Funktion wurde abgebrochen: Ungültiges Kommando in der Operandenliste.
00	40	0024	Die Funktion wurde abgebrochen: Fehler bei der Substitution der Kommandoeingabe.
02	00	000C	Die Funktion wurde abgebrochen: Das Empfangsfeld ist zu kurz. Der letzte Ausgabesatz, der in das Empfangsfeld gebracht wurde, ist abgeschnitten.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Layout der DSECT für VER=4

```
           CMD MF=D,PARMOD=31,VER=4
1          #INTF REFTYPE=REQUEST,INTNAME=CMD,INTCOMP=4
1          DS    0F
1          MFCHK DMACID=CLP,PREFIX=M,MACID=CLP,MF=D,DNAME=DMCLP
2 MDMCLP   DSECT ,
2                *,##### PREFIX=M, MACID=CLP #####
1 MCLPSTRT DS    0F
1          FHDR  MF=(C,MCLP)
2          DS    0A
2 MCLPFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
2 MCLPIFID DS    0A              0   INTERFACE IDENTIFIER
2 MCLPFCTU DS    AL2             0   FUNCTION UNIT NUMBER
2 *                                  BIT 15    HEADER FLAG BIT,
2 *                                  MUST BE RESET UNTIL FURTHER NOTICE
2 *                                  BIT 14-12 UNUSED, MUST BE RESET
2 *                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
2 MCLPFCT  DS    AL1             2   FUNCTION NUMBER
2 MCLPFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 MCLPRET  DS    0A              4   GENERAL RETURN CODE
2 *
2 * GENERAL_RETURN_CODE CLEARED (X'00000000') MEANS
2 * REQUEST SUCCESSFUL PROCESSED AND NO ADDITIONAL INFORMATION
2 *
2 MCLPSRET DS    0AL2            4   SUB RETURN CODE
2 MCLPSR2  DS    AL1             4   SUB RETURN CODE 2
2 * ALWAYS CLEARED (X'00') IF MAIN_RETURN_CODE IS X'FFFF'
2 * Standard subcode2 values as defined by convention:
2 MCLPR2OK EQU   X'00'               All correct, no additional info
2 MCLPR2NA EQU   X'01'               Successful, no action was necessary
2 MCLPR2WA EQU   X'02'               Warning, particular situation
2 MCLPSR1  DS    AL1             5   SUB RETURN CODE 1
2 *
2 * GENERAL INDICATION OF ERROR CLASSES
2 *
2 * CLASS A    X'00'           FUNCTION WAS SUCCESSFULLY PROCESSED
2 * CLASS B    X'01' - X'1F'   PARAMETER SYNTAX ERROR
2 * CLASS C    X'20'           INTERNAL ERROR IN CALLED FUNCTION
2 * CLASS D    X'40' - X'7F'   NO CLASS SPECIFIC REACTION POSSIBLE
2 * CLASS E    X'80' - X'82'   WAIT AND RETRY
2 *
2 MCLPRFSP EQU   X'00'               FUNCTION SUCCESSFULLY PROCESSED
2 MCLPRPER EQU   X'01'               PARAMETER SYNTAX ERROR
2 *  3 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'01' - X'1F'
2 MCLPRFNS EQU   X'01'               CALLED FUNCTION NOT SUPPORTED
```

```
2 MCLPRFNA EQU   X'02'               CALLED FUNCTION NOT AVAILABLE
2 MCLPRVNA EQU   X'03'               INTERFACE VERSION NOT SUPPORTED
2 *
2 MCLPRAER EQU   X'04'               ALIGNMENT ERROR
2 MCLPRIER EQU   X'20'               INTERNAL ERROR
2 MCLPRCAR EQU   X'40'               CORRECT AND RETRY
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'40' - X'7F'
2 MCLPRECR EQU   X'41'               SUBSYSTEM (SS) MUST BE CREATED
2 *                                  EXPLICITELY BY CREATE-SS
2 MCLPRECN EQU   X'42'               SS MUST BE EXPLICITELY CONNECTED
2 *
2 MCLPRWAR EQU   X'80'               WAIT FOR A SHORT TIME AND RETRY
2 MCLPRWLR EQU   X'81'                    "     LONG       "
2 MCLPRWUR EQU   X'82'               WAIT TIME IS UNCALCULABLY LONG
2 *                                  BUT RETRY IS POSSIBLE
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'80' - X'82'
2 MCLPRTNA EQU   X'81'               SS TEMPORARILY NOT AVAILABLE
2 MCLPRDH  EQU   X'82'               SS IN DELETE / HOLD
2 *
2 MCLPMRET DS    0AL2           6    MAIN RETURN CODE
2 MCLPMR2  DS    AL1            6    MAIN RETURN CODE 2
2 MCLPMR1  DS    AL1            7    MAIN RETURN CODE 1
2 *
2 * SPECIAL LAYOUT OF LINKAGE_MAIN_RETURN_CODE (YYYY IN X'00XXYYYY')
2 *
2 MCLPRLNK EQU   X'FFFF'             LINKAGE ERROR / REQ. NOT PROCESSED
2 MCLPFHL  EQU   8              8    GENERAL OPERAND LIST HEADER LENGTH
2 *
1 MCLPFLAG DS    X                     Flag bits
1 *              Flag bits:.0......   : UNUSED
1 *                        0.......   : reserved for TPR usage
1 MCLPDIA  EQU   X'20'      ..1.....   : DIALOG possible (w. SDF only)
1 MCLPNSYS EQU   X'10'      ...1....   : no SYSOUT logging
1 MCLPLIST EQU   X'08'      ....1...   : LIST of commands
1 MCLPNOUT EQU   X'04'      .....10.   : no output buffer
1 MCLPLONG EQU   X'02'      .....01.   : long output buffer
1 *                         .....00.   : short (old) output buffer
1 MCLPREG  EQU   X'01'      .......1    buffer @ given via a register/
1 MCLPADDR EQU   X'00'      .......0    buffer @ given directly
1 MCLPNSRG EQU   X'11'      ...1...1    no SYSOUT, buff.@ in a reg.
1 MCLPNSAD EQU   X'10'      ...1...0    no SYSOUT, buff.@ given directly
1 MCLPFLA3 DS    XL1                   FLAG 3
1 MCLPSJV  EQU   X'80'      10......    substitute jv only
1 MCLPSALL EQU   X'40'      01......    substitute all
1 MCLPDATE EQU   X'20'      ..1.....    data var buf extend
1 MCLPMSGE EQU   X'10'      ...1....    msg var buf extend
1 MCLPCUOR EQU   X'04'      .....1..    origin=current
1 MCLPUNUS DS    XL1                   reserved (unused)
```

```
1 MCLPROUT DS    XL1                    reg.# if buffer @ in reg.
1 MCLPOUT  DS    A                      buffer @ or 0
1 MCLPCMD@ DS    A                      @ of command (-> V-field)
1 MCLPRC@  DS    A                      @ return code of command
1 MCLPTPR2 DS    A              reserved for TPR usage.
1 MCLPDAV@ DS    A                      @ of var name for data
1 MCLPMSV@ DS    A                      @ of var name for message
1 MCLPDAVL DS    H                      length of var name
1 MCLPMSVL DS    H                      length of var name
1 MCLPTPR3 DS    A              reserved for TPR usage.
1 MCLPHLN  EQU   *-MCLPSTRT           CMD p.l. length
1 MCLPLNTH EQU   *       command's V-field when CLPCMD@ points here
1 MCLPCMD  EQU   *+4     cmd start when MCLPCMD@ points to MCLPLNTH
```

Beispiel

Mit dem Makro **CMD** wird das Kommando SHOW-JOB-STATUS in zwei Ausprägungen ausgeführt. Die Programmausführung erfolgt im 31-Bit-Adressierungsmodus unterhalb der 16-MB-Grenze.

```
CMD      START
         PRINT NOGEN
CMD      AMODE ANY
         BALR  3,0
         USING *,3
         CMD   MF=(E,LFORMAD1),PARMOD=31 ------------------------------------  (1)
         CMD   MF=(E,LFORMAD2),PARMOD=31 ------------------------------------  (2)
         MVC   MESSAGE(4),PROTCONT
         MVC   MESSTXT,PROTCONT+4
         WROUT MESSAGE,END,PARMOD=31
END      TERM
LFORMAD1 CMD   'SHOW-JOB-STA','INF=(*STD,*PROGRAM)',MF=L,PARMOD=31 ------  (3)
LFORMAD2 CMD   'SHOW-JOB-STA','INF=*STD',PROT,SYSOUT=NO,MF=L,PARMOD=31    (4)
         DS    OF
PROT     DC    Y(PROTEND-PROT) ------------------------------------------  (5)
         DC    X'4040'
PROTCONT DS    CL2500
PROTEND  EQU   *
MESSAGE  DC    Y(ENDMESS-MESSAGE)      Record length
         DS    CL2                     Reserved
         DC    X'01'                   Print feed control character
MESSTXT  DS    CL255                   Contents
ENDMESS  EQU   *
         END
```

(1) Der Makroaufruf ist in Befehlsteil und Datenbereich aufgespalten (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29). Hier steht nur der Befehlsteil (SVC) mit einem Verweis auf den Datenbereich LFORMADR1 im Datenteil des Programms.

(2) Der Makro wird mit einem Verweis auf den Datenbereich LFORMADR2 aufgerufen.

(3) Das Kommando SHOW-JOB-STATUS wird mit dem Operanden INF=(*STD,*PROGRAM) aufgerufen. Die Ausgabe erfolgt an der Datensichtstation.

(4) Das Kommando SHOW-JOB-STATUS wird mit dem Operanden INF=*STD aufgerufen. Die Information wird in den Bereich `PROT` ausgegeben.

(5) Der Bereich für die Kommando-Ausgabe beginnt an einer Wortgrenze mit dem Eintrag der Bereichslänge in den beiden ersten Byte.

Ablaufprotokoll:

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,cmd), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,cmd))
%  ASS6011 ASSEMBLY TIME: 344 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 80 MSEC
//end
%  ASS6012 END OF ASSEMBH
/start-executable-program library=macexmp.lib,element-or-symbol=cmd
%  BLS0523 ELEMENT 'CMD', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'CMD', VERSION ' ' OF '<date> <time>' LOADED
NAME      TSN TYPE       PRI      CPU-USED CPU-MAX ACCOUNT# ------------  (6)
MACTEST  2QSE 3 DIALOG1  0 210     58.7710    9000 89002
NAME      TSN TYPE        SIZE CURR-CMD
MACTEST  2QSE 3 DIALOG1      1 START-EXECUTABLE-PROGRAM
              PROG::2OSG:$QM212.MACEXMP.LIB(CMD,@,L)
NAME      TSN TYPE       PRI      CPU-USED CPU-MAX ACCOUNT# ------------  (7)
MACTEST  2QSE 3 DIALOG1  0 210     58.7856    9000 89002
```

(6) Ausgabe des ersten **CMD**-Aufrufs direkt auf Datensichtstation.

(7) Der Ausgabebereich PROT des zweiten **CMD**-Aufruf wird mit **WROUT** ausgegeben.

CONTXT – Auf Prozessdaten zugreifen

Allgemeines

Anwendungsgebiet: Contingency-Verfahren; siehe Seite 111
STXIT-Verfahren; siehe Seite 133
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

Wird ein Basis- oder Contingency-Prozess durch einen Contingency-Prozess unterbrochen, so wird u.a. der Inhalt seiner Register und des Befehlszählers im PCB (Process Control Block) abgespeichert.

Makrobeschreibung

Mit dem Makro **CONTXT** hat ein Contingency-Prozess Zugriff auf den Kontext (PCB, Process Control Block) eines unterbrochenen Prozesses.

CONTXT unterstützt alle aktuellen BS2000-Server. Demzufolge kann der unterbrochene Prozess in /390-Code mit /390-Kontext (/390-Server) oder in x86-Code mit x86-Kontext (x86-Server) vorliegen. Da sich die Kontexte unterscheiden, wird der Operand LAYOUT zur Unterscheidung verwendet.

Für LAYOUT=COMPATIBLE (Voreinstellung) werden für einen unterbrochenen /390-Prozess der vollständige Kontext (Register und PC) gelesen. Der gesamte Kontext kann verändert und geschrieben werden.
Für einen unterbrochenen x86-Prozess werden äquivalente Teile des Kontextes auf die entsprechenden Bereiche des Layouts abgebildet. Der Kontext wird also nicht vollständig ausgegeben. Er kann auch nur in Teilen geschrieben werden.

Für LAYOUT=FCONTXT wird der vollständige Kontext eines unterbrochenen x86-Prozesses gelesen. ILC wird gebildet. CC und PM sind ohne Bedeutung.
Der vollständige Kontext eines unterbrochenen /390-Prozesses wird auf die entsprechenden Teilbereiche des Layouts abgebildet.
Der gesamte Kontext eines Prozesses kann verändert und geschrieben werden. Für x86-Prozesse können CC, ILC und PM nicht geschrieben werden.

Siehe auch „PCB-Zugriffe im Prozess-Modus /390“ auf Seite 346.

Wenn x86-Code in Benutzeranwendungen (/390-Code) eingebunden oder nachgeladen wird (z. B. Nachladen des portierten Produktes SORT), kann an dieser Schnittstelle sowohl der Kontext von x86-Code als auch der Kontext von /390-Code sichtbar werden, je nachdem in welchem Modus die Unterbrechung stattfand. Programme, die vollständig im /390-Modus ablaufen, sind nicht betroffen.

Folgende Unterschiede im Kontext werden sichtbar, wenn ein x86-Programm unterbrochen wurde und der Kontext des unterbrochenen (x86-)Prozesses betrachtet wird (`CONTXT SAVE=...,LAYOUT=FCONTXT`):

– der Kontext enthält die /390-äquivalenten Teile (Register, PC, Gleitpunkregister, usw.), die auch über die DSECT adressiert werden können und

– einen HSI-abhängigen Bereich, der in Blockform gelesen oder geschrieben werden kann

Makroaufrufformat und Operandenbeschreibung

CONTXT

[{ SAVE=adr / (r) }]
[{ STACKR=(x1, x2, ...),OWNR=(y1, y2,...) }

[, { SAVACR=adr / (r) }]
[, { STKACR=(x1, x2, ...),OWNACR=(y1, y2,...) }

,FPR=NO / YES

,PROCESS=MAIN / LAST

,FUNCT=READ / WRITE

,LAYOUT=COMPATIBLE / FCONTXT

,LAYOUTF=DSECT[,PREFIX=p]

[,ILC=adr / (r)]

[,CC=adr / (r)]

[,PM=adr / (r)]

[,PMODE=adr / (r)]

[,MODE=adr / (r)]

[,ASCMOD=adr / (r)]

[,PRGCODE=adr / (r)]

[,PRGCODL=adr / (r)]

[,PARMOD=24 / 31]

[,MF=L / (E,..)]

Die Operanden ILC, CC, PM sind für x86-Server ohne Bedeutung.

SAVE=
Beschreibt die Adresse eines Feldes für den Datenaustausch mit dem PCB des angegebenen Prozesses (Operand PROCESS). Aufbau und Inhalt dieses Feldes hängen vom Operanden LAYOUT ab.

– *LAYOUT=COMPATIBLE (Voreinstellung)*
Das Datenaustauschfeld hat denselben Aufbau wie bei der bisherigen CONTXT-Schnittstelle im /390-Modus.
Feldlänge = 68 Byte; Ausrichtung auf Wortgrenze nötig.
Feldaufbau und Zuordnung:

```
Byte  0 bis Byte 63: Register R0 bis R15 des /390-PCB
                     äquivalente Register des x86-PCB
Byte 64 bis Byte 67: Adresse des nächsten Befehls (PC / NIA)
```

– *LAYOUT=FCONTXT*
Zur Unterstützung des x86-Modus hat das Datenaustauschfeld einen eigenen Aufbau. Es enthält neben dem /390-PCB auch den gesamten x86-PCB und damit auch die Gleitpunktregister. Diese werden unabhängig vom Operanden FPR zwischen dem Datenaustauschfeld und dem PCB des angegebenen Prozesses übertragen.

Falls der Prozess, auf dessen PCB zugegriffen wird, jedoch im /390-Modus abläuft, findet eine eingeschränkte Übertragung statt. Es werden nur die äquivalenten Bereiche des Datenaustauschfeldes versorgt bzw. gelesen.

Die Feldlänge des Datenaustauschfeldes kann mit dem Makro **STXIT**, Operand CONTXTL, dynamisch ermittelt werden. Die Validierung des Datenaustauschfeldes geschieht aber immer in der jeweils benutzten Länge. Das Datenaustauschfeld muss auf Doppelwortgrenze ausgerichtet sein.

Durch Aufruf des Makros **CONTXT** mit dem Operanden LAYOUTF=DSECT kann eine DSECT erzeugt werden, die den neuen Aufbau des Datenaustauschfeldes beschreibt. Mit dieser DSECT ist es möglich, die einzelnen Teilfelder des Datenaustauschfeldes symbolisch zu adressieren.

Bei FUNCT=READ wird der Kontext aus dem PCB des angegebenen Prozesses in das spezifizierte Feld übertragen.

Bei FUNCT=WRITE wird der Inhalt des spezifizierten Feldes in den Kontext des angegebenen Prozesses übertragen. Der Befehlszähler (PC / NIA = Next Instruction Address) kann nur dann geschrieben werden, wenn er in ein /390-Modul zeigt oder LAYOUT= FCONTXT angegeben wurde.

adr
symbolische Adresse (Name) des Feldes für den Datenaustausch

(r)
r = Register mit dem Adresswert von adr

STACKR=
Bezeichnet eine Folge ausgewählter Register (einschließlich Befehlszähler) des angegebenen Prozesses für den Datenaustausch. Für x86-Kontexte werden nur die Äquivalente der /390-Register R0 bis R15 unterstützt. Der Befehlszähler (PC / NIA = Next Instruction Address) darf nur dann geschrieben werden, wenn er in ein /390-Modul zeigt oder wenn LAYOUT=FCONTXT angegeben wurde. Für das Lesen des Befehlszählers gibt es keine derartige Einschränkung.

Hinweis
Bei den Operanden STACKR und OWNR muss jeweils die gleiche Anzahl von Operanden angegeben werden. Die Angaben in den Klammern werden für die Übertragung einander paarweise zugeordnet. Es erfolgt also eine Übertragung zwischen x1 und y1, x2 und y2, x3 und y3, usw.

(x1,x2,...)
x1,x2 = Elemente aus der Menge (0,1,2,.....,15,PC).
Die Zahlen 0, ...,15 stehen für die allgemeinen Register und PC für den Befehlszähler.
Bezüglich des Befehlszählers wird nur die Adresse des nächsten Befehls übertragen.

OWNR=
bezeichnet eine Folge ausgewählter Register des aufrufenden (Contingency-)Prozesses, die die gelesenen Werte aufnehmen sollen oder die die zu schreibenden Werte enthalten (siehe auch Hinweis für STACKR).

(y1,y2,...)
y1,y2 = Elemente aus der Menge (0,1,2,.....,15,PC).
Die Zahlen 0, ...,15 stehen für die allgemeinen Register und PC für den Befehlszähler.
Bei der Verwendung der Operanden STACKR und OWNR ist zu beachten, dass die Register R1 und R15 durch den Makroaufruf zerstört werden (Register R1: Adresse des Datenbereichs, Register R15: Rücksprunginformation). Daher ist „Schreiben“ des Inhalts von Register R1 und „Lesen“ in das Register R15 nicht möglich.

SAVACR=
Beschreibt die Adresse eines Feldes für den Datenaustausch mit dem PCB des angegebenen Prozesses (Operand PROCESS).
Feldlänge = 64 Byte; Ausrichtung auf Wortgrenze erforderlich.
Bei FUNCT=READ wird der Inhalt der Zugriffsregister AR0 bis AR15 aus dem PCB des angegebenenen Prozesses in das spezifizierte Feld übertragen.
Bei FUNCT=WRITE wird der Inhalt des spezifizierten Feldes entsprechend obiger Zuordnung in die Zugriffsregister des angegebenen Prozesses übertragen.

adr
symbolische Adresse (Name) des Feldes für den Datenaustausch

(r)
r = Register mit dem Adresswert von adr

STKACR=
Bezeichnet eine Folge ausgewählter Zugriffsregister des angegebenen Prozesses für den Datenaustausch.

Hinweis
Bei den Operanden STKACR und OWNACR muss jeweils die gleiche Anzahl von Operanden angegeben werden. Die Angaben in den Klammern werden für die Übertragung einander paarweise zugeordnet. Es erfolgt also eine Übertragung zwischen x1 und y1, x2 und y2, x3 und y3, usw.

(x1,x2,...)
x1,x2 = Elemente aus der Menge (0,1,2,.....,15).
Die Zahlen 0, ...,15 stehen für die Zugriffsregister.

OWNACR=
bezeichnet eine Folge von Zugriffsregistern des aufrufenden (Contingency-)Prozesses, die die gelesenen Werte aufnehmen sollen oder die die zu schreibenden Werte enthalten (siehe auch Hinweis für STKACR).

(y1,y2,...)
y1,y2 = Elemente aus der Menge (0,1,2,.....,15).
Die Zahlen 0, ...,15 stehen für die Zugriffsregister.

FPR=
gibt an, ob die Inhalte der Gleitpunktregister übertragen werden sollen.
Dieser Operand ist nur in Verbindung mit LAYOUT=COMPATIBLE (Voreinstellung) von Bedeutung.
Läuft einer der beteiligten Prozesse im x86-Modus, sind von dessen Gleitpunktregistern nur diejenigen an der Übertragung beteiligt, die Äquivalente der /390-Gleitpunktregister darstellen.

NO
Die Inhalte der Gleitpunktregister werden nicht übertragen.

YES
Bei FUNCT=READ wird der Inhalt der Gleitpunktregister des angegebenen Prozesses in die Gleitpunktregister des aufrufenden (Contingency-) Prozesses übertragen.
Bei FUNCT=WRITE wird der Inhalt der Gleitpunktregister des aufrufenden (Contingency-) Prozesses in die Gleitpunktregister des angegebenen Prozesses übertragen.

Bei LAYOUT=FCONTXT findet grundsätzlich keine direkte Übertragung der Gleitpunktregister des aufrufenden Prozesses in die Gleitpunktregister des angegebenen Prozesses oder umgekehrt statt. Stattdessen werden die Gleitpunktregister über das Datenaustauschfeld übertragen, das mit dem Operanden SAVE festgelegt ist. Die Angabe des Operanden FPR wird ignoriert.
Dabei werden unabhängig vom Ablaufmodus des aufrufenden Prozesses alle Gleitpunktregister des x86-Modus übertragen, falls der angegebene Prozess im x86-Modus abläuft.

PROCESS=
gibt an, zu welchem Prozess der Zugriff gewünscht wird.

MAIN
Der Zugriff erfolgt zum Basisprozess, auch wenn der Basisprozess nicht durch den aufrufenden (Contingency-)Prozess unterbrochen wurde.

LAST
Der Zugriff erfolgt zu dem Prozess, der durch den aufrufenden Prozess unterbrochen wurde; das kann ein Contingency- oder der Basisprozess sein.

FUNCT=
gibt an, ob ein Lese- oder Schreibzugriff ausgeführt werden soll.
Siehe auch „PCB-Zugriffe im Prozess-Modus /390“ auf Seite 346.

READ
Die Inhalte der spezifizierten Register und ggf. des Befehlszählers bzw. der komplette x86-Kontext (bei LAYOUT=FCONTEXT auf x86-Servern) werden aus dem PCB des angegebenen Prozesses gelesen und in die angegebenen Felder übertragen.

WRITE
Die spezifizierten Register und ggf. der Befehlszähler bzw. der komplette x86-Kontext (bei LAYOUT=FCONTEXT auf x86-Servern) des angegebenen Prozesses werden mit den Werten überschrieben, die der aufrufende Prozess angibt.
Der Schreibzugriff ist nur möglich, wenn der Speicherschlüssel im PCB des aufrufenden Prozesses mit dem im PCB des angegebenen Prozesses übereinstimmt.

Im /390-Prozess-Modus: Der Inhalt der durch ILC/CC/PM/ASCMOD beschriebenen Felder wird in den PCB des angegebenen Prozesses (unterbrochener Prozess oder Basisprozess) eingetragen. Die Information ist in den korrespondierenden Bits zu hinterlegen.

LAYOUT=
legt Umfang und Aufbau des Datenaustauschfeldes fest, dessen Adresse im Operanden SAVE angegeben wird. Er steuert auch die Funktion „Schreiben NIA“ über den Operanden STACKR=(PC), siehe Seite 334.

COMPATIBLE
Umfang und Layout des /390-Kontexts werden erwartet. Die Gleitpunktregister werden nur übertragen, wenn gleichzeitig FPR=YES angegeben wurde.

FCONTXT
Umfang und Layout des jeweiligen x86-Kontextes werden erwartet. Für x86-Kontexte werden die Gleitpunktregister, unabhängig vom Operanden FPR, über das Datenaustauschfeld übertragen. Die Länge der dafür vom Benutzer bereitzustellenden SAVE-Area kann mit dem Makro **STXIT**, Operand CONTXTL, dynamisch ermittelt werden. Die tatsächlich benötigte Länge hängt vom Prozess-Modus ab. Zur Länge und zum Layout des jeweiligen Datenaustauschfeldes siehe "Layout der DSECT" auf Seite 341.

LAYOUTF=DSECT
veranlasst die Erzeugung einer DSECT für das Datenaustauschfeld wie beim Operanden LAYOUT=FCONTXT angegeben. Die Adresse des Datenaustauschfeldes wird beim Aktionsaufruf im Operanden SAVE angegeben. Bei Angabe dieses Operanden wird als einziger weiterer Operand PREFIX ausgewertet. Die DSECT ist auf Seite 341 abgedruckt.

PREFIX=p
Angabe eines Buchstabens, mit dem das erste Zeichen der Feldnamen und der Equates bestimmt wird. Dieser Operand wird nur berücksichtigt, wenn gleichzeitig LAYOUTF= DSECT angegeben ist.

ILC, CC, PM=
Diese Operanden sind für einen x86-PCB ohne Bedeutung. Sie können nicht in einen x86-PCB geschrieben werden.

ILC=
beschreibt die Adresse eines Feldes für den Befehlslängencode (im PCR-Format). Feldlänge = 1 Byte. Eintrag in Bit 0-1.

adr
symbolische Adresse (Name) des Feldes für den Befehlslängencode

(r)
r = Register mit dem Adresswert von adr

CC=
beschreibt die Adresse eines Feldes für den Condition Code (im PCR-Format). Feldlänge = 1 Byte. Eintrag in Bit 2-3.

adr
symbolische Adresse (Name) des Feldes für den Condition Code

(r)
r = Register mit dem Adresswert von adr

PM=
beschreibt die Adresse eines Feldes für die Programmaske (im PCR-Format).
Feldlänge = 1 Byte. Eintrag in Bit 4-7.

adr
symbolische Adresse (Name) des Feldes für die Programmaske

(r)
r = Register mit dem Adresswert von adr

PMODE=
Beschreibt die Adresse eines Feldes für den Prozessormodus.
Feldlänge = 1 Byte.
Der Prozessormodus kann auf x86-Servern gelesen oder geändert werden. Auf anderen BS2000-Servern kann er nur gelesen werden.

Es bedeuten:
X'00': /390-Modus (native auf /390-Servern oder unter /390-Firmware auf x86-Servern)
X'01': x86-Modus native

adr
symbolische Adresse (Name) des Feldes für den Prozessormodus

(r)
r = Register mit dem Adresswert von adr

MODE=
beschreibt die Adresse eines Feldes für den Adressierungsmodus. Feldlänge = 1 Byte.
Der Adressierungsmodus des angegebenen Prozesses kann gelesen oder geändert werden.
Es bedeuten:
X'00': 24-Bit-Adressierungsmodus
X'01': 31-Bit-Adressierungsmodus

Bei FUNCT=READ wird der Adressierungsmodus aus dem PCB in das angegebene Feld übertragen.
Bei FUNCT=WRITE wird der angegebene Adressierungsmodus in den PCB übertragen.
Bei einer Änderung des Adressierungsmodus muss der Anwender auch für einen gültigen 24-Bit- oder 31-Bit-Kontext sorgen.

adr
symbolische Adresse (Name) des Feldes für den Adressierungsmodus

(r)
r = Register mit dem Adresswert von adr

ASCMOD=
Beschreibt die Adresse eines Feldes für den ASC-Modus (Address Space Control). Feldlänge = 1 Byte. Der Adressierungsmodus des angegebenen Prozesses kann gelesen oder geändert werden.
Es bedeuten:
X'00': Program-Space-Mode
X'40': Access-Register-Mode (ASC-Modus), das Programm läuft im AR-Modus, siehe auch Abschnitt „Erweiterung durch Datenräume“ auf Seite 61.

adr
symbolische Adresse (Name) des Feldes für den ASC-Modus

(r)
r = Register mit dem Adresswert von adr

PRGCODE=
beschreibt die Adresse eines Feldes für den Programm-Code. In diesem Feld wird der unterbrochene Befehl bzw. der durch NIA (**N**ext **I**nstruction **A**ddress) festgelegte Befehl (aus dem spezifizierten Prozess: LAST- oder MAIN-PCB) linksbündig zur Verfügung gestellt. Nur Lesezugriff erlaubt. Feldlänge = 6 Byte. Angabe nur zusammen mit dem Operanden PRGCODL.

adr
symbolische Adresse (Name) des Feldes für den Programm-Code

(r)
r = Register mit dem Adresswert von adr

PRGCODL=
beschreibt die Adresse eines Feldes für den Programm-Code. In diesem Feld wird die Länge dieses Befehls abgelegt. Feldlänge = 1 Byte.
Angabe nur zusammen mit dem Operanden PRGCODE.
Folgende Werte sind möglich:
> 0: (=2 oder =4 oder =6): Länge des ermittelten Programm-Codes.
= 0: Anzeige: Der Programm-Code konnte nicht ermittelt werden. Das Feld `PRGCODE` bleibt unverändert.

adr
symbolische Adresse (Name) des Feldes für die Länge des Programm-Codes.

(r)
r = Register mit dem Adresswert von adr

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB).

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Layout der DSECT für LAYOUT=FCONTXT

```
CONTXT LAYOUTF=DSECT
1 *
1 ********************************************************************
1 *           DSECT  FOR  SAVE-FIELD  BY  LAYOUT = FCONTXT            *
1 ********************************************************************
1 *
1 SFCONTXT   DSECT
1 *
1 SAVEHSI    DS    X            HSI INDICATOR
1 SHSI390    EQU   X'01'          /390 HSI
1 SHSI390E   EQU   X'03'          /390 HSI + ESA
1 SHSIRISC   EQU   X'04'          RISC HSI  (NOT USED)
1 SHSISPAC   EQU   X'08'          SPARC HSI
1 SHSISPME   EQU   X'0A'          SPARC HSI + ESA
1 SHSISXI    EQU   X'10'          IA64 HSI
1 SHSISXE    EQU   X'12'          IA64 HSI+ ESA
1 SHSIX86    EQU   X'20'          X86 HSI
1 SHSIX86E   EQU   X'22'          X86 HSI+ ESA
1 *
1 SAVEAMOD   DS    X            ADDRESS MODE
1 SAMODE24   EQU   X'00'          24-BIT ADDRESS MODE
1 SAMODE31   EQU   X'01'          31-BIT ADDRESS MODE
1 *
1 SAVEPMOD   DS    X            PROCESSOR MODE
1 SPMODE1    EQU   X'00'          "/390" (/390 - MASCHINE)  RESP.
1 *                               "/390-EMULATION" (SPARC-MASCHINE)
1 SPMODE2    EQU   X'01'          "SPARC " (SPARC-MASCHINE)
1 *
1 SAVEILC    DS    X            INSTRUCT LENGTH CODE (PCR-FORMAT:BIT 0-1)
1 SAVECC     DS    X            CONDITION CODE (PCR-FORMAT: BIT 2-3)
1 SAVEPM     DS    X            PROGRAM MASK (PCR-FORMAT: BIT 4-7)
1 *
1 SAVEASCM   DS    X            ASC-MODE (ESA)  >> NOT USED ON RISC
1 *
1            DS    XL1          UNUSED
1 *
1 *-----------------------------------------------------------------
1 *          PROCESS MODE DEPENDENT AREA
1 *-----------------------------------------------------------------
1 *
1 SAV390A    DS    0D         AREA FOR PMODE = /390 AND SPARC
1 *
1 *                           ( RISC )  &  /390   GENERAL REGISTERS
1 *
1 SAVERR0    DS    2F           R0    )       (HARD-WIRED TO ZERO ! )
1 SAVERR1    DS    2F           R1    )
```

```
1 SAVERR2    DS    2F          R2    )
1 SAVERR3    DS    2F          R3    )
1 SAVERR4    DS    2F          R4    >> RISC ONLY
1 SAVERR5    DS    2F          R5    )
1 SAVERR6    DS    2F          R6    )
1 SAVERR7    DS    2F          R7    )
1 *
1 SAVERR8    DS    2F          R8
1 SAVEGR0    EQU   SAVERR8+4     /390:  R0  EQUIVALENT
1 SAVERR9    DS    2F          R9
1 SAVEGR1    EQU   SAVERR9+4     /390:  R1  EQUIVALENT
1 SAVERR10   DS    2F          R10
1 SAVEGR2    EQU   SAVERR10+4    /390:  R2  EQUIVALENT
1 SAVERR11   DS    2F          R11
1 SAVEGR3    EQU   SAVERR11+4    /390:  R3  EQUIVALENT
1 SAVERR12   DS    2F          R12
1 SAVEGR4    EQU   SAVERR12+4    /390:  R4  EQUIVALENT
1 SAVERR13   DS    2F          R13
1 SAVEGR5    EQU   SAVERR13+4    /390:  R5  EQUIVALENT
1 SAVERR14   DS    2F          R14
1 SAVEGR6    EQU   SAVERR14+4    /390:  R6  EQUIVALENT
1 SAVERR15   DS    2F          R15
1 SAVEGR7    EQU   SAVERR15+4    /390:  R7  EQUIVALENT
1 SAVERR16   DS    2F          R16
1 SAVEGR8    EQU   SAVERR16+4    /390:  R8  EQUIVALENT
1 SAVERR17   DS    2F          R17
1 SAVEGR9    EQU   SAVERR17+4    /390:  R9  EQUIVALENT
1 SAVERR18   DS    2F          R18
1 SAVEGR10   EQU   SAVERR18+4    /390:  R10 EQUIVALENT
1 SAVERR19   DS    2F          R19
1 SAVEGR11   EQU   SAVERR19+4    /390:  R11 EQUIVALENT
1 SAVERR20   DS    2F          R20
1 SAVEGR12   EQU   SAVERR20+4    /390:  R12 EQUIVALENT
1 SAVERR21   DS    2F          R21
1 SAVEGR13   EQU   SAVERR21+4    /390:  R13 EQUIVALENT
1 SAVERR22   DS    2F          R22
1 SAVEGR14   EQU   SAVERR22+4    /390:  R14 EQUIVALENT
1 SAVERR23   DS    2F          R23
1 SAVEGR15   EQU   SAVERR23+4    /390:  R15 EQUIVALENT
1 *
1 SAVERR24   DS    2F          R24   )
1 SAVERR25   DS    2F          R25   )
1 SAVERR26   DS    2F          R26   )
1 SAVERR27   DS    2F          R27   >> RISC ONLY
1 SAVERR28   DS    2F          R28   )
1 SAVERR29   DS    2F          R29   )
1 SAVERR30   DS    2F          R30   )       (ADDRESS MODE MASK)
1 SAVERR31   DS    2F          R31   )
```

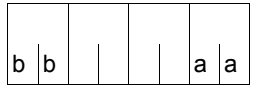
```
1 *
1 *
1 SAVENIA    DS    2F          NIA/PC (NEXT INSTRUCTION ADDRESS)
1 SVNIA390   EQU   SAVENIA+4     /390:  NIA-EQUIVALENT
1 *
1 *
1 SAVEHI     DS    2F          MULTIPLE/DIVIDE REG HI RESULT (RISC ONLY)
1 SAVELO     DS    2F          MULTIPLE/DIVIDE REG LO RESULT (RISC ONLY)
1 *
1 *
1 *                          ( RISC )  &  /390  FLOATING POINT REGISTERS
1 *
1 SAVEF0     DS    F           F0 : F0/1   = /390: EXT FPR 8  EQUIVALENT
1 SAVEF1     DS    F           F1
1 SAVEF2     DS    F           F2 : F2/3   = /390: EXT FPR 10 EQUIVALENT
1 SAVEF3     DS    F           F3
1 SAVEF4     DS    F           F4 : F4/5   = /390: EXT FPR 12 EQUIVALENT
1 SAVEF5     DS    F           F5
1 SAVEF6     DS    F           F6 : F6/7   = /390: EXT FPR 14 EQUIVALENT
1 SAVEF7     DS    F           F7
1 SAVEF8     DS    F           F8 : F8/9   = /390: EXT FPR 1  EQUIVALENT
1 SAVEF9     DS    F           F9
1 SAVEF10    DS    F           F10: F10/11 = /390: EXT FPR 3  EQUIVALENT
1 SAVEF11    DS    F           F11
1 SAVEF12    DS    F           F12: F12/13 = /390: EXT FPR 5  EQUIVALENT
1 SAVEF13    DS    F           F13
1 SAVEF14    DS    F           F14: F14/15 = /390: EXT FPR 7  EQUIVALENT
1 SAVEF15    DS    F           F15
1 SAVEF16    DS    F           F16: F16/17 = /390: EXT FPR 9  EQUIVALENT
1 SAVEF17    DS    F           F17
1 SAVEF18    DS    F           F18: F18/19 = /390: EXT FPR 11 EQUIVALENT
1 SAVEF19    DS    F           F19
1 *
1 SAVEF20    DS    2F          F20/21  =   /390:  FPR 0  EQUIVALENT
1 SAVEF22    DS    2F          F22/23  =   /390:  FPR 2  EQUIVALENT
1 SAVEF24    DS    2F          F24/25  =   /390:  FPR 4  EQUIVALENT
1 SAVEF26    DS    2F          F26/27  =   /390:  FPR 6  EQUIVALENT
1 *
1 SAVEF28    DS    F           F28: F28/29 = /390: EXT FPR 13 EQUIVALENT
1 SAVEF29    DS    F           F29
1 SAVEF30    DS    F           F30: F30/31 = /390: EXT FPR 15 EQUIVALENT
1 SAVEF31    DS    F           F31
1 *
1 SAVEFCR    DS    F         FP-CONTROL-/STATUS REG >> NOT USED ON RISC
1 *
1 *
1 *                          /390-ESA  ACCESS REGISTERS
1 *
```

```
1 SAVEAR0    DS    F            ACR0       )
1 SAVEAR1    DS    F            ACR1       )
1 SAVEAR2    DS    F            ACR2       )
1 SAVEAR3    DS    F            ACR3       >> NOT USED ON RISC
1            DS    9F           ACR4-ACR12 )
1 SAVEAR13   DS    F            ACR13      )
1 SAVEAR14   DS    F            ACR14      )
1 SAVEAR15   DS    F            ACR15      )
1 *
1 SWOSPARC   EQU   *-SFCONTXT   LENGTH   -  WITHOUT SPARC-BLOCK
1 *
1 *-------------------------------------------------------------
1 *                          SPARC  CONTEXT  BLOCK
1 *-------------------------------------------------------------
1 *
1            DS    0D                              )
1 SSPARCB    DS    100D         SPARC-AREA: BEGIN  )
1            DS    100D                            >> /390: NOT USED
1            DS    15D                             )
1 SSPARCE    DS    0D           SPARC-AREA: END    )
1 SLSPARCB   EQU   SSPARCE-SSPARCB       LENGTH OF SPARC-BLOCK (SPARC)
1 *
1 *-------------------------------------------------------------
1 *
1 SAVLNGTH   EQU   *-SFCONTXT   LENGTH  OF  SAVE-FIELD
1 *
1 *-------------------------------------------------------------
1 *         IA64 CONTEXT  BLOCK: FCONTEXT
1 *-------------------------------------------------------------
1 *
1            ORG   SSPARCB      REDEFINITION OF NATIVE AREA
1 SSXIB      DS    0D         IA64-AREA: BEGIN
1 SLIA64     EQU   SWOSPARC+4064        LENGTH OF IA64 CONTEXT
1 *
1 *
1 *
1 *-------------------------------------------------------------
1 *         X86  CONTEXT  BLOCK: FCONTEXT
1 *-------------------------------------------------------------
1 *
1            ORG   SSPARCB      REDEFINITION OF NATIVE AREA
1 SX86E      DS    0D         X86-AREA: BEGIN
1 SLX86E     EQU   SWOSPARC+4096        LENGTH OF X86 CONTEXT
1 *
1 *-------------------------------------------------------------
```

Rückinformation und Fehleranzeigen

Während der Makrobearbeitung enthält Register R1 die Adresse der Operandenliste.

R15:



Über die Ausführung des Makros CONTXT wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.
aa=X'00': normale Ausführung
aa≠X'00': Funktion nicht ausgeführt.

X'bb'	X'aa'	Erläuterung
X'00'	X'00'	normale Ausführung. PCB wurde (im aktuellen Unterbrechungszustand) noch nicht verändert.
X'04'	X'00'	normale Ausführung. PCB wurde (im aktuellen Unterbrechungszustand) bereits mit CONTXT verändert.
X'04'	X'04'	Funktion nicht ausgeführt. Ungültige Operanden.
X'04'	X'08'	Funktion nicht ausgeführt. Makroaufruf erfolgte im Basisprozess.
X'04'	X'18'	Schreiben nicht zulässig Folgende Ursachen können vorliegen: – Schreiben von PRGCODE – Schreiben von CC, ILC, PM in x86-PCB – Schreiben eines Befehlszählers, der in ein x86-Modul zeigt, bei LAYOUT=COMPATIBLE
X'04'	X'1C'	Funktion nicht ausgeführt. Bei FUNCT=WRITE stimmt der Speicherschlüssel im zu ändernden PCB nicht mit dem im aktuellen PCB überein
X'04'	X'20'	Funktion nicht ausgeführt: kein Zugriff auf die Zugriffsregister möglich.

PCB-Zugriffe im Prozess-Modus /390

Die folgende Tabelle zeigt das Ergebnis der Zugriffe auf den PCB in Abhängigkeit von den CONTXT-Operanden, vom Ablaufmodus des unterbrochenen Prozesses und der Angabe im Operanden LAYOUT:

CONTXT Operand	**CONTXT FUNCT=**	**LAYOUT=COMPATIBLE**		**LAYOUT=FCONTXT**	
		/390-PCB	**x86-PCB**	**/390-PCB**	**x86-PCB**
SAVE	READ	(1)	(2)	(3)	(4)
	WRITE		RC=X'18' (*)		
OWNR/STACKR (ohne PC/NIA)	READ	wie bisher	R0-R15	wie bisher	R0-R15
	WRITE				
STACKR=(PC)	READ	wie bisher	x86 native NIA	wie bisher	x86 native NIA
	WRITE		RC=X'18' (*)		
SAVACR/ OWNACR / STKACR / ASCMOD	READ	wie bisher	wie bisher	wie bisher	wie bisher
	WRITE				
FPR	READ	(5)	(5)	(6)	(7)
	WRITE				
ILC	READ	wie bisher	Wert: 0 oder 4	wie bisher	Wert: 0 oder 4
	WRITE		RC=X'18'		RC=X'18'
CC	READ	wie bisher	Wert: X'00'	wie bisher	Wert: X'00'
	WRITE		RC=X'18'		RC=X'18'
PM	READ	wie bisher	Wert: X'00'	wie bisher	Wert: X'00'
	WRITE		RC=X'18'		RC=X'18'
MODE	READ	wie bisher	wie bisher	wie bisher	wie bisher
	WRITE				
PMODE	READ	X'00'	X'01'	X'00'	X'01'
	WRITE	(8)	(8)	(8)	(8)

Bedeutung der in der Tabelle verwendeten Abkürzungen:

R0 - R15: allgemeine Register
RC: Returncode

Bedeutung der in der Tabelle verwendeten Anmerkungen:

(*): Der Befehlszähler (PC / NIA = Next Instruction Address) darf nur dann geschrieben werden, wenn er in ein /390-Modul zeigt (mit `LAYOUT= COMPATIBLE`) oder wenn `LAYOUT=FCONTXT` angegeben wurde.

(1): Das Datenaustauschfeld hat den gleichen Aufbau wie bisher.

(2): Das Datenaustauschfeld hat den gleichen Aufbau wie bisher.
Es werden nur die zu /390-Registern äquivalenten Register und NIA gelesen oder geschrieben.

(3): Das Datenaustauschfeld hat den neuen Aufbau. Es werden nur die zu den /390-Registern äquivalenten Register, die zu den /390-Gleitpunktregistern äquivalenten x86-Gleitpunktregister, NIA und die weiteren Prozeßstati gelesen oder geschrieben (soweit überschreibbar). Der Austausch erfolgt über die entsprechenden Felder `SAVExxxx`

(4): Zusätzlich zu (3) wird der vollständige x86-Kontext gelesen oder geschrieben (über den Austauschbereich `SSPARCB` in der Länge `SLSPARCB`). Das Ändern einzelner Daten im Kontext ist nur über die Felder `SAVExxxx` möglich.

(5): Die Datenübertragung findet zwischen den /390-Gleitpunktregistern des angegebenen PCB und denen des Contingency-PCB statt.

(6): Nur über den Operanden SAVE, siehe (3)

(7): Nur über den Operanden SAVE, siehe (4)

(8): Das Schreiben des PMODE wird erlaubt, um z.B. den unterbrochenen PCB auf einer zentralen Beendigungsroutine wieder aufzusetzen, deren PMODE nicht der unterbrochene PMODE ist. Sinnvoll ist das Schreiben des PMODE nur, wenn gleichzeitig NIA verändert wird.

CRYPT – Wörter verschlüsseln

Allgemeines

Anwendungsgebiet: Verschlüsselung von Wörtern; siehe Seite 165
Makrotyp: S-Typ, MF-Format **3**: D-/C-/M-/E-/L-Form; siehe Seite 29

Makrobeschreibung

Der Makro **CRYPT** dient der Einweg-Verschlüsselung von Wörtern mit einer maximalen Länge von 8 Byte. Einweg-Verschlüsselung heißt, dass eine Entschlüsselung der mit **CRYPT** verschlüsselten Wörter nicht möglich ist. Im Ergebnis der Makroausführung wird eine 4 Byte oder 8 Byte lange Zeichenkette zurückgeliefert.

Makroaufrufformat und Operandenbeschreibung

```
CRYPT
INSTRA=<var: pointer> / (<reg: pointer>)
,INSTRL=4 / 8 / <var: int:1>
,OUSTRA=<var: pointer> / (<reg: pointer>)
,CRYALG=*SCA / *SCAVK / *OLD / <var: enum-of _ecrt_s:1>
,CRCL2OP=*YES / *NO / <var: enum-of _cl2op_s:1>
,VKEYA=<var: pointer> / (<reg: pointer>)
,XPAND=*INPAR / *KEYPAR
,MF=D / C / M / E / L
[,PARAM = adr / (r)]
,PREFIX=S / p
,MACID=RME / macid
```

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

CRCL2OP=
verschlüsselt das Eingabewort in Abhängigkeit vom Systemparameter ENCRYPTION.

***YES**
Es wird abhängig vom Systemparameter ENCRYPTION verschlüsselt.

***NO**
Es wird immer verschlüsselt, also unabhängig vom Systemparameter ENCRYPTION.

<var: enum-of _cl2op_s:1>
Name des Feldes mit der Art der Verschlüsselung.

CRYALG=
wählt den Verschlüsselungsalgorithmus aus.

<u>*SCA</u>
Es wird der Verschlüsselungsalgorithmus SCA verwendet.

***SCAVK**
Es wird der Verschlüsselungsalgorithmus SCA und der im Schlüsselfeld (Operand VKEYA) angegebene Schlüssel verwendet.

***OLD**
Der bisher verwendete Verschlüsselungsalgorithmus wird verwendet.

<var: enum-of _ecrt_s:1>
Name des Feldes mit dem Verschlüsselungsalgorithmus.

INSTRA=
bestimmt die Adresse eines Feldes, dass das zu verschlüsselnde Wort (Eingabewort) enthält. Die Länge des Feldes wird mit dem Operanden INSTRL angegeben. Das Eingabewort muss eine Zeichenkette vom Typ X-String oder C-String mit maximal 8 Zeichen sein.
Die Angabe dieses Operanden ist Pflicht bei MF=L.

<var: pointer>
Name des Feldes mit der Adresse des Eingabewortes; nur in Verbindung mit MF=M zulässig.

(<reg: pointer>)
Register mit der Adresse des Eingabewortes; nur in Verbindung mit MF=M zulässig.

INSTRL=
gibt die Länge des Feldes an, das für das zu verschlüsselnde Eingabewort reserviert werden muss. Die Länge darf den Wert 8 nicht übersteigen. Das Feld, in dem das Eingabewort spezifiziert ist, darf nur entweder 4 Byte oder 8 Byte lang sein.

<u>4</u>
Es werden 4 Byte für die Feldlänge reserviert.

8
Es werden 8 Byte für das Feld reserviert.

<var: int:1>
Name des Feldes mit der Angabe der zu reservierenden Feldlänge.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Bei der C-Form oder D-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

OUSTRA=
bestimmt die Adresse eines Feldes, dass das verschlüsselte Wort (Ausgabewort) enthalten soll. Zurückgeliefert wird ein 4 Byte langes Ausgabewort, wenn das Eingabewort ≤ 4 Byte ist. Beträgt die Länge des Eingabewortes zwischen 5 Byte und 8 Byte, wird ein 8 Byte langes Ausgabewort zurückgeliefert.
Die Angabe dieses Operanden ist Pflicht bei MF=L.

<var: pointer>
Name des Feldes mit der Adresse des Ausgabewortes; nur in Verbindung mit MF=M zulässig.

(<reg: pointer>)
Register mit der Adresse des Ausgabewortes; nur in Verbindung mit MF=M zulässig.

VKEYA=
Adresse des verwendeten variablen Schlüssels, der bei der Verschlüsselungseinstellung CRYALG=*SCAVK verwendet werden soll.

<var: pointer>
Name des Feldes mit der Adresse des variablen Schlüssels; nur in Verbindung mit MF=M zulässig.

(<reg: pointer>)
Register mit der Adresse des variablen Schlüssels; nur in Verbindung mit MF=M zulässig.

XPAND=
steuert die Makroauflösung.

***INPAR**
Es wird die Parameterstruktur expandiert.

***KEYPAR**
Es wird nur der Datenbereich für den variablen Schlüssel expandiert.

Auswahl des variablen Schlüssels

Die Verwendung eines variablen Schlüssels ist nur bei einer Verschlüsselung mit dem SCA-Algorithmus (Operand CRYALG=*SCA) möglich. Der variable Schlüssel ist 44 Byte lang und besteht aus folgenden vier Teilen (PREFIX und MACID sind jeweils mit ihren Voreinstellungen belegt):

```
SRMEVK      DS      0F
SRMECC      DS      F       Iterationsanzahl
SRMEKEE1    DS      XL16    Schlüsselkomponente EE1
SRMEKEE2    DS      XL16    Schlüsselkomponente EE2
SRMEKEE3    DS      XL8     Schlüsselkomponente EE3
SRMEVK#     EQU     *-SRMEVK
```

Die Verschlüsselung mit dem SCA-Algorithmus beruht auf der iterativen Anwendung einer Basisverschlüsselung. Um eine genügend große Sicherheit zu bieten, muss die Anzahl der Iterationen (Feld `SRMECC`) zwischen 128 und 8192 liegen. Eine Verschlüsselung mit der Iterationsanzahl 128 benötigt ca. 15000 Operationen. Die Anzahl der Operationen steigt linear mit der Anzahl der Iterationen.

Die Schlüsselkomponenten EE1 und EE2 (Felder `SRMEKEE1` und `SRMEKEE2`) repräsentieren Permutationen der Zahlen 0 bis 15. Für einen sog. „sicheren“ Schlüssel muss jedes Byte von EE1 und EE2 eine Zahl zwischen 0 und 15 enthalten, und zwar so, dass jede dieser Zahlen jeweils einmal in EE1 bzw. EE2 vorkommt.
Die Schlüsselkomponente EE3 (Feld `SRMEKEE3`) enthält beliebige Zeichen in der Länge 8 Byte. Es dürfen jedoch keine 2 gleichen Byte enthalten sein.

Die Verschlüsselungsroutine prüft die o.g. Bedingungen für einen „sicheren“ variablen Schlüssel nicht ab. Bei Einweg-Verschlüsselungsverfahren ist die Verwendung von „schwachen“ Schlüsseln oft nicht zu vermeiden. Der SCA-Verschlüsselungsalgorithmus ist daher auch mit „schwachen“ Schlüsseln ablauffähig. Aus Sicherheitsgründen sollte die Verwendung von „schwachen“ Schlüsseln vermieden werden.

Verschlüsselung von Eingabeworten einer Länge > 8 Byte

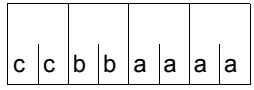
Ist eine Verschlüsselung von Worten mit einer Länge > 8 Byte nötig, kann das Eingabewort in 8 Byte-Worte zerlegt und die Wortsegmente separat verschlüsselt werden. Die Verschlüsselung von Wörtern mit einer Länge > 8 Byte ist mit den vom Makro **CRYPT** angebotenen Algorithmen nicht sicherer, als die Verschlüsselung eines Wortes mit einer Länge ≤ 8 Byte.

Layout der DSECT

Das Layout der DSECT finden Sie auf Seite 353.

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros CRYPT wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Funktion erfolgreich ausgeführt.
X'01'	X'01'	X'0001'	Funktion wegen Operandenfehler nicht ausgeführt: Keine Zuweisung des Eingabewortes.
X'02'	X'01'	X'0001'	Funktion wegen Operandenfehler nicht ausgeführt: Keine Zuweisung des Ausgabewortes.
X'03'	X'01'	X'0001'	Funktion wegen Operandenfehler nicht ausgeführt: Keine Zuweisung für den variablen Schlüssel.
X'04'	X'01'	X'0001'	Funktion wegen Operandenfehler nicht ausgeführt: Unzulässige Angabe für den Verschlüsselungsalgorithmus.
X'05'	X'01'	X'0001'	Funktion wegen Operandenfehler nicht ausgeführt: Unzulässige Angabe zur Abhängigkeit vom Systemparameter.
X'06'	X'01'	X'0001'	Funktion wegen Operandenfehler nicht ausgeführt: Unzulässige Längenangabe für das Eingabewort.
X'08'	X'01'	X'0001'	Funktion wegen Operandenfehler nicht ausgeführt: Unzulässiger variabler Schlüssel.
X'09'	X'01'	X'0002'	Funktion nicht ausgeführt wegen Fehler bei der Speicherbereichs-anforderung: Kein Zugriff auf das Feld mit dem Eingabewort möglich.
X'0A'	X'01'	X'0002'	Funktion nicht ausgeführt wegen Fehler bei der Speicherbereichs-anforderung: Kein Zugriff auf das Feld mit dem Ausgabewort möglich.
X'0B'	X'01'	X'0002'	Funktion nicht ausgeführt wegen Fehler bei der Speicherbereichsanfor-derung: Kein Zugriff auf das Feld mit dem variablen Schlüssel möglich.
X'0C'	X'01'	X'0002'	Funktion nicht ausgeführt wegen Fehler bei der Speicherbereichs-anforderung: Kein Zugriff auf den Datenbereich möglich.
X'0D'	X'01'	X'0002'	Funktion nicht ausgeführt wegen Fehler bei der Speicherbereichs-anforderung: Kein Zugriff auf das Feld SRMEPOE möglich.
	X'20'	X'0003'	Funktion nicht ausgeführt: Interner Fehler.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Beispiel

```
PRINT NOGEN
   CRYPT    START
            BALR  10,0
            USING *,10
            CRYPT MF=E,PARAM=PARLIST ------------------------------------  (1)
 1          MFCHK MF=E,PREFIX=S,MACID=RME,PARAM=PARLIST,
 1                SVC=16,
 1                DMACID=RME,SUPPORT=(D,L,C,M,E)
 2          LA    1,PARLIST
 2          SVC   16
            CLI   SRMEMR1,SRMEOK      * Error query
            BNE   ERREXIT
            UNPK  OUTPUTX(9),OUTPUT(5)
            UNPK  OUTPUTX+8(9),OUTPUT+4(5)
            TR    OUTPUTX,CODETAB-C'0'
            WROUT CODE,0              * Output
            TERM
   *
   ERREXIT  WROUT TEXT,0
            TERM
   ****
   CODE     DC    Y(CODEEND-CODE)
            DS    CL3
            DC    C'OUTPUT OF THE ENCRYPTED WORD '
   INPUT    DC    C'SUPERMAN'         * Input word *
            DC    C': '
   OUTPUT   DS    CL8                 * Output word
            DC    C'  '
   OUTPUTX  DS    CL16                * Output word hex
   CODEEND  EQU   *
            DS    C
   TEXT     DC    Y(TEXTEND-TEXT)
            DS    CL3
            DC    C'ERROR !!'
   TEXTEND  EQU   *
   KEY      DS    0F
            DC    F'250'              * Number of iterations
            DC    X'0203040506070809' * EE1
            DC    X'0A0B0C0D0E0F0001'   EE1 *
            DC    X'0100030205040706' * EE2
            DC    X'09080B0A0D0C0F0E'   EE2 *
            DC    X'A1A2A3A4A5A6A7A8' * EE3 *
   PARLIST  CRYPT MF=L,INSTRL=8,CRYALG=*SCAVK,CRCL2OP=*NO, -
                  VKEYA=KEY,INSTRA=INPUT,OUSTRA=OUTPUT ------------------  (1)
            ORG   PARLIST
```

```
           CRYPT MF=C ------------------------------------------------  (2)
1 *
1 SRMEPA     DS    OF          BEGIN of PARAMETERAREA
1         FHDR  MF=(C,SRME),EQUATES=NO                    STANDARD HEADER
2          DS    0A
2 SRMEFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
2 SRMEIFID DS    0A              0   INTERFACE IDENTIFIER
2 SRMEFCTU DS    AL2             0   FUNCTION UNIT NUMBER
2 *                                  BIT 15    HEADER FLAG BIT,
2 *                                  MUST BE RESET UNTIL FURTHER NOTICE
2 *                                  BIT 14-12 UNUSED, MUST BE RESET
2 *                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
2 SRMEFCT  DS    AL1             2   FUNCTION NUMBER
2 SRMEFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 SRMERET  DS    0A              4   GENERAL RETURN CODE
2 SRMESRET DS    0AL2            4   SUB RETURN CODE
2 SRMESR2  DS    AL1             4   SUB RETURN CODE 2
2 SRMESR1  DS    AL1             5   SUB RETURN CODE 1
2 SRMEMRET DS    0AL2            6   MAIN RETURN CODE
2 SRMEMR2  DS    AL1             6   MAIN RETURN CODE 2
2 SRMEMR1  DS    AL1             7   MAIN RETURN CODE 1
2 SRMEFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 *       RETURN CODE EQUATES FOR MAIN-CODE 1
1 SRMEOK   EQU   X'00'               NOERROR
1 SRMEIOP  EQU   X'01'               INVALID OPERAND
1 SRMEIAR  EQU   X'02'               INVALID AREA
1 SRMEINE  EQU   X'03'               INTERNAL ERROR
1 *
1 SRMEIN   DS    F                   INPUT STRING ADDRESS
1 SRMEOUT  DS    F                   OUTPUT STRING ADDRESS
1 SRMELEN  DS    X                   INPUT STRING LENGTH
1 SRMEECR  DS    X                   SELECT ENCRYPTION ROUTINE
1 *
1 *       EQUATES FOR ENCRYPTION ROUTINE SELECT
1 SRMEECRS EQU   X'01'                SCA ENCRYPTION ROUTINE
1 SRMEECRO EQU   X'02'                OLD ENCRYPTION ROUTINE
1 SRMEECRV EQU   X'03'                SCA ENCR. ROUT. (V. KEY)
1 *
1 SRMEC2O  DS    X                  SEL. CLASS 2 OPTION YES/NO
1 *       EQUATES FOR CLASS 2 OPTION
1 SRMEC2OY EQU   X'01'                CLASS 2 OPTION YES
1 SRMEC2ON EQU   X'02'                CLASS 2 OPTION NO
1 *
1 SRMEPOE  DS    X                   PROGRESS OF EXECUTION
1 *       RETURN FOR PROGRESS OF EXECUTION
```

```
 1 SRMEUV  EQU   X'01'               UNCRYPTED
 1 SRMESCA EQU   X'02'               SCA ENCRYPTED
 1 SRMEOLD EQU   X'03'               OLD ENCRYPTED
 1 SRMESVK EQU   X'04'               SCA ENCRYPTED (V. KEY)
 1 *
 1 SRMESVK@ DS    F                   ADR. VARIABLE KEY
 1 SRME#   EQU   *-SRMEPA    LENGTH OF PARAMETERAREA
   *
   CODETAB  DC    C'0123456789ABCDEF'
            END   CRYPT
```

Ablaufprotokoll:

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
%//compile source=*library-element(lib.srpmencp,crypt), -
%//        compiler-action=module-generation(module-format=llm), -
%//        module-library=lib.srpmencp, -
%//        listing=parameters(output=*library-element(lib.srpmencp,crypt))
%  ASS6011 ASSEMBLY TIME: 360 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 65 MSEC
%//end
%  ASS6012 END OF ASSEMBH
/start-executable-program library=lib.srpmencp,element-or-symbol=crypt
%  BLS0523 ELEMENT 'CRYPT', VERSION '@', TYPE 'L' FROM LIBRARY
   ':2OSC:$EVA.LIB .SRPMENCP' IN PROCESS
%  BLS0524 LLM 'CRYPT', VERSION ' ' OF '<date> <time>' LOADED
OUTPUT OF THE ENCRYPTED WORD SUPERMAN: |ÑzÄkè~]  4F69A9639254FFBD ------  (3)
```

(1) Ein Wort der Länge 8 Bytes soll mit dem SCA-Algorithmus unabhängig von der CLASS-2-OPTION mit dem Makro **CRYPT** verschlüsselt werden. Es soll ein variabler Schlüssel verwendet werden. Das Eingabewort wird aus dem Feld `INPUT` eingelesen, das Ausgabewort im Feld `OUTPUT` abgelegt.

(2) Layout der Parameterliste.

(3) Ausgabe des Feldes `CODE` durch den Makro **WROUT**. Das Eingabewort `SUPERMAN` wird verschlüsselt zu |ÑzÄkè~]. Hexadezimal ausgedrückt lautet das Verschlüsselungsergebnis `4F69A9639254FFBD`.

CSTAT – Seitenstatus ändern

Allgemeines

Anwendungsgebiet: Arbeiten mit virtuellem Speicher; siehe Seite 55
Memory Pool Technik; siehe Seite 55
Makrotyp: S-Typ, MF-Format **1**: Standardform/E-Form/L-Form; siehe Seite 29

Das Betriebssystem verwaltet den virtuellen Arbeitsspeicher seitenweise (das betrifft u.a. Seitenwechsel (Paging), Speicherschutz, Anfordern und Freigabe von Speicherseiten). Eine Speicherseite umfasst 4 KB (=4096 Byte). Das Programm des Anwenders belegt Speicherplatz in Einheiten von je einer Speicherseite im Klasse-6-Speicher.

Makrobeschreibung

Mit dem Makro **CSTAT** verändert der Anwender die Attribute seiner Klasse-6-Speicherseiten in Bezug auf

– Seitenwechsel (Speicherseiten resident oder seitenwechselbar)
– Zugriffsart (Lese-/Schreibzugriff)
– spezieller Zugriffsschutz (Zugriffe über AID, DUMP nur mit besonderer Privilegierung)

Für Speicherseiten in einem Memory Pool ist zu beachten:

– Speicherseiten eines residenten Memory Pools können mit einem **CSTAT**-Aufruf nicht seitenwechselbar gemacht werden (s. auch Makro **ENAMP**). Es werden bei Makroausführung nur die Speicherseiten berücksichtigt, die außerhalb des Memory Pools liegen.
– Speicherseiten eines nichtresidenten Memory Pools können von jedem Memory-Pool-Teilnehmer mit **CSTAT** resident und im weiteren Ablauf auch wieder pageable gemacht werden.
– Bezüglich der Änderungen der Zugriffsrechte hat der Makro **CSTAT** eine niedrigere Priorität als der Makro **CSTMP**:
Mit **CSTMP** eingerichteter Schreibschutz für einen Memory Pool kann mit **CSTAT** nicht aufgehoben werden; auch nicht für die Seiten, deren Schreibschutz mit **CSTAT** vor Aufruf von **CSTMP** eingerichtet wurde.

Makroaufrufformat und Operandenbeschreibung

```
CSTAT

PGNUM=wert / ALL / (r)
[,ACCESS=READ / WRITE / (r)]
[,PAGE=YES / NO / (r)]
[,PROTECT=YES]
[,MF=(E,..) / L]
```

PGNUM=
bezeichnet die Klasse-6-Speicherseite(n), deren Zustand in Bezug auf Seitenwechsel, Zugiffsart oder speziellen Zugriffsschutz verändert werden soll. Zusätzlich zu PGNUM muss mindestens ein weiterer Operand angegeben werden. **CSTAT**-Aufruf nur mit PGNUM=.... ist sinnlos und führt zu der Anzeige (Returncode) X'0C'.

wert
(virtuelle) Seitennummer der Speicherseite, deren Zustand verändert werden soll

ALL
Der Zustand aller Speicherseiten des Klasse-6-Speichers soll geändert werden. Die Angabe PGNUM=ALL darf nur mit der Angabe PAGE=... kombiniert werden.

(r)
r = Register, das die Seitennummer (wert) oder die Angabe ALL (X'40C1D3D3') enthält; ($2 \leq r \leq 12$).

ACCESS=
beschreibt, ob auf die angegebene Speicherseite nur Lesezugriffe oder auch Schreibzugriffe erlaubt sind (Erlaubnis für Schreibzugriff impliziert auch Lesezugriff).

Hinweise
- Die Angabe ACCESS=... darf nicht mit der Angabe PGNUM=ALL kombiniert werden.
- Memory Pool: Mit **CSTMP** eingerichteter Schreibschutz kann nicht mit **CSTAT** aufgehoben werden.

READ
Es ist nur Lesezugriff erlaubt.

WRITE
Schreibzugriff erlaubt.

(r)
r = Register mit der Angabe, ob Schreibzugriff erlaubt ist oder nicht.
YES: Schreibzugriff erlaubt (X'40E8C5E2).
NO: nur Lesezugriff erlaubt (X'4040D5D6').

PAGE=
gibt an, ob die Speicherseite(n) seitenwechselbar (pageable) oder resident sein soll(en).

YES
Die Speicherseite(n) soll(en) seitenwechselbar sein.
Speicherseiten eines residenten Memory Pools bleiben unberücksichtigt.

NO
Die Speicherseite(n) soll(en) resident sein.
Es werden maximal nur so viele Seiten resident gemacht, wie mit dem Operanden RESIDENT-PAGES=PARAMETERS(MINIMUM=...) im Kommando START-PROGRAM oder LOAD-PROGRAM vereinbart wurden.

Hinweis
Bei Speichersättigung können folgende Maßnahmen wirksam werden:
– Noch anstehende **CSTAT**-Anforderungen nach residenten Speicherseiten werden abgewiesen (es erfolgt trotzdem Returncode X'00').
– Seiten, die mit **CSTAT** resident gemacht wurden, können durch das Betriebssystem seitenwechselbar gemacht werden (sie werden später nicht automatisch wieder resident gemacht).

(r)
r = Register mit der Angabe YES (X'40E8C5E2') oder NO (X'4040D5D6'); ($2 \leq r \leq 12$).

PROTECT=
vereinbart einen besonderen Zugriffsschutz für die Speicherseite.

Hinweis
– Die Angabe PROTECT=YES kann nicht mit der Angabe PGNUM=ALL oder ACCESS=.... kombiniert werden.
– Der mit PROTECT vereinbarte Zugriffsschutz kann nur durch Freigabe der Speicherseite (**RELM**) rückgängig gemacht werden. Der Inhalt der Seite wird dabei gelöscht!

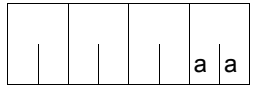
YES
Zugriffe auf die Speicherseite über die Testhilfe AID sind nur mit besonderer Privilegierung möglich. Dasselbe gilt für einen Speicherabzug (Dump).

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Rückinformation und Fehleranzeigen

Nach der Makroausführung enthält Register R1 die Adresse der Operandenliste.

R15:



Über die Ausführung des Makros CSTAT wird im rechtsbündigen Byte des Registers R15 ein Returncode übergeben.

X'aa'	**Erläuterung**
X'00'	Funktion ausgeführt.
X'04'	Die angegebene Seite ist nicht allokiert.
X'0C'	Operandenfehler
X'10'	Das Programm hat versucht, insgesamt mehr Seiten resident zu machen, als im Kommando START- oder LOAD-EXECUTABLE-PROGRAM (Operand RESIDENT-PAGES) vereinbart war.

CSTMP – Schreib- bzw. Lesezugriff für Memory Pool festlegen

Allgemeines

Anwendungsgebiet: Memory Pool Technik; siehe Seite 55
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/D-/E-Form; siehe Seite 29

Ein Memory Pool (MP) ist ein Speicherbereich im Klasse-6-Speicher, der von mehreren Anwendern gemeinsam benutzt werden kann. Seine Größe (und Lage) wird von dem ersten Anwender festgelegt. Ein Memory Pool kann mit Schreibschutz ausgestattet werden (**CSTMP** besitzt Vorrang vor **CSTAT**).

Beispiel für das Anlegen eines „Read Only Memory Pool“:

1. MP anlegen und Speicherseiten anfordern (**ENAMP**, **REQMP**),
2. In den MP schreiben (z.B. Laden „Shared Code“),
3. Schreibschutz einrichten: MP auf „Read Only“ setzen (**CSTMP**)

Makrobeschreibung

Mit dem Makro **CSTMP** kann der (dazu berechtigte) Anwender einen Memory Pool mit Schreibschutz (nur Lesezugriff erlaubt) versehen oder denselben aufheben. Der geforderte Zugriffsschutz gilt für alle Seiten und alle Teilnehmer des Memory Pools. Die Funktion wird nur ausgeführt, wenn der Anwender die geforderte Berechtigung (CSTMP-MACRO-ALLOWED=*YES) im Benutzerkatalog besitzt.

Hinweise

- Ein Memory Pool wird über den Pool-Namen oder über seine Kurzkennung angesprochen (siehe **ENAMP**).
- **CSTMP** überlagert die Wirkung des Makros **CSTAT**:
  - **CSTAT** wird abgewiesen, wenn der Schreibschutz schon mit **CSTMP** eingerichtet wurde.
  - mit **CSTAT** eingerichteter Schreibschutz kann mit **CSTMP** aufgehoben oder auf alle Poolseiten erweitert werden. In letzterem Fall kann der Schreibschutz nur mit **CSTMP** wieder aufgehoben werden.
- Hierarchisch abgestufte Schutzmöglichkeiten können mit **CSTMP** nicht eingerichtet werden.
- Für einen MP mit Schreibschutz können weder Speicherseiten angefordert (**REQMP**) noch freigegeben werden (**RELMP**).

Makroaufrufformat und Operandenbeschreibung

```
CSTMP

{ { MPNAME=name                          }
{ { MPNAMAD=adr [,MPNAMLN=länge]         } ,SCOPE=LOCAL / GROUP / USER_GROUP / GLOBAL }
{ MPID=adr                               }

,ACCESS=WRITE / READ
[,PARMOD=24 / 31]
[,MF=L / (E,..) / (D,pre) / D]
```

MPNAME=
beschreibt den Namen des Memory Pools.

name
Name des Memory Pools (Verbindung mit Operand SCOPE beachten).

MPNAMAD=
gibt die Adresse des Feldes an, in dem der Name der Memory Pools steht.

adr
symbolische Adresse (Name) des Feldes (Verbindung mit Operand SCOPE beachten).

MPNAMLN=
bezeichnet die Länge des unter MPNAMAD angegebenen Namens.
Wenn nicht spezifiziert: Längenattribut des Feldes adr.

länge
Länge in Byte.

MPID=
beschreibt die Adresse eines Feldes (Länge = 4 Byte) mit der Kurzkennung für den Memory Pool (siehe auch **ENAMP**). Die Kurzkennung identifiziert den Memory Pool eindeutig; die Angabe beschleunigt die Verarbeitung.

adr
symbolische Adresse (Name) des Feldes mit der Kurzkennung

SCOPE=
beschreibt den Geltungsbereich (Teilnehmerkreis) des Memory Pools. Die Angabe dient der eindeutigen Identifizierung des Memory Pools und muss immer in Verbindung mit den Operanden MPNAME bzw. MPNAMAD spezifiziert werden.

LOCAL
Der Memory Pool wird nur von dem einrichtenden Teilnehmer benutzt.

GROUP
Teilnehmer können alle Tasks mit der Benutzerkennung des einrichtenden Teilnehmers sein.

USER_GROUP
Teilnehmer können alle Tasks sein, deren Benutzerkennungen der gleichen Benutzergruppe angehören wie die Benutzerkennung des einrichtenden Teilnehmers.
Der Operandenwert setzt die Existenz von Benutzergruppen voraus und kann daher nur angegeben werden, wenn die Funktionseinheit SRPM des Software-Produkts SECOS im System vorhanden ist. Vor einem Makroaufruf mit SCOPE=USER_GROUP muss deshalb mit dem Makro GETUGR (siehe Handbuch „SECOS“ [14]) geprüft werden, ob SRPM zur Verfügung steht; abhängig vom Ergebnis (Returncode) ist im Programm zu reagieren.

GLOBAL
Teilnehmer können alle im System laufenden Tasks sein.

ACCESS=
beschreibt, ob auf den Memory Pool nur Lesezugriffe oder auch Schreibzugriffe zulässig sind (Berechtigung zum Schreibzugriff impliziert auch Lesezugriff). Der Zugriffsschutz gilt für alle Teilnehmer.

WRITE
Schreibzugriff erlaubt.

READ
es ist nur Lesezugriff erlaubt.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Bei der D-Form des Makroaufrufs kann ein Präfix (pre = 1..3 Buchstaben), wie im Aufrufformat dargestellt, angegeben werden.
Voreinstellung: pre = CST

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

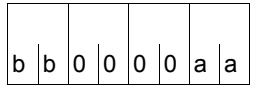
24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

Rückinformation und Fehleranzeigen

Nach der Makrobearbeitung enthält Register R1 die Adresse des Datenbereichs.

R15:



Über die Ausführung des Makros CSTMP wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'bb'	X'aa'	Erläuterung
X'00'	X'00'	normale Ausführung
X'04'	X'04'	Funktion nicht ausgeführt. Aufrufer ist kein Memory Pool-Teilnehmer (kein ENAMP-Aufruf).

X'bb'	X'aa'	Erläuterung
X'1C'	X'04'	Funktion nicht ausgeführt. Operandenfehler: – unzulässige Adresse des Datenbereichs – fehlerhafter Aufbau des Datenbereichs – unzulässige Adresse für MPNAMAD oder MPID im Datenbereich – Benennung des Memory Pools: – Name enthält unzulässige Zeichen – ungültige Längenangabe (MPNAMLN) – Memory Pool nicht benannt (MPNAME,MPNAMAD,MPID nicht spezifiziert). – MPNAMLN angegeben, aber MPNAMAD nicht spezifiziert – SCOPE angegeben, aber MPNAME/MPNAMAD nicht spezifiziert. – Benennung nicht eindeutig. Es wurde mehr als nur ein Operand zur Benennung spezifiziert (MPNAME/MPNAMAD/MPID). – ungültige SCOPE-Angabe – SCOPE=USER_GROUP wurde angegeben, obwohl SRPM nicht im System vorhanden ist – ungültige ACCESS-Angabe – PARMOD=24 in Verbindung mit 31-Bit-Adressierungsmodus (AMODE 31) angegeben
X'24'	X'04'	Funktion nicht ausgeführt. Berechtigungsfehler: – Aufrufer ist nicht berechtigt, den CSTMP-Makro aufzurufen (fehlender Eintrag in der Benutzerkennung) – Aufrufer ist nicht berechtigt, den Zugriffsschutz eines privilegierten oder Klasse-5-MPs zu ändern. – Der Memory Pool enthält DIV- oder FASTPAM-Fenster.

31-Bit-Schnittstelle:

– Bei fehlerhafter Ausrichtung oder Initialisierung des Standardheaders werden im Register R15 zusätzlich die Returncodes X'0001FFFF' / X'0003FFFF'/ X'0004FFFF' übergeben; siehe Tabelle „Standard-Returncodes“ auf Seite 43.

– Im Standardheader werden keine Returncodes übergeben.

CTIME – Mit Zeitstempeln rechnen

Allgemeines

Anwendungsgebiet: Abfragen und Zugriff zu Listen und Tabellen; siehe Seite 158
Makrotyp: S-Typ, MF-Format **3**: C-/D-/E-/L-/M-Form; siehe Seite 29

Makrobeschreibung

Der Makro **CTIME** bietet folgende Funktionen:

– die Änderung der Darstellung von Zeitstempeln (FUNCT=*CONV)
– die Addition und Subtraktion einer Zeitspanne von einem Zeitstempel (FUNCT=*ADD)
– die Ermittlung der Zeitspanne zwischen zwei Zeitstempeln (FUNCT=*DIFF)

CTIME stellt die Ergebnisse im Datenbereich wahlweise in abdruckbarer, binärer oder TODR/TODX-Form zur Verfügung.

Hinweise

– Der Makro **CTIME** benutzt keinen SVC.
  Bei MF=D (Voreinstellung) wird vom System kein Standardheader initialisiert, d.h. der Anwender muss vor Aufruf des Makros den Standardheader selbst definieren (zum Aufbau des Standardheaders siehe Seite 43).
– Da die Funktionen des Makros **CTIME** nicht über einen SVC sondern über eine Unterprogrammschnittstelle ausgelöst werden, muss ein Programm, das **CTIME** aufruft, einen 18 Worte langen Sicherstellungsbereich zur Verfügung stellen. Vor dem Makroaufruf ist die Adresse dieses Sicherstellungsbereiches in Register R13 zu laden.

Der Datenbereich enthält Speicherbereiche für:

1. Eingabezeitstempel	NTICS1I	Länge:	48 Byte
2. Eingabezeitstempel	NTICS2I		48 Byte
Eingabezeitspanne	NTICD1I		32 Byte
Ausgabezeitstempel	NTICS1O		48 Byte
Ausgabezeitspanne	NTICD1O		32 Byte

wobei die Feldnamen für PREFIX=N und MACID=TIC gelten.

Über Zeitstempel und Zeitspannen lassen sich die DSECTs NTICS (für Zeitstempel) und NTICD (für Zeitspannen) legen, um die Datenfelder mit den Zeitangaben anzusprechen.

Ein Zeitstempel besteht aus Datum (Jahr, Monat, Tag, Wochentag) und Uhrzeit (Stunde, Minute, Sekunde, Millisekunde, Mikrosekunde). Er bezieht sich immer auf eine bestimmte Zeitbasis und kann in verschiedenen Formaten angegeben werden. Zeitspannen bestehen aus Angaben von Tagen und Uhrzeit und können ebenfalls in verschiedenen Formaten angegeben werden.

Mögliche **Zeitbasen** sind:

UTC: Universal Time Coordinate ≙ Greenwich Time (Weltzeit)
LTI: Local Time (lokale Zeit im aufrufenden System)
FZ: Foreign Zone („fremde“, beliebige Zeitbasis)
Bei Angabe der beliebigen Zeitbasis ist der Zeitstempel nur dann eindeutig, wenn Zonen-Informationen mitgegeben werden. Diese Informationen geben Auskunft darüber, welche Weltzeit-Zonen-Differenz und/oder welche Sommer-Winterzeit-Umstellungsdifferenz beachtet werden muss und ob der Zeitstempel in Sommer- oder Winterzeit angegeben wurde.

Die Darstellung eines **Zeitstempels** ist in folgenden Formaten möglich:

ISO4 abdruckbares Format in Dezimalzahlen mit dem Aufbau

```
yyyy-mm-ddjjj wwhh:mm:ssvhz:mz-hs:ms-smlsmcs
```

Es bedeutet in folgender Reihenfolge:

`yyyy`	Jahr	(4 Byte)
`mm`	Monat des Jahres	(2 Byte)
`dd`	Tag des Monats	(2 Byte)
`jjj`	julianischer Tag des Jahres	(3 Byte)
`ww`	Wochentag	(2 Byte)
`hh`	Stunde	(2 Byte)
`mm`	Minute	(2 Byte)
`ss`	Sekunde	(2 Byte)
`v`	Vorzeichen zur Zonen-Differenz	(1 Byte)
`hz`	Stunde der Zonen-Differenz	(2 Byte)
`mz`	Minute der Zonen-Differenz	(2 Byte)
`hs`	Stunde der Sommer-Winterzeit-Differenz	(2 Byte)
`ms`	Minute der Sommer-Winterzeit-Differenz	(2 Byte)
`s`	Sommer-Winterzeit-Angabe	(1 Byte)
`mls`	Millisekunde	(3 Byte)
`mcs`	Mikrosekunde	(3 Byte)

Der Datenbereich zur Aufnahme des Zeitstempels hat folgenden Aufbau (Makroauflösung mit MF=D und PREFIX=N):

```
NTICSISO4       DS    OXL48        timestamp in iso4
NTICSDATE_U     DS    OXL10        date union
*
NTICSMD         DS    OXL10        date
NTICSIDY        DS    CL4          year
NTICSID1        DS    CL1          hyphen1
NTICSIDM        DS    CL2          month
NTICSID2        DS    CL1          hyphen2
NTICSIDD        DS    CL2          day
*
                ORG   NTICSDATE_U
NTICSDATE_CHAR  DS    CL10         date_char
                ORG   NTICSDATE_U+10
NTICSIDJ        DS    CL3          julian
NTICSIDB        DS    CL1          blank
*
NTICSMW         DS    OCL2         begin of weekday
NTICSIWD        DS    CL2          weekday
*
NTICSMT         DS    OCL8         time: "hh:mm:ss"
NTICSITH        DS    CL2          hour
NTICSIT1        DS    CL1          colon1
NTICSITM        DS    CL2          minute
NTICSIT2        DS    CL1          colon2
NTICSITS        DS    CL2          second
*
NTICSMZ         DS    OCL14        zone: "shh:mm-hh:mm-a"
NTICSIZS        DS    CL1          zonesign
NTICSIZH        DS    CL2          zonehour
NTICSIZ1        DS    CL1          colon3
NTICSIZM        DS    CL2          zoneminute
NTICSIZ2        DS    CL1          hyphen3
NTICSISH        DS    CL2          seasonhour
NTICSIS1        DS    CL1          colon4
NTICSISM        DS    CL2          seasonminute
NTICSIS2        DS    CL1          hyphen4
NTICSISA        DS    FL1          actualseason
*
NTICSMF         DS    OXL6         begin of fraction of second
NTICSIF         DS    OCL6         fraction of second : "mmmuuu"
NTICSIFM        DS    CL3          millisecond
NTICSIFN        DS    CL3          microsecond
NTICSILE        EQU   *-NTICSISO4  LENGTH OF ISO4 TIMESTAMP
*                                  (without address of CHDATE List)
NTICSCDL        DS    A            chdates_addr
```

BINAR Die numerischen Werte werden binär in Halbworten angegeben.

Der Datenbereich zur Aufnahme des Zeitstempels hat folgenden Aufbau (Makroauflösung mit MF=D und PREFIX=N):

```
NTICSBINAR      DS    0XL42        timestamp in binar
NTICSBDY        DS    H            year
NTICSBDM        DS    H            month
NTICSBDD        DS    H            day
NTICSBDJ        DS    H            julian
NTICSFILL1      DS    CL6          fill1
NTICSBWD        DS    H            weekday
*
NTICSBTH        DS    H            hour
NTICSBTM        DS    H            minute
NTICSBTS        DS    H            second
NTICSBL1        EQU   *-NTICSBINAR LENGTH OF BINARY TIMESTAMP
*                                     PART1 (date & time)
NTICSFILL2      DS    CL2          fill2
NTICSBZH        DS    H            zonehour
NTICSBZM        DS    H            zoneminute
NTICSBSH        DS    H            seasonhour
NTICSBSM        DS    H            seasonminute
NTICSBSA        DS    FL1          actualseason
NTICSFILL3      DS    CL5          fill3
*                                  fraction of second
NTICSBFM        DS    H            millisecond
NTICSBFN        DS    H            microsecond
NTICSBL2        EQU   *-NTICSBFM   LENGTH OF BINARY TIMESTAMP
*                                     PART2 (fraction of second)
NTICSBLE        EQU   *-NTICSBINAR LENGTH OF BINARY TIMESTAMP
```

TODR Time-Of-Day-Register-Format: Das TOD-Register hat die Länge eines Doppelwortes und enthält:
((Anzahl der Mikrosekunden seit 1.1.1900) * 4096) modulo 2^{64}

Es wird durch die Hardware laufend hochgezählt. Mit dem STCK-Befehl kann das TOD-Register abgefragt werden. Die Interpretation des TODR-Inhaltes hängt von der für den Systemlauf eingestellten Epoche ab, siehe Handbuch „Systembetreuung“ [10]).

Der Datenbereich zur Aufnahme des Zeitstempels hat folgenden Aufbau (Makroauflösung mit MF=D und PREFIX=N):

```
NTICST          DS    0XL8         timestamp in TODR
NTICSTMS        DS    F            most significant word:
*                                     approx sec
NTICSTLS        DS    F            least significant word:
*                                     micro_sec * 2**12
```

TODX Erweitertes Time-Of-Day-Register-Format: es hat die Länge eines Doppelwortes und enthält die Anzahl der Mikrosekunden seit dem 1. Januar 1900 0 Uhr.

Der Datenbereich zur Aufnahme des Zeitstempels hat folgenden Aufbau (Makroauflösung mit MF=D und PREFIX=N):

```
NTICSX          DS    OXL8         timestamp in TODX
NTICSTHW        DS    F            high word
NTICSTLW        DS    F            low word
```

Die Darstellung von **Zeitspannen** ist in folgenden Formaten möglich:

ISO4 / ISO4MIC abdruckbares Format in Dezimalzahlen
Der Datenbereich zur Aufnahme der Zeitspanne hat folgenden Aufbau (Makroauflösung mit MF=D und PREFIX=N):

```
NTICDI          DS    OXL32        tdiff in iso4
NTICDIS         DS    CL1          sign
NTICDIDD        DS    CL10         day
NTICDID1        DS    CL1          hyphen
NTICDITH        DS    CL2          hour
NTICDIT1        DS    CL1          colon1
NTICDITM        DS    CL2          minute
NTICDIT2        DS    CL1          colon2
NTICDITS        DS    CL2          second
NTICDIFP        DS    CL1          point
NTICDIFM        DS    CL3          millisecond
NTICDIFN        DS    CL3          microsecond
NTICDILE        EQU   *-NTICDI     LENGTH OF ISO4 TIMEDIFFERENCE
*                                  (without unused field)
                DS    CL5          unused
```

BINAR / BINARMIC binäre Angabe.
Der Datenbereich zur Aufnahme der Zeitspanne hat folgenden Aufbau (Makroauflösung mit MF=D und PREFIX=N):

```
NTICDB          DS    OXL16        tdiff in binar
NTICDBDD        DS    F            day
NTICDBTH        DS    H            hour
NTICDBTM        DS    H            minute
NTICDBTS        DS    H            second
NTICDBFM        DS    H            millisecond
NTICDBFN        DS    H            microsecond
NTICDBLE        EQU   *-NTICDB     LENGTH OF BINARY TIMESTAMP
*                                  (without the fill field)
                DS    H            to fill the gap
```

TODR Time-Of-Day-Register-Format.
Der Datenbereich zur Aufnahme der Zeitspanne hat folgenden Aufbau (Makroauflösung mit MF=D und PREFIX=N):

```
NTICDT      DS    0XL8        tdiff in TODR (for C)
NTICDTMS    DS    F           most significant word:
*                                 approx. sec
NTICDTLS    DS    F           least significant word:
*                                 micro_sec * 2**12
```

TODX Erweitertes Time-Of-Day-Register-Format.
Der Datenbereich zur Aufnahme der Zeitspanne hat folgenden Aufbau (Makroauflösung mit MF=D und PREFIX=N):

```
NTICDX      DS    0XL8        tdiff in TODX (for C)
NTICDTHW    DS    F           high word
NTICDTLW    DS    F           low word
```

Wertebereiche

Es gelten die folgenden Wertebereiche für die Arbeit mit dem **CTIME**-Makro. Zu beachten ist die Unterscheidung in der Format-Darstellung bei den Ein- und Ausgabe-Zeitstempeln und Zeitspannen.

BINAR- oder ISO4-Format:

```
01.01.1900 00:00:00,000000  <  Zeitstempel  <  31.12.9999 23:59:59,999999
```

erlaubte Zeitdifferenz: ± 2147483647 Tage

TODR-Format:

```
t0   <  Zeitstempel  <  t1
```

erlaubte Zeitdifferenz: ± 26062 Tage

`t0` und `t1` sind dabei abhängig von der im Parameterservice (Parametersatz GTIME, siehe Handbuch „Systembetreuung“ [10]) für den Systemlauf eingestellten Epoche für das TOD-Register. Für die Standardepoche (GTIME-Parameter EPOCH=00) ist das TODR-Format:
`01.01.1900 00:00:00.000000  <  Zeitstempel  <  17.09.2042 23:53:47.370495`

TODX-Format:

```
01.01.1900 00:00:00  <  Zeitstempel  <  18.03.4317 02:44:48.587775
```

erlaubte Zeitdifferenz: ± 882867 Tage

Makroaufrufformat und Operandenbeschreibung

CTIME
FUNCT=*CONV / *ADD / *ADDLL / *DIFF / adr / (r) ,BASE1IN=*UTC / *LTI / *FZ / adr / (r) ,FRM1IN=*ISO4 / *ISO4MIC / *BINAR / *BINARMIC / *TODR / *TODX / adr / (r) ,INF1IN=*CALEND / *JULIAN / adr / (r) ,FRM1ZIN=*NONE / *ISO4 / *ISO4LST / *BINAR / *BINARLST / adr / (r) ,CHDL1IN=*NONE / adr / (r) ,BASE2IN=*UTC / *LTI / *FZ / adr / (r) ,FRM2IN=*ISO4 / *ISO4MIC / *BINAR / *BINARMIC / *TODR / *TODX / adr / (r) ,INF2IN=*CALEND / *JULIAN / adr / (r) ,FRM2ZIN=*NONE / *ISO4 / *ISO4LST / *BINAR / *BINARLST / adr / (r) ,CHDL2IN=*NONE / adr / (r) ,BASEOUT=*LTI / *UTC / *FZ / adr / (r) ,FRMOUT=*ISO4 / *ISO4MIC / *BINAR / *BINARMIC / *TODR / *TODX / adr / (r) ,FRMZOUT=*NONE / *ISO4 / *ISO4LST / *BINAR / *BINARLST / adr / (r) ,CHDLOUT=*NONE / adr / (r) ,FRMDIN=*ISO4 / *ISO4MIC / *BINAR / *BINARMIC / *TODR / *TODX / adr / (r) ,FRMDOUT=*ISO4 / *ISO4MIC / *BINAR / *BINARMIC / *TODR / *TODX / adr / (r) ,LINKADR=*NONE / linkadr ,MF=D / C / L / M / E [,PARAM=adr / (r)] ,PREFIX=N / p ,MACID=TIC / macid

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

BASE1IN=
BASE2IN=
BASEOUT=
gibt die Zeitbasis an, die für den 1. oder 2. Eingabe-Zeitstempel (bei BASE1IN bzw. BASE2IN) oder für den Ausgabe-Zeitstempel (bei BASEOUT) gültig ist.

***UTC**
Der Zeitstempel wird in der Zeitbasis der Weltzeit UTC (Universal Time Coordinate, entspricht der Greenwich-Zeit) angegeben.
Dieser Wert ist Voreinstellung bei BASE2IN.

***LTI**
Der Zeitstempel wird in der im Aufrufer-System gültigen lokalen Zeitbasis LTI (Local TIme) angegeben.
Dieser Wert ist Voreinstellung bei BASEOUT.

***FZ**
Der Zeitstempel wird in einer fremden bzw. beliebigen Zeitbasis (Foreign Zone) angegeben. Die diese Zeitbasis charakterisierenden Informationen müssen in der Zonen-Information des betreffenden Zeitstempels mitgegeben und mit den jeweiligen Operanden spezifiziert werden (Angabe des Operanden FRMxZIN bzw. FRMZOUT erforderlich).

adr
symbolische Adresse (Name) eines Feldes, das zuvor mit dem entsprechenden Equate geladen werden muss. Länge = 1 Byte.

(r)
r= Register mit dem Wert des entsprechenden Equates.

CHDL1IN=
CHDL2IN=
CHDLOUT=
gibt für FZ-Zeitstempel eine Tabelle (Liste) von Umstellungszeitpunkten mit, mit deren Hilfe die Einordnung der Zeitstempel in Sommer- oder Winterzeit erfolgt.
Eine Angabe ungleich *NONE ist nur bei MF=M und bei „FZ“-Zeitstempeln erlaubt.

***NONE**
Es wird keine Tabelle mitgegeben.

adr
symbolische Adresse (Name) eines Feldes, das die Tabelle von Umstellungszeitpunkten enthält. Dieser Operandenwert ist nur mit bei MF=M erlaubt und darf nur in Verbindung mit BASExxx=*FZ angegeben werden. Die gewünschte Verarbeitung dieser Tabelle muss mit den Werten FRMxZIN=*ISO4LST/*BINARLST bzw. FRMZOUT= *ISO4LST/*BINARLST eingestellt werden. Zur eindeutigen Darstellung eines Zeitstempels gehört auch, dass die Felder Zonendifferenz und Umstellungsdifferenz der Zoneninformation belegt sind.

Die Tabelle muss auf Wortgrenze ausgerichtet und in folgender Form aufgebaut sein:
Es werden aufeinander folgend Doppelworte erwartet, die die Informationen über die Umstellungszeitpunkte enthalten. Die CHDATE-Informationen liegen als STCK-Werte auf UTC-Basis vor, die um 8 Bits logisch nach rechts verschoben wurden (SRL-Befehl). Das niederwertigste Bit zeigt dabei die Art der Umstellung an:
Ist das Bit 0, wird von Winter- auf Sommerzeit umgestellt, bzw. umgekehrt, wenn das Bit 1 ist.
Die Erstellung der Tabelle sollte mit **GTIME** CHDATE=... erfolgen. Die Werte in diesen Doppelworten müssen aufsteigend sein.
Die Tabelle muss mit einem Doppelwort D'0' beendet werden.
Zur Arbeitsweise siehe Beispiel auf Seite 382.

(r)
r = Register mit dem Adresswert von adr.

FRMDIN=
FRMDOUT
gibt an, welches Format der Eingabe-Zeitspanne vorliegt (bei FRMDIN als Eingabe-Zeitspanne) oder gewünscht wird (bei FRMDOUT als Ausgabe-Zeitspanne).

***ISO4**
Die Zeitspanne wird im abdruckbaren Format dargestellt.
Bei FRMDOUT werden auch die Felder für Milli- und Mikrosekunden versorgt.

***ISO4MIC**
Die Zeitspanne wird im abdruckbaren Format dargestellt. Bei FRMDIN wird die erhöhte Auflösung mit Milli- und Mikrosekunden erwartet.

***BINAR**
Die Zeitspanne wird binär mit Festpunktzahlen dargestellt. Dabei werden die Tage mit 4 Byte, die Stunden, Minuten und Sekunden mit je 2 Byte dargestellt. Bei FRMDOUT werden auch die Felder für Milli- und Mikrosekunden versorgt.

***BINARMIC**
Die Zeitspanne wird binär mit Festpunktzahlen dargestellt. Bei FRMDIN wird die erhöhte Auflösung mit Milli- und Mikrosekunden erwartet.

***TODR**
Die Zeitspanne wird im Time-Of-Day-Register-Format dargestellt.

***TODX**
Die Zeitspanne wird im erweiterten Time-Of-Day-Register-Format dargestellt.

adr
symbolische Adresse (Name) eines Feldes, das zuvor mit dem entsprechenden Equate geladen werden muss. Länge = 1 Byte.

(r)
r= Register mit dem Wert des entsprechenden Equates.

FRM1IN=
FRM2IN=
FRMOUT=
gibt an, welches Zeitstempel-Format vorliegt (bei FRM1IN, FRM2IN als Eingabe-Zeitstempel) oder gewünscht wird (bei FRMOUT als Ausgabe-Zeitstempel).

***ISO4**
Der Zeitstempel wird im abdruckbaren Format dargestellt. Dieser Wert ist Voreinstellung bei FRM2IN und FRMOUT. Bei FRMOUT werden auch die Felder für Milli- und Mikrosekunden versorgt.

***ISO4MIC**
Der Zeitstempel wird im abdruckbaren Format dargestellt. Bei den Eingabe-Zeitstempeln wird die erhöhte Auflösung mit Milli- und Mikrosekunden erwartet.

***BINAR**
Der Zeitstempel wird binär mit Halbwort-Festpunktzahlen dargestellt. Bei FRMOUT werden auch die Felder für Milli- und Mikrosekunden versorgt.

***BINARMIC**
Der Zeitstempel wird binär mit Halbwort-Festpunktzahlen dargestellt. Bei den Eingabe-Zeitstempeln wird die erhöhte Auflösung mit Milli- und Mikrosekunden erwartet.

***TODR**
Der Zeitstempel wird im Time-Of-Day-Register-Format dargestellt.

***TODX**
Die Zeitspanne wird im erweiterten Time-Of-Day-Register-Format dargestellt.

adr
symbolische Adresse (Name) eines Feldes, das zuvor mit dem entsprechenden Equate geladen werden muss. Länge = 1 Byte.

(r)
r= Register mit dem Wert des entsprechenden Equates.

FRM1ZIN=
FRM2ZIN=
FRMZOUT=
gibt das Format der mitgegebenen (bei FRMxZIN) bzw. der gewünschten (bei FRMZOUT) zusätzlichen Zonen-Information an, die die Zeitbasis des spezifizierten Zeitstempels charakterisiert.
Die Angabe des Operanden ist zwingend, wenn für den betreffenden Zeitstempel die Zeitbasis „FZ“ angegeben wurde.

Für einen LTI-Eingabe-Zeitstempel kann vorgegeben werden, ob die für den Zeitstempel mitgegebene Information Sommer- oder Winterzeit ausgewertet werden soll. „*NONE“ bedeutet dabei keine Auswertung, andere Werte verlangen, dass das entsprechende Feld der Zonen-Information mit einem gültigen Wert gefüllt ist.

Im Falle des Ausgabe-Zeitstempels gibt der Operand FRMZOUT an, in welchem Format die Ausgabe der Zonen-Information erfolgt. Bei „*NONE“ wird das Format des Zeitstempels selbst übernommen (liegt das Format „*TODR“ vor, wird die Zonen-Information binär ausgegeben).

***NONE**
Es wird keine Information mitgegeben.

***ISO4**
Die Zonen-Information wird im abdruckbaren Format angegeben.

***ISO4LST**
Die Zonen-Information wird im abdruckbaren Format angegeben. Die Adresse der CHDATE-Tabelle muss durch den Operanden CHDLxIN bzw. CHDLOUT bekanntgegeben werden.
Die Angabe in der Zonen-Information, ob es sich um einen Zeitstempel für Sommer- oder Winterzeit handelt, wird dabei nicht berücksichtigt. Der FZ-Zeitstempel wird mit dieser Tabelle verglichen und daraus die Sommer-/Winterzeit-Information gewonnen (siehe auch Beispiel auf Seite 382).

***BINAR**
Die Zonen-Information wird binär als Halbwort-Festpunktzahl angegeben.

***BINARLST**
Die Zonen-Information wird binär als Halbwort-Festpunktzahl angegeben. Die Adresse der CHDATE-Tabelle muss durch den Operanden CHDLxIN bzw. CHDLOUT bekanntgegeben werden.
Die Angabe in der Zonen-Information, ob es sich um einen Zeitstempel für Sommer- oder Winterzeit handelt, wird dabei nicht berücksichtigt. Der FZ-Zeitstempel wird mit dieser Tabelle verglichen und daraus die Sommer-/Winterzeit-Information gewonnen (siehe auch Beispiel auf Seite 382).

adr
symbolische Adresse (Name) eines Feldes, das zuvor mit dem entsprechenden Equate geladen werden muss. Länge = 1 Byte.

(r)
r= Register mit dem Wert des entsprechenden Equates.

FUNCT=
gibt an, welche Funktion des **CTIME**-Makros ausgeführt werden soll.

***CONV**
Der 1. Eingabe-Zeitstempel wird in eine andere Darstellung konvertiert. Das Ergebnis wird in den Ausgabezeitstempel geschrieben.

***ADD**
Die Eingabezeitspanne wird zum 1. Eingabe-Zeitstempel addiert. Das Ergebnis steht im Ausgabe-Zeitstempel. Die Eingabe-Zeitspanne kann auch negativ sein. Das Format der Eingabe-Zeitspanne wird mit dem Operanden FRMDIN angegeben.

***ADDLL**
Wie bei *ADD wird die Eingabezeitspanne zum 1. Eingabe-Zeitstempel addiert.
Es wird jedoch berücksichtigt, dass zwischen dem Eingabe-Zeitstempel und dem Ausgabe-Zeitstempel ein Sommer-Winterzeit-Umstellungszeitpunkt ist. (Solche Umstellungstage sind nicht wie üblich 24 Stunden lang, sondern um die Umstellungsdifferenz verlängert oder verkürzt. Bei der Addition werden sie im Gegensatz zu *ADD trotzdem als 24-Stunden-Tage betrachtet.) Das Ergebnis steht im Ausgabe-Zeitstempel. Die Eingabe-Zeitspanne kann auch negativ sein.

Beispiel
Die Umstellung von Winter- auf Sommerzeit fand am 30.03.2008 um 02:00:00 statt.

```
Eingabe-Zeitstempel     2008-03-29,23:00:00
Eingabe-Zeitspanne          +00001-00:00:00 (1 Tag)

Ergebnis von *ADD       2008-03-31,00:00:00
Ergebnis von *ADDLL     2008-03-30,23:00:00
```

***DIFF**
Es wird die Zeitdifferenz vom 2. Eingabe-Zeitstempel zum 1. Eingabe-Zeitstempel berechnet. Das Ergebnis steht in der Ausgabe-Zeitspanne.

adr
symbolische Adresse (Name) eines Feldes, das zuvor mit dem entsprechenden Equate geladen werden muss. Länge = 1 Byte.

(r)
r= Register mit dem Wert des entsprechenden Equates.

INF1IN=
INF2IN=
gibt an, wie bei dem jeweiligen Eingabe-Zeitstempel der Tag des Jahres angegeben ist.

***CALEND**
Die Angabe erfolgt in der üblichen Darstellung von Monat und Tag des Monats.

***JULIAN**
Die Angabe des Tages erfolgt in der julianischen Schreibweise: die Tage des Jahres werden, beginnend mit dem 1. Januar, fortlaufend durchgezählt.

adr
symbolische Adresse (Name) eines Feldes, das zuvor mit dem entsprechenden Equate geladen werden muss. Länge = 1 Byte.

(r)
r= Register mit dem Wert des entsprechenden Equates.

LINKADR=
gibt an, auf welche Weise dem Anwenderprogramm die Adresse des Einsprungpunktes IGTCTI für die **CTIME**-Routine im Subsystem GET-TIME zur Verfügung gestellt wird. Bei MF=E muss LINKADR angegeben werden, in allen anderen Fällen ist die Angabe von LINKADR wirkungslos.

***NONE**
Bei der Übersetzung generiert der Assembler einen Externverweis für die Einsprungstelle IGTCTI, der beim Binden über die Autolink-Funktion des BLS aufgelöst wird. Dieser Wert kann verwendet werden, wenn das Modul, das den **CTIME**-Aufruf enthält,

– immer mit dem Dynamischen Bindelader DBL gebunden und geladen wird (in diesem Fall lässt man den **CTIME** in der E-Form eine V-Konstante absetzen, welche beim Ladevorgang durch das BLS versorgt wird) oder
– mit dem BINDER des neuen BLS (siehe Handbuch „BINDER“ [5]) unter der BINDER-Anweisung SET-EXTERN-RESOLUTION RESOLUTION=STD gebunden wird.

linkadr
symbolische Adresse (Name) eines Wortes, in dem der Anwender vor dem **CTIME**-Aufruf die Adresse der Einsprungstelle IGTCTI bereitgestellt hat.
Das folgende Beispiel zeigt, wie dem Programm die Adresse der Einsprungstelle IGTCTI zunächst durch einen geeigneten **BIND**-Aufruf im Register R1 zur Verfügung gestellt und anschließend für den **CTIME**-Aufruf in das mit linkadr bezeichnete Wort übertragen werden kann:

```
          BIND   MF=E,PARAM=BINDPL
          :
          CTIME  MF=E,PARAM=OPLIST,LINKADR=AENTRY
          :
AENTRY    DS     F
OPLIST    CTIME  MF=L,...
BINDPL    BIND   MF=L,SYMBOL=IGTCTI,SYMBLAD=AENTRY
```

Auf diese Weise muss dem Anwenderprogramm die Einsprungadresse der **CTIME**-Routine immer dann mitgeteilt werden, wenn keiner der unter LINKADR=*NONE erwähnten Fälle vorliegt, z.B. also insbesondere dann, wenn das Modul mit dem **CTIME**-Aufruf z.B. mit dem BINDER unter der BINDER-Anweisung SET-EXTERN-RESOLUTION RESOLUTION=MANDATORY gebunden wird.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

Bei der E-Form und der M-Form des Makroaufrufs wird das Label des Datenbereichs im Operanden PARAM angegeben. Der Datenbereich muss auf Doppelwort ausgerichtet sein.
Voreinstellung: NTICPA
Beim Aufruf des Makros mit MF=L muss der Anwender dieses Label explizit angeben, sonst wird eine `MNOTE` ausgegeben.

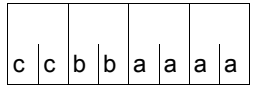
Registerverwendung

Beim Aufruf des Makros **CTIME** werden folgende Register benötigt:

R1	wird vom Makro mit der Adresse des Datenbereichs geladen.
R13	ist vor dem Makroaufruf mit der Adresse eines 18 Worte langen Sicherstellungsbereiches zu laden, den das aufrufende Programm zur Verfügung stellen muss.
R14	wird vom Makro mit der Rückkehradresse des Anwenderprogramms geladen.
R15	wird von der (über **CTIME**) gerufenen Routine überschrieben.

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros CTIME wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

cc	bb	aaaa	Erläuterung
00	00	0000	Funktion erfolgreich ausgeführt.
	01	0001	Die Funktion wurde abgebrochen: Ein angegebener Zeitstempel im Datenbereich ist fehlerhaft. *Maßnahme:* Zeitstempel korrigieren.
	01	0002	Die Funktion wurde abgebrochen: Die Zoneninformation zu einem angegebenen Zeitstempel ist fehlerhaft. *Maßnahme:* Zonen-Information korrigieren.
	01	0003	Die Funktion wurde abgebrochen: Die Angabe einer Zeitspanne ist fehlerhaft. *Maßnahme:* Zeitspannen-Angabe korrigieren.
	01	0006	Die Funktion wurde abgebrochen: Die Spezifikation der Ein- oder Ausgabedaten durch die Operanden im Datenbereich ist fehlerhaft. *Maßnahme:* Spezifikation korrigieren.
	01	000D	Die Funktion wurde abgebrochen: Die Adresse der CHDATEs-Tabelle wurde nicht auf Wortgrenze ausgerichtet oder das Adressfeld ist leer. *Maßnahme:* Programmkorrektur.
	01	000E	Die Funktion wurde abgebrochen: Der Aufbau der CHDATEs-Tabelle ist fehlerhaft. Die gewünschte Verarbeitung der Daten ist nicht möglich. Mögliche Fehler sind: 1. Das erste Byte jedes Eintrags ist nicht X'00' . 2. Das letzte Byte jedes Eintrags ist nicht alternierend X'00' und X'01' . 3. Die Einträge sind nicht monoton aufsteigend in Bezug auf den einzuordnenden Zeitstempel. Das kann daran liegen, dass – der Tabellenaufbau mit GTIME fehlerhaft erfolgte – z.B. nur CHDATEs für die kommenden zwei Jahre eingetragen wurden, aber ein Zeitstempel in drei Jahren eingeordnet werden soll; die der Tabelle folgenden Daten werden dann als Fortsetzung der Tabelle betrachtet. 4. Die zeitliche Differenz zwischen zwei Einträgen (die ersten beiden Einträge ausgenommen) liegt nicht im Bereich von 4 bis 8 Monaten *Maßnahme:* Korrektur der Tabelle oder des tabellenaufbauenden Programms.
00	04	FFFF	Die Funktion wurde abgebrochen: Der Datenbereich ist nicht auf Doppelwortgrenze ausgerichtet. *Maßnahme:* Programmkorrektur.

cc	bb	aaaa	Erläuterung
02	00	0007	Warnung: Die Funktion wurde ausgeführt, der Ausgabe-Zeitstempel liegt jedoch vor oder nach den Umstellungszeitpunkten. Es kann nicht entschieden werden, ob es sich um einen Sommer- oder Winter-Zeitstempel handelt. Es wird angenommen, dass ein Winterzeitstempel gewünscht wurde. Um die Eindeutigkeit des Ausgabe-Zeitstempels zu erhalten, muss die Zoneninformation mit abgespeichert werden. Diese Warnung kann nur bei einem LTI-Zeitstempel auftreten oder bei einen FZ-Zeitstempel, dem eine CHDATE-Tabelle mitgegeben wurde.
02		0008	Warnung: Die Funktion wurde ausgeführt, die Darstellung des angegebenen Zeitstempels liegt jedoch in einem Umstellungszeitintervall, das eigentlich nicht vorkommen kann. Es wird ein Winter-Zeitstempel angenommen. Wenn bekannt ist, ob es sich um einen Sommer- oder Winter-Zeitstempel handelt, kann diese Information mitgegeben werden (siehe Operand FRMnZIN). Diese Warnung kann nur bei einem LTI-Eingabe-Zeitstempel auftreten oder bei einem FZ-Zeitstempel, dem eine CHDATE-Tabelle mitgegeben wurde. *Maßnahme:* Wenn die Information vorhanden ist, ob es sich um einen Sommer- oder Winter-Zeitstempel handelt, kann sie mitgegeben werden. Zur Verarbeitung dieser Information muss der Operand FRMxZIN entsprechend gesetzt werden.
02		0009	Warnung: Eingabe-Zeitstempel nicht eindeutig. Die Funktion wurde ausgeführt, die Darstellung des angebenen Zeitstempels liegt jedoch in einem Umstellungsintervall, in dem die Darstellung doppelt vorkommt. Es wird ein Sommer-Zeitstempel angenommen. Wenn bekannt ist, ob es sich um einen Sommer- oder Winter-Zeitstempel handelt, kann diese Information mitgegeben werden (siehe Operand FRMnZIN). Diese Warnung kann nur bei einem LTI-Eingabe-Zeitstempel auftreten oder bei einem FZ-Zeitstempel, dem eine CHDATE-Tabelle mitgegeben wurde. *Maßnahme:* Wenn die Information vorhanden ist, ob es sich um einen Sommer- oder Winter-Zeitstempel handelt, kann sie mitgegeben werden. Zur Verarbeitung dieser Information muss der Operand FRMxZIN entsprechend gesetzt werden.
02		000A	Warnung: Ausgabe-Zeitstempel nicht eindeutig. Die Funktion wurde ausgeführt, der Ausgabe-Zeitstempel liegt jedoch in einem Zeitintervall am Umstellungszeitpunkt Sommer- auf Winterzeit, der doppelt vorkommt. Um die Eindeutigkeit des Ausgabe-Zeitstempels zu erhalten, muss die Zoneninformation mit abgespeichert werden. Diese Warnung kann nur bei einem LTI-Ausgabe-Zeitstempel auftreten oder bei einem FZ-Zeitstempel, dem eine CHDATE-Tabelle mitgegeben wurde.

cc	bb	aaaa	Erläuterung
02		000B	Warnung: Die Funktion wurde ausgeführt, bei der Addition von einer Zeitspanne zu einem Zeitstempel ist jedoch die obere Bereichsgrenze (s. Wertebereiche) überschritten worden. Das Ausgabedatum erhält als Wert die obere Bereichsgrenze.
02		000C	Warnung: Die Funktion wurde ausgeführt, bei der Subtraktion von einer Zeitspanne von einem Zeitstempel ist jedoch die untere Bereichsgrenze (s. Wertebereiche) unterschritten worden. Das Ausgabedatum erhält als Wert die untere Bereichsgrenze.
02		000F	Warnung: Die Funktion wurde ausgeführt, der Ausgabe-Zeistempel liegt jedoch in einem Umstellungszeitintervall, das eigentlich nicht vorkommen darf. Diese Warnung kann nur bei Ausgabe-Zeitstempeln der *ADDLL-Funktion vorkommen.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Wenn Subcode1 ungleich Null ist, wird nur der Returncode im Standardheader gesetzt und keine weiteren Daten übertragen.

Das aufrufende Programm wird beendet, wenn folgende Fehler auftreten:
– Der Datenbereich ist dem Aufrufer nicht zugewiesen.
– Der Datenbereich ist nicht auf Doppelwortgrenze ausgerichtet.
– Der Datenbereich ist gegen Schreibzugriff geschützt.

Beispiel für die Nutzung einer anwenderspezifischen Tabelle von CHDATEs

```
         TITLE 'GTIME and CTIME'
*
         PRINT NOGEN,BASE
         GPARMOD 31
*
CGTIME   @ENTR TYP=M
         BIND MF=E,PARAM=BINDPLG
         BIND MF=E,PARAM=BINDPLC
*
         LA    R1,GTPAR1
         USING NTIGPL,R1
         MVC   NTIGCHD(NTIGCHDL),=A(0,128) ------------------------------  (1)
         LA    R3,CHDATES ------------------------------------------------  (2)
*
         @CYCL ,
         GTIME MF=E,PARAM=(R1),LINKADR=AENTRYG ---------------------------  (3)
         MVC   0(NTIGCHDL,R3),NTIGCHD ------------------------------------  (4)
         LA    R3,NTIGCHDL(R3) ---------------------------------------------  (5)
*
         @WHEN NE
         CLC NTIGRET,=A(0)
         @BREA , ---------------------------------------------------------  (6)
*
         @BEND ,
*
*********************************************************************
* Following this part of the program, the system's CHDATES are      *
* already stored in the CHDATES memory area.                        *
*********************************************************************
*
DTH1     LA    R1,CTPAR2
         USING NTICPL,R1
         CTIME MF=M,CHDLOUT=CHDATES
*
         RDATA MF=(E,READDATE) -------------------------------------------  (7)
         LA    R1,CTPAR2
         USING NTICPL,R1
         MVC   NTIC1MD(L'DATEDATE),DATEDATE ------------------------------  (8)
         MVC   NTIC1MT(L'DATETIME),DATETIME
         MVC   NTIC3MZ(14),=C'+01:00:-01:00-'
         CTIME MF=E,PARAM=(R1),LINKADR=AENTRYC ---------------------------  (9)
*
END      @EXIT
*
```

```
*********************************************************************
* With the above section of the program, RDATA is used to read in a  *
* date in the format "yyyy mm dd hh mm ss" from the terminal and to  *
* convert it using the CTIME from UTC to FZ, taking into account the *
* CHDATEs table. The event is shown in ISO4 format in the parameter  *
* list in the output time stamp.                                     *
*********************************************************************
*
READDATE RDATA INDATE,0,MF=L ------------------------------------------ (10)
*
INDATE   DS    0CL30
DATELEN  DS    CL2
DATERES  DS    CL2
DATEDATE DS    CL10
DATESPAC DS    CL1
DATETIME DS    CL8
*
GTPAR1   GTIME MF=L,CHDATE=NEXT ---------------------------------------(3)
*
CTPAR2   CTIME MF=L,FUNCT=*CONV,BASE1IN=*UTC,FRM1IN=*ISO4,           C
               BASEOUT=*LTI,FRMOUT=*ISO4 ------------------------------ (9)
*
CHDATES  DS    100D ------------------------------------------------------ (11)
*
BINDPLG  BIND MF=L,SYMBOL=I@GTIME,SYMBLAD=AENTRYG
BINDPLC  BIND MF=L,SYMBOL=IGTCTI,SYMBLAD=AENTRYC
*
AENTRYG  DS    A
AENTRYC  DS    A
*
         @END   ,
*********************************************************************
* DSECTs                                                            *
*********************************************************************
*
NTICPL   CTIME MF=D
NTIGPL   GTIME MF=D
         END
```

(1) Vorbelegung der **GTIME**-Parameterliste mit einem Wert, der sichert, dass die folgenden CHDATEs in chronologischer Reihenfolge geliefert werden.

(2) In Register R3 wird die Speicheradresse des CHDATEs geladen.

(3) Mit dem **GTIME**-Aufruf wird das erste bzw. nächstes CHDATE geholt.

(4) Es wird im Speicherbereich gesichert.

(5) Der Zeiger auf die CHDATEs-Tabelle wird um X'08' hochgezählt.

(6) Die Schleife soll verlassen werden, wenn es kein nächstes CHDATE mehr gibt (oder im Fehlerfall).

(7) Der **RDATA** fordert die Eingabe eines Datums über Datensichtstation an. Das Datum muss im Format `yyyy-mm-dd hh:mm:ss` eingegeben werden. Die Eingabe der Trennzeichen ist nicht erforderlich, es genügt jeweils ein Leerzeichen.

(8) Der **CTIME**-Datenbereich wird mit Datum und Zone versorgt.

(9) Der Eingabezeitstempel wird von UTC in LTI konvertiert. Der Ausgabezeitstempel soll - ebenso wie der Eingabezeitstempel - im ISO4-Format vorliegen.

(10) **RDATA**-Datenbereich mit Definition des Eingabefeldes `INDATE`.

(11) Bereichsdefinition für maximal 100 CHDATEs.

Ablaufprotokoll

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,cgtime), -
//        compiler-action=module-generation(module-format=llm), -
//        macro-library=$tsos.syslib.assembh.012, -  ------------------- (12)
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,cgtime)), -
//        test-support=*aid
%  ASS6011 ASSEMBLY TIME: 1605 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 101 MSEC
//end
%  ASS6012 END OF ASSEMBH
/add-file-link link-name=blslib00,file-name=$tsos.syslib.assembh.012  -- (13)
/load-executable-program library=macexmp.lib,element-or-symbol=cgtime, -
/     dbl-parameters=*par(resolution=*par(alternate-libraries=*yes))
%  BLS0523 ELEMENT 'CGTIME', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'CGTIME', VERSION ' ' OF '<date> <time>' LOADED
/%in dth1 <%d %@(chdates) -> %xl96>  ------------------------------------ (14)
/%in end <%d indate,%1 -> %xl232>  ------------------------------------- (15)
/%r
```

```
*** TID: 005000D8 *** TSN: 2QSE ****************************************
CURRENT PC: 0000006A    CSECT: CGTIME  ***********************************
V'00000208' = CGTIME   + #'00000208'  ------------------------------------ (16)
00000208 (00000208) 008FF960 489C4000 0090D566 AC464001    ..9-.. ...N... .
00000218 (00000218) 0091BA3A 1E2A4000 00929F0D 900E4001    .j.... ..k.... .
00000228 (00000228) 009383E1 01F24000 009468B4 73D64001    .lc..2 ..m...O .
00000238 (00000238) 00954D87 E5BA4000 0096325B 579E4001    .n(gV. ..o.$.. .
00000248 (00000248) 0097172E C9824000 009804CF 49A04001    .p..Ib ..q.... .
00000258 (00000258) 00FFFFFF FFFFFF00 0099CE76 2D684001    .~~~~~~..r.... .
*2012-01-20 14:36:35 ------------------------------------------------------ (17)
SRC_REF:   230 SOURCE: CGTIME  PROC: CGTIME  *******************************
INDATE          = |....2012-01-20 14:36:35.......|
CURRENT PC: 00000098    CSECT: CGTIME  ***********************************
V'00000120' = CGTIME   + #'00000120' ------------------------------------- (18)
00000120 (00000120) 00050702 00000000 01010001 01010001    ................
00000130 (00000130) 02010000 01010100 F2F0F1F2 60F0F160    ........2012-01-
00000140 (00000140) F2F0F0F0 F0404040 F1F47AF3 F67AF3F5    20000   14:36:35
00000150 (00000150) 00000000 00000000 00000000 00000000    ................
00000160 (00000160) 00000000 00000000 F0F0F0F0 60F0F060    ........0000-00-
00000170 (00000170) F0F0F0F0 F0404040 F0F07AF0 F07AF0F0    00000   00:00:00
00000180 (00000180) 00000000 00000000 00000000 00000000    ................
00000190 (00000190) 00000000 00000000 4EF0F0F0 F0F0F0F0    ........+0000000
000001A0 (000001A0) F0F0F060 F0F07AF0 F07AF0F0 00000000    000-00:00:00....
000001B0 (000001B0) 00000000 00000000 F2F0F1F2 60F0F160    ........2012-01-
000001C0 (000001C0) F2F0F0F2 F040C6D9 F1F57AF3 F67AF3F5    20020 FR15:36:35
000001D0 (000001D0) 4EF0F17A F0F060F0 F17AF0F0 60E6F0F0    +01:00-01:00-W00
000001E0 (000001E0) F0F0F0F0 00000208 00000000 0000016C    0000...........%
000001F0 (000001F0) 00000000 00000000 00000000 00000000    ................
00000200 (00000200) 00000000 00F8C180                      .....8A.
```

(12) Bei der Assemblierung wird die Bibliothek angegeben, die die für strukturierte Programme benötigten @-Makros enthält.

(13) Für den Ablauf von strukturierten Programmen wird ebenfalls die Bibliothek mit den @-Makros benötigt. Sie wird hier mit dem Linknamen BLSLIB00 und die Angabe ALTERNATE-LIBRARIES=*YES im Ladeaufruf zugewiesen.

(14) An der symbolischen Adresse DTH1 soll die Tabelle der CHKDATEs ausgegeben werden.

(15) An der symbolischen Adresse END sollen das Feld INDATE und die Operandenliste ausgegeben werden. Die Anfangsadresse der Operandenliste steht nach einem erfolgreichen **CTIME**-Aufruf im Register R1. Die Länge der Operandenliste beträgt 232 Byte (diese Information erhält man durch die Auflösung der DSECT mit **CTIME** MF=D).

(16) Die symbolische Adresse DTH1 wurde erreicht. Die Tabelle der CHDATEs wird ausgegeben. Im System sind z.Zt. 10 CHDATEs bekannt. Sie liegen im TODR-Format vor, um 1 Byte nach rechts verschoben.
Das Ende der CHDATEs-Tabelle markiert das System mit X'00FFFFFF FFFFFF00'.

(17) Die Eingabe eines Zeitstempels wird angefordert (Prompt „*“).

(18) Die symbolische Adresse END wurde erreicht. Der Inhalt des Feldes INDATE wird ausgegeben: Der eingegebene Zeitstempel wurde von **RDATA** korrekt eingelesen.

Ausgabe der Operandenliste:

- Der 1. Eingabezeitstempel der Operandenliste entspricht dem von **RDATA** eingelesenen Zeitstempel „2012-01-20 14:36:35“. Die Zeitbasis ist UTC. Der Zeitstempel liegt im ISO4-Format vor.
- Ein 2. Eingabezeitstempel wurde - ebenso wie die Eingabezeitspanne - nicht angegeben.
- Der Ausgabezeitstempel (Zeitbasis LTI) liegt im ISO4-Format vor und enthält folgende Informationen:
  Der angegebene Tag ist der 20 Tag des Jahres und ein Freitag.
  Die aktuelle Zeit wurde neu berechnet. Die LTI-Zeit lautet „15:36:35“.
  Die Zonen-Information „+01:00-01:00-W“ gibt darüber Auskunft, dass die Zonen-Differenz zwischen UTC und LTI eine Stunde und die Differenz zwischen Sommer- und Normalzeit ebenfalls eine Stunde beträgt, und dass Winterzeit eingestellt ist („W“ wie Winter, Normalzeit; Sommerzeit würde mit „S“ wie „Sommer“ angezeigt werden).
- Es wurde keine Ausgabezeitspanne berechnet.

CUPAB – Datenbereich adressieren (24-Bit-Schnittstelle)

Allgemeines

Anwendungsgebiet: Abfragen und Zugriff zu Listen und Tabellen; siehe Seite 158
Ein-/Ausgabe; siehe Seite 159
Makrotyp: O-Typ; siehe Seite 28

Stand der Beschreibung: TIAM V13.2A

Der Makro **CUPAB** ist nur im 24-Bit-Adressierungsmodus anzuwenden.
Im 31-Bit-Adressierungsmodus muss die Form MF=C/D des jeweiligen Ein-/Ausgabemakros (**RDATA**, **WROUT**, **WRTRD**) verwendet werden.

Makrobeschreibung

Mit dem Makro **CUPAB** kann der Benutzer die Felder und Flags in den Operandenlisten der Makros **RDATA**, **WROUT** und **WRTRD** symbolisch ansprechen. Der Makroaufruf **CUPAB** generiert hierzu eine Dummy Section (DSECT) zum Datenbereich mit 24-Bit-Adressen.

Makroaufrufformat und Operandenbeschreibung

[name] CUPAB
D

name
Wenn im Namensfeld ein Eintrag angegeben ist, wird er als der DSECT-Name verwendet. Enthält das Namensfeld keinen Eintrag, dann wird CUPAB als DSECT-Name generiert.

D
Bewirkt die Generierung einer DSECT für den Datenbereich. Wenn dieser Operand nicht angegeben ist, wird eine MNOTE-Meldung ausgegeben.
Die Generierung der DSECT wird davon aber nicht beeinträchtigt.

– Für die Operandentabelle des Aufrufs **RDATA** gelten folgende Feldnamen:

Feldname	Byte	Bedeutung
CURAREAW	0 - 3	Ganzwort, das CUREDIT1 und CURAREA enthält
CUREDIT1	0	Eingabe-Aufbereitungsbyte 1
CURAREA	1 - 3	Adresse des Benutzer-Einlesebereichs
CURFTB	4	Flag
CUREDIT2	5	Eingabe-Aufbereitungsbyte 2
CURALEN	6 - 7	Länge des Benutzer-Einlesebereiches
CURERRW	8 - 11	Ganzwort, das CURACI und CURERROR enthält
CURACI	8	Indikator für SYSDTA-Zuweisungsänderung
CURERROR	9 - 11	Fehleradresse
L@RDATAB		Länge der Operandentabelle für RDATA

– Für die Operandentabelle des Aufruf **WROUT** gelten folgende Feldnamen:

Feldname	Byte	Bedeutung
CUWMSGW	0 - 3	Ganzwort, das CUWEDIT1 und CUWMSG enthält
CUWEDIT1	0	Ausgabe-Aufbereitungsbyte 1
CUWMSG	1 - 3	Adresse der Nachricht im Benutzerprogramm
CUWERRW	4 - 7	Ganzwort, das CUWEDIT2 und CUWERROR enthält
CUWEDIT2	4	Ausgabe-Aufbereitungsbyte 2
CUWERROR	5 - 7	Fehleradresse
L@WROUTB		Länge der Operandentabelle für WROUT

– Für die Operandentabelle des Aufrufs **WRTRD** gelten folgende Feldnamen:

Feldname	Byte	Bedeutung
CUBMSGW	0 - 3	Ganzwort, das CUBOEDT1 und CUBMSG enthält.
CUBOEDT1	0	Ausgabe-Aufbereitungsbyte 1
CUBMSG	1 - 3	Adresse des Nachrichtenausgabebereiches
CUBAREAW	4 - 7	Ganzwort, das CUBIEDT1 und CUBAREA enthält
CUBIEDT1	4	Eingabe-Aufbereitungsbyte 1
CUBAREA	5 - 7	Adresse des Eingabereiches
CUBOEDT2	8	Ausgabe-Aufbereitungsbyte 2
CUBIEDT2	9	Eingabe-Aufbereitungsbyte 2
CUBALEN	10 - 11	Länge des Eingabereiches
CUBERRW	12 - 15	Ganzwort, das CUBERROR enthält
reserviert	12	-
CUBERROR	13 - 15	Fehleradresse
L@WRTRDB		Länge der Operandentabelle für WRTRD

Symbolische Konstanten für die Aufbereitungsbytes 1 und 2

CUPAB definiert außer den Feldnamen für die Operandentabellen zu **RDATA, WROUT** und **WRTRD** auch symbolische Konstanten für die Werte der Operanden edit, edit1 und edit2.

Die folgenden Tabellen geben eine Übersicht über die Namen der durch **CUPAB** definierten symbolischen Konstanten, ihre aktuellen Werte, ihre Entsprechungen in den symbolischen Aufbereitungsoperanden bei Angabe des Operanden MODE und über ihre Zulässigkeit in den Mode-Arten.

a) Ausgabe-Aufbereitungsbyte 1 (Output Edit Option Byte 1)

CUPAB-Name	Bit	entsprechender MODE-Operand	zulässig (X) bzw. obligatorisch (1) bei MODE=			
			COMP	LINE	FORM	PHYS
CWR1CODE	2^0	OTRSUP=	X	0	0	0
CWR1LNET	2^1	OLINEND=	X	0	1	1
	2^2	reserviert für MODE=	0	1	0	1
CWR1RSET	2^3	OMANUAL=	X	0	0	0
CWR1HOM	2^4	OHOM=	0	X	0	0
CWR1PTPE	2^5	OPTAPE=	X	0	0	0
	2^6	reserviert für MODE=	0	0	1	1
CWR1HARD	2^7	OHCOPY=	X	X	0	X

b) Ausgabe-Aufbereitungsbyte 2 (Output Edit Option Byte 2)

CUPAB-Name	Bit	entsprechender MODE-Operand	zulässig (X) bzw. obligatorisch (1) bei MODE=			
			COMP	LINE	FORM	PHYS
CWR2HDR	2^0	OHDR=	X	0	1	X
CWR2NOLC	2^1	ONOLOGC=	0	X	0	0
CWR2EXT	2^2	EXTEND=	0	X	0	0
CWR2INFO	2^3	OINFO=	0	X	0	0
	2^4	reserviert	0	0	0	0
CWR2POSN	2^5	ONOPOSN=	0	X	0	0
CWR2TRAN	2^5	OTRANS=	0	0	0	X
CWR2BEL	2^6	OBELL=	0	X	0	0
CWR2ETB	2^7	OETB=	0	0	0	X

c) Eingabe-Aufbereitungsbyte 1 (Input Edit Option Byte 1)

CUPAB-Name	Bit	entsprechender MODE-Operand	zulässig (X) bzw. obligatorisch (1) bei MODE=			
			COMP	LINE	FORM	PHYS
CRD1CODE	2^0	ITRSUP=	X	0	0	X
CRD1LNET	2^1	ILINEND=	X	0	1	1
CRD1BACK	2^2	IGETBS=	X	X	X	X
CRD1RSET	2^3	IMANUAL=	X	0	0	0
CRD1LCT	2^4	ILCASE=	X	X	X	X
	2^5	reserviert für MODE=	0	1	0	1
	2^6	reserviert für MODE=	0	0	1	1
CRD1HDR	2^7	IHDR=	X	0	1	X

d) Eingabe-Aufbereitungsbyte 2 (Input Edit Option Byte 2)

CUPAB-Name	Bit	entsprechender MODE-Operand	zulässig (X) bzw. obligatorisch (1) bei MODE=			
			COMP	LINE	FORM	PHYS
CRD2GFC	2^0	IGETFC=	0	X	0	0
	2^1	reserviert	0	0	0	0
CRD2CFD	2^2	ICFD=	0	X	0	0
CRD2GIC	2^3	IGETIC=	0	X	0	0
	2^4	reserviert	0	0	0	0
CRD2EXT	2^5	EXTEND=	0	X	0	1
	2^6	reserviert	0	0	0	0
	2^7	reserviert	0	0	0	0

Bei allen nicht reservierten Bits gilt für die Operanden:

MODE-Operand	zugehöriges Bit
= Y	gesetzt (1)
= N	rückgesetzt (0)

Die Bedeutung der MODE-Operanden (und damit der zugehörigen Bits der Eingabe- und Ausgabe-Aufbereitungsbytes) kann aus den Operandenbeschreibungen der Aufrufe **RDATA**, **WROUT** und **WRTRD** ersehen werden.

DCSTA – Operandentabelle für Datenstationseigenschaften generieren

Allgemeines

Anwendungsgebiete: Abfragen und Zugriff zu Listen und Tabellen; siehe Seite 158
Ein-/Ausgabe; siehe Seite 159
Makrotyp: O-Typ; siehe Seite 28

- Stand der Beschreibung: VTSU V13.3A

Makrobeschreibung

Der Makro **DCSTA** unterstützt den Makro **TSTAT**.
DCSTA generiert Empfangsfelder oder symbolische Feldnamen (DSECTs) für die Informationen, die man durch den Aufruf von **TSTAT** erhält.

Makroaufrufformat und Operandenbeschreibung

[name] DCSTA
D / C [,prefix] ,TYPE=TCHAR / PHDIM / LIDIM / VDT[YP] / EDOPT / OFLOW / STNAM / PRNAM / ALL / MONCS / PERPH / BASIC

name
wird zum symbolischen Namen der ersten DS-Anweisung in der Makroauflösung, wenn der Operand C angegeben ist (das Längenattribut ist NULL).
Wird der Operand D angegeben, so gibt der Operand name den Namen des Pseudoabschnittes (DSECT) an.

Fehlt der Operand name, so bildet das System einen Namen aus dem geltenden Präfix (siehe Operand prefix) und dem Operandenwert für TYPE.

C
Ein Speicherbereich mit symbolischen Adressen wird generiert. Dieser Bereich kann im Aufruf des Makros **TSTAT** als Empfangsfeld angegeben werden. Es wird keine CSECT-Anweisung generiert.

D
Ein Pseudoabschnitt (DSECT) wird zusammen mit einer DSECT-Anweisung generiert. Mit den generierten symbolischen Namen kann man ein Empfangsfeld adressieren, das man für den Makro **TSTAT** definiert hat.

prefix
1 bis 3 Zeichen, mit denen die generierten Feldnamen beginnen sollen. Die prefix-Angabe ersetzt die Zeichen „STA“, mit denen standardmäßig die Feldnamen beginnen.

TYPE=
legt fest, welches Empfangsfeld bzw. welche Feldnamen generiert werden sollen:

TCHAR
Abfrage der Charakteristik der Datenstation

PHDIM
Abfrage der physikalischen Eigenschaft der Datenstation

LIDIM
Abfrage der logischen Eigenschaften der Datenstation

VDT[YP]
Abfrage des logischen Typs der Datenstation

EDOPT
Statische Edit-Options

OFLOW
Abfrage der Art der Überlaufsteuerung

STNAM
Abfrage des Datenstationsnamens

PRNAM
Abfrage des Prozessorsnamens

ALL
Abfrage der Informationen von TCHAR bis PRNAM

MONCS
Beschreibung von Monitor und Zeichensätzen

PERPH
Abfrage der angeschlossenen Peripherie

BASIC
Grundinformationen der Datenstation

Funktionsweise

Will man mit dem Makro **TSTAT** Informationen anfordern, so kann man das Empfangsfeld explizit definieren oder mit dem Makroaufruf **DCSTA C,...** definieren lassen. Hat man das Empfangsfeld explizit definiert, so kann man mit dem Aufruf **DCSTA D,...** einen Pseudoabschnitt (DSECT) generieren lassen, der eine Adressenstruktur für das Empfangsfeld darstellt. Der Pseudoabschnitt muss durch eine USING-Anweisung mit dem Empfangsfeld verknüpft werden. Für die Abfrage von Bitwerten generiert der Makro **DCSTA** symbolische Konstanten, mit denen der Feldinhalt verglichen werden kann.
Die Feldnamen, die **DCSTA** standardmäßig generiert, können der Makroauflösung im Beispiel entnommen werden. Sie beginnen einheitlich mit den Buchstaben „STA“. Man kann diese Buchstaben durch eine selbstgewählte Zeichenfolge ersetzen lassen.

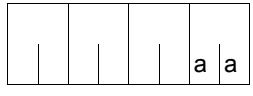
Die Anfangsadressen der Empfangsfelder, die der Aufruf **DCSTA C,...** generiert, müssen im Aufruf des Makros **TSTAT** angegeben werden.

Sie lauten standardmäßig:

STATCHAR	STASTNAM
STAPHDIM	STAPRNAM
STALIDIM	STAALL
STAVDT	STAMONCS
STAEDOPT	STAPERPH
STAOFLOW	STABASIC

Rückinformation und Fehleranzeigen

R15:



Über die Ausführung des Makros DCSTA wird im Register R15 rechtsbündig ein Returncode angegeben.

X'aa'	**Erläuterung**
X'00'	Normale Beendigung
X'04'	nicht behebbarer Fehler
X'08'	Fehler im Datenbereich
X'0C'	Keine Teilnehmerdatenstation vorhanden
X'10'	Länge des Empfangsfeldes ist zu klein: Es wurde nur ein Teil der Information geliefert, wenn der Operand ALL angegeben wurde. Bei allen anderen Operanden wird keine Information übertragen.
X'14'	Die gewünschte Information ist (teilweise) nicht verfügbar.

Beschreibung der übergebenen Information:

- TCHAR: Physikalischer Typ (Bereichslänge: 8 Byte)

Byte	Symb. Name	Bedeutung
0	STAPTTYP (STADCAMP) (STADCAMT)	Partnertyp: Partner ist ein DCAM-Programm Partner ist eine Datenstation
1	STADVTYP (STAD1000) (STAD100E) (STADT100) (STADPT80) (STAD8110) (STAD8151) (STAD8152) (STAD8160) (STAD8162) (STAD9731) (STAD9750) (STAD9751) (STAD9752) (STAD9753) (STAD9754) (STAD9755) (STAD9763) (STAD8122) (STAD8121) (STAD9001) (STAD9002) (STAD9003) (STAD9004) (STAD9012) (STAD9013) (STAD0131) (STAD0189) (STAD9022) (STAD1118) (STAD1119) (STAD3270) (STADHOST) (STADAP) (STAD9021) (STAD3287) (STAD9014) (STAD9026) (STADFE)	Behandelter Gerätetyp (z.B.): Schreibstation T1000 Schreibstation FS100-E Schreibstation T100 Schreibstation PT80 Schreibstation 8110 Datensichtstation 8151 Datensichtstation 8152 Datensichtstation 8160 Datensichtstation 8162 Grafikstation 9731 Datensichtstation 9750/9749 Datensichtstation 9751 Datensichtstation 9752 Datensichtstation 9753 Datensichtstation 9754 Datensichtstation 9755 Datensichtstation 9763 Drucker 8122 Drucker 8121 Drucker 9001 Drucker 9002 Drucker 9003 Drucker 9004 Drucker 9012 Drucker 9013 Drucker 9001-31 Drucker 9001-8931 Drucker 9022 Drucker 9011-18 Drucker 9011-19 Datensichtstation 3270 Programm im Server AP-Station Drucker 9021 Drucker 3287 Drucker 9014 Drucker 9026 (HDLC, kompatibel 9025) Front-End Datensichtstation (FHS-DOORS)

Byte	Symb. Name	Bedeutung
2	STATCHR2	Zeichenvorrat:
	(STATC2EX)	zweiter Zeichenvorrat vorhanden
	(STATC2LC)	Kleinbuchstaben vorhanden
	(STATC2DT)	deutsche (statt internationale) Tastatur generiert
	(STATC2DF)	Byte 2 ist definiert
3	STATCHR3	Zusätze an der Datensichtstation:
	(STATC3H1)	Lokaler Hardcopy-Drucker generiert oder mit TCHNG zugewiesen
	(STATC3H2)	zentraler Hardcopy-Drucker generiert oder mit TCHNG zugewiesen
	(STATC3IC)	Ausweisleser generiert oder mit TCHNG zugewiesen
	(STATC3AP)	APL Zusatz generiert oder mit TCHNG zugewiesen
	(STATC3GF)	Grafik Zusatz generiert oder mit TCHNG zugewiesen
	(STATC3DZ)	Dezentrale Formatierung
	(STATC3DF)	Byte 3 ist definiert
4	STATCHR4	Datensichtstationsfunktionen
	(STATC4CO)	4 Farben
	(STATC4ZF)	Zeichen- und Feldattribute
	(STATC4ST)	Status von der Station möglich
	(STATC4HI)	Hardware Systemzeile vorhanden
	(STATC4C8)	8 Farben
	(STATC4HP)	HP Laser Jet II
	(STATC4DF)	Byte 4 ist definiert
5	STATTCHRS	Informationen aus der Statusmeldung
	(STATTCSDT)	deutsche Tastatur angeschlossen
	(STATTCSHC)	lokales Hardcopygerät angeschlossen
	(STATTCSIC)	Ausweisleser angeschlossen
	(STATTCSDF)	Status von der Station vorhanden
6	STACTRLU	Gerätetyp der Druckersteuerung, wenn die Datenstation ein Drucker ist, X' 00' bei Druckersteuerung 8112, generierter Gerätetyp (siehe oben) bei Datensichtstationen
7	STACHCAD	Kanaladresse des zentralen Hardcopy-Gerätes

Hinweise

– Die DSS 9749 kann im PDN als eigener Gerätetyp generiert werden. Für Anwenderprogramme jedoch wird sie beim **TSTAT**-Makro immer als 9750 ausgewiesen.

– Die DSS 9758 M4 kann im PDN als DSS 9755 oder DSS 9763 generiert werden. Für Anwendungsprogramme wird sie beim **TSTAT**-Makro jedoch immer als DSS 9755 ausgewiesen.

- PHDIM: Physikalische Eigenschaften (Bereichslänge: 8 Byte)

Byte	Symb. Name	Bedeutung
0 - 1	STALLEN	Physikalische Zeilenlänge
2 - 3	STANOLIN	Physikalische Zeilenanzahl (bei Schreibstationen unbeschränkt: X' 7FFF')
4 - 5	STAMAXDB	Größter physikalischer Gerätepuffer, d.h. größte Anzahl von Zeichen, die mit einem Ausgabeaufruf zur Datenstation gesendet werden kann. X' 7FFF' (unbeschränkt): Beschränkung nur durch Zugriffsmethode oder Leitung.
6 - 7	-	reserviert

Bit 2^{15} = 1 bedeutet jeweils: Wert nicht verfügbar.

- LIDIM: Logische Eigenschaften (Zeilenmodus) (Bereichslänge: 8 Byte)

Byte	Symb. Name	Bedeutung
0 - 1	STALLLEN	Anzahl der Zeichen je physikalische Zeile im Line-Mode, wobei ' NL' im Text als zwei Zeichen zu zählen ist.
2 - 3	STALNOLN	Anzahl der physikalischen Zeilen, die im Line-Mode ausgegeben werden können, ohne dass die Überlaufkontrolle anspricht.
4 - 5	STALMAXB	Anzahl der Zeichen, die im Line-Mode in einer Nachricht gesendet werden können, ohne dass die Überlaufkontrolle anspricht (in der Regel Zeilen mal Spalten minus 1).
6 - 7	-	reserviert.

Bit 2^{15} = 1 bedeutet jeweils: Wert nicht verfügbar.

- VDT[YP]: Logischer Typ (Bereichslänge: 8 Byte)

Byte	Symb. Name	Bedeutung
0	STAVDT	logischer Gerätetyp:
	(STALINCP)	Zeilen-/Seitendatenstation (LINE MODE)
	(STAFORCP)	Formatdatenstation (FORM MODE)
	(STACMPCP)	Datensichtstationsunterstützung kompatibel zu früheren Betriebssystem-versionen (COMP MODE)
	(STAFYSCP)	Physikalische Datenstationsunterstützung (PHYS MODE)
	(STAEXLCP)	Zeilen-/Seitendatenstation (Ext. LINE MODE)
	(STAAUTLF)	Automatischer Zeilenvorschub für Drucker
	(STANOINP)	Datenstation ist ein Drucker
	(STAEOM=0)	Datenstation ist eine Schreibstation
	(STAEOM=1)	Datenstation ist eine Datensichtstation
1	STAVDTPR	Logisches Geräteprotokoll:
	(STATD810)	810-Protokoll
	(STAT3270)	3270-Protokoll
2 - 7	-	reserviert.

- EDOPT: Statische Edit-Options (Bereichslänge: 8 Byte)

Zur Darstellung werden die symbolischen Operanden des Makros **WRTRD** verwendet.

Byte	Symb. Name	Bedeutung
0	STASEWR1	Ausgabe-Aufbereitungsbyte 1
	(STAWR1MM)	Maske für Ausgabe-Aufbereitungsmodus =STAWR1CO: MODE=COMP =STAWR1LI: MODE=LINE =STAWR1FO: MODE=FORM =STAWR1FY: MODE=PHYS
	(STAWR1CD)	=1: OTRSUP =Y =0: =N
	(STAWR1LE)	=1: ONLINEND =Y =0: =N
	(STAWR1RE)	=1: OMANUAL =Y =0: =N
	(STAWR1HO)	=1: OHOM =Y =0: =N
	(STAWR1PT)	=1: OPTAPE =Y =0: =N
	(STAWR1HC)	=1: OHCOPY =Y =0: =N

Byte	Symb. Name	Bedeutung
1	STASEWR2	Ausgabe-Aufbereitungsbyte 2
	(STAWR2HD)	=1: OHDR =Y =0: =N
	(STAWR2NO)	=1: ONOLOGC =Y =0: =N
	(STAWR2EX)	=1: EXTEND =Y =0: =N
	(STAWR2ET)	=1: OETB =Y =0: =N
	(STAWR2BL)	=1: OBELL =Y =0: =N
	(STAWR2TP)	=1: OTRANS =Y =0: =N
	(STAWR2IM)	=1: OINFO =Y =0: =N
	(STAWR2PN)	=1: ONOPOSN =Y =0: =N
2	STASERD1	Eingabe-Aufbereitungsbyte 1
	(STARD1MM)	Maske für Eingabe-Aufbereitungsmodus =STARD1CO: MODE=COMP =STARD1LI: MODE=LINE =STARD1FO: MODE=FORM =STARD1FY: MODE=PHYS
	(STARD1CD)	=1: ITRSUP =Y =0: =N
	(STARD1LE)	=1: ILINEND =Y =0: =N
	(STARD1BS)	=1: IGETBS =Y =0: =N
	(STARD1PT)	=1: IMANUAL =Y =0: =N
	(STARD1LC)	=1: ILCASE =Y =0: =N
	(STARD1HD)	=1: IHDR =Y =0: =N
3	STASERD2	Eingabe-Aufbereitungsbyte 2
	(STARD2FC)	=1: IGETFC =Y =0: =N
	(STARD2IC)	=1: IGETIC =Y =0: =N
	(STARD2CF)	=1: ICFD =Y =0: =N
	(STARD2EX)	=1: EXTEND =Y =0: =N

Byte	Symb. Name	Bedeutung
4 - 7	-	reserviert

- OFLOW: Steuerung bei Bildschirmüberlauf (Bereichslänge: 8 Byte)

Byte	Symb. Name	Bedeutung
0	STAOFLOW	Art der Überlaufsteuerung:
	(STAOFCTM)	Bit 2^0=1 Überlaufsteuerung bei Benutzung des Timers. Wartezeit mit n Sekunden (sedezimale Angabe) gemäß STAOFTIM
	(STAOFCAK)	Bit 2^1=1 Überlaufsteuerung mit Quittungsanforderung bei Überlauf
	(STAOFCTL)	Bit 2^0=0 keine Überlaufsteuerung Bit 2^1=0 keine Überlaufsteuerung
	(STAOFPGM)	Bit 2^5=0 Überlaufsteuerung durch das System Bit 2^5=1 Überlaufsteuerung durch das Benutzerprogramm (siehe Makro TCHNG, Kommando MODIFY-TERMINAL-OPTIONS)
1	STAOFTIM	Wartezeit (sedezimale Angabe in Sekunden).
2 - 7	-	reserviert

- STNAM: Name der Station, wie in der PDN angegeben (Bereichslänge: 8 Byte)

Byte	Symb. Name	Bedeutung
0 - 7	STASTNAM	Name der Station

- PRNAM: Name des Servers, an den die Station angeschlossen ist, wie in der RDF-Generierung angegeben (Bereichslänge: 8 Byte)

Byte	Symb. Name	Bedeutung
0 - 7	STAPRNAM	Name des Servers

- ALL

Die symbolischen Adressen der Tabellenfelder sind wie oben festgelegt. Die Reihenfolge der Teilbereiche ist:

TCHAR, PHDIM, LIDIM, VDTYP, EDOPT, OFLOW, STNAM, PRNAM

Hinweis

Bei der Angabe TYPE=ALL hat der Bereich eine Größe von 64 Byte.
TYPE=ALL umfasst nicht MONCS, PERPH und BASIC.

- MONCS: Beschreibung von Monitor und Zeichensätzen
  (Bereichslänge: mindestens 14 Byte)

Informationen werden über die Statusmeldung der Datenstation geliefert, wenn diese vorliegt, ansonsten werden Standardwerte angenommen.

Byte	Symb. Name	Bedeutung
0	STAMOCPR	Statusmeldung von der Station
	(STAMOCY)	Statusmeldung liegt vor. Die folgenden Informationen sind der Status-Meldung entnommen.
	(STAMOCN)	Statusmeldung liegt nicht vor. Für die folgenden Informationen werden VTSU-Standardwerte angenommen
1	STAMOTYP	Monitortyp der Station
	(STAMONO)	Datenstation mit einem Monochrom-Bildschirm
	(STACOLOR)	Datenstation mit einem Farb-Bildschirm
	(STAPRINT)	Datenstation ist ein Drucker
2	STAFAT	Feldattribute
	(STAFATY)	Feldattribute können verwendet werden
	(STAFATN)	Feldattribute können nicht verwendet werden
3	-	reserviert
4	STADIM1	Bildschirmgröße 24 Zeilen x 80 Zeichen
	(STADIMY)	Format wird unterstützt (ist mit DIM ansprechbar)
	(STADIMN)	Format wird nicht unterstützt (ist nicht mit DIM ansprechbar)
5	STADIM2	Bildschirmgröße 32 Zeilen x 80 Zeichen
	(STADIMY)	Format wird unterstützt (ist mit DIM ansprechbar)
	(STADIMN)	Format wird nicht unterstützt (ist nicht mit DIM ansprechbar)
6	STADIM3	Bildschirmgröße 43 Zeilen x 80 Zeichen
	(STADIMY)	Format wird unterstützt (ist mit DIM ansprechbar)
	(STADIMN)	Format wird nicht unterstützt (ist nicht mit DIM ansprechbar)
7	STADIM4	Bildschirmgröße 27 Zeilen x 132 Zeichen
	(STADIMY)	Format wird unterstützt (ist mit DIM ansprechbar)
	(STADIMN)	Format wird nicht unterstützt (ist nicht mit DIM ansprechbar)
8-11	-	reserviert
12-13	STACSNO	Anzahl der ansprechbaren Zeichensätze
14	STACS0T	Art des Zeichensatzes 0
	(STACSSIN)	ladbarer Monochromzeichensatz
	(STACSTRI)	ladbarer Farbzeichensatz
	(STACSNO)	nicht ladbarer Zeichensatz

Byte	Symb. Name	Bedeutung
15	STACSOS	Zustand des Zeichensatzes 0
	(STACSNLO)	Zeichensatz kann geladen werden
	(STACSDSS)	Zeichensatz wurde von der Datensichtstation belegt
	(STACSDVN)	Zeichensatz wurde von der DVA geladen
	(STACSDVA)	Zeichensatz wurde von der DVA geladen und zugewiesen
16	STACS1T	Art des Zeichensatzes 1
	(STACSSIN)	ladbarer Monochromzeichensatz
	(STACSTRI)	ladbarer Farbzeichensatz
	(STACSNO)	nicht ladbarer Zeichensatz
17	STACS1S	Zustand des Zeichensatzes 1
	(STACSNLO)	Zeichensatz kann geladen werden
	(STACSDSS)	Zeichensatz wurde von der Datensichtstation belegt
	(STACSDVN)	Zeichensatz wurde von der DVA geladen
	(STACSDVA)	Zeichensatz wurde von der DVA geladen und zugewiesen
18	STACS2T	Art des Zeichensatzes 2
	(STACSSIN)	ladbarer Monochromzeichensatz
	(STACSTRI)	ladbarer Farbzeichensatz
	(STACSNO)	nicht ladbarer Zeichensatz
19	STACS2S	Zustand des Zeichensatzes 2
	(STACSNLO)	Zeichensatz kann geladen werden
	(STACSDSS)	Zeichensatz wurde von der Datensichtstation belegt
	(STACSDVN)	Zeichensatz wurde von der DVA geladen
	(STACSDVA)	Zeichensatz wurde von der DVA geladen und zugewiesen
20	STACS3T	Art des Zeichensatzes 3
	(STACSSIN)	ladbarer Monochromzeichensatz ladbarer Farbzeichensatz
	(STACSTRI)	nicht ladbarer Zeichensatz
	(STACSNO)	
21	STACS3S	Zustand des Zeichensatzes 3
	(STACSNLO)	Zeichensatz kann geladen werden
	(STACSDSS)	Zeichensatz wurde von der Datensichtstation belegt
	(STACSDVN)	Zeichensatz wurde von der DVA geladen
	(STACSDVA)	Zeichensatz wurde von der DVA geladen und zugewiesen
22	STACS4T	Art des Zeichensatzes 4
	(STACSSIN)	ladbarer Monochromzeichensatz
	(STACSTRI)	ladbarer Farbzeichensatz
	(STACSNO)	nicht ladbarer Zeichensatz

Byte	Symb. Name	Bedeutung
23	STACS4S	Zustand des Zeichensatzes 4
	(STACSNLO)	Zeichensatz kann geladen werden
	(STACSDSS)	Zeichensatz wurde von der Datensichtstation belegt
	(STACSDVN)	Zeichensatz wurde von der DVA geladen
	(STACSDVA)	Zeichensatz wurde von der DVA geladen und zugewiesen
24	STACS5T	Art des Zeichensatzes 5
	(STACSSIN)	ladbarer Monochromzeichensatz
	(STACSTRI)	ladbarer Farbzeichensatz
	(STACSNO)	nicht ladbarer Zeichensatz
25	STACS5S	Zustand des Zeichensatzes 5
	(STACSNLO)	Zeichensatz kann geladen werden
	(STACSDSS)	Zeichensatz wurde von der Datensichtstation belegt
	(STACSDSS)	Zeichensatz wurde von der Datensichtstation belegt
	(STACSDVN)	Zeichensatz wurde von der DVA geladen
	(STACSDVA)	Zeichensatz wurde von der DVA geladen und zugewiesen
26	STACS6T	Art des Zeichensatzes 6
	(STACSSIN)	ladbarer Monochromzeichensatz
	(STACSTRI)	ladbarer Farbzeichensatz
	(STACSNO)	nicht ladbarer Zeichensatz
27	STACS6S	Zustand des Zeichensatzes 6
	(STACSNLO)	Zeichensatz kann geladen werden
	(STACSDSS)	Zeichensatz wurde von der Datensichtstation belegt
	(STACSDVN)	Zeichensatz wurde von der DVA geladen
	(STACSDVA)	Zeichensatz wurde von der DVA geladen und zugewiesen
28	STACS7T	Art des Zeichensatzes 7
	(STACSSIN)	ladbarer Monochromzeichensatz
	(STACSTRI)	ladbarer Farbzeichensatz
	(STACSNO)	nicht ladbarer Zeichensatz
29	STACS7S	Zustand des Zeichensatzes 7
	(STACSNLO)	Zeichensatz kann geladen werden
	(STACSDSS)	Zeichensatz wurde von der Datensichtstation belegt
	(STACSDVN)	Zeichensatz wurde von der DVA geladen
	(STACSDVA)	Zeichensatz wurde von der DVA geladen und zugewiesen

Byte 14-29 werden bei zu kleiner Länge des Statusbereichs weggelassen, ohne einen Return-Code zu liefern. Die Information über die Zeichensätze wird nur geliefert, wenn die Zeichensätze über Byte 12-13 ansprechbar sind.

- PERPH: angeschlossene Peripherie (Bereichslänge: 8 Byte)

Informationen werden über die Statusmeldung der Datenstation geliefert, wenn diese vorliegt, ansonsten über die Generierung.

Byte	Symb. Name	Bedeutung
0	STAPERPR	Status von der Station
	(STAPERY) (STAPERN)	Statusmeldung der Datenstation vorhanden keine Statusmeldung der Datenstation vorhanden
1-2	-	reserviert
3	STALOCHC	lokales Hardcopygerät
	(STALHCY) (STALHCN)	lokales Hardcopygerät angeschlossen kein lokales Hardcopygerät angeschlossen
4-5	-	reserviert
6	STAIDCAR	Ausweisleser
	(STAIDCY) (STAIDCN)	Ausweisleser angeschlossen kein Ausweisleser angeschlossen
7	STACKT	Chipkartenterminal
	(STACKTY) (STACKTN)	Chipkartenterminal angeschlossen kein Chipkartenterminal angeschlossen

- BASIC: Grundinformationen der Datenstation (Bereichslänge: 24 Byte)

Informationen werden über die Statusmeldung der Datenstation geliefert, wenn diese vorliegt, ansonsten werden sie aus der Generierung geliefert bzw. VTSU-B Standardwerte genommen.

Wenn Sie eine Länge größer als 24 Byte und kleiner als 33 Byte festlegen, werden nur Informationen zu Byte 0-23 geliefert.
Wenn Sie eine Länge gleich 33 Byte festlegen, werden nur Informationen zu Byte 0-32 geliefert. Ist XHCS nicht geladen oder unterstützt die Datenstation keinen erweiterten Zeichensatz, werden keine Informationen zum Datenstationstyp und zum erweiterten Standardnamen geliefert.
Wenn Sie eine Länge größer 33 Byte und kleiner 52 Byte festlegen und die Datensichtstation im 8-bit-Modus arbeitet, wird das erste Byte dieses Bereichs auf X'00' gesetzt. Die zu empfangende Information wird abgeschnitten, da der angegebene Bereich zu klein ist.

Bei einer angegebene Länge von 52 Byte werden Informationen zu Byte 0-51 geliefert. Nicht benötigte Byte werden auf X'00' gesetzt. Für Drucker werden keine Informationen bereitgestellt.
Bei einer angegebene Länge von 60 Byte werden Informationen zu Byte 0-59 geliefert. Nicht benötigte Byte werden auf X'00' gesetzt.

Bei einer angegebene Länge von 64 Byte, ist die zurückgelieferte Information vollständig (Byte 0-63). Nicht benötigte Byte werden auf X'00' gesetzt.

Byte	Symb. Name	Bedeutung
0	STAINFO	Status von der Station
	(STAINFOY) (STAINFON)	Statusmeldung von der Station liegt vor keine Statusmeldung von der Station
1	STAINFP	Status von der Station
	(STAINFPY) (STAINFPN)	Statusmeldung von der Station ist möglich es wird keine Statusmeldung von der Station erwartet
2-9	STAPTNAM	Abdruckbarer Datenstationstyp
	TYP00	unbekannter Datenstationstyp
	DSS-X.29	Datensichtgerät X.29
	RECHNER	RECHNER
	SS-8102	Drucker 8102
	DSS-8151	Datensichtstation 8151
	DSS-8152	Datensichtstation 8152
	SS-8110	Fernschreiber 8110
	SS-8121	Drucker 8121
	FS100	Fernschreiber T100
	FS100-E	Fernschreiber FS100-E
	DRS90037	Drucker 90037
	DRS-8122	Drucker 8122
	DSS-8162	Datensichtstation 8162
	DSS-8160	Datensichtstation 8160
	DRS-8124	Drucker 8124
	AP	Anwendungsprogramm
	SST-X.29	Fernschreiber X.29
	DSS-9750	Datensichtstation 9750 oder 9749
	DRS-9003	Drucker 9003
	DSS-9770	Datensichtstation 9770
	DRS-9002	Drucker 9002
	DSS-3974	Datensichtstation 3974
	DSS-9751	Datensichtstation 9751
	DSS-9752	Datensichtstation 9752
	DSS-9753	Datensichtstation 9753
	DRS-9001	Drucker 9001
	DSS-9731	Datensichtstation 9731
	DSS9770R	Datensichtstation 9770R
	DRS-9004	Drucker 9004
	DSS-9754	Datensichtstation 9754
	DSS-9755	Datensichtstation 9755
	DSS-9763	Datensichtstation 9763
	DRS-9012	Drucker 9012
	DRS-9013	Drucker 9013

Byte	Symb. Name	Bedeutung
2-9 (Forts.)	STAPTNAM	Abdruckbarer Datenstationstyp
	DSS-3270	Datensichtstation 3270
	DRS-0131	Drucker 9001-31
	DRS-0189	Drucker 9001-8931
	DRS-9022	Drucker 9022
	DRS-1118	Drucker 9011-18
	DRS-1119	Drucker 9011-19
	DRS-3287	Drucker 3287
	TCP-IP	TCP-IP-Anwendung
	DRS-9021	Drucker 9021
	DRS-9014	Drucker 9014
	DRS-9026	Drucker 9026 (HDLC, kompatibel 9025)
	DSS-FE	Front-End Datensichtstation (FHS-DOORS)
10	STAHCOPY	lokales Hardcopygerät
	(STABLHCY)	lokales Hardcopygerät angeschlossen
	(STABLHCN)	kein lokales Hardcopygerät angeschlossen
11	STAIDCR	Ausweisleser
	(STAIDCRY)	Ausweisleser angeschlossen
	(STAIDCRN)	kein Ausweisleser angeschlossen
12	STACOL	Anzahl der Farben an der Datenstation
	(STACOLNO)	keine Farben
	(STACOL4)	4 Farben
	(STACOL8)	8 Farben
13-15	-	reserviert
16-19	STALINES	Physikalische Zeilenzahl abdruckbar (dezimal) (aus der Generierung oder Standardwerte)
20-23	STACOLUM	Physikalische Zeichenzahl pro Zeile abdruckbar (dezimal) (aus der Generierung oder Standardwerte)
24	STATTYPE	Datenstationstyp
	(STATYPE7)	Datenstation kann nur im 7-bit-Modus arbeiten
	(STATYPE8)	Datenstation kann im 7- oder 8-bit-Modus arbeiten
25-32	STACURCH	erweiterter Standard-Name. Wenn die Datensichtstation 8-bit-Modus unterstützt, wird ein Wert geliefert.
33	STACCSNN	Anzahl der unterstützten 8-bit-Zeichensätze
	(STATRINF)	Bei X' 00' und gleichzeitig STATYPE8 überschreitet die Länge der auszugebenden Information die angegebene Länge. Die Information wird abgeschnitten.
34	STACSS1	1-ter unterstützter Zeichensatz Die Variantennummer wird hexadezimal angegeben.

Byte	Symb. Name	Bedeutung
35	STACSS2	2-ter unterstützter Zeichensatz Die Variantennummer wird hexadezimal angegeben.
36	STACSS3	3-ter unterstützter Zeichensatz Die Variantennummer wird hexadezimal angegeben.
37	STACSS4	4-ter unterstützter Zeichensatz Die Variantennummer wird hexadezimal angegeben.
38	STACSS5	5-ter unterstützter Zeichensatz Die Variantennummer wird hexadezimal angegeben.
39	STACSS6	6-ter unterstützter Zeichensatz Die Variantennummer wird hexadezimal angegeben.
40	STACSS7	7-ter unterstützter Zeichensatz Die Variantennummer wird hexadezimal angegeben.
41	STACSS8	8-ter unterstützter Zeichensatz Die Variantennummer wird hexadezimal angegeben.
42	STACSS9	9-ter unterstützter Zeichensatz Die Variantennummer wird hexadezimal angegeben.
43	STACSS10	10-ter unterstützter Zeichensatz Die Variantennummer wird hexadezimal angegeben.
44	STACSS11	11-ter unterstützter Zeichensatz Die Variantennummer wird hexadezimal angegeben.
45	STACSS12	12-ter unterstützter Zeichensatz Die Variantennummer wird hexadezimal angegeben.
46	STACSS13	13-ter unterstützter Zeichensatz Die Variantennummer wird hexadezimal angegeben.
47	STACSS14	14-ter unterstützter Zeichensatz Die Variantennummer wird hexadezimal angegeben.
48	STACSS15	15-ter unterstützter Zeichensatz Die Variantennummer wird hexadezimal angegeben.
49	STACSS16	16-ter unterstützter Zeichensatz Die Variantennummer wird hexadezimal angegeben.
50-51	-	reserviert
52-59	STAACTCH	Name des aktivierten erweiterten Zeichensatzes. Name wird nur geliefert, wenn die Datensichtstation den 8-Bit-Modus unterstützt.
60	STARMODE (STARMODM) (STARMODU)	Physikalischer Lese-Modus Es werden nur die modifizierten Felder gelesen. Es werden alle ungeschützten Felder gelesen.

Byte	Symb. Name	Bedeutung
61	STALLECH	logisches Zeilenende-Zeichen für Datensichtstationen ohne die äquivalenten Hardware-Funktionen.
62	STASUBCH	Ersatzzeichen für Zeichen, die kleiner X' 40' und keine logischen Steuerzeichen sind.
63	STAPERHC (STAPERHY) (STAPERHN)	Permanentes Hardcopy Alle Ausgabenachrichten für eine Datensichtstation werden gleichzeitig über ein angeschlossenes Hardcopy-Gerät ausgedruckt. Ausgabenachrichten werden nicht zusätzlich als Hardcopy protokolliert.

Werden bei STACCSn (Byte 34-49) weniger als 16 unterschiedliche Varianten unterstützt, werden die verbleibenden Variantennummern (16-n) auf X'00' gesetzt.

Beispiele siehe Makro **TSTAT**.

DELFEI – SOLSIG- oder POSSIG-Eintrag löschen

Allgemeines

Anwendungsgebiet: (optimierte) Ereignissteuerung; siehe Seite 95
Makrotyp: R-Typ; siehe Seite 28

Forward Eventing (FEV) ist eine optimierte Form der sychronen Ereignissteuerung (synchrones Eventing). FEV vermeidet für wiederholte **POSSIG**- bzw. **SOLSIG**-Aufrufe in einem Programm die wiederholte Validation der angegebenen Operanden.

Makrobeschreibung

Der Makro **DELFEI** nimmt Bezug auf einen mit **DSOFEI** oder **DPOFEI** erzeugten Eintrag in der EVENTLST und löscht diesen in der EVENTLST. Ein **DISEI**-Aufruf löscht ebenfalls die zugehörigen Forward-Events (impliziter **DELFEI**).

Makroaufrufformat und Operandenbeschreibung

DELFEI
REFNUM=(r)

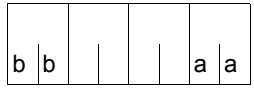
REFNUM=
bezeichnet ein Register, das (direkt) die Referenznummer für den POSSIG-Eintrag enthält.

(r)
r = Register mit der Referenznummer.

Rückinformation und Fehleranzeigen

Während der Makrobearbeitung enthält Register R1 die Referenznummer; Register R0 wird mit einem internen Funktionscode überschrieben..

R15:



Über die Ausführung des Makros DELFEI wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'bb'	X'aa'	Erläuterung
X'00'	X'00'	Funktion ausgeführt: Der Eintrag in der EVENTLST wurde gelöscht.
X'04'	X'04'	Keine Aktion: Falsche Referenznummer oder Eintrag bereits gelöscht.

DEQAR – Belegung einer Serialisierungskennung aufheben

Allgemeines

Anwendungsgebiet: (Task-)Serialisierung; siehe Seite 92
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

DEQAR generiert je nach Angabe die 24-Bit- oder die 31-Bit-Schnittstelle. Bei Verkettung der Makroaufrufe muss in jedem Aufruf der Operand PARMOD mit demselben Operandenwert angegeben werden.

Makrobeschreibung

Der Makro **DEQAR** beendet die Belegung der angegebenen Serialisierungskennung (angefordert durch **ENQAR**) durch die Task des Aufrufers. Befindet sich in der Warteschlange für diese Serialisierungskennung eine andere Task, so wird diese aktiviert und belegt nun die Serialisierungskennung.
Als wahlweise Funktion kann mit diesem Makroaufruf auch die Zuordnung der Serialisierungskennung (**ENASI**) zu der Task des Aufrufers aufgehoben werden.
Durch den CONTINU-Operanden können bis zu 255 **DEQAR**-Aufrufe gekettet werden.

Makroaufrufformat und Operandenbeschreibung

```
DEQAR

{ { SINAME=name                          }                                  }
{ { SINAMAD=adr / (r) [,SINAMLN=länge]   } ,SCOPE=LOCAL / GROUP / USER_GROUP / GLOBAL }
{ SIID=adr / (r)                                                            }

,DISSI=NO / YES
,HOLDER=SELF / ANY
,CONTINU=NO / YES
[,PARMOD=24 / 31]
[,MF=L / (E, ..)]
```

SINAME=name
gibt den Namen der Serialisierungskennung an. Zur eindeutigen Bezeichnung der Serialisierungskennung muss zusätzlich „SCOPE“ angegeben werden.

SINAMAD=
bezeichnet den Namen der Serialisierungskennung. Die Kennung ist nur durch die zusätzliche Angabe von „SCOPE“ eindeutig bezeichnet.

adr
symbolische Adresse des Feldes, das den Namen enthält

(r)
r = Register, das die Adresse enthält

SINAMLN=
gibt die Länge des Namens der Serialisierungskennung in Byte an. Die Länge muss mindestens 1 Byte sein und darf 54 Byte nicht überschreiten.
Fehlt der Operand, so wird das Längenattribut des SINAMAD-Operanden angenommen, wenn SINAMAD=adr angegeben ist;
bei SINAMAD=(r) wird die maximale Länge (54) angenommen.

länge
Länge des Namens der Serialisierungskennung.

SCOPE=
beschreibt den Geltungsbereich (Teilnehmerkreis) für die Serialisierungskennung.

LOCAL
Die Serialisierungskennung wird nur von der Task des Aufrufers benutzt.

GROUP
Teilnehmer sind alle Tasks unter der Benutzerkennung des Aufrufers.

USER_GROUP
Teilnehmer können alle Tasks sein, deren Benutzerkennungen der gleichen Benutzergruppe angehören wie die Benutzerkennung des einrichtenden Teilnehmers.
Der Operandenwert setzt die Existenz von Benutzergruppen voraus und kann daher nur angegeben werden, wenn die Funktionseinheit SRPM des Software-Produkts SECOS im System vorhanden ist.

Vor einem Makroaufruf mit SCOPE=USER_GROUP muss deshalb mit dem Makro GETUGR (siehe Handbuch „SECOS“ [14]) geprüft werden, ob SRPM zur Verfügung steht; abhängig vom Ergebnis (Returncode) ist im Programm zu reagieren.

GLOBAL
Teilnehmer sind alle Tasks im System.

SIID=
bezeichnet die Kurzkennung der Serialisierungskennung. Diese Kurzkennung wird dem Benutzer durch den **ENASI**-Makroaufruf zur Verfügung gestellt. Die Verwendung der Kurzkennung an Stelle des Namens zur Identifizierung einer Serialisierungskennung beschleunigt die Verarbeitung. Die Serialsierungskennung ist durch die Kurzkennung eindeutig bezeichnet.

adr
symbolische Adresse eines 4 Byte langen Feldes, das die Kurzkennung enthält

(r)
r = Register, das die Adresse enthält

DISSI=
gibt an, ob die Zuordnung der angegebenen Serialisierungskennung durch die Task des Aufrufers aufgehoben werden soll (siehe Makro **DISSI**).

NO
Die Zuordnung wird nicht aufgehoben.

YES
Die Zuordnung wird aufgehoben.

HOLDER=
gibt an, ob die Beendigung des Zugriffs durchgeführt werden soll, wenn die Task des Aufrufers den derzeit wirksamen Zugriff angefordert hat.

SELF
Der Zugriff wird beendet, wenn die Task des Aufrufers auch den derzeit wirksamen Zugriff angefordert hat.

ANY
Bei Angabe von HOLDER=ANY wird der Zugriff beendet, unabhängig davon, von welcher Task der Zugriff angefordert wurde.

CONTINU=
Durch diesen Operanden kann eine Kettung von bis zu 255 **DEQAR**-Aufrufen erfolgen.

NO
Dies ist der letzte (bzw. einzige) Aufruf einer Folge.

YES
YES bedeutet, dass unmittelbar nach diesem **DEQAR**-Aufruf ein weiterer folgt.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB).

Hinweise zum Makroaufruf

– Bei der Verwendung der Listenform des Makroaufrufes (Operand MF=L) ist zu beachten:
Zur Ausführung muss nur ein Makroaufruf mit MF=E gegeben werden, unabhängig davon, ob dieser Aufruf für eine einzelne Anforderung oder für eine Folge von Anforderungen gilt.

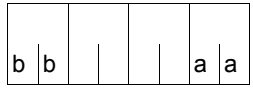
– Der Datenbereich bei einer Folge von Anforderungen wird durch Kettung der Makroaufrufe (MF=L) durch den CONTINU-Operand erzeugt.

– Bei Programmbeendigung werden alle Zugriffsanforderungen dieses Programms beendet.

Rückinformation und Fehleranzeigen

Während der Makrobearbeitung enthält Register R1 die Adresse der Operandenliste.

R15:



Über die Ausführung des Makros DEQAR wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'bb'	X'aa'	Erläuterung
X'00'	X' 00'	Alle Dequeue-Aufrufe wurden ausgeführt. Keine zusätzlichen Aktionen wurden ausgeführt.
X'04'	X'00'	Alle Dequeue-Aufrufe wurden ausgeführt. Mindestens eine Serialisierungskennung wurde gelöscht.
X'08'	X'00'	Alle Dequeue-Aufrufe wurden ausgeführt. Mindestens ein Disable (siehe DISSI-Makroaufruf) wurde ausgeführt.
X'0C'	X'04'	Nicht alle Dequeue-Aufrufe wurden ausgeführt. Es wurde mindestens ein 'Dequeue' nicht ausgeführt, weil entsprechende Serialisierungskennungen nicht der Task des Aufrufers zugeordnet waren.
X'10'	X'04'	Nicht alle Dequeue-Aufrufe wurden ausgeführt. Es wurden ungültige Operanden angegeben: – ungültige Adresse; z.B. Adresse innerhalb einer DSECT – ungültige Länge – ungültiger Name – nichtdefinierter Geltungsbereich, CONTINU-, DISSI- oder HOLDER-Wert.
X'14'	X'04'	Nicht alle Dequeue-Aufrufe wurden ausgeführt. Es wurde eine ungültige Kennung angegeben (die Kennung ist dem System nicht bekannt, bzw. der Aufrufer hat keinen Enable-Aufruf gegeben, siehe ENASI-Makroaufruf).
X'20'	X'04'	Nicht alle Dequeue-Aufrufe wurden ausgeführt. Für mindestens eine der Serialisierungskennungen lag keine Zugriffsanforderung mehr vor oder für mindestens eine der Serialisierungskennungen sollte der Zugriff mit HOLDER=SELF beendet werden und die Zugriffsanforderung wurde von einer anderen Task gestellt. Werden mehrere DEQAR-Aufrufe gekettet, so werden die Dequeue-Operationen für diejenigen Serialisierungskennungen ausgeführt, für die sie erlaubt sind. Der Zustand der Serialisierungskennungen, für die der DEQAR-Aufruf ungültig war, bleibt unverändert. Wurde der Operand DISSI=YES angegeben, wird ein Disable ausgeführt.

DISCO – Contingency-Definition schließen

Allgemeines

Anwendungsgebiet: Contingency-Verfahren; siehe Seite 111
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

Makrobeschreibung

Mit dem Makro **DISCO** wird einer Routine die Möglichkeit zum Ablauf als Contingency-Prozess entzogen. Der Contingency-Prozess ist nicht mehr definiert und kann in nachfolgenden **POSSIG**- oder **SOLSIG**-Makroaufrufen nicht mehr angegeben werden.

Noch folgende **SOLSIG**- und **POSSIG**-Makroaufrufe, die sich auf eine gelöschte Contingency-Prozess-Definition beziehen, werden zurückgewiesen (Rücksprunginformation). Bereits in den Warteschlangen sich befindende **SOLSIG**- und **POSSIG**-Anforderungen (also bereits zuvor gegebene Makroaufrufe) sind von der Löschung der Contingency-Definition nicht berührt, d.h. bei Eintreten der Bedingungen wird der entsprechende Contingency-Prozess noch aktiviert.

Makroaufrufformat und Operandenbeschreibung

```
DISCO

{ { CONAME=name                                  }
{ { CONAMAD=adr / (r) [,CONAMLN=länge]           } }
{ COID=adr / (r)                                 }

[,PARMOD=24 / 31]
[,MF=L / (E,..)]
```

CONAME=name
gibt den Namen des Contingency-Prozesses an.

CONAMAD=
bezeichnet den Namen des Contingency-Prozesses.

adr
symbolische Adresse des Feldes, das den Namen enthält

(r)
r = Register, das die Adresse enthält

CONAMLN=
gibt die Länge des Namens des Contingency-Prozesses in Byte an.

Die Länge muss mindestens 1 Byte sein und darf 54 Byte nicht überschreiten.
Fehlt der Operand, so wird das Längenattribut des CONAMAD-Operanden angenommen, wenn CONAMAD=adr angegeben ist; bei CONAMAD=(r) wird die maximale Länge (54) angenommen.

länge
Länge des Namens des Contingency-Prozesses.

COID=
bezeichnet die Kurzkennung des Contingency-Prozesses. Diese Kurzkennung wird dem Benutzer durch den **ENACO**-Makroaufruf zur Verfügung gestellt.

adr
die symbolische Adresse eines 4 Byte langen Feldes, das die Kurzkennung enthält

(r)
r = Register, das die Adresse enthält

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

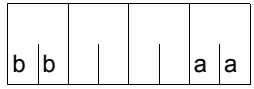
24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB).

Rückinformation und Fehleranzeigen

Register R1 enthält die Adresse des Datenbereichs.

R15:



Über die Ausführung des Makros DISCO wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'bb'	X'aa'	Erläuterung
X'04'	X'00'	Die Contingency-Prozess-Definition wurde gelöscht.
X'10'	X'04'	Es wurden ungültige Operanden angegeben. Keine Aktion.
X'14'	X'04'	Ungültiger Name bzw. ungültige Kurzkennung. Es existiert kein Contingency-Prozess mit spezifizierter Identifikation. Keine Aktion.

Beispiele enthalten der Abschnitt „Contingency-Prozesse“ (Seite 111) und die Beschreibung des Makros **POSSIG** (Seite 754).

DISEI – Ereignisgesteuerte Verarbeitung beenden

Allgemeines

Anwendungsgebiet: Ereignissteuerung; siehe Seite 95
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

Makrobeschreibung

Der Makroaufruf **DISEI** wird benutzt, um die Verwendung der angegebenen Ereigniskennung durch die Task des Aufrufers zu beenden. Die Ereigniskennung wird gelöscht, wenn sie von keiner weiteren Task verwendet wird.

Makroaufrufformat und Operandenbeschreibung

```
DISEI

{ { EINAME=name                                                    }
{ { EINAMAD=adr / (r) [,EINAMLN=länge] },SCOPE=LOCAL / GROUP / USER_GROUP / GLOBAL }
{ EIID=adr / (r)                                                   }

[,PARMOD=24 / 31]
[,MF=L / (E,..)]
```

EINAME=name
gibt den Namen der Ereigniskennung an, die der aufrufende Prozess nicht mehr verwenden wird. Die Bezeichnung der Ereigniskennung ist nur mit der zusätzlichen Angabe des Operanden SCOPE eindeutig.

EINAMAD=
bezeichnet die Ereigniskennung. Zur eindeutigen Bezeichnung der Ereigniskennung muss zusätzlich der Operand SCOPE angegeben werden.

adr
symbolische Adresse des Feldes, das den Namen der Ereigniskennung enthält

(r)
r = Register, das die Adresse enthält

EINAMLN=
gibt die Länge des Namens der Ereigniskennung in Byte an.
Die Länge muss mindestens 1 Byte sein und darf 54 Byte nicht überschreiten.
Fehlt der Operand, so wird das Längenattribut des EINAMAD-Operanden angenommen, wenn EINAMAD=adr angegeben ist;
bei EINAMAD=(r) wird die maximale Länge (54) angenommen.

länge
Länge des Namens der Ereigniskennung.

SCOPE=
beschreibt den Geltungsbereich (Teilnehmerkreis) für die Ereigniskennung.

LOCAL
Die Ereigniskennung wird nur von der Task des Aufrufers benutzt.

GROUP
Teilnehmer sind alle Tasks unter der Benutzerkennung des Aufrufers.

USER_GROUP
Teilnehmer können alle Tasks sein, deren Benutzerkennungen der gleichen Benutzergruppe angehören wie die Benutzerkennung des einrichtenden Teilnehmers.

Der Operandenwert setzt die Existenz von Benutzergruppen voraus und kann daher nur angegeben werden, wenn die Funktionseinheit SRPM des Software-Produkts SECOS im System vorhanden ist. Vor einem Makroaufruf mit SCOPE=USER_GROUP muss deshalb mit dem Makro GETUGR (siehe Handbuch „SECOS“ [14]) geprüft werden, ob SRPM zur Verfügung steht; abhängig vom Ergebnis (Returncode) ist im Programm zu reagieren.

GLOBAL
Teilnehmer sind alle Tasks im System

EIID=
bezeichnet die Kurzkennung der Ereigniskennung. Diese Kurzkennung wird dem Benutzer durch den **ENAEI**-Makroaufruf zur Verfügung gestellt. Die Verwendung der Kurzkennung an Stelle des Namens zur Identifizierung der Ereigniskennung beschleunigt die Verarbeitung. Diese Angabe ist eindeutig.

adr
symbolische Adresse des Feldes, das die Kurzkennung enthält

(r)
r = Register, das die Adresse des Feldes enthält

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB).

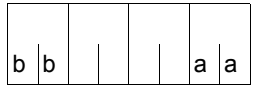
Funktionsweise

Stehen bei Aufruf des Makros **DISEI** noch **SOLSIG**- bzw. **POSSIG**-Anforderungen des aufrufenden Programms in den Warteschlangen der Ereigniskennung (siehe Abschnitt „Ereignisgesteuerte Verarbeitung (Eventing)“ auf Seite 95), so werden sie gelöscht; ebenso werden die zugeord-neten Forward-Events aus der EVENTLST gelöscht (impliziter **DELFEI**). Bei **SOLSIG**-Anforderungen (synchron/asynchron) wird das aufrufende Programm über den Returncode in Register R15 bzw. über den Ereignis-Informationscode davon benachrichtigt. Dasselbe gilt für **POSSIG**-Anforderungen, zu denen ein Contingency-Prozess angegeben war.

Rückinformation und Fehleranzeigen

Register R1 enthält die Adresse des Datenbereichs.

R15:



Über die Ausführung des Makros DISEI wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'bb'	**X'aa'**	**Erläuterung**
X'04'	X'00'	Funktion ausgeführt: Die Ereigniskennung wurde gelöscht.
X'08'	X'00'	Funktion ausgeführt: Die Verwendung der Ereigniskennung wurde für die Task des Aufrufers beendet. Die Ereigniskennung wurde nicht gelöscht.
X'0C'	X'04'	Keine Aktion: Die vom System erstellte Ereigniskennung ist der Task des Aufrufers nicht zugeordnet.
X'10'	X'04'	Keine Aktion: Es wurden ungültige Operanden angegeben.
X'14'	X'04'	Keine Aktion: Ungültiger Name bzw. ungültige Kurzkennung. Es existiert keine Ereigniskennung mit spezifizierter Identifikation.
X'18'	X'04'	Keine Aktion: Die zu löschende Ereigniskennung wird noch von FASTPAM benutzt

Beispiele enthalten der Abschnitt „Ereignisgesteuerte Verarbeitung (Eventing)“ (Seite 107) und der Abschnitt „Contingency-Prozesse“ (Seite 119).

DISMP – Memory Pool schließen

Allgemeines

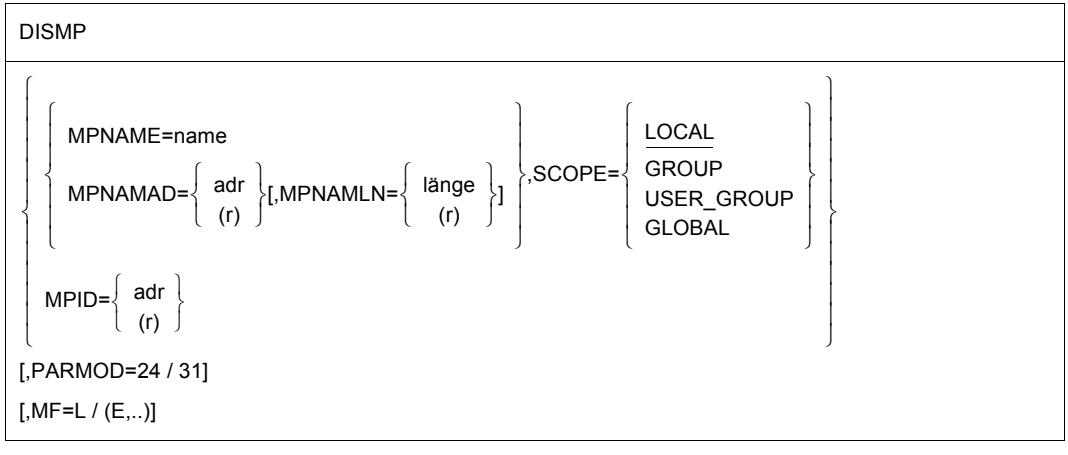
Anwendungsgebiet: Memory Pool Technik; siehe Seite 55
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

Ein Memory Pool ist ein Speicherbereich im Klasse-6-Speicher, der von mehreren Anwendern gemeinsam benutzt werden kann. Ein Anwender kann mit **ENAMP** einen Memory Pool einrichten oder seine Teilnahme an einem bestehenden erklären. Mit **DISMP** kann ein Anwender explizit seine Teilnahme an einem Memory Pool beenden; implizit mit Programmbeendigung.
Ein Memory Pool wird über den Pool-Namen **oder** über seine Kurzkennung (siehe **ENAMP**) angesprochen.
Nach **DISMP** muss die erneute Teilnahme an demselben (noch bestehenden) Memory Pool wieder mit **ENAMP** erklärt werden; der Aufrufer erhält eine neue Kurzkennung für den Memory Pool.

Makrobeschreibung

Mit dem Makro **DISMP** kann ein Memory Pool-Teilnehmer die Verbindung zu dem Memory Pool lösen. Der Memory Pool wird aufgelöst, wenn der Aufrufer der letzte (einzige) Teilnehmer ist. In diesem Fall werden alle Seiten des Pools implizit freigegeben.

Makroaufrufformat und Operandenbeschreibung



MPNAME=
bezeichnet den Namen des Memory Pools (Verbindung mit Operand SCOPE beachten).

name
Name des Memory Pools

MPNAMAD=
beschreibt die Adresse des Feldes mit name.(Verbindung mit Operand SCOPE beachten).

adr
symbolische Adresse (Name) des Feldes

(r)
r = Register mit dem Adresswert des Feldes

MPNAMLN=
beschreibt die Länge des unter MPNAMAD angegebenen Namens. Wenn nicht spezifiziert: Längenattribut des Feldes adr bzw. 54 Byte, wenn MPNAMAD=(r) angegeben wurde.

länge
Länge in Byte

(r)
r = Register, das länge enthält

SCOPE=
beschreibt den Geltungsbereich (Teilnehmerkreis) des Memory Pools. Die Angabe dient der eindeutigen Identifizierung des Memory Pools und muss immer in Verbindung mit den Operanden MPNAME bzw. MPNAMAD spezifiziert werden.

LOCAL
Der Memory Pool wird nur von dem einrichtenden Teilnehmer benutzt.

GROUP
Teilnehmer können alle Tasks unter der Benutzerkennung des einrichtenden Teilnehmers sein.

USER_GROUP
Teilnehmer können alle Tasks sein, deren Benutzerkennungen der gleichen Benutzergruppe angehören wie die Benutzerkennung des einrichtenden Teilnehmers.
Der Operandenwert setzt die Existenz von Benutzergruppen voraus und kann daher nur angegeben werden, wenn die Funktionseinheit SRPM des Software-Produkts SECOS im System vorhanden ist. Vor einem Makroaufruf mit SCOPE=USER_GROUP muss deshalb mit dem Makro GETUGR (siehe Handbuch „SECOS“ [14]) geprüft werden, ob SRPM zur Verfügung steht; abhängig vom Ergebnis (Returncode) ist im Programm zu reagieren.

GLOBAL
Teilnehmer können alle im System laufenden Tasks sein.

MPID=
beschreibt die Adresse eines Feldes (Länge = 4 Byte) mit der Kurzkennung für den Memory Pool (siehe auch **ENAMP**). Die Kurzkennung identifiziert den Memory Pool eindeutig. Die Bezeichnung des Memory Pools mittels MPID beschleunigt die Verarbeitung.

adr
symbolische Adresse (Name) des Feldes mit der Kurzkennung

(r)
r = Register mit dem Adresswert des Feldes

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

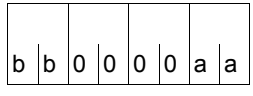
24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

Rückinformation und Fehleranzeigen

Nach der Makrobearbeitung enthält Register R1 die Adresse des Datenbereichs.

R15:



Über die Ausführung des Makros DISMP wird im Register R15 ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) übergeben.
aa=X'00' : normale Ausführung;
aa=X'04' : Funktion nicht ausgeführt.

X'bb'	X' aa'	Erläuterung
X'0C'	X'00'	Normale Ausführung. Aufrufer = letzter oder einziger Teilnehmer; Memory Pool aufgelöst.
X'10'	X'00'	Normale Ausführung. Teilnahme am Memory Pool beendet; Memory Pool bleibt bestehen.
X'04'	X'04'	Funktion nicht ausgeführt. Aufrufer ist kein Teilnehmer des genannten Memory Pools (kein ENAMP-Aufruf).
X'1C'	X'04'	Funktion nicht ausgeführt. Operandenfehler: – unzulässige Adresse des Datenbereichs – fehlerhafter Aufbau des Datenbereichs – unzulässige Adresse für MPNAMAD oder MPID im Datenbereich – Benennung des Memory Pools – Name enthält unzulässige Zeichen – ungültige Längenangabe (MPNAMLN) – Memory Pool nicht benannt (MPNAME, MPNAMAD, MPID nicht spezifiziert) – MPNAMLN angegeben, aber MPNAMAD nicht spezifiziert – SCOPE angegeben, aber MPNAME/MPNAMAD nicht spezifiziert. – Benennung nicht eindeutig. Es wurde mehr als nur ein Operand zur Benennung des MPs spezifiziert (MPNAME/MPNAMAD/MPID) – ungültige SCOPE-Angabe – SCOPE=USER_GROUP wurde angegeben, obwohl SRPM nicht im System vorhanden ist. – ungültiges Register (R1) angegeben. – PARMOD=24 in Verbindung mit 31-Bit-Adressierungsmodus (AMODE 31) angegeben.
X'24'	X'04'	Funktion nicht ausgeführt. Berechtigungsfehler: – Die (eigene) Task hält noch eine (Seitenfreigabe-) Sperre für den Memory Pool. Die Sperre wurde in einem privilegierten Zustand gesetzt. – Aufrufer ist nicht berechtigt, die Teilnahme an einem privilegierten oder Klasse-5-MP zu beenden. – Der Memory Pool wird noch durch FASTPAM belegt.

31-Bit-Schnittstelle:

– Bei fehlerhafter Ausrichtung oder Initialisierung des Standardheaders werden im Register R15 zusätzlich die Returncodes X'0001FFFF' / X'0003FFFF' / X'0004FFFF' übergeben, siehe Tabelle „Standard-Returncodes“ auf Seite 43.

– Im Standardheader werden keine Returncodes übergeben.

DISSI – Zuordnung zu einer Serialisierungskennung lösen

Allgemeines

Anwendungsgebiet: (Task-)Serialisierung; siehe Seite 92
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

DISSI generiert je nach Angabe die 24-Bit- oder die 31-Bit-Schnittstelle. Bei Makroverkettung müssen alle Makros der Kette dieselbe Schnittstelle (24-Bit oder 31-Bit-Schnittstelle) benutzen.

Makrobeschreibung

Dieser Makroaufruf wird benutzt, um die Verwendung der angegebenen Serialisierungskennung durch die Task des Aufrufers zu beenden. Verwendet keine weitere Task diese Serialisierungskennung, so wird sie gelöscht.
Vor dem **DISSI**-Makroaufruf muss die Belegung der Serialisierungskennung wieder aufgehoben sein (Makro **DEQAR**).
Mit dem CONTINU-Operanden können bis zu 255 **DISSI**-Aufrufe gekettet werden.

Makroaufrufformat und Operandenbeschreibung

```
DISSI

{ { SINAME=name                                  }                                                  }
{ { SINAMAD=adr / (r) [,SINAMLN=länge]           },SCOPE=LOCAL / GROUP / USER_GROUP / GLOBAL        }
{ SIID=adr / (r)                                                                                    }

,CONTINU=NO / YES
[,PARMOD=24 / 31]
[,MF=L / (E,..)]
```

SINAME=name
gibt den Namen der Serialisierungskennung an. Diese Angabe ist nur zusammen mit dem Operanden SCOPE eindeutig.

SINAMAD=
bezeichnet den Namen der Serialisierungskennung. Diese Angabe ist nur zusammen mit dem Operanden SCOPE eindeutig.

adr
symbolische Adresse des Feldes, das den Namen enthält

(r)
r = Register, das die Adresse enthält.

SINAMLN=
gibt die Länge des Namens der Serialisierungskennung in Byte an. Die Länge muss mindestens 1 Byte sein und darf 54 Byte nicht überschreiten.
Fehlt der Operand, so wird das Längenattribut des SINAMAD-Operanden angenommen, wenn SINAMAD=adr angegeben ist; bei SINAMAD=(r) wird die maximale Länge (54) angenommen.

länge
Länge des Namens der Serialisierungskennung.

SCOPE=
beschreibt den Geltungsbereich (Teilnehmerkreis) für die Serialisierungskennung.

LOCAL
Die Serialisierungskennung wird nur von der Task des Aufrufers benutzt.

GROUP
Teilnehmer sind alle Tasks unter der Benutzerkennung des Aufrufers.

USER_GROUP
Teilnehmer können alle Tasks sein, deren Benutzerkennungen der gleichen Benutzergruppe angehören wie die Benutzerkennung des einrichtenden Teilnehmers.

Der Operandenwert setzt die Existenz von Benutzergruppen voraus und kann daher nur angegeben werden, wenn die Funktionseinheit SRPM des Software-Produkts SECOS im System vorhanden ist. Vor einem Makroaufruf mit SCOPE=USER_GROUP muss deshalb mit dem Makro GETUGR (siehe Handbuch „SECOS“ [14]) geprüft werden, ob SRPM zur Verfügung steht; abhängig vom Ergebnis (Returncode) ist im Programm zu reagieren.

GLOBAL
Teilnehmer sind alle Tasks im System

SIID=
bezeichnet die Kurzkennung der Serialisierungskennung. Diese Kurzkennung wird dem Benutzer durch den **ENASI**-Makroaufruf zur Verfügung gestellt. Die Verwendung der Kurzkennung, an Stelle des Namens zur Identifizierung einer Serialisierungskennung beschleunigt die Verarbeitung. Diese Angabe ist eindeutig.

adr
symbolische Adresse eines 4 Byte langen Feldes, das die Kurzkennung enthält

(r)
r = Register, das die Adresse enthält

CONTINU=
Durch diesen Operanden kann eine Kettung von bis zu 255 **DISSI**-Aufrufen erfolgen.

NO
Dieser Aufruf ist der letzte (bzw. einzige) Aufruf einer Folge.

YES
unmittelbar nach diesem **DISSI**-Aufruf erfolgt ein weiterer.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB).

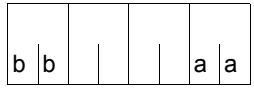
Hinweise zum Makroaufruf

- Die Disable-Funktion kann auch als Teil eines **DEQAR**-Makroaufrufes angefordert werden (siehe **DEQAR**-Makroaufruf).
- Bei Programmbeendigung wird für alle Serialisierungskennungen, die von der Task noch verwendet werden, vom System ein Disable ausgeführt.
- Bei der Verwendung der Listenform des Makroaufrufes (Operand MF=L) ist zu beachten:
  Zur Ausführung muss nur ein Makroaufruf mit MF=E gegeben werden, unabhängig davon, ob dieser Aufruf für eine einzelne Anforderung oder für eine Folge von Anforderungen gilt. Der Datenbereich bei einer Folge von Anforderungen wird durch Kettung der Makroaufrufe (MF=L) durch den CONTINU-Operanden erzeugt.

Rückinformation und Fehleranzeigen

Register R1 enthält die Adresse des Datenbereichs.

R15:



Über die Ausführung des Makros DISSI wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'bb'	X'aa'	Erläuterung
X'04'	X'00'	Alle Disable-Aufrufe wurden ausgeführt: Mindestens eine Serialisierungskennung wurde gelöscht.
X'08'	X'00'	Alle Disable-Aufrufe wurden ausgeführt: Die Verwendung aller angegebenen Serialisierungskennungen wurde beendet.
X'0C'	X'04'	Nicht alle Disable-Aufrufe wurden ausgeführt: Mindestens eine Serialisierungskennung wurde nicht von der Task des Aufrufers verwendet.
X'10'	X'04'	Nicht alle Disable-Aufrufe wurden ausgeführt: Es wurden ungültige Operanden angegeben: – ungültige Adresse; z.B. Adresse innerhalb einer DSECT – ungültige Länge – ungültiger Name – nichtdefinierter Geltungsbereich oder CONTINU-Wert
X'14'	X'04'	Nicht alle Disable-Aufrufe wurden ausgeführt: Es wurde eine ungültige Kennung angegeben (die Kennung ist dem System nicht bekannt, bzw. der Aufrufer hat keinen zugehörigen, gültigen ENABLE-Aufruf gegeben; siehe ENASI-Makroaufruf).
X'24'	X'04'	Nicht alle Disable-Aufrufe wurden ausgeführt: Die Disable-Anforderung wurde für eine Serialisierungskennung gegeben, für die eine Enqueue-Anforderung erfüllt, jedoch noch keine zugehörige Dequeue-Aktion ausgeführt wurde.

DJINF – Datenliste oder DSECT für Makro JINF erstellen

Allgemeines

Anwendungsgebiet: Abfragen und Zugriff zu Listen und Tabellen; siehe Seite 158
Makrotyp: Definitionsmakro; siehe Seite 28

Makrobeschreibung

Der Makro **DJINF** generiert eine Beschreibung des Ausgabebereichs für den Makro **JINF**. Die Beschreibung wird als DSECT oder als Datenabschnitt angelegt. Die DSECT/Datenliste wird für die 24-Bit-Schnittstelle durch Angabe der Operanden JATTR/RUNTIME erzeugt; für die 31-Bit-Schnittstelle durch Angabe des Operanden PARLIST. Im letzteren Fall beginnt die Beschreibung des Ausgabebereichs mit dem Standardheader. Die Initialisierungswerte sind in die Datenliste eingetragen.

Makroaufrufformat und Operandenbeschreibung

```
DJINF
DSECT=YES / NO
[,PREFIX=p]
{ ,JATTR=NO / YES ,RUNTIME=NO / YES }
{ ,PARLIST=NO / YES                 }
```

DSECT=
gibt an, ob eine DSECT zu dem Ausgabebereich oder ein Datenabschnitt (Datenliste) als Ausgabebereich generiert wird.

YES
Eine DSECT wird generiert.

NO
Eine Datenliste wird generiert.

PREFIX=
bezeichnet eine Zeichenfolge, mit der die symbolischen Namen der DSECT/Datenliste beginnen.

p
Präfix für die symbolischen Namen; Länge ≤ 2 Zeichen. Voreinstellung: p = JI.

JATTR=
gibt an, ob die DSECT/Datenliste für die Jobdaten generiert wird.

NO
Eine DSECT/Datenliste wird nicht generiert.

YES
eine DSECT/Datenliste wird generiert.

RUNTIME=
gibt an, ob die DSECT/Datenliste für aktuelle Jobdaten (Job-Startzeit, Anzahl Job-Wiederholungen) generiert wird.

NO
Eine DSECT/Datenliste wird nicht generiert.

YES
Eine DSECT/Datenliste wird generiert.

PARLIST=
gibt an, ob eine DSECT/Datenliste für die 31-Bit-Schnittstelle generiert wird.

NO
Eine DSECT/Datenliste wird nicht generiert.

YES
eine DSECT/Datenliste wird generiert.

Layout der DSECT für die 31-Bit-Schnittstelle

```
          DJINF DSECT=YES,PARLIST=YES
1 *--------P A R A M E T E R L I S T
1          #INTF REFTYPE=REQUEST,                                        C
1                INTNAME=JINF,                                         C
1                INTCOMP=002
1 JIJOBDPL  DSECT
1          FHDR  UNIT=43,FUNCT=2,VERS=1
2          DS    0A
2          DS    0XL8               GENERAL OPERAND LIST HEADER
2          DC    AL2(43)            FUNCTION UNIT NUMBER
2          DC    AL1(2)             FUNCTION NUMBER
2          DC    AL1(1)             FUNCTION INTERFACE VERSION NUMBER
2          DC    X'FFFFFFFF'         Returncode is virgin
1 JIJOUID   DS    CL8                    USER ID
1 JIJOACC   DS    CL8                    ACCOUNT NUMBER
1 JIJOJCLA  DS    CL8                    JOB CLASS
1 JIJOJNAM  DS    CL8                    JOB NAME
1 JIJOTSN   DS    CL4                    TASK SEQUENCE NUMBER
1 JIJOJPRI  DS    X                      JOB PRIORITY
```

```
1 JIJORP    DS    X                   REPEAT-TYPE
1 JIJORPNO  EQU   X'01'                - NO
1 JIJORPAS  EQU   X'02'                - AT-STREAM-STARTUP
1 JIJORPDL  EQU   X'03'                - DAILY
1 JIJORPWK  EQU   X'04'                - WEEKLY
1 JIJORPPD  EQU   X'05'                - PERIOD
1 JIJORPIT  DS    H                   REPEAT-INTERVAL
1 JIJOST    DS    X                   START-TYPE
1 JIJOSTSO  EQU   X'01'                - SOON
1 JIJOSTEA  EQU   X'02'                - EARLIEST
1 JIJOSTAT  EQU   X'03'                - AT
1 JIJOSTLA  EQU   X'04'                - LATEST
1 JIJOSTWI  EQU   X'05'                - WITHIN
1 JIJOSTBU  EQU   X'06'                - BYUSER
1 JIJOSTBO  EQU   X'07'                - BYOPERATOR
1 JIJOSTIM  EQU   X'08'                - IMMEDIATE
1 JIJOSTAS  EQU   X'09'                - AT-STREAM-STARTUP
1 JIJOSTDR  DS    CL6                 START DATE REQUESTED
1 JIJOSTTR  DS    CL4                 START TIME REQUESTED
1 JIJORER   DS    X                   RERUN INDICATOR
1 JIJORERN  EQU   X'00'                - NO
1 JIJORERY  EQU   X'80'                - YES
1 JIJOFLU   DS    X                   FLUSH INDICATOR
1 JIJOFLUN  EQU   X'00'                - NO
1 JIJOFLUY  EQU   X'80'                - YES
1 JIJOTIME  DS    F                   CPU TIME REQUESTED
1 JIJONTL   DS    X                   NO TIME LIMIT INDICATOR
1 JIJONTLN  EQU   X'00'                - NTL NOT REQUESTED
1 JIJONTLY  EQU   X'80'                - NTL REQUESTED
1 JIJOMONJ  DS    CL54                MONITORING JOB VARIABLE
1 JIJOSTDA  DS    CL6                 DATE JOB ACTUALLY STARTED
1 JIJOSTTI  DS    CL4                 TIME JOB ACTUALLY STARTED
1 JIJORPCO  DS    H                   REPEAT-COUNT
1 JIJOLEN   EQU   *-JIJOBDPL          LENGTH OF PARAMETERLIST
```

DJSI – Datenbereiche oder DSECTs für Jobscheduler-Makros erstellen (24-Bit-Schnittstelle)

Allgemeines

Anwendungsgebiet: Jobscheduler (Systemverwaltermakro); siehe Seite 169
Makrotyp: Definitionsmakro; siehe Seite 28

- JMS = Job Management System; JSS = Job Scheduling Supports.
  JSS ist Bestandteil von JMS.
- Für die 31-Bit-Schnittstelle siehe Makro **DJSIPL**.

Makrobeschreibung

Der Makro **DJSI** erstellt Namensdefinitionen, DSECTs oder Datenbereiche für das 24-Bit-Schnittstellenformat der Jobscheduler-Makros

JSATTCH	Job Scheduler mit Job Management System verbinden
JSDETCH	Job Scheduler von Job Management System lösen
JSEXPCT	nächstes Ereignis für Jobscheduler anfordern
JSRUNJB	übergibt Job an Klassenscheduler
JSINFO	überträgt die STREAM-PARAMETER
JSWAKE	spezifiziert nächstes Zeitereignis für Jobscheduler

DSECTs und Datenlisten beginnen mit dem Standardheader. Die Initialisierungswerte für den Standardheader sind eingetragen.

Makroaufrufformat und Operandenbeschreibung

DJSI
DSECT=YES / NO [,PREFIX=p] ,EVENT=NO / YES ,CLOCK=NO / YES ,JSINF=NO / YES ,STRTINF=NO / YES ,WAKE=NO / YES

DSECT=
gibt an, ob eine Dummy Section (DSECT) zu den nachfolgend spezifizierten Datenbereichen generiert wird, oder ob die Datenbereiche und Definitionen direkt im Anwenderprogramm angelegt werden.

YES
Eine Dummy Section wird generiert.

NO
Die Datenbereiche und Definitionen werden direkt im Anwenderprogramm an- bzw. abgelegt (unmittelbarer Zugriff über die symbolischen Namen).

EVENT=...
gibt an, ob der Ausgabebereich des Makros **JSEXPCT** und eine Liste der JMS-Ereignisse erstellt werden soll.

CLOCK=...
gibt an, ob der Ausgabebereich des Makros **JSATTCH** erstellt werden soll.

STRINF=...
gibt an, ob der Eingabebereich des Makros **JSRUNJB** erstellt werden soll.

JSINF=...
gibt an, ob der Ausgabebereich des Makros **JSINFO** erstellt werden soll.

WAKE=...
gibt an, ob der Eingabebereich des Makros **JSWAKE** erstellt wird.

PREFIX=
bezeichnet eine Zeichenfolge, mit der die symbolischen Namen der DSECT/Datenliste beginnen.

p
Präfix für die symbolischen Namen; Länge ≤ 2 Zeichen. Voreinstellung: p = JS.

Hinweis zum Makroaufruf

Jedem Subset (EVENT, CLOCK, ...) werden Equates für den Returncode der Jobscheduler-Makros hinzugefügt.

DJSIPL – Datenbereiche oder DSECTs für Jobscheduler-Makros erstellen (31-Bit-Schnittstelle)

Allgemeines

Anwendungsgebiet: Jobscheduler (Systemverwaltermakro); siehe Seite 169

Makrotyp: Definitionsmakro; siehe Seite 28

- Für die 24-Bit-Schnittstelle siehe Makro **DJSI**.

Makrobeschreibung

Der Makro **DJSIPL** erstellt Namensdefinitionen, DSECTs oder Datenbereiche für die 31-Bit-Schnittstelle der Jobscheduler-Makros

JSATTCH	Job Scheduler mit Job Management System verbinden
JSDETCH	Job Scheduler von Job Management System lösen
JSEXPCT	nächstes Ereignis für Jobscheduler anfordern
JSRUNJB	übergibt Job an Klassenscheduler
JSINFO	überträgt die STREAM-PARAMETER
JSWAKE	nächstes Zeitereignis für Jobscheduler spezifizieren.

DSECTs und Datenlisten beginnen mit dem Standardheader. Die Initialisierungswerte für den Standardheader sind eingetragen.

Makroaufrufformat und Operandenbeschreibung

```
DJSIPL
```

```
DSECT=YES / NO
[,PREFIX=p]
,JSATTCH=NO / YES
,JSEXPCT=NO / YES
,JSINFO=NO / YES
,JSDETCH=NO / YES
,JSRUNJB=NO / YES
,JSWAKE=NO / YES
```

DSECT=
gibt an, ob Dummy Sections (DSECTs) zu den nachfolgend angegebenen Ein-/Ausgabebereichen generiert werden, oder ob die Datenbereiche und Definitionen direkt im Anwenderprogramm angelegt werden.

YES
Eine Dummy Section wird generiert.

NO
Die Datenbereiche und Definitionen werden direkt im Anwenderprogramm an- bzw. abgelegt.

JSATTCH=...
gibt an, ob der Ausgabebereich des Makros **JSATTCH** erstellt werden soll.

JSDETCH=...
gibt an, ob der Eingabebereich des Makros **JSDETCH** erstellt werden soll.

JSEXPCT=...
gibt an, ob der Ausgabebereich des Makros **JSEXPCT** und eine Liste der JMS-Ereignisse erstellt werden soll.

JSRUNJB=...
gibt an, ob der Eingabebereich des Makros **JSRUNJB** erstellt werden soll.

JSINFO=...
gibt an, ob der Ausgabebereich des Makros **JSINFO** erstellt werden soll.

JSWAKE=...
gibt an, ob der Eingabebereich des Makros **JSWAKE** erstellt werden soll.

PREFIX=
bezeichnet eine Zeichenfolge, mit der die symbolischen Namen der DSECTs bzw. der Datenbereiche beginnen.

p
Präfix für die symbolischen Namen; Länge ≤ 2 Zeichen. Voreinstellung: präfix=`JI`.

DPOFEI – POSSIG-Eintrag erzeugen

Allgemeines

Anwendungsgebiet: (optimierte) Ereignissteuerung, siehe Seite 95
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

Forward Eventing (FEV) ist eine optimierte Form der synchronen Ereignissteuerung (synchrones Eventing). FEV vermeidet für wiederholte **POSSIG**- bzw. **SOLSIG**-Makroaufrufe in einem Programm die wiederholte Validation der angegebenen Operanden. Stattdessen wird eine Ereignisliste EVENTLST angelegt und z.B. für das Senden von Signalen zu einer Ereigniskennung (POSSIG-Funktion) einmalig ein POSSIG-Eintrag eingetragen. Im weiteren Programmverlauf wird bei der (realen) Sendeanforderung nur noch auf diesen Eintrag Bezug genommen (**RPOFEI**). Der Eintrag kann explizit wieder gelöscht werden (**DELFEI**).

Pro Teilnehmer können maximal 2047 Einträge in der EVENTLST erzeugt werden. Die Task des Aufrufers muss der Ereigniskennung zugeordnet sein (**ENAEI**).

Makrobeschreibung

Der Makro **DPOFEI** erzeugt einen POSSIG-Eintrag in der Ereignisliste EVENTLST. Die benötigten Angaben werden in den Eintrag übernommen (Name der Ereigniskennung oder Kurzkennung, Postcode, maximale Abholzeit für das Signal (Ereignis)). Dem Aufrufer wird eine Referenznummer für den Eintrag übergeben.
Es ist möglich, mehrere aufeinander folgende Aufrufe **DPOFEI** miteinander zu verketten. Die Kette kann auch mit dem Makro **DSOFEI** beendet werden. Bei der Verkettung ist zu beachten:

– **DSOFEI** muss immer als letzter Makro in der Kette angegeben werden.
– Der erste fehlerhafte Makroaufruf beendet die Kette; schon erzeugte Einträge werden wieder gelöscht.
– Der Operand REFNUM (für die Übergabe der Referenznummer) kann nur im ersten Makroaufruf der Kette angegeben werden.
– Es können maximal 5 Makroaufrufe miteinander verkettet werden. Alle Makros der Kette müssen dasselbe Schnittstellenformat benutzen.

Makroaufrufformat und Operandenbeschreibung

```
DPOFEI

{ { EINAME=name                                  }                                                  }
{ { EINAMAD=adr / (r) [,EINAMLN=länge]           },SCOPE=LOCAL / GROUP / USER_GROUP / GLOBAL        }
{ EIID=adr / (r)                                                                                    }

[,REFNUM=adr / (r)]

[, { SPOSTAD=adr / (r) } ]
   { SPOSTR=r            }

,SPOSTL=1 / 2

,CONTINU=NO / YES / DSOFEI

[,LIFETIM=sec / (r)]

[,PARMOD=24 / 31]

[,MF=L / (E,..)]
```

EINAME=
bezeichnet den Namen der Ereigniskennung, der sich der Aufrufer mit dem Makro **ENAEI** zugeordnet hat (Verbindung mit dem Operanden SCOPE beachten).

name
Name der Ereigniskennung

EINAMAD=
bezeichnet das Feld, das den Namen der Ereigniskennung enthält (Verbindung mit dem Operanden SCOPE beachten).

adr
symbolische Adresse des Feldes, das den Namen der Ereigniskennung enthält

(r)
r = Register mit dem Adresswert adr

EINAMLN=
bezeichnet die Länge des Namens der Ereigniskennung.
Bei Nichtangabe wird das Längenattribut des bei EINAMAD=adr angegebenen Feldes angenommen bzw. 54 Byte, wenn EINAMAD=(r) angegeben wurde.

länge
Länge des Namens in Byte

SCOPE=
beschreibt den Geltungsbereich (Teilnehmerkreis) für die Ereigniskennung. Der Name der Ereigniskennung ist nur in Verbindung mit dem Geltungsbereich eindeutig. SCOPE muss immer in Verbindung mit EINAME bzw. EINAMAD spezifiziert werden.

LOCAL
Die Ereigniskennung wird nur von der Task des Aufrufers benutzt.

GROUP
Teilnehmer sind alle Tasks unter der Benutzerkennung des Aufrufers.

USER_GROUP
Teilnehmer können alle Tasks sein, deren Benutzerkennungen der gleichen Benutzergruppe angehören wie die Benutzerkennung des einrichtenden Teilnehmers.

Der Operandenwert setzt die Existenz von Benutzergruppen voraus und kann daher nur angegeben werden, wenn die Funktionseinheit SRPM des Software-Produkts SECOS im System vorhanden ist. Vor einem Makroaufruf mit SCOPE=USER_GROUP muss deshalb mit dem Makro GETUGR (siehe Handbuch „SECOS“ [14]) geprüft werden, ob SRPM zur Verfügung steht; abhängig vom Ergebnis (Returncode) ist im Programm zu reagieren.

GLOBAL
Teilnehmer sind alle Tasks im System.

EIID=
bezeichnet die Kurzkennung für die Ereigniskennung.
Die Kurzkennung wird dem Anwender während der Ausführung des Makros **ENAEI** übergeben. Sie bezeichnet die Ereigniskennung eindeutig; ihre Verwendung beschleunigt die Makroausführung.

adr
symbolische Adresse eines 4 Byte langen Feldes, das die Kurzkennung enthält

(r)
r = Register mit dem Adresswert adr

REFNUM=
bezeichnet die Adresse eines Feldes, in das dem Aufrufer eine Referenznummer für den Eintrag in der EVENTLST übergeben wird.
Feldlänge = 4 Byte; Ausrichtung auf Wortgrenze.

adr
symbolische Adresse des Feldes, in das die Referenznummer übergeben werden soll

(r)
r = Register mit dem Adresswert adr

SPOSTAD=
bezeichnet ein Feld mit einem Postcode, der dem Empfänger des Signals (Ereignisses) übergeben werden soll (siehe Makro **SOLSIG** bzw. **DSOFEI**). Der Postcode ist 4 bzw. 8 Byte lang.
Feldlänge = 4 bzw. 8 Byte; Ausrichtung auf Wortgrenze.
Ein Postcode der Form X'00000000' wird nicht übertragen. Der Operand RPOSTR bietet dieselbe Funktion mit schnellerer Verarbeitung.

adr
symbolische Adresse des Feldes, das den Postcode enthält

(r)
r = Register mit dem Adresswert adr

SPOSTR=
bezeichnet ein Register, das den Postcode (direkt) enthält. Bei 2 Worten (8 Byte) Post Code muss das dem angegebenen Register folgende Register (der Zahl nach) das zweite Wort des Post Codes enthalten. Post Code X'00000000' wird nicht übertragen.

r
Register mit dem Postcode

SPOSTL=
bezeichnet die Länge des Post Codes in Worten.

1
Der Post Code ist 1 Wort (4 Byte) lang.

2
Der Post Code ist 2 Worte lang.

CONTINU=
ermöglicht die Verkettung des Makros **DPOFEI** mit weiteren, unmittelbar folgenden Aufrufen **DPOFEI** oder **DSOFEI**.

NO
Es folgt kein weiterer Aufruf **DPOFEI** oder **DSOFEI**.

YES
Es folgt ein Aufruf des Makros **DPOFEI**.

DSOFEI
Es folgt ein Aufruf des Makros **DSOFEI**.

LIFETIM=
ermöglicht die Angabe eines Zeitintervalls (in Sekunden), in dem das signalisierte Ereignis abgeholt werden muss (Makro **SOLSIG** oder **DSOFEI**). Nach Ablauf des Zeitintervalls wird das Ereignis aus der Ereigniswarteschlange entfernt. Die Ausführung erfolgt mit einer Genauigkeit von +10 Sekunden.

sec
Zeitangabe in Sekunden; 1 ≤ sec ≤ 43200. Voreinstellung: sec = 600.

(r)
r = Register, das den Wert für sec enthält.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

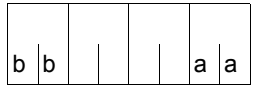
24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB).

Rückinformation und Fehleranzeigen

Während der Makrobearbeitung enthält Register R1 den Adresswert für die Operandenliste; Register R0 wird überschrieben.

R15:



Über die Ausführung des Makros DPOFEI wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'bb'	X'aa'	Erläuterung
X'00'	X'00'	Normale Ausführung: Ein POSSIG-Eintrag wurde erzeugt. Alle spezifizierten Operanden sind gültig.
X'04'	X'04'	Keine Aktion: Die maximale Anzahl von 2047 Aufrufen wurde überschritten.
X'0C'	X'04'	Keine Aktion: Die Ereigniskennung ist der Task des Aufrufers nicht zugeordnet.
X'10'	X'04'	Keine Aktion: Ungültige Operandenangabe.
X'14'	X'04'	Keine Aktion: Es existiert keine Ereigniskennung mit den angegebenen Namen oder der angegebenen Kurzkennung.

Beispiel einer verketteten Struktur:

```
DPOFEI   START
         BALR
         USING
         :
         DPOFEI EIID=IDP1,REFNUM=REFNR,CONTINU=YES
         DPOFEI EIID=IDP2,CONTINU=YES
         DPOFEI EIID=IDP3,CONTINU=DSOFEI
         DSOFEI EIID=IDS1
         :
         TERM
*****  DEFINITIONS ****************************
         :
IDP1     DS    CL4
IDP2     DS    CL4
IDP3     DS    CL4
IDS1     DS    CL4
REFNR    DS    F
         :
         END
```

DSHARE – Shared Code des Benutzers aus Common Memory Pools entladen

Allgemeines

Anwendungsgebiet: Binden und Laden; siehe Seite 47
Makrotyp: S-Typ, MF-Format **2**: Standardform/C-/D-/L-/E-/M-Form; siehe Seite 29

Zum dynamischen Bindelader DBL siehe auch Handbuch „BLSSERV“ [4].

Makrobeschreibung

Der Makroaufruf **DSHARE** entlädt ein einzelnes gemeinsam benutzbares Programm aus einem Common Memory Pool, das zuvor mit dem **ASHARE**-Makro geladen wurde. Die aufrufende Task muss an den Common Memory Pool angeschlossen sein, in dem sich das gemeinsam benutzbare Programm befindet. Wenn sich der letzte Teilnehmer von einem Memory Pool trennt, werden alle gemeinsam benutzbaren Programme dieses Memory Pools entladen und der Memory Pool wird aufgelöst (siehe Makro **DISMP**).

Makroaufrufformat und Operandenbeschreibung

```
DSHARE

 { PROGRAM=name   }
 { PROG@=adr / (r) }

[, { PGMVERS=*STD / version } ]
   { PGMVER@=adr / (r)      }

,MF=S / C / D / E / L / M

[,PARAM=adr / (r)]

,PREFIX=P / p

,MACID=BDS / macid
```

PROGRAM=name
Identifiziert das Programm, das mit dem **ASHARE**-Makro in den Common Memory Pool geladen wurde. Der angegebene Name muss eindeutig und darf nicht länger als 32 Zeichen sein.

PROG@=
Gibt die Adresse eines Feldes an, das den Programmnamen enthält. Angabe nur mit MF=M.

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.

(r)
r = Register mit der gesuchten Feldadresse.

PGMVERS=
Gibt die Programmversion an.

***STD**
Die Programmversion wird beim Entladen nicht berücksichtigt, d.h. das erste gefundene Programm mit dem angegebenen Namen wird entladen.

version
Die Versionsangabe darf maximal 24 Zeichen lang sein. Wenn diese Version des Programms nicht im Common Memory Pool existiert, wird nichts entladen und der entsprechende Returncode übergeben.

PGMVER@=
Gibt die Adresse eines Feldes an, das die Programmversion enthält. Angabe nur mit MF=M.

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.

(r)
r = Register mit der gesuchten Feldadresse.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. angegebenen Operanden PREFIX, MACID und PARAM siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

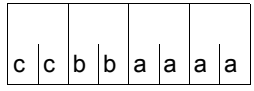
Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

Hinweise zum Makroaufruf

- Vor dem Aufruf von **DSHARE** muss sich der Benutzer an den Memory Pool anschließen.
- Shared Code kann nur mit dem Makro **DSHARE** aus Memory Pools entladen werden. Der Makro **UNBIND** kann dafür nicht verwendet werden.
- Es liegt in der Verantwortung des Benutzers, zu überprüfen, ob das zu entladende Programm noch von anderen Benutzern ausgeführt wird.

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros DSHARE wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Der Makro wurde normal ausgeführt.
X'00'	X'01'	X'0001'	Der Programmname fehlt.
X'00'	X'01'	X'0010'	Das Programm wurde in keinem für den Benutzer zugreifbaren Memory Pool gefunden.
X'00'	X'01'	X'0011'	Das Programm wurde gefunden, aber der Benutzer ist nicht an den betreffenden Memory Pool angeschlossen.
X'00'	X'20'	X'0100'	Systemfehler
X'00'	X'20'	X'0101'	DBL-Fehler während des Entladens
X'00'	X'20'	X'0103'	Fehler des DBL-Lock-Managers während der Abarbeitung des DSHARE-Makros.
X'00'	X'01'	X'FFFF'	Die Funktion wird nicht mehr oder noch nicht unterstützt.
X'00'	X'03'	X'FFFF'	Die Version der Schnittstelle wird nicht unterstützt.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

DSOFEI – SOLSIG-Eintrag erzeugen

Allgemeines

Anwendungsgebiet: (optimierte) Ereignissteuerung; siehe Seite 95
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

Forward Eventing (FEV) ist eine optimierte Form der synchronen Ereignissteuerung (synchrones Eventing). FEV vermeidet für wiederholte **SOLSIG**- bzw. **POSSIG**-Makroaufrufe in einem Programm die wiederholte Validation der angegebenen Operanden. Stattdessen wird eine Ereignisliste EVENTLST angelegt und z.B. für Signalanforderungen von einer Ereigniskennung (SOLSIG-Funktion) einmalig ein SOLSIG-Eintrag eingetragen. Im weiteren Programmfortschritt wird bei der (realen) Signalanforderung nur noch auf diesen Eintrag Bezug genommen (**RSOFEI**). Der Eintrag kann explizit wieder gelöscht werden (**DELFEI**).

Pro Teilnehmer können maximal 2047 Einträge in der EVENTLST erzeugt werden. Die Task des Aufrufers muss der Ereigniskennung zugeordnet sein (**ENAEI**).

Makrobeschreibung

Der Makro **DSOFEI** erzeugt einen SOLSIG-Eintrag in der Ereignisliste EVENTLST. Die benötigten Angaben werden in den Eintrag übernommen (Name der Ereigniskennung oder Kurzkennung, Postcode, maximale Wartezeit bis zum Eintreffen des Signals (Ereignisses)). Dem Aufrufer wird eine Referenznummer für den Eintrag übergeben.

Makroaufrufformat und Operandenbeschreibung

```
DSOFEI

{ { EINAME=name                          }                                          }
{ { EINAMAD=adr / (r) [,EINAMLN=länge]   } ,SCOPE=LOCAL / GROUP / USER_GROUP / GLOBAL }
{ EIID=adr / (r)                                                                    }

,REFNUM=adr / (r)

[,LIFETIM=sec / (r)]

[, { RPOSTAD=adr / (r) } ], RPOSTL=1 / 2, RPOSTNUM=anzahl / (r)
   { RPOSTR=r          }

[,PARMOD=24 / 31]

[,MF=L / (E,..)]
```

EINAME=
bezeichnet den Namen der Ereigniskennung, der sich der Aufrufer mit dem Makro **ENAEI** zugeordnet hat (Verbindung mit dem Operanden SCOPE beachten).

name
Name der Ereigniskennung

EINAMAD=
bezeichnet das Feld, das den Namen der Ereigniskennung enthält (Verbindung mit dem Operanden SCOPE beachten).

adr
symbolische Adresse des Feldes, das den Namen der Ereigniskennung enthält

(r)
r = Register mit dem Adresswert adr

EINAMLN=
bezeichnet die Länge des Namens der Ereigniskennung.
Bei Nichtangabe wird das Längenattribut des bei EINAMAD=adr angegebenen Feldes angenommen bzw. 54 Byte, wenn EINAMAD=(r) angegeben wurde.

länge
Länge des Namens in Byte

SCOPE=
beschreibt den Geltungsbereich (Teilnehmerkreis) für die Ereigniskennung. Der Name der Ereigniskennung ist nur in Verbindung mit dem Geltungsbereich eindeutig. SCOPE muss immer in Verbindung mit EINAME bzw. EINAMAD spezifiziert werden.

LOCAL
Die Ereigniskennung wird nur von der Task des Aufrufers benutzt.

GROUP
Teilnehmer sind alle Tasks unter der Benutzerkennung des Aufrufers.

USER_GROUP
Teilnehmer können alle Tasks sein, deren Benutzerkennungen der gleichen Benutzergruppe angehören wie die Benutzerkennung des einrichtenden Teilnehmers.

Der Operandenwert setzt die Existenz von Benutzergruppen voraus und kann daher nur angegeben werden, wenn die Funktionseinheit SRPM des Software-Produkts SECOS im System vorhanden ist. Vor einem Makroaufruf mit SCOPE=USER_GROUP muss deshalb mit dem Makro GETUGR (siehe Handbuch „SECOS“ [14]) geprüft werden, ob SRPM zur Verfügung steht; abhängig vom Ergebnis (Returncode) ist im Programm zu reagieren.

GLOBAL
Teilnehmer sind alle Tasks im System.

EIID=
bezeichnet die Kurzkennung für die Ereigniskennung.
Die Kurzkennung wird dem Anwender während der Ausführung des Makros **ENAEI** übergeben. Sie bezeichnet die Ereigniskennung eindeutig; ihre Verwendung beschleunigt die Makroausführung.

adr
symbolische Adresse eines 4 Byte langen Feldes, das die Kurzkennung enthält

(r)
r = Register mit dem Adresswert adr

REFNUM=
bezeichnet die Adresse eines Feldes, in das dem Aufrufer eine Referenznummer für den Eintrag in der EVENTLST übergeben wird.
Feldlänge = 4 Byte; Ausrichtung auf Wortgrenze.

adr
symbolische Adresse des Feldes, in das die Referenznummer übergeben werden soll

(r)
r = Register mit dem Adresswert adr

LIFETIM=
ermöglicht die Angabe eines Zeitintervalls (in Sekunden) als maximale Wartezeit für die Task des Aufrufers bis zum Eintreffen des Ereignisses (Signals). Die Ausführung erfolgt mit einer Genauigkeit von +10 Sekunden.

sec
Zeitangabe in Sekunden. $1 \leq sec \leq 43200$. Voreinstellung: sec = 600.

(r)
r = Register, das den Wert für sec enthält.

RPOSTAD=
bezeichnet ein Feld, in das ein Post Code übertragen werden soll. Der Post Code kann 4 oder 8 Byte lang sein. Der Operand RPOSTL legt fest, ob nur das erste Wort oder beide Wörter des Post Codes in das Feld eingetragen werden.
Der Operand RPOSTR führt dieselbe Funktion aus wie RPOSTAD; die Verarbeitung erfolgt aber schneller.

adr
symbolische Adresse des Feldes, in das der Post Code eingetragen werden soll.
Feldlänge = 4 oder 8 Byte.

(r)
r = Register mit dem Adresswert adr

RPOSTR=
bezeichnet ein Register, in das der Post Code direkt eingetragen werden soll. Ein Post Code aus 2 Worten bestehend, wird in das angegebene und das nachfolgende Register (der Zahl nach) eingetragen, wenn RPOSTL=2 angegeben wurde.

r
Register, in das der Post Code eingetragen werden soll

RPOSTL=
bezeichnet die Anzahl der Worte des Post Codes, die empfangen werden sollen.

1
Von dem Post Code soll nur 1 Wort (das erste Wort) übertragen werden.

2
Der Post Code soll in voller Länge (2 Worte) übertragen werden.

RPOSTNUM=
bezeichnet die maximale Anzahl Post Codes, die übertragen werden sollen. Der Bereich, der bei RPOSTAD angegeben ist, muss groß genug sein, um alle Post Codes empfangen zu können. Benötigte Länge: `RPOSTNUM*2+1` (1 Byte Indikator X'FF', Listenende).

anzahl
Anzahl Post Codes, die übertragen werden sollen.

(r)
r = Register mit der Anzahl Post Codes, die übertragen werden sollen.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

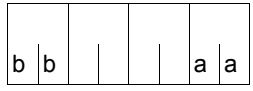
24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB).

Rückinformation und Fehleranzeigen

Während der Makrobearbeitung enthält Register R1 den Adresswert für den Datenbereich; Register R0 wird überschrieben.

R15:



Über die Ausführung des Makros DSOFEI wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'bb'	**X'aa'**	**Erläuterung**
X'00'	X'00'	Normale Ausführung: Ein SOLSIG-Eintrag wurde erzeugt. Alle spezifizierten Operanden sind gültig.
X'04'	X'04'	Keine Aktion: Die maximale Anzahl von 2047 Aufrufen wurde überschritten.
X'0C'	X'04'	Keine Aktion: Die Ereigniskennung ist der Task des Aufrufers nicht zugeordnet.
X'10'	X'04'	Keine Aktion: Ungültige Operandenangabe.
X'14'	X'04'	Keine Aktion: Es existiert keine Ereigniskennung mit den angegebenen Namen oder der angegebenen Kurzkennung.

DSPSRV – Datenraum verwalten

Allgemeines

Anwendungsgebiet: Erweiterung durch Datenräume; siehe Seite 61
Makrotyp: S-Typ, MF-Format **3**: C-/D-/L-/M-/R-/E-Form; siehe Seite 29

Der Makro **DSPSRV** kann auf allen BS2000-Servern verwendet werden (siehe Abschnitt „Erweiterung durch Datenräume“ auf Seite 61).

Makrobeschreibung

Mit dem Makro **DSPSRV** kann der Anwender unter Angabe des Typs, eines Namens, eines Geltungsbereiches und der gewünschten Größe einen Datenraum anlegen. Damit wird dieser Anwender zum Eigentümer des Datenraumes. Das System gibt eine Identifikation zurück (die SPID), die den Datenraum sessionweit eindeutig kennzeichnet.
Um auf einen Datenraum zugreifen zu können, muss das Programm mit ihm über Zugriffslisten verbunden werden. Diese Verbindung wird durch den Makro **ALESRV** realisiert.

Der Datenraum-Typ bestimmt die Art der Allokierung/Deallokierung des Speichers innerhalb des Datenraums und die Funktionen, die dazu zur Verfügung stehen.

Die Funktionen des **DSPSRV**-Makros ermöglichen:
– einen Datenraum anzulegen (FCT=CREATE),
– einen bestehenden Datenraum wieder freizugeben (FCT=DESTROY),
– Informationen über einen bestehenden Datenraum abzufragen (FCT=INFORM),

für einen Datenraum vom Typ STACK:
– einen bestehenden Datenraum um Speicherseiten zu erweitern (FCT=EXTEND),
– den Inhalt eines bestehenden Datenraumes (auch teilweise in 4KB-Einheiten) zu löschen, d.h. mit binären Nullen zu überschreiben (FCT=CLEAR),
– die aktuelle Größe eines Datenraums zu verringern (FCT=REDUCE),

für einen Datenraum vom Typ HEAP:
– einen Bereich für einen Datenraum anzufordern (FCT=GETAREA) und
– einen Bereich für einen Datenraum freizugeben (FCT=RETAREA).

Makroaufrufformat und Operandenbeschreibung

```
DSPSRV

FCT=( CREATE, NAME='name'/name_adr, INISIZE=zahl / (r)
        ,MAXSIZE=zahl / (r), DIAPROT=NO/YES,
        ,SCOPE=LOCAL / GROUP / USER_GROUP / GLOBAL
        ,TYPE=STACK / HEAP
    | DESTROY,SPID=spid_adr
    | INFORM,IDENT=NAME / SPID [,SPID=spid_adr][,NAME='name' / name_adr]
        ,SCOPE=LOCAL / GROUP / USER_GROUP / GLOBAL
    | EXTEND,SPID=spid_adr ,SIZE=zahl / (r)
    | CLEAR,SPID=spid_adr,AREA=area_adr / (r) ,SIZE=zahl / (r)
    | REDUCE,SPID=spid_adr, SIZE=zahl / (r)
    | GETAREA,SPID=spid_adr, SIZE=zahl / (r)
    | RETAREA,SPID=spid_adr, AREA=area_adr / (r) ,SIZE=zahl / (r) )

[,MF=C / D / L / M / E / R[,SPID=spid_adr][,EXTADDR=ext_adr][.AREA=area_adr]]

[,PARAM=adr / (r)]

,PREFIX=N / p

,MACID=VDD / macid
```

In der folgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

AREA=
legt die Startadresse eines Datenraumbereiches fest. Dieser Operand kann sowohl Eingabeoperand (FCT=RETAREA, CLEAR) als auch Ausgabeoperand (FCT=GETAREA) sein. Die Adresse muss auf Seitengrenze (4KB) ausgerichtet werden und im Bereich des angelegten Datenraumes liegen.

area_adr
symbolische Adresse (Name) eines Feldes (4 Byte), das die Startadresse des Bereiches enthält.

(r)
r = Register, dessen Inhalt die Startadresse ist.

DIAPROT=
gibt an, ob der Datenraum gegen Zugriffe durch Diagnose-Tools (z.B. AID, USERDUMP, CDUMP2) geschützt werden soll.

NO
Der Datenraum wird nicht geschützt, d.h. Zugriffe durch Diagnose-Tools sind erlaubt.

YES
Der Datenraum soll besonders geschützt werden. Zugriffe durch Diagnose-Tools werden nicht erlaubt.

EXTADDR=
legt die Startadresse des zu erweiterenden Datenraums fest. Dieser Operand ist ein Ausgabeoperand (nur bei MF=R).

ext_addr
Adresse zur Ausgabe der Startadresse der neuen Speicherplatzerweiterung (bei FCT=EXTEND).

FCT=
bestimmt die auszuführende Funktion des Makros **DSPSRV**.

CREATE
legt eine neuen Datenraum an. Der Aufrufer wird der Eigentümer des Datenraumes. Das System gibt bei Returncode X'aaaa'=X'0000' die SPID zurück, die mit MF=R aus dem Datenbereich gelesen werden kann.

DESTROY
gibt einen bestehenden Datenraum wieder frei, sofern der Aufrufer auch der Eigentümer des Datenraumes ist.

EXTEND
erweitert einen bestehenden Datenraum vom Typ STACK um Speicherseiten zu je 4KB. Zur Größe des Datenraums siehe Hinweise zur Adressraumgröße.
Das System gibt bei Returncode X'aaaa'=X'0000' die Startadresse der Speicherplatzerweiterung (EXTADDR) zurück, die mit MF=R aus dem Datenbereich gelesen werden kann. Der allokierte Bereich wird mit binären Nullen überschrieben.

CLEAR
löscht Inhalte eines Datenraumbereiches vom Typ STACK, indem Speicherseiten in Einheiten zu je 4KB mit binären Nullen überschrieben werden. Die so gelöschten Speicherseiten bleiben nicht länger im realen Speicher (Seitenwechselspeicher) erhalten.

INFORM
informiert über den mit seinem Namen und Geltungsbereich oder seiner SPID angegebenen Datenraum. Die mit MF=D generierte DSECT beinhaltet alle Informationen.

REDUCE
verringert die aktuelle Größe eines Datenraums vom Typ STACK um Speicherseiten zu je 4 KB.

GETAREA
weist einen allokierten Bereich innerhalb eines Datenraums vom Typ HEAP zu. Das System gibt bei Returncode X'aaaa'=X'0000' die Startadresse des Bereichs (AREA) zurück, die mit MF=R aus dem Datenbereich gelesen werden kann.. Der allokierte Bereich wird mit binären Nullen überschrieben.

RETAREA
gibt einen allokierten Bereich innerhalb eines Datenraums vom Typ HEAP frei.

IDENT=
bestimmt, durch welche Operanden (NAME und SCOPE oder SPID) der Datenraum identifiziert werden soll (bei FCT=INFORM).

NAME
Der Datenraum wird durch Angabe seines Namens und seines Geltungsbereiches identifiziert.

SPID
Der Datenraum wird durch die Angabe seiner SPID identifiziert.

INISIZE=
gibt die Anfangsgröße des angeforderten Datenraumes in Einheiten zu je 4KB an. Zur Größe des Datenraums siehe Hinweise zur Adressraumgröße und zur Allokierungsgröße (siehe Seite 457).

zahl
positiver, ganzzahliger Wert (X'01' .. X'80000'), der die Anfangsgröße des Datenraums angibt.

(r)
r = Register, dessen Inhalt zahl ist.

MAXSIZE=
gibt die gewünschte maximale Größe des angeforderten Datenraumes in Einheiten zu je 4KB an. Für einen Datenraum vom Typ HEAP wird die angegebene Größe auf die nächste MB-Grenze aufgerundet.
Ein Datenraum muss seine angegebene maximale Größe nicht erreichen, sie stellt nur eine Begrenzung nach oben dar. Die maximal zulässige Größe ist abhängig von der im Benutzerkatalog angegebenen maximal erlaubten Adressraumgröße (ADDRESS-SPACE-LIMIT), siehe Hinweise zur Adressraumgröße.

zahl
positiver, ganzzahliger Wert (X'01' .. X'80000'), der die maximale Größe des Datenraums angibt.

(r)
r = Register, dessen Inhalt zahl ist.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form, D-Form, R-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form, R-Form oder M-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

Mit MF=R können die Ausgabeparameter der Funktionen CREATE (SPID=), INFORM (SPID=), EXTEND (EXTADDR=) und GETAREA (AREA=) aus dem Parameterbereich ausgelesen werden.

NAME=
benennt den Datenraum. Länge = 1..54 alphanumerische Zeichen, wobei das erste Zeichen ein Buchstabe oder eines der Zeichen # oder @ sein muss.
Der Name eines Datenraumes ist nur innerhalb seines Geltungsbereiches (siehe Operand SCOPE) eindeutig, d.h. dass Datenräume mit gleichem Namen, aber unterschiedlichen Geltungsbereichen gleichzeitig existieren können.

'name'
name = Name des Datenraumes.

name_adr
symbolische Adresse (Name) eines Feldes (54 Bytes), das den Namen des Datenraumes in alphanumerischen Zeichen enthält.

SCOPE=
bestimmt den Geltungsbereich des angegebenen Datenraumes. Ein Datenraum ist nur innerhalb seines Geltungsbereiches eindeutig durch seinen Namen gekennzeichnet.
Der Geltungsbereich bestimmt, welche Tasks am angegebenen Datenraum teilhaben, auf ihn zugreifen können.

LOCAL
Der Datenraum wird nur von der einrichtenden Task genutzt. Andere Tasks haben keinen Zugriff.

GROUP
Alle Tasks mit der Benutzerkennung der einrichtenden Task können sich an den Datenraum anschließen.

USER_GROUP
Alle Tasks, deren Benutzerkennung derselben Benutzergruppe angehören wie die Benutzerkennung der einrichtenden Task, können den Datenraum nutzen.
Voraussetzung ist, dass das Subsystem SRPM geladen wurde.

GLOBAL
Alle Tasks im laufenden System können sich an den Datenraum anschließen.

SIZE=
legt die Größe des Bereichs bei den Funktionen EXTEND, REDUCE, CLEAR, GETAREA und RETAREA fest. Dieser Operand wird in Einheiten zu je 4KB angegeben.
Zu beachten ist:

- bei FCT=EXTEND:
  Der um SIZE erweiterte Datenraum darf nicht größer werden, als bei MAXSIZE angegeben ist, siehe Hinweise zur Adressraumgröße auf Seite 457.
- bei FCT=CLEAR/RETAREA:
  Der durch AREA und SIZE bestimmte Bereich muss innerhalb des Datenraumes liegen, der durch SPID bestimmt ist.
- bei FCT=REDUCE:
  Der Wert von SIZE darf nicht größer als die aktuelle Größe des Datenraums sein.
- bei FCT=GETAREA:
  Die Summe aller allokierten Bereiche darf nicht größer werden als bei MAXSIZE angegeben.

zahl
positiver, ganzzahliger Wert ≥ 1, der die Anzahl der Einheiten zu je 4KB bestimmt, um die der Datenraum erweitert werden soll (FCT=EXTEND) bzw. die gelöscht werden sollen (FCT=CLEAR).

(r)
r = Register, dessen Inhalt zahl ist.

SPID=
kennzeichnet einen Datenraum eindeutig im Gesamtsystem.
Sie wird beim Anlegen eines Datenraumes vom System vergeben.
Dieser Operand kann sowohl Ein- als auch Ausgabeoperand sein.

spid_adr
symbolische Adresse (Name) eines Feldes (8 Byte), das die SPID des Datenraumes enthält.

TYPE=
bestimmt die Art der Allokierung/Deallokierung des Speichers innerhalb des Datenraums. Er wird zum Zeitpunkt der Erzeugung bestimmt.

STACK
Ein Datenraum vom Typ STACK ist ein virtuell zusammenhängender allokierter Bereich, beginnend ab Adresse 0 bis zur aktuellen Größe.
Die verfügbaren Allokierungsfunktionen sind EXTEND, REDUCE und CLEAR. Die Funktionen GETAREA und RETAREA werden abgewiesen.

HEAP
Ein Datenraum vom Typ HEAP ist ein virtueller Adressraum, in dem dynamisch beliebig große Bereiche bis zur maximalen Größe des Datenraums allokiert werden können.
Die verfügbaren Allokierungsfunktionen sind GETAREA und RETAREA. Die Funktionen EXTEND, REDUCE und CLEAR und der Parameter INISIZE werden abgewiesen.

Hinweis

DIV (Data in Virtual)-Funktionalität wird nur in Datenräumen vom Typ STACK unterstützt.

Hinweise zum Makroaufruf

– Der Operand SPID kann sowohl Ein- als auch Ausgabeoperand sein, d.h. dass er durch FCT=CREATE oder FCT=INFORM in dem erzeugten Datenbereich als Ausgabeoperand vorliegt und bei nochmaliger Verwendung dieses Datenbereichs auch als Eingabeoperand gültig ist. Dasselbe gilt für den Operand AREA bei den Funktionen GETAREA und RETAREA.

– Die Summe der von der Task schon belegten und durch den Makro **DSPSRV** zusätzlich angeforderten Speicherseiten darf die im Benutzerkatalog eingetragene maximale Größe des Adressraums (ADDRESS-SPACE-LIMIT) des Eigentümers nicht überschreiten. Dasselbe gilt für die Angabe von MAXSIZE.
Mit dem Kommando SHOW-USER-ATTRIBUTES PUBSET=*HOME kann sich der Anwender über seine Adressraumgröße informieren. Ist die Adressraumgröße ausgeschöpft, wird ein Speicherplatzfehler gemeldet und die Funktion abgebrochen.

– Bei Beendigung des Programms, das den Datenraum erzeugt hat, wird der Datenraum automatisch, d.h. auch ohne FCT=DESTROY, freigegeben. Die SPID verliert ihre Gültigkeit, noch bestehende ALETs werden jedoch nicht gelöscht. Greift ein anderes Programm mit einem solchen ALET auf den nicht mehr existierenden Datenraum zu, wird es abgebrochen.

– Wenn in einem zu löschenden Datenraum noch DIV-Fenster existieren, so müssen diese vor der Freigabe geschlossen werden.

Hinweise zur Adressraumgröße

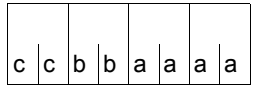
– Die im Benutzerkatalog angegebene maximale Adressraumgröße darf von keiner Angabe zum Erzeugen oder Erweitern von Datenräumen überschritten werden. Das bedeutet, dass die Summe aller Speicheranforderungen kleiner/gleich der im Feld ADDRESS-SPACE-LIMIT stehenden Adressraumgröße sein muss. Die Summe aller Speicheranforderungen ergibt sich aus:

  – den im Programmraum allokierten Klasse-6-Speicherseiten,
  – den allokierten Seiten, die für bereits erzeugte Datenräume angefordert wurden,
  – den mit dem aktuellen **DSPSRV**-Aufruf angeforderten Seiten (mit INISIZE und MAXSIZE beim Erzeugen oder mit SIZE beim Erweitern).

– Die für einen Datenraum beim Erzeugen angegebene maximale Größe (MAXSIZE) muss größer oder gleich der bei INISIZE angegebenen Anfangsgröße des Datenraums sein.
Die beim Erweitern eines Datenraums anzugebende Größe SIZE darf (zusammen mit INISIZE und evtl. schon vorher mit FCT=EXTEND hinzugefügten Speicherseiten) MAXSIZE nicht überschreiten.

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros DSPSRV wird im Standardheader ein Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Funktion erfolgreich ausgeführt. Mit MF=R kann aus dem Datenbereich folgender Wert gelesen werden: – bei FCT=CREATE: spid_adr (Operand SPID) – bei FCT=EXTEND: ext_adr (Operand EXTADDR) – bei FCT=INFORM: spid_adr (Operand SPID) – bei FCT=GETAREA: area_adr (Operand AREA)
X'02'	X'00'	X'0001'	Warnung: Der angegebene Datenraum wurde freigegeben, obwohl noch andere Programme mit ihm verbunden sind (bei FCT=DESTROY).
X'00'	X'01'	X'0003'	Ungültiger Operand FCT.
X'01'	X'01'	X'0003'	Ungültiger Operand NAME.
X'02'	X'01'	X'0003'	Ungültiger Operand SCOPE.
X'04'	X'01'	X'0003'	Ungültiger Operand TYPE.
X'05'	X'01'	X'0003'	Ungültiger Operand IDENT.
X'06'	X'01'	X'0003'	Ungültiger Operand MAXSIZE.
X'07'	X'01'	X'0003'	Ungültiger Operand INISIZE.
X'0A'	X'01'	X'0003'	Ungültiger Operand DIAPROT.
X'0C'	X'01'	X'0003'	Ungültiger Operand AREA.
X'0D'	X'01'	X'0003'	Ungültiger Operand SIZE.
X'FF'	X'01'	X'0003'	Ungültige Kombination von Operanden.
	X'20'	X'0005'	Interner Fehler.
X'00'	X'40'	X'000D'	DIV-Anwendung läuft: Der Datenraum enthält DIV-Fenster (bei FCT=DESTROY).
X'00'	X'40'	X'0102'	Die Angabe des Operanden NAME ist fehlerhaft. Es existiert bereits ein Datenraum mit dem angegebenen Namen (nur bei FCT=CREATE).
X'00'	X'40'	X'0104'	Ungültige Angabe von NAME und/oder SCOPE (bei FCT=INFORM).
X'00'	X'40'	X'0106'	Max. Seitenwechselspeicher-Auslastung erreicht.
X'00'	X'40'	X'0107'	Der im Benutzerkatalog eingetragene maximale Adressraum für die Task (ADDRESS-SPACE-LIMIT) wurde überschritten. Ungültige Angabe von INISIZE oder MAXSIZE (bei FCT=CREATE) oder SIZE (bei FCT=EXTEND).

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'40'	X'0202'	Fehler bei SCOPE=USER_GROUP: Das Subsystem SRPM ist nicht geladen (bei FCT=CREATE).
X'00'	X'40'	X'0206'	Max. Hauptspeicher-Auslastung erreicht.
X'00'	X'40'	X'0302'	Die aufrufende Task ist nicht der Eigentümer des Datenraumes und darf diesen nicht löschen (bei FCT=DESTROY).
X'00'	X'40'	X'0304'	Die Angabe des SPID-Operanden ist fehlerhaft: Der angegebene Datenraum existiert nicht oder der Aufrufer hat keine Berechtigung, auf diesen Datenraum zuzugreifen.
X'00'	X'40'	X'0306'	Max. erlaubte Anzahl von Datenräumen erreicht (FCT=CREATE).
X'00'	X'40'	X'0404'	Fehler bei TYPE. Der Typ des Datenraums ist für die angegebene Funktion ungültig.
X'00'	X'40'	X'0406'	Adressraum-Sättigung. Nicht genügend freier und zusammenhängender Adressraum innerhalb des Datenraums vorhanden, um die Anforderung zu erfüllen (bei FCT=GETAREA)
X'00'	X'40'	X'0604'	Ungültiger Operand (MAX)SIZE: – Die max. Größe des Datenraumes wird überschritten (bei FCT=EXTEND). – Die angegebene Größe liegt über der aktuellen Größe des Datenraums (bei FCT=REDUCE). – Die angegebene Größe liegt über der maximalen Größe des Datenraums (bei FCT=GETAREA)
X'00'	X'40'	X'0C04'	Der angegebene Bereich ist nicht Bestandteil des Datenraumes (bei FCT=CLEAR und FCT=RETAREA).
X'00'	X'40'	X'0F04'	Allokierungsfehler. Der angegebene Bereich ist nicht innerhalb des Datenraums allokiert (bei FCT=RETAREA).
X'00'	X'81'	X'0106'	Max. Seitenwechselspeicher-Auslastung erreicht.
X'00'	X'81'	X'0306'	Interner Resourcen-Engpass.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Beispiel siehe im Abschnitt „Erweiterung durch Datenräume“ auf Seite 68.

DTMODE – Datenliste oder DSECT für Makro TMODE erstellen

Allgemeines

Anwendungsgebiet: Abfragen und Zugriff zu Listen und Tabellen; siehe Seite 158
Makrotyp: Definitionsmakro; siehe Seite 28

Makrobeschreibung

Der Makro **DTMODE** generiert eine Beschreibung des Ein-/Ausgabebereichs für den Makro **TMODE** im 31-Bit-Adressierungsmodus. Die Beschreibung wird als DSECT oder als Datenabschnitt (Datenliste) angelegt und beginnt mit dem Standardheader.
Die Initialisierungswerte sind in die Datenliste eingetragen.

Makroaufrufformat und Operandenbeschreibung

DTMODE
DSECT=YES / NO [,PREFIX=p]

DSECT=
gibt an, ob eine DSECT zu dem Ausgabebereich oder ein Datenabschnitt (Datenliste) als Ausgabebereich angelegt wird.

YES
Eine DSECT wird angelegt.

NO
Eine Datenliste wird angelegt.

PREFIX=
bezeichnet eine Zeichenfolge, mit der die symbolischen Namen der DSECT/Datenliste beginnen.

=p
Präfix für die symbolischen Namen. Länge ≤ 2 Zeichen; Voreinstellung: p = TM.

Layout der DSECT zu dem Ein-/Ausgabebereich

```
          DTMODE DSECT=YES
1         #INTF REFTYPE=REQUEST,INTNAME=TMODE,INTCOMP=002
1 *---------------- P A R A M E T E R L I S T
1 TMODPL    DSECT
1           FHDR  UNIT=43,FUNCT=1,VERS=1
2          DS    0A
2          DS    0XL8               GENERAL OPERAND LIST HEADER
2          DC    AL2(43)            FUNCTION UNIT NUMBER
2          DC    AL1(1)             FUNCTION NUMBER
2          DC    AL1(1)             FUNCTION INTERFACE VERSION NUMBER
2          DC    X'FFFFFFFF'         Returncode is virgin
1 TMODTSN   DC    CL4' '                 TASK SEQUENCE NUMBER
1 TMODUSER  DC    CL8' '                 USER IDENTIFICATION NUMBER
1 TMODACCT  DC    CL8' '                 TASK ACCOUNT NUMBER
1 TMODTIME  DC    F'0'                   TASK CPU TIME
1 TMODPRIV  DC    AL1(0)                 TASK PRIVELEDGE CODE
1 TMODPRSA  EQU   1                      SYSTEM ADMINISTRATOR BIT
1 TMODPRUS  EQU   2                      USER BIT
1 TMODLLEN  DC    AL1(0)                 PHYSICAL LINE LENGTH (TERMINAL)
1 TMODVDT   DC    AL1(0)                 VIRTUAL DEVICE TYPE
1 TMODLINC  EQU   1                      LINE MODE CAPABILITY
1 TMODFORC  EQU   2                      FORMAT MODE CAPABILITY
1 TMODCMPC  EQU   4                      COMPATIBLE MODE CAPABILITY
1 TMODFYSC  EQU   8                      PHYSICAL MODE CAPABILITY
1 TMODEXLC  EQU   16                     EXTENDED LINE MODE CAPABILITY
1 TMODEOM   EQU   64                     EVANESCENT OUTPUT MESSAGES(VDU)
1 TMODTYPE  DC    AL1(0)                 TASK OR TERMINAL TYPE
1 TMODPRI   DC    AL1(0)                 TASK PRIORITY
1 TMODMSG   DC    X'00'                  MSG OPTIONS              :*
1 TMODMSGF  EQU   X'01'                    |
1 TMODMSGC  EQU   X'02'                    |
1 TMODMSGH  EQU   X'04'                    > SEE /OPTION COMMAND
1 TMODMSGT  EQU   X'08'                    |
1 TMODMSGL  EQU   X'20'                    |
1 TMODBUFS  DC    H'0'                   BUFFERSIZE
1 TMODPNAM  DC    CL8' '                 PROGRAM NAME
1 TMODNAME  DC    CL8' '                 JOB NAME FROM /LOGON
1 TMODPLL   EQU   *-TMODPL                LENGTH OF PARAMETERLIST
```

Erläuterung der Feldinhalte

TMODTSN: Auftragsnummer (TSN); 4 Zeichen
TMODUSER: Benutzerkennung (userid); 8 Zeichen
TMODACCT: Abrechnungsnummer; 8 Zeichen
TMODTIME: Von der Task verbrauchte CPU-Zeit, angegeben als Vielfaches von 100 Mikrosekunden; 8-stellige Dezimalzahl. Werte > 204800 Sekunden werden nicht ausgegeben; bei weiteren Aufrufen wird dieser Maximalwert wiederholt.
TMODPRIV: Taskprivilegierung. Es bedeuten:
X'01' ≙ Task unter der Kennung der Systemverwaltung (TSOS).
X'02' ≙ Task unter der Kennung des (nichtprivilegierten) Aufrufers.
TMODLLEN: (physikalische) Zeilenlänge der Datensichtstation; nur wenn diese im Zeilenmodus arbeitet.
TMODVDT: Eigenschaften der Datensichtstation.
TMODTYPE: Typ der Datensichtstation. Für einen Batch-Auftrag wird X'00' ausgegeben. Sonst:
X'02' ≙ Schreibstation 8103
X'04' ≙ Datensichtstation 8150
X'11' ≙ TRANSDATA 8415, 8418
X'15' ≙ Datensichtstation 8151
X'16' ≙ Datensichtstation 8152
X'17' ≙ Schreibstation 8110
X'18' ≙ Datensichtstation 8161 mit 54 Zeichen je Zeile
X'19' ≙ Datensichtstation 8161 mit 64 Zeichen je Zeile
X'1A' ≙ Datensichtstation 8161 mit 80 Zeichen je Zeile
X'2C' ≙ Datensichtstation 8162
X'2D' ≙ Datensichtstation 8160
X'35' ≙ Datensichtstation 9750
X'4F' ≙ Datensichtstation 9763
TMODPRI: Runpriorität der Task; 2-stellige Sedezimalzahl.
TMODBUFS: Länge des physikalischen Ein-/Ausgabepuffers der Datenstation; 4-stellige Sedezimalzahl. Nicht bei Batch-Aufträgen.
TMODPNAM: Programmname, wenn das Modul mit dem statischen Lader (ELDE) geladen wurde; 8 Zeichen. X'00...0', wenn das Modul mit dem dynamischen Bindelader (DBL) geladen wurde.
TMODNAME: Jobname aus dem SET-LOGON-PARAMETERS-Kommando.

ENACO – Contingency-Definition eröffnen

Allgemeines

Anwendungsgebiet: Contingency-Verfahren; siehe Seite 111
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

Makrobeschreibung

Durch den Aufruf des Makros **ENACO** wird eine Routine als Contingency-Prozess definiert. Der Contingency-Prozess erhält dabei einen Namen. Er muss definiert sein, bevor er in einem **SOLSIG**- oder **POSSIG**-Makroaufruf angegeben werden kann. Durch den Aufruf **ENACO** wird eine Kurzkennung zur Verfügung gestellt, die in weiteren Aufrufen, die sich auf den Contingency-Prozess beziehen, zu verwenden ist. Ein Programm kann maximal 400 Contingency-Prozesse gleichzeitig verwenden. Der Geltungsbereich eines Contingency-Prozesses ist lokal: Die Verwendung ist auf die Task des Aufrufers beschränkt.

Der zum Zeitpunkt des **ENACO**-Aufrufs bestehende Adressierungsmodus (AMODE) muss auch bei Ablauf der Contingency-Routine eingeschaltet sein.

Makroaufrufformat und Operandenbeschreibung

```
ENACO

{ CONAME=name                          }
{ CONAMAD=adr / (r) [,CONAMLN=länge]   }
,COADAD=adr / (r) ,COIDRET=adr / (r)
[,COMAD=adr / (r) ] [,LEVEL=prio / (r)]
[,PARMOD=24 /  31] [,MF=L / (E, ..)]
```

CONAME=
bezeichnet den Namen des Contingency-Prozesses.

name
Name des Contingency-Prozesses. 1 ≤ Namenslänge ≤ 54.

```
Namensbildung:
1. Zeichen:     Buchstabe, #, @
2.-54. Zeichen: beliebige Kombination aus der Zeichenmenge
                (A,...,Z,0,...,9,$,#,@).
Das erste Blank (X'40') beendet den Namen.
```

CONAMAD=
bezeichnet die Adresse des Felds mit dem Namen des Contingency-Prozesses (Namensbildung siehe CONAME).

adr
symbolische Adresse (Name) des Feldes

(r)
r = Register mit dem Adresswert adr

CONAMLN=
beschreibt die Länge des Namens des Contingency-Prozesses, wenn CONAMAD=... spezifiziert wurde.

länge
Länge des Namens in Byte
Voreinstellung:
– Längenattribut des bei CONAMAD=... angegebenen Feldes
– 54 Byte, wenn CONAMAD=(r) spezifiziert wurde

COADAD=
bezeichnet ein Feld, das die Startadresse des Contingency-Prozesses enthält. Das Feld ist auf Wortgrenze auszurichten.

adr
symbolische Adresse (Name) eines Feldes, das die Startadresse enthält

(r)
r = Register mit der Adresse adr

COIDRET=
bezeichnet die Adresse eines Feldes, in das eine Kurzkennung übergeben wird. Die Kurzkennung ist in weiteren Makroaufrufen, die sich auf den Contingency-Prozess beziehen, zu verwenden. Das Feld ist auf Wortgrenze auszurichten.

adr
symbolische Adresse (Name) des Feldes für die Kurzkennung

(r)
r = Register mit dem Adresswert des Feldes

COMAD=
bezeichnet ein Feld, das eine Contingency-Mitteilung enthält. Die Mitteilung wird in das Register R1 des Contingency-Prozesses übertragen. Eine hier gegebene Mitteilung kann durch eine eventuell beim **SOLSIG**- oder **POSSIG**-Makroaufruf gegebene ersetzt werden.

adr
symbolische Adresse (Name) des Feldes mit der Mitteilung

(r)
r = Register mit dem Adresswert adr

LEVEL=
benennt die Priorität (Verarbeitungsebene) des Contingency-Prozesses.

prio
Priorität des Contingency-Prozesses; 1 ≤ prio ≤ 127
Voreinstellung: prio = 1.

(r)
r = Register mit dem Wert für prio.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

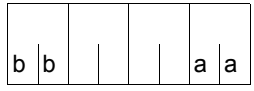
24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB).

Rückinformation und Fehleranzeigen

Während der Makrobearbeitung enthält Register R1 die Adresse der Operandenliste.

R15:



Über die Ausführung des Makros ENACO wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'bb'	X'aa'	Erläuterung
X'04'	X'00'	Der Contingency-Prozess wurde für den aufrufenden Prozess definiert.
X'0C'	X'04'	Der Contingency-Prozess war bereits für den aufrufenden Prozess definiert. Keine Aktion.
X'10'	X'04'	Es wurden ungültige Operanden angegeben. Keine Aktion.
X'18'	X'04'	Die maximal erlaubte Anzahl von gleichzeitig verwendeten Contingency-Prozessen wurde überschritten. Keine Aktion.

Beispiele enthalten der Abschnitt „Contingency-Prozesse“ (siehe Seite 111) und die Beschreibung des Makros **POSSIG** (siehe Seite 754).

ENAEI – Ereignisgesteuerte Verarbeitung eröffnen

Allgemeines

Anwendungsgebiet: Ereignissteuerung; siehe Seite 95
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

Makrobeschreibung

Der Makro **ENAEI** richtet für die Task des Aufrufers eine Ereigniskennung ein. Wenn eine Ereigniskennung mit den angegebenen Namen in dem definierten Geltungsbereich bereits besteht (eingerichtet durch einen **ENAEI**-Makroaufruf in einer anderen Task), bewirkt der Aufruf nur eine Zuordnung zwischen der Task des Aufrufers und der Ereigniskennung. Andernfalls wird die Ereigniskennung vom System erstellt und der Task zugeordnet.

Durch diesen Aufruf wird außerdem eine Kurzkennung für die Ereigniskennung zur Verfügung gestellt, die in weiteren Aufrufen zur Beschleunigung der Verarbeitung verwendet werden kann.
In einem Programm können maximal 2000 Ereigniskennungen gleichzeitig verwendet werden.

Makroaufrufformat und Operandenbeschreibung

```
ENAEI

{ EINAME=name                          }
{ EINAMAD=adr / (r) [,EINAMLN=länge]   },SCOPE=LOCAL / GROUP / USER_GROUP / GLOBAL

,EIIDRET=adr / (r)

,SOSIGQ=FIFO / LIFO

[,PARMOD=24 / 31]

[,MF=L / (E,..)]
```

EINAME=
bezeichnet den Namen der Ereigniskennung.

name
Name der Ereigniskennung. 1 ≤ Namenslänge ≤ 54

```
Namensbildung:
1. Zeichen:     Buchstabe, #, @
2.-54. Zeichen: beliebige Kombination aus der Zeichenmenge
                (A,...,Z,0,...,9,$,#,@).
Das erste Blank (X'40') beendet den Namen.
```

EINAMAD=
bezeichnet die Adresse des Felds mit dem Namen der Ereigniskennung (Namensbildung siehe oben).

adr
symbolische Adresse (Name) des Feldes

(r)
r = Register mit dem Adresswert adr

EINAMLN=
beschreibt die Länge des Namens der Ereigniskennung, wenn EINAMAD=... spezifiziert wurde.

länge
Länge des Namens in Byte

Voreinstellung:
– Längenattribut des bei EINAMAD angegebenen Feldes.
– 54 Byte, wenn EINAMAD=(r) spezifiziert wurde.

SCOPE=
beschreibt den Geltungsbereich (Teilnehmerkreis) für die Ereigniskennung.

LOCAL
Die Ereigniskennung wird nur von der Task des Aufrufers benutzt.

GROUP
Teilnehmer sind alle Tasks unter der Benutzerkennung des Aufrufers.

USER_GROUP
Teilnehmer können alle Tasks sein, deren Benutzerkennungen der gleichen Benutzergruppe angehören wie die Benutzerkennung des einrichtenden Teilnehmers.

Der Operandenwert setzt die Existenz von Benutzergruppen voraus und kann daher nur angegeben werden, wenn die Funktionseinheit SRPM des Software-Produkts SECOS im System vorhanden ist. Vor einem Makroaufruf mit SCOPE=USER_GROUP

muss deshalb mit dem Makro GETUGR (siehe Handbuch „SECOS“ [14]) geprüft werden, ob SRPM zur Verfügung steht; abhängig vom Ergebnis (Returncode) ist im Programm zu reagieren.

GLOBAL
Teilnehmer sind alle Tasks im System.

EIIDRET=
bezeichnet ein Feld, in das die Kurzkennung für die Ereigniskennung übergeben werden soll.

adr
symbolische Adresse (Name) des Feldes für die Kurzkennung;
Feldlänge = 4 Byte; das Feld ist auf Wortgrenze auszurichten.

(r)
r = Register mit dem Adresswert adr.

SOLSIGQ=
bezeichnet das Warteschlangen-Prinzip für die Einreihung der SOLSIG-Anforderungen. Alle **ENAEI**-Makros unter einer bestimmten Ereigniskennung müssen dasselbe Warteschlangen-Prinzip für die SOLSIG-Anforderungen vereinbaren.

FIFO
FIFO = First In First Out.

LIFO
LIFO = Last In First Out.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

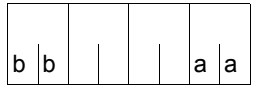
24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB).

Rückinformation und Fehleranzeigen

Während der Makrobearbeitung enthält Register R1 die Adresse der Operandenliste.

R15:



Über die Ausführung des Makros ENAEI wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'bb'	X'aa'	Erläuterung
X'04'	X'00'	Funktion ausgeführt: Die Ereigniskennung wurde vom System erstellt und der Task des Aufrufers zugeordnet.
X'08'	X'00'	Funktion ausgeführt: Die vom System bereits erstellte Ereigniskennung wurde der Task des Aufrufers zugeordnet.
X'0C'	X'04'	Keine Aktion: Die Ereigniskennung war bereits der Task des Aufrufers zugeordnet. Die Kurzkennung der Ereigniskennung wird übergeben.
X'10'	X'04'	Keine Aktion: Es wurden ungültige Operanden angegeben.
X'18'	X'04'	Keine Aktion: Die maximal erlaubte Anzahl von gleichzeitig verwendeten Ereigniskennungen wurde überschritten.
X'1C'	X'04'	Keine Aktion: Für die Ereigniskennung wurden unterschiedliche Vereinbarungen (FIFO und LIFO) bei der SOLSIG-Warteschlange behandlung getroffen (SOSLIGQ-Operand).

Beispiele enthalten der Abschnitt „Ereignisgesteuerte Verarbeitung (Eventing)“ (Seite 107), der Abschnitt „Contingency-Prozesse“ (Seite 119) sowie die Beschreibung der Makros **POSSIG** (Seite 760) und **SOLSIG** (Seite 850).

ENAMP – Memory Pool eröffnen

Allgemeines

Anwendungsgebiet: Memory Pool Technik; siehe Seite 55
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

Ein Memory Pool (MP) ist ein Speicherbereich im Klasse-6-Speicher, der von mehreren Anwendern gemeinsam benutzt werden kann. Seine Größe (Lage), Bezeichnung (Name) und Speicherattribute werden von dem Anwender festgelegt, der den Memory Pool einrichtet.

Es können sowohl Memory Pools aus 64KB-Einheiten als auch aus 1MB-Einheiten angelegt werden. Memory Pools aus 64KB-Einheiten werden immer unterhalb der 16MB-Grenze angelegt und langfristig nicht unterstützt. Aus diesem Grund und wegen möglicher Performanceeinbußen sollten Memory Pools aus 64KB-Einheiten nicht mehr angelegt werden. Die Größe des Memory Pools und die Belegung der Speicherseiten können mit dem Makro **MINF** abgefragt werden.

Makrobeschreibung

Mit dem Makro **ENAMP** kann ein Anwender einen Memory Pool einrichten oder seine Teilnahme an einem existierenden Memory Pool erklären. Dem Aufrufer wird vom System eine Kurzkennung für den Memory Pool zurückgegeben. Die Kurzkennung kann zur Beschleunigung der Verarbeitung in weiteren Pool-Aufrufen (**CSTMP, REQMP, RELMP, DISMP**) verwendet werden; sie kann für verschiedene Pool-Teilnehmer verschieden sein.
Für einen neu einzurichtenden Memory Pool legt der Aufrufer mit **ENAMP** folgende Poolattribute unveränderlich fest:

- Name des Memory Pools
- Geltungsbereich (Teilnehmerkreis: nur der Aufrufer, alle Tasks unter der Benutzerkennung des Aufrufers, alle Tasks aus der Benutzergruppe des Aufrufers, alle Tasks im System)
- Größe
- Anfangsadresse (einheitlich oder beliebig für alle Pool-Teilnehmer)
- Lage (unterhalb der 16 MB-Grenze oder beliebig für alle Pool-Teilnehmer)
- Seitenverwaltung (resident oder pageable (seitenwechselbar))

Mit **ENAMP** wird der angegebene Speicherplatz im Adressraum des Aufrufers reserviert. Die Anforderung der Seiten erfolgt mit **REQMP**.

Für die Teilnahme an einem existierenden Memory Pool gilt:

- Ein Memory Pool kann nur über seinen Namen in Verbindung mit dem Geltungsbereich (Operand SCOPE) eindeutig identifiziert werden (in nachfolgenden Aufrufen genügt die Kurzkennung).
- Der Teilnehmer darf keine abweichenden Poolattribute (siehe oben) spezifizieren (am besten ist es, die Voreinstellung zu benutzen).
- Jeder Teilnehmer kann mit **REQMP** Speicherplatz (innerhalb des Memory Pools) anfordern und mit **RELMP** Speicherplatz freigeben.
- Jeder Teilnehmer mit der Berechtigung CSTMP=YES im Benutzerkatalog kann für die Speicherseiten einen Zugriffsschutz vereinbaren oder aufheben (**CSTMP**).
- Jeder Teilnehmer kann mit **DISMP** seine Teilnahme an einem Memory Pool beenden.

Hinweise

- **WRCPT**-Makro, Kommandos HOLD-TASK und RESTART-PROGRAM werden abgewiesen, wenn eine Task Teilnehmer an einem Memory-Pool ist.
- dynamisch versorgte Datenbereiche und Ein-/Ausgabebereiche sollten nicht in Memory Pools abgelegt werden (sonst ist unübersichtliche Synchronisation erforderlich).

Makroaufrufformat und Operandenbeschreibung

```
ENAMP

{ MPNAME=name                                  },SCOPE=LOCAL / GROUP / USER_GROUP / GLOBAL
{ MPNAMAD=adr / (r) [,MPNAMLN=länge / (r)]     }

[,MPIDRET=adr / (r)]

,MODE=ANY / NEW / OLD

[, { BSIZE=größe / (r) } ]
   { PSIZE=größe / (r) }

[,LOC=BELOW]

[,PAGE=adr / (r)]

[,FIXED=YES]

,RES=NO / YES

,INHERIT=YES / NO

[,PARMOD=24 / 31]

[,MF=L / (E,..)]
```

MPNAME=
bezeichnet den Namen des Memory Pools (Verbindung mit Operand SCOPE beachten).

name
Name des Memory Pools; 1 ≤ Namenslänge ≤ 54 Zeichen

```
Namensbildung:
1. Zeichen:     Buchstabe oder Sonderzeichen #,@.
2.–54. Zeichen: beliebige Kombination aus der Zeichenmenge
                (A,...,Z,0,...,9,$,#,@).
Das erste Blank (X'40') beendet den Namen.
```

MPNAMAD=
beschreibt die Adresse des Feldes mit name (Verbindung mit Operand SCOPE beachten).

adr
symbolische Adresse (Name) des Feldes

(r)
r = Register mit dem Adresswert des Feldes

MPNAMLN=
beschreibt die Länge des unter MPNAMAD angegebenen Namens. Wenn nicht spezifiziert: Längenattribut des Feldes adr bzw. 54 Byte, falls MPNAMAD=(r) angegeben wurde.

länge
Länge in Byte

(r)
r = Register, das länge enthält

SCOPE=
beschreibt den Geltungsbereich (Teilnehmerkreis) des Memory Pools.
Die Angabe dient der eindeutigen Bezeichnung des Memory Pools und muss in Verbindung mit den Operanden MPNAME und MPNAMAD spezifiziert werden. Voreinstellung beachten!

LOCAL
Der Memory Pool wird nur von dem einrichtenden Teilnehmer benutzt.

GROUP
Teilnehmer können alle Tasks mit der Benutzerkennung des einrichtenden Teilnehmers sein.

USER_GROUP
Teilnehmer können alle Tasks sein, deren Benutzerkennungen der gleichen Benutzergruppe angehören wie die Benutzerkennung des einrichtenden Teilnehmers.
Der Operandenwert setzt die Existenz von Benutzergruppen voraus und kann daher nur angegeben werden, wenn die Funktionseinheit SRPM des Software-Produkts SECOS im System vorhanden ist. Vor einem Makroaufruf mit SCOPE=USER_GROUP

muss deshalb mit dem Makro GETUGR (siehe Handbuch „SECOS“ [14]) geprüft werden, ob SRPM zur Verfügung steht; abhängig vom Ergebnis (Returncode) ist im Programm zu reagieren.

GLOBAL
Teilnehmer können alle im System laufenden Tasks sein.

MPIDRET=
bezeichnet eine Kurzkennung für den Memory Pool. Die Kurzkennung wird dem Anwender vom System nach der Makroausführung zurückgegeben und kann in nachfolgenden Aufrufen (**REQMP, RELMP, DISMP, CSTMP**) zur eindeutigen Bezeichnung des Memory Pools verwendet werden. Die Verwendung der Kurzkennung beschleunigt die Verarbeitung. Jeder Memory Pool-Teilnehmer erhält durch **ENAMP** eine eigene Kurzkennung. Eine Ausnahme sind sog. vererbbare Memory Pools, bei denen die „Sohntask“ dieselbe Kurzkennung nutzen kann wie die „Vatertask“.

adr
symbolische Adresse des 4 Byte-Feldes für die Kurzkennung.

(r)
r = Register mit dem Adresswert des Feldes.

MODE=
gibt an, ob der Aufrufer einen Memory Pool neu einrichten oder sich an einen schon bestehenden anschließen will. Im letzteren Fall ist zu beachten, dass ein Memory Pool nur über den Namen und seinen Geltungsbereich (Operand SCOPE) eindeutig identifiziert werden kann.

ANY
Der Aufrufer will sich dem genannten Memory Pool anschließen, falls dieser existiert; wenn nicht, wird ein Memory Pool mit den spezifizierten Attributen für den Aufrufer eingerichtet.

NEW
Der Aufrufer will einen neuen Memory Pool mit den spezifizierten Attributen einrichten. Der Aufruf wird abgewiesen, wenn bereits ein Memory Pool mit dem genannten Namen und Geltungsbereich (SCOPE) existiert.

OLD
Der Aufrufer will sich einem schon bestehenden Memory Pool anschließen. Der Aufruf wird abgewiesen, wenn
- der Memory Pool nicht existiert,
- die spezifizierten Poolattribute nicht mit denen übereinstimmen, die beim Einrichten des Pools verbindlich festgelegt wurden,
- SCOPE=LOCAL angegeben wurde.

BSIZE=
beschreibt die Größe des Memory Pools in 4KB-Einheiten (=Speicherseiten). Der Bereich wird zusammenhängend und je nach Angabe für PAGE bzw. LOC und in Abhängigkeit vom Adressierungsmodus unterhalb bzw. oberhalb der 16MB-Grenze angelegt.
BSIZE kann nicht angegeben werden, wenn beim Einrichten des Memory Pools der Operand PSIZE spezifiziert wurde.

größe
Anzahl der Speicherseiten (4KB); größe = 0 ist unzulässig; der Aufruf wird abgewiesen. Aus Performance-Gründen kann die Größe des Memory Pools vom Betriebssystem aufgerundet werden (s. Hinweis).

(r)
r = Register mit der Anzahl der Speicherseiten
Voreinstellung:
– Ein (neuer) Memory Pool wird aus Kompatibilitätsgründen mit der Voreinstellung für den Operanden PSIZE eingerichtet, wenn weder BSIZE noch PSIZE (explizit) spezifiziert werden.
– aktueller Wert für bestehenden Memory Pool.

Hinweise
– Aus Performance-Gründen kann die Größe eines MP wie folgt aufgerundet werden: Der Memory Pool wird in 1MB-Einheiten angelegt und auf 1MB-Grenze ausgerichtet. Die Aufrundung erfolgt so, dass ein Vielfaches (n) von 1MB-Einheiten erreicht wird (Größe MP = n * 1MB ≥ größe * 4KB).
– Die Aufrundung ist nur in der BS2000-Version 9.0 garantierter Bestandteil der Funktionsausführung.

PSIZE=
beschreibt die Größe des Memory Pools in 64KB-Einheiten. Der Bereich wird unterhalb der 16MB-Grenze zusammenhängend angelegt und auf 64KB-Grenze ausgerichtet.
PSIZE kann nicht angegeben werden, wenn beim Einrichten des Memory Pools der Operand BSIZE spezifiziert wurde.

größe
Anzahl der Speichereinheiten mit je 64 KB; größe = 0 ist unzulässig; der Aufruf wird abgewiesen.

(r)
r = Register mit dem Wert für größe
Voreinstellung (nur, wenn beim Einrichten des MP der Operand BSIZE nicht spezifiziert wurde):
– größe = 1 für neu einzurichtenden Memory Pool,
– aktueller Wert für bestehenden Memory Pool.

Hinweis

Der Operand PSIZE und Memory Pools mit 64KB-Einheiten werden langfristig nicht mehr unterstützt.

LOC=
bezeichnet den Teil des Adressraums, in den der Memory Pool platziert werden soll. Die Angabe ist nur sinnvoll bei 31-Bit-Adressierung und in Verbindung mit dem Operanden BSIZE.

BELOW
Der Memory Pool wird unterhalb der 16MB-Grenze im Adressraum des Aufrufers platziert.

Hinweis

Ein Memory Pool, der mit LOC=BELOW und nicht verbindlicher Anfangsadresse eingerichtet wurde, kann für einen anderen MP-Teilnehmer auch in den Bereich oberhalb der 16MB-Grenze seines Adressraums platziert werden.

PAGE=
gibt die Anfangsadresse des Memory Pools im Adressraum des Aufrufers an. Die Anfangsadresse ist wie folgt auszurichten:

– Ausrichtung auf 64KB-Grenze, wenn der Memory Pool aus 64KB-Einheiten besteht (Operand PSIZE) bzw.
– Ausrichtung auf 1MB-Grenze, wenn der Memory Pool aus 1MB-Einheiten besteht (Operand BSIZE).

Voreinstellung:
Der erste ausreichend große und zusammenhängende Bereich, auf 64KB- bzw. 1MB-Grenze beginnend, wird ausgewählt.
Im 31-Bit-Adressierungsmodus wird der MP oberhalb der 16MB-Grenze platziert, wenn nicht LOC=BELOW angegeben wurde. Wurde der Memory Pool mit FIXED=YES eingerichtet, gilt für alle weiteren Teilnehmer die verbindlich festgelegte Anfangsadresse. Voreinstellung in diesem Fall: die Anfangsadresse wird übernommen (der Memory Pool wird im Adressraum des Aufrufers so platziert, wie im Adressraum des Aufrufers, der den Memory Pool eingerichtet hat). Die Anfangsadresse wird dem Aufrufer im Register R1 übergeben.

adr
Anfangsadresse.

(r)
r = Register mit dem Adresswert adr

Hinweis

Der Operand PAGE sollte nach Möglichkeit nicht benutzt werden. Ist seine Verwendung jedoch erforderlich, sollte vorher Größe und Lage des Klasse-6-Speichers mit dem Makro **MINF** abgefragt werden.

FIXED=YES
Der Memory Pool hat für alle Teilnehmer dieselbe virtuelle Anfangsadresse (wird ab dieser Adresse im Adressraum des Aufrufers platziert).

Hinweise

- Wenn FIXED=YES nicht angegeben wird (vom ersten Aufrufer), dann darf jeder weitere Teilnehmer eine andere Anfangsadresse in seinem Adressraum angeben.
- Ein Aufrufer, der sich einem Memory Pool anschließen will, kann die Angabe FIXED=YES nicht rückgängig machen.
- Die Angabe des Operanden ist nur für den Aufrufer sinnvoll, der den Memory Pool einrichtet. Bei einem Pool-Teilnehmer wird FIXED=YES zurückgewiesen, wenn der Operand nicht schon beim Einrichten des Memory Pools spezifiziert wurde.

RES=
gibt an, ob die Speicherseiten des Pools seitenwechselbar oder resident sein sollen. Diese Eigenschaft wird von dem einrichtenden Aufrufer bestimmt (die maximale Anzahl residenter Speicherseiten ist im Benutzerkatalog festgelegt).

NO
die Speicherseiten sollen seitenwechselbar sein.

YES
die Speicherseiten sollen resident sein.

Hinweis

Bei RES=YES wird die Angabe für den Operanden PSIZE oder BSIZE nicht gegen die Angabe für den Operanden RESIDENT-PAGES im START-PROGRAM- oder LOAD-PROGRAM-Kommando geprüft. Die Prüfung erfolgt erst, wenn mit **REQMP** Speicherplatz belegt wird.

INHERIT=
spezifiziert die Vererbbarkeit eines Memory Pools, wenn die teilnehmende Anwendertask („Vatertask“) eine neue Task („Sohntask“) erzeugt: „fork()“. Dieser Operand wird nicht unterstützt, wenn ein lokaler und residenter Klasse-6-Memory Pool angegeben oder PARMOD=24 gesetzt wurde. Siehe auch Hinweise auf Seite 478.

YES
Der Memory Pool soll vererbbar sein, d.h. im Fall eines fork() ist die Sohntask implizit an einen nicht-lokalen Memory Pool der Vatertask angeschlossen.

Für einen lokalen (nicht-residenten) Memory Pool wird der Speicherplatz von der Vatertask in die Sohntask kopiert. Das bedeutet, dass der Speicherplatz für jede Task lokal zur Verfügung steht und die Task über den Inhalt des Memory Pools und die auszuführenden Funktionen frei entscheiden kann.

NO
Im Fall eines fork() ist die Sohntask nicht an den Memory Pool der Vatertask angeschlossen. Der Speicherbereich ist in der Sohntask nicht zugewiesen.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB) und beginnen mit dem Standardheader.

Hinweise zur Vererbbarkeit von Memory Pools

– Für vererbbare, *lokale* Memory Pools wird für die „Sohntask“ (nach der Taskerzeugung durch die Vatertask: „fork()“) implizit ein neuer Memory Pool angelegt. Er hat die gleichen Eigenschaften (Name, Zugriffsrechte, usw.) und den gleichen Inhalt wie der Memory Pool der Vatertask zum Zeitpunkt der Taskerzeugung der Sohntask (realisiert durch den Copy-On-Write-Mechanismus). Das ermöglicht, dass sowohl Vater- als auch Sohntask auf einer gemeinsamen Basis auf den Inhalt des Memory Pools zugreifen können, ihre jeweils eingebrachten Änderungen aber nur lokal im „eigenen“ Memory Pool gelten.

– Für vererbbare, *nicht-lokale* Memory Pools geschieht die Vererbung durch einen impliziten **ENAMP**-Aufruf zum bestehenden Memory Pool während des fork(). Das bedeutet, dass Vater- und Sohntask im selben Speicherbereich arbeiten.

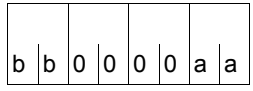
– Die Kurzkennung des Memory Pools (Operand MPIDRET) wird von der Vatertask auf die Sohntask vererbt.

– Nicht vererbbar dagegen sind das Kontingent zur Anforderung von residenten Speicherseiten und - im Falle eines lokalen Memory Pools - die mit **CSTAT** PAGE=NO priorisierten, residenten Speicherseiten.

Rückinformation und Fehleranzeigen

Nach Funktionsausführung wird die virtuelle Anfangsadresse des Memory Pools im Register R1 abgespeichert.

R15:



Über die Ausführung des Makros ENAMP wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'bb'	X'aa'	Erläuterung
X'04'	X'00'	Normale Ausführung. Ein neuer Memory Pool wurde eingerichtet (MODE=NEW/ANY).
X'08'	X'00'	Normale Ausführung. Aufrufer ist neuer Teilnehmer des angegebenen Memory Pools (MODE=OLD/ANY).
X'04'	X'04'	Funktion nicht ausgeführt. Memory Pool nicht vorhanden (MODE=OLD).
X'08'	X'04'	Funktion nicht ausgeführt. Aufruf bezieht sich auf einen existierenden Memory Pool. – Aufrufer ist schon Teilnehmer des Memory Pools (MODE=OLD/ANY). Register R1 enthält die Anfangsadresse und MPIDRET die ID des MPs. – Memory Pool existiert schon (MODE=NEW). – abweichende Poolattribute angegeben (MODE=OLD/ANY): – PSIZE/BSIZE – PAGE (bei „fixed“ MP: die Anfangsadresse des MPs wird in Register R1 zurückgegeben) – LOC (bei „fixed“ MP) – FIXED – RES
X'14'	X'04'	Funktion nicht ausgeführt. Nicht ausreichend freier Platz – im Adressraum des Aufrufers – im Adressraum unterhalb der 16MB-Grenze (in Verbindung mit LOC=BELOW oder PSIZE=....).
X'18'	X'04'	Funktion nicht ausgeführt. Ungültige Speicheradresse: – Anfangsadresse oder eine Adresse des angegebenen Bereichs liegt außerhalb des Adressraums des Aufrufers. – Anfangsadresse oder eine Adresse des angegebenen Bereichs zeigt auf die 16MB-Grenze oder darüber und LOC=BELOW oder PSIZE=... wurde angegeben. – Anfangsadresse nicht auf 64KB/1MB-Grenze ausgerichtet. – der angegebene Adressraum ist nicht durchgehend frei.

X'bb'	X'aa'	Erläuterung
X'1C'	X'04'	Funktion nicht ausgeführt. Operandenfehler: – unzulässige Adresse des Datenbereichs – fehlerhafter Aufbau des Datenbereichs – unzulässige Adressen für MPNAMAD oder MPIDRET im Datenbereich – Benennung des Memory Pools: – Name enthält unzulässige Zeichen – ungültige Längenangabe (MPNAMLN) – Name nicht angegeben (MPNAME, MPNAMAD nicht spezifiziert) – MPNAME und MPNAMAD spezifiziert – MPNAMLN angegeben, aber MPNAMAD nicht spezifiziert – SCOPE angegeben, aber MPNAME/MPNAMAD nicht spezifiziert. – ungültige Angaben bei SCOPE/MODE/BSIZE/PSIZE/LOC/FIXED/RES – der MP-Name und das Feld, in das die Kurzkennung übergeben wird, überlappen sich. – ungültiges Register (R1) angegeben. – PARMOD=24 in Verbindung mit 31-Bit-Adressierungsmodus (AMODE 31) angegeben. – SCOPE=USER_GROUP wurde angegeben, obwohl SRPM nicht im System vorhanden ist. – Bei einem ENAMP-Aufruf für einen bereits existierenden Memory-Pool stimmt der aktuelle Zugriffsschlüssel nicht mit dem überein, der bei der Einrichtung des Pools gültig war, weil eine der beteiligten Benutzerkennungen immer noch das obsolete Privileg SECURE-OLTP besitzt. – Es wurde INHERIT=YES angegeben, obwohl der betroffene Memory Pool lokal und resident ist oder PARMOD=24 angegeben wurde.
X'20'	X'04'	Funktion nicht ausgeführt. Aufruf kann infolge Speichersättigung momentan nicht ausgeführt werden. Ein späterer Aufruf kann erfolgreich sein.

31-Bit-Schnittstelle:

– Bei fehlerhafter Ausrichtung oder Initialisierung des Standardheaders werden im Register R15 zusätzlich die Returncodes X'0001FFFF' / X'0003FFFF' / X'0004FFFF' übergeben; siehe Tabelle „Standard-Returncodes“ auf Seite 43.
– Im Standardheader wird kein Returncode übergeben.

ENASI – Serialisierungskennung zuordnen

Allgemeines

Anwendungsgebiet: (Task-)Serialisierung; siehe Seite 92
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

ENASI generiert je nach Angabe die 24-Bit- oder die 31-Bit-Schnittstelle. Bei Makrokettung müssen alle Makros der Kette dieselbe Schnittstelle (24-Bit oder 31-Bit-Schnittstelle) benutzen.

Makrobeschreibung

Der Makro **ENASI** richtet für die Task des Aufrufers eine Serialisierungskennung ein. Wenn eine Serialisierungskennung mit dem angegebenen Namen in dem definierten Geltungsbereich bereits besteht (eingerichtet durch einen **ENASI**-Makroaufruf in einer anderen Task), bewirkt der Makroaufruf nur eine Zuordnung zwischen der Task des Aufrufers und der Serialisierungskennung. Andernfalls wird die Serialisierungskennung vom System erstellt und der Task des Aufrufers zugeordnet.
Durch diesen Makroaufruf wird außerdem eine Kurzkennung für die Serialisierungskennung zur Verfügung gestellt, die in weiteren Aufrufen zur Beschleunigung der Verarbeitung verwendet werden kann.

Ein Programm kann maximal 2000 Serialisierungskennungen gleichzeitig verwenden.
Mit dem CONTINU-Operanden können bis zu 255 **ENASI**-Aufrufe gekettet werden.

Makroaufrufformat und Operandenbeschreibung

```
ENASI

{ SINAME=name                          }
{ SINAMAD=adr / (r) [,SINAMLN=länge]   },SCOPE=LOCAL / GROUP / USER_GROUP / GLOBAL

,SIIDRET=adr / (r)

,CONTINU=NO / YES

[,PARMOD=24 / 31]

[,MF=L / (E,..)]
```

SINAME=name
gibt den Namen der Serialisierungskennung an. Dieser Name ist eine 1 bis 54 Byte lange Zeichenfolge. Der erste auftretende Zwischenraum (X'40') beendet den Namen. Als Zeichen können alle Buchstaben und Ziffern sowie die Sonderzeichen $, # und @ verwendet werden. Ziffern und das Zeichen $ sind als erstes Zeichen nicht erlaubt.

SINAMAD=
bezeichnet den Namen der Serialisierungskennung. Die Regeln für den Aufbau des Namens sind bei dem Operanden SINAME beschrieben.

adr
symbolische Adresse des Feldes, das den Namen enthält

(r)
r = Register, das die Adresse enthält

SINAMLN=
gibt die Länge des Namens der Serialisierungskennung in Byte an. Die Länge muss mindestens 1 Byte sein und darf 54 Byte nicht überschreiten.
Fehlt der Operand, so wird das Längenattribut des SINAMAD-Operanden angenommen, wenn SINAMAD=adr angegeben ist; bei SINAMAD=(r) wird die maximale Länge (54) angenommen.

länge
Länge des Namens der Serialisierungskennung.

SCOPE=
beschreibt den Geltungsbereich (Teilnehmerkreis) für die Serialisierungskennung.

LOCAL
Die Serialisierungskennung wird nur von der Task des Aufrufers benutzt.

GROUP
Teilnehmer sind alle Tasks unter der Benutzerkennung des Aufrufers.

USER_GROUP
Teilnehmer können alle Tasks sein, deren Benutzerkennungen der gleichen Benutzergruppe angehören wie die Benutzerkennung des einrichtenden Teilnehmers.

Der Operandenwert setzt die Existenz von Benutzergruppen voraus und kann daher nur angegeben werden, wenn die Funktionseinheit SRPM des Software-Produkts SECOS im System vorhanden ist. Vor einem Makroaufruf mit SCOPE=USER_GROUP muss deshalb mit dem Makro GETUGR (siehe Handbuch „SECOS“ [14]) geprüft werden, ob SRPM zur Verfügung steht; abhängig vom Ergebnis (Returncode) ist im Programm zu reagieren.

GLOBAL
Teilnehmer sind alle Tasks im System.

SIIDRET=
bezeichnet die Kurzkennung der Serialisierungskennung. Diese Kurzkennung kann in weiteren Makroaufrufen (**ENQAR**, **DEQAR**, **CHKSI** und **DISSI**), die sich auf die angegebene Serialisierungskennung beziehen, zur Beschleunigung der Verarbeitung verwendet werden.

adr
symbolische Adresse (Name) eines 4 Byte langen Feldes, in dem die Kurzkennung dem aufrufenden Programm übergeben wird

(r)
r = Register mit dem Adresswert adr

CONTINU=
Durch diesen Operanden kann eine Kettung von bis zu 255 **ENASI**-Aufrufen erfolgen.

NO
Dies ist der letzte (bzw. einzige) Aufruf einer Folge.

YES
Unmittelbar nach diesem **ENASI**-Aufruf folgt ein weiterer.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

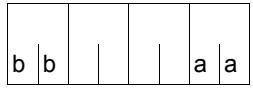
Hinweise zum Makroaufruf

- Die dem aufrufenden Programm übergebene Kurzkennung für die Serialisierungskennung kann in weiteren Aufrufen an Stelle des Namens verwendet werden, um die Verarbeitung zu beschleunigen.
- Ein explizites Enable für eine Serialisierungskennung ist nur erforderlich, wenn der Benutzer die beschleunigte Verarbeitung wünscht, die durch die Verwendung der Kurzkennung ermöglicht wird. Ist dies nicht der Fall, kann der Benutzer von der impliziten Enable-Funktion Gebrauch machen (s. **ENQAR**-/**DEQAR**-Makroaufrufe).
- Eine Task kann eine Serialisierungskennung nur verwenden, wenn ein zugehöriges Enable (explizit oder implizit) für diese Task verarbeitet wurde.
- Jeder zweite Enable-Aufruf (explizit oder implizit) in einem Programm für eine Serialisierungskennung, die bereits vorher in einem expliziten Enable angegeben und deren Verwendung wieder beendet wurde (siehe **DISSI**-Aufruf), bekommt nicht notwendigerweise dieselbe Kurzkennung wie im ersten Enable-Aufruf.
- Ist derselbe Name für eine Serialisierungskennung in zwei Enable-Aufrufen mit zwei verschiedenen Geltungsbereichen angegeben, werden zwei unterscheidbare Serialisierungskennungen vom System verarbeitet.
- Der Geltungsbereich einer Kurzkennung ist derselbe wie der Geltungsbereich des zugeordneten Namens.
- Bei der Verwendung von MF=L ist zu beachten:
  Zur Ausführung muss nur ein Makroaufruf mit MF=E gegeben werden, unabhängig davon, ob dieser Aufruf für eine Anforderung oder für eine Folge von Anforderungen gilt. Der Datenbereich bei einer Folge von Anforderungen wird durch Kettung der Makroaufrufe (MF=L) durch den CONTINU-Operand erzeugt.

Rückinformation und Fehleranzeigen

Register R1 enthält die Adresse des Datenbereichs.

R15:



Über die Ausführung des Makros ENASI wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'bb'	**X'aa'**	**Erläuterung**
X'04'	X'00'	Alle Enable-Aufrufe wurden ausgeführt: Mindestens eine Serialisierungskennung wurde eingerichtet.
X'08'	X'00'	Alle Enable-Aufrufe wurden ausgeführt: Die Verwendung mindestens einer Serialisierungskennung wurde ermöglicht.
X'0C'	X'04'	Nicht alle Enable-Aufrufe wurden ausgeführt: Mindestens eine Serialisierungkennung wurde von der Task des Aufrufers bereits verwendet.
X'10'	X'04'	Nicht alle Enable-Aufrufe wurden ausgeführt: Es wurden ungültige Operanden angegeben – ungültige Adresse – ungültige Länge – ungültiger Name – nichtdefinierter Geltungsbereich oder CONTINU-Wert
X'18'	X'04'	Nicht alle Enable-Aufrufe wurden ausgeführt: Die maximal erlaubte Anzahl von gleichzeitig verwendeten Serialisierungskennungen wurde überschritten.

ENQAR – Belegung einer Serialisierungskennung anfordern

Allgemeines

Anwendungsgebiet: (Task-)Serialisierung; siehe Seite 92
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

ENQAR generiert je nach Angabe die 24-Bit- oder die 31-Bit-Schnittstelle. Bei Makroverkettung müssen alle Makros der Kette dieselbe Schnittstelle benutzen.

Makrobeschreibung

Dieser Makro fordert den Zugriff zu der angegebenen Serialisierungskennung an. Die Zugriffsanforderung wird in die Warteschlange dieser Serialisierungskennung eingetragen und das Programm solange in den Wartezustand versetzt, bis es das erste Programm in dieser Warteschlange ist. Dann läuft das Programm weiter und belegt die Serialisierungskennung solange, bis ein **DEQAR**-Makroaufruf für diese Serialisierungskennung gegeben wird. Ist keine Serialisierungskennung mit dem angegebenen Namen im definierten Geltungsbereich vorhanden, so wird sie eingerichtet und eine implizite Enable-Funktion (siehe **ENASI**-Makroaufruf) ausgeführt. Durch den COND-Operanden bestimmt der Benutzer, ob die Zugriffsanforderung auf jeden Fall bearbeitet wird oder nur dann, wenn sie sofort befriedigt werden kann. Mit dem LIFETIM-Operanden kann die Zeit begrenzt werden, die ein Programm auf die Erfüllung einer Zugriffsanforderung wartet.
Durch den CONTINU-Operanden können bis zu 255 **ENQAR**-Aufrufe gekettet werden. Eine solche Folge von Anforderungen wird erst dann verarbeitet, wenn gleichzeitig alle Einzelanforderungen erfüllt werden können.

Makroaufrufformat und Operandenbeschreibung

```
ENQAR

{ { SINAME=name                                  }                                                  }
{ { SINAMAD=adr / (r) [,SINAMLN=länge] },SCOPE=LOCAL / GROUP / USER_GROUP / GLOBAL                  }
{ SIID=adr / (r)                                                                                    }

,CONTINU=NO / YES
,COND=UNCOND / IMMED
[,LIFETIM=sec / (r)]
[,PARMOD=24 / 31]
[,MF=L / (E,..)]
```

SINAME=name
gibt den Namen der Serialisierungskennung an. Diese Angabe ist nur zusammen mit dem Geltungsbereich (Operand SCOPE) eindeutig.

SINAMAD=
bezeichnet den Namen der Serialisierungskennung. Dieser Name ist nur zusammen mit dem Geltungsbereich (Operand SCOPE) eindeutig.

adr
symbolische Adresse des Feldes, das den Namen enthält

(r)
r = Register, das die Adresse enthält

SINAMLN=
gibt die Länge des Namens der Serialisierungskennung in Byte an. Die Länge muss mindestens 1 Byte sein und darf 54 Byte nicht überschreiten.
Fehlt der Operand, so wird das Längenattribut des SINAMAD-Operanden angenommen, wenn SINAMAD=adr angegeben ist; bei SINAMAD=(r) wird die maximale Länge (54) angenommen.

länge
Länge des Namens der Serialisierungskennung.

SCOPE=
beschreibt den Geltungsbereich (Teilnehmerkreis) für die Serialisierungskennung.

LOCAL
Die Serialisierungskennung wird nur von der Task des Aufrufers benutzt.

GROUP
Teilnehmer sind alle Tasks unter der Benutzerkennung des Aufrufers.

USER_GROUP
Teilnehmer können alle Tasks sein, deren Benutzerkennungen der gleichen Benutzergruppe angehören wie die Benutzerkennung des einrichtenden Teilnehmers. Der Operandenwert setzt die Existenz von Benutzergruppen voraus und kann daher nur angegeben werden, wenn die Funktionseinheit SRPM des Software-Produkts SECOS im System vorhanden ist.

Vor einem Makroaufruf mit SCOPE=USER_GROUP muss deshalb mit dem Makro GETUGR (siehe Handbuch „SECOS“ [14]) geprüft werden, ob SRPM zur Verfügung steht; abhängig vom Ergebnis (Returncode) ist im Programm zu reagieren.

GLOBAL
Teilnehmer sind alle Tasks im System.

SIID=
bezeichnet die Kurzkennung der Serialisierungskennung. Diese Kurzkennung wird dem Benutzer durch den **ENASI**-Makroaufruf zur Verfügung gestellt. Die Verwendung der Kurzkennung an Stelle des Namens zur Indentifizierung der Serialisierungskennung beschleunigt die Verarbeitung und ist eindeutig.

adr
symbolische Adresse eines 4 Byte langen Feldes, das die Kurzkennung enthält

(r)
r = Register, das die Adresse enthält

CONTINU=
Durch diesen Operanden kann eine Kettung von bis zu 255 **ENQAR**-Aufrufen erfolgen.

NO
Dies ist der letzte (bzw. einzige) Aufruf einer Folge.

YES
YES bedeutet, dass unmittelbar nach diesem **ENQAR**-Aufruf ein weiterer folgt.

COND=
beschreibt die Verarbeitungsweise der Zugriffsanforderung. Kann die Anforderung sofort erfüllt werden, wird dies getan; wenn nicht, wird die Anforderung in die Warteschlange für die angesprochene Serialisierungskennung eingereiht. Das aufrufende Programm muss dann solange warten, bis die vollständige Anforderung bearbeitet werden kann oder die im LIFETIM-Operanden angegebene Wartezeit abgelaufen ist. In einer mit CONTINU geketteten Folge von **ENQAR**-Aufrufen muss der Operand COND im letzten **ENQAR**-Aufruf angegeben werden; er gilt jedoch für die gesamte Folge.

UNCOND
Die Anforderung ist ohne Bedingung.

IMMED
Die Zugriffsanforderung wird nur dann verarbeitet, wenn die vollständige Anforderung erfüllt werden kann.

LIFETIM=
gibt die Zeit an, die die Task auf die Erfüllung der Zugriffsanforderungen warten soll. Ein Returncode zeigt an, ob die Anforderung erfüllt oder die maximale Wartezeit überschritten wurde. In einer mit CONTINU geketteten Folge von **ENQAR**-Aufrufen muss der Operand LIFETIM im letzten **ENQAR**-Aufruf angegegeben werden; er gilt jedoch für die gesamte Folge.

sec
Zeitangabe in Sekunden. 1 sec ≤ Wartezeit ≤ 43200 sec
Die Genauigkeit der Bearbeitung beträgt +10 sec.
Voreinstellung: 600 sec

(r)
r = Register, das die Zeitangabe in Sekunden enthält.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

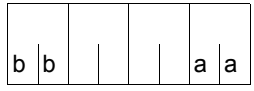
Hinweise zum Makroaufruf

- Ein implizites Enable wird ausgeführt, wenn der Aufrufer für diese Serialisierungskennung noch kein explizites Enable (siehe **ENASI**-Makroaufruf) angefordert hatte. Es wird jedoch bei einem impliziten Enable keine Kurzkennung zur Verfügung gestellt.
- Jeder weitere Enable-Aufruf (explizit oder implizit) in einem Programm für eine Serialisierungskennung, die bereits vorher in einem expliziten Enable angegeben und deren Verwendung wieder beendet wurde (siehe **DISSI**-Aufruf), bekommt nicht notwendigerweise dieselbe Kurzkennung wie im ersten Enable-Aufruf.
- Ein Enable, das als Teil eines **ENQAR**-Aufrufes ausgeführt wird, und das Einreihen in die Warteschlange sind eine zusammengehörende Operation.
- Ist derselbe Name für eine Serialisierungskennung in zwei Enable-Aufrufen mit zwei verschiedenen Geltungsbereichen angegeben, werden zwei unterschiedliche Serialisierungskennungen vom System verarbeitet.
- Bei der Verwendung von MF=L ist zu beachten:
  Zur Ausführung muss nur ein Makroaufruf mit MF=E gegeben werden, unabhängig davon, ob dieser Aufruf für eine einzelne Anforderung oder für eine Folge von Anforderungen gilt. Der Datenbereich bei einer Folge von Anforderungen wird durch Kettung der Makroaufrufe (MF=L) durch den CONTINU-Operanden erzeugt.

Rückinformation und Fehleranzeigen

Register R1 enthält die Adresse des Datenbereichs.

R15:



Über die Ausführung des Makros ENQAR wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'bb'	X'aa'	Erläuterung
X'04'	X'00'	Die Enqueue-Anforderungen wurden erfüllt: Mindestens eine Serialisierungskennung wurde eingerichtet und der Task des Aufrufers zugeordnet.
X'08'	X'00'	Die Enqueue-Anforderungen wurden erfüllt: Mindestens eine Serialisierungskennung wurde der Task des Aufrufers zugeordnet.
X'0C'	X'00'	Die Enqueue-Anforderungen wurden erfüllt: Alle Serialisierungskennungen waren bereits der Task des Aufrufers zugeordnet.
X'3C'	X'00'	Die Enqueue-Anforderungen wurden erfüllt: Es wurde bereits ein DEQAR-Aufruf aus einer Task verarbeitet, die nicht ' Halter' der Serialisierungskennung war. Dieser DEQAR-Aufruf wurde verarbeitet, da HOLDER=ANY angegeben war. Diese Rückinformation hat Priorität über alle anderen, deren Rücksprungschalter RS=X' 00' ist.
X'10'	X'04'	Die Enqueue-Anforderungen wurden nicht erfüllt. Es wurden ungültige Operanden angegeben: – ungültige Adresse (z.B. Adresse innerhalb einer DSECT) – ungültige Länge, ungültiger Name, ungültiger LIFETIM-Wert – nichtdefinierter Geltungsbereich, CONTINU- oder COND-Wert – COND- oder LIFETIM-Angabe für eine Kennung, die nicht die letzte in einer mit CONTINU geketteten Folge ist.
X'14'	X'04'	Die Enqueue-Anforderungen wurden nicht erfüllt: Es wurde eine ungültige Kennung angegeben.
X'18'	X'04'	Die Enqueue-Anforderungen wurden nicht erfüllt: Die maximale Anzahl gleichzeitig verwendbarer Serialisierungskennungen wurde überschritten.
X'1C'	X'04'	Die Enqueue-Anforderungen wurden nicht erfüllt: Die gesamte Anforderung konnte nicht – sofort erfüllt werden (bei COND=IMMED) – innerhalb der Wartezeit erfüllt werden (bei COND=UNCOND).
X'24'	X'04'	Die Enqueue-Anforderungen wurden nicht erfüllt: Das aufrufende Programm belegt bereits die angeforderte Serialisierungskennung.
X'40'	X'04'	Die Enqueue-Anforderungen wurden nicht erfüllt: Es war kein Klasse-5-Speicher zur Verarbeitung der Anforderung verfügbar.
X'44'	X'04'	Die Enqueue-Anforderungen wurden nicht erfüllt: Es wurde eine Disable-Anforderung für mindestens eine Serialisierungskennung gegeben (durch einen Contingency-Prozess mit höherer Priorität), während die Enqueue-Anforderung noch anstand.

ENTER – ENTER-Auftrag (ENTER-Job) einleiten

Allgemeines

Anwendungsgebiet: Starten, Unterbrechen und Beenden; siehe Seite 72
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-/C-/D-Form; siehe Seite 29

Makrobeschreibung

Mit dem Makroaufruf **ENTER** wird über den Makroanschluss des Kommandosprachübersetzers MCLP das Kommando **ENTER-JOB** gegeben, ohne dass der Programm-Modus verlassen wird (siehe Abschnitt „Makroaufrufe an den Kommandosprachübersetzer“ auf Seite 45). Meldungen bezüglich der Kommandobearbeitung werden auf SYSOUT ausgegeben und auf Wunsch auch in einen Bereich des aufrufenden Programms übertragen. Mit dem ENTER-JOB-Kommando kann der Anwender einen Batch-Auftrag, der in einer (ENTER-)Datei gespeichert ist, dem Betriebssystem zur Verarbeitung übergeben.
Die (ENTER-)Datei ist eine katalogisierte Datei oder ein Bibliothekselement. Das Kommando ENTER-JOB kann sowohl im Programm-Modus als auch im Kommandomodus abgesetzt werden (siehe auch Handbuch „Kommandos“ [19]). Der neue Auftrag (Job) erhält eine eigene Auftragsnummer (TSN) und wird in einer eigenen Task - unabhängig von der aufrufenden Task - ausgeführt. Die Angaben im ENTER-JOB-Kommando bezeichnen die (ENTER-)Datei, identifizieren den Aufrufer (Zugriffsberechtigung und Abrechnung) und charakterisieren den Job und die Protokollführung über den Joblauf.

Die Angaben zur Zugriffsberechtigung werden gegen den Eintrag im Benutzerkatalog geprüft; weitere Angaben zur Jobklasse und zu den Jobattributen (Job-, Runpriorität, Systemressourcen) auch gegen den Eintrag in der Jobklassendefinition. Diese Einträge sind dem Anwender über die Kommandos SHOW-USER-ATTRIBUTES oder SHOW-JOB-CLASS zugänglich.
Stimmen die Angaben für PRIORITY (Priorität) und NTL (No Time Limit) im Benutzerkatalog und der Jobklassendefinition nicht überein, wird der für den Anwender günstigere Wert zugelassen.
Die Operanden PRIORITY und MSG werden nur noch aus Kompatibilitätsgründen unterstützt. Stattdessen sollten die Operanden RUN-PRIO bzw. RUN-PRIO in Verbindung mit START=IMMEDATELY (für PRIORITY=(p,EXPRESS)) und LOG verwendet werden.

Die Liste der Kommando-Operanden muss in Apostrophen als Zeichenkette angegeben werden ('pfadname [,userid1, ... ,JOB-PAR=..]'). Bei innerhalb dieser Operandenliste auftretenden Zeichenketten (z.B. bei Angabe des Operanden HOST='hostid') müssen die Apostrophe zur Entwertung verdoppelt werden.

Eine (ENTER-)Datei beginnt mit dem SET-LOGON-PARAMETERS- und endet mit dem EXIT-JOB-Kommando. Die Operanden im SET-LOGON-PARAMETERS-Kommando werden nur bei Angabe von FROM-LOGON=YES ausgewertet.

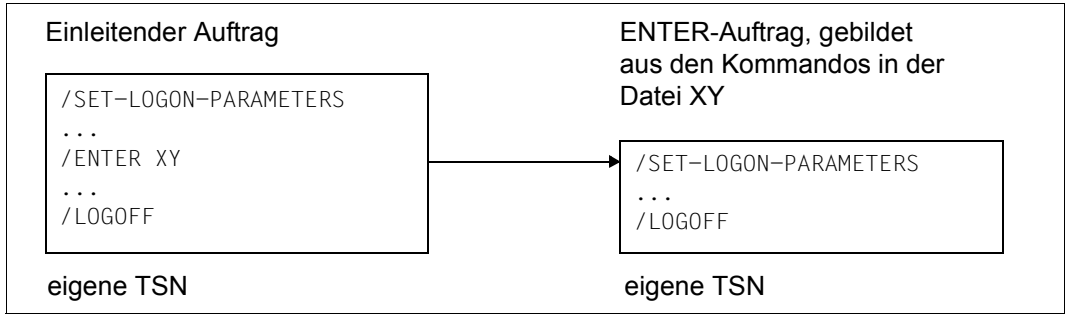


Bild 24: Einleitung eines ENTER-Jobs

Makroaufrufformat und Operandenbeschreibung

ENTER

'<text>'

wobei <text> aus den folgenden Operanden besteht (Hochkomma müssen doppelt angegeben werden):

pfadname
[,userid1,abrechnr[,kennwort]]
,FROM-LOGON=NO / YES
[,FPASS=kennwort]
[,CRPASS=kennwort]
,ERASE=NO / YES
[,HOST=*ANY / ' 'hostid' ' / jvname1] [,CAT=' 'catid' ' / jvname2]
[,JOB-CLASS=*STD / jobklasse]
[,MONJV=jvname] [,JVPASS=kennwort]
[,JOB-PRIO=STD / jprio]
[,RERUN=NO / YES]
[,FLUSH=NO / YES]

[,START= { STD | SOON | WITHIN({ HOURS=stunde[,MINUTES=minute] | [HOURS=stunde,]MINUTES=minute }) | AT([DATE=yy-mm-dd,]TIME=hh:mm) | EARLIEST([DATE=yy-mm-dd,]TIME=hh:mm) | LATEST([DATE=yy-mm-dd,]TIME=hh:mm) | AT-STREAM-STARTUP | IMMEDIATELY }]

ENTER (Fortsetzung)

```
,REPEAT= { STD                                                    } [ ]
         { NO                                                     }
         { PERIOD( { HOURS=stunde[,MINUTES=minute]  } )           }
         {         { [HOURS=stunde,]MINUTES=minute  }             }
         { DAILY                                                  }
         { WEEKLY                                                 }
         { AT-STREAM-STARTUP                                      }
,CALENDAR=' 'pfadname' ' ,SYMDATE=symdatname]
,LIMIT=STD / anzahl / (DATE=yy-mm-dd,TIME=hh:mm)
[,RUN-PRIO=STD / rprio]
[,TIME=STD / NTL / t]
,PROTECTION=NONE / CANCEL
[,PRINT=STD / NO-LIMIT / anzahl]
[,LOG=(LISTING=NO / YES)]
[,JOB-PAR=*NO / ' 'attribute' ']
[,PRIORITY=p / ([p],EXP[RESS])]
,MSG=[F / C] [L] [H]
[,adr / (r)]
[,PARMOD=24 / 31]
[,MF=C / D / L / (E,..)]
```

Die Variable `pfadname` steht für: [:catid:] [$userid.] { dateiname | bibliothek(element) }

wobei

catid
Katalogkennung des Pubsets, auf dem die Datei gespeichert ist.
Voreinstellung: die der Benutzerkennung im Benutzerkatalog zugeordnete Katalogkennung.

userid
Benutzerkennung, der die Datei zugeordnet ist.
Voreinstellung: Benutzerkennung, unter der der Makroaufruf stattfindet.

Wenn `pfadname` ohne Katalog- und Benutzerkennung angegeben wird und er nicht in der eigenen Benutzerkennung katalogisiert ist, dann versucht das System, auf eine gleichnamige Datei bzw. Bibliothek in der System-Standardkennung zuzugreifen (Funktion „Secondary-Read“, siehe Handbuch „Einführung in das DVS“ [8]).

dateiname
Name der katalogisierten Datei mit dem Batch-Auftrag.

- `dateiname` kann auch der Name einer temporären Datei sein (siehe Handbuch „DVS Einführung“ [8])
- es kann keine Dateigeneration oder Dateigenerationsgruppe angegeben werden
- die Angabe einer Dateigruppe (Format „datei(gruppe)“!) ist nur für Banddateien zulässig
- Gehört die Datei nicht zur eigenen Benutzerkennung, muss sie mehrbenutzbar sein

bibliothek
Name einer PLAM-Bibliothek auf Platte (siehe Handbuch „LMS“ [29]).

(element)
element = Name des Bibliothekelementes mit dem Batch-Auftrag

Der Ausdruck bibliothek(element) darf ohne Katalog- und Benutzerkennung maximal 41 Zeichen lang sein. Mit vollem Pfadnamen einschließlich Katalog- und Benutzerkennung darf er maximal 54 Zeichen lang sein.

userid1
Benutzerkennung für den einzuleitenden ENTER-Auftrag oder `*FROMCA`, wenn der Makroaufruf aus einer Benutzerkennung mit dem Privileg OPERATING erfolgt.

***FROMCA**
Die Benutzerkennung des Aufrufers wird verwendet.

abrechnr
Abrechnungsnummer für den ENTER-Auftrag.

Die Operanden `userid1` und `abrechnr` dürfen im ENTER-Makro nur gemeinsam angegeben oder weggelassen werden. Fehlen sie im ENTER-Makro, so gelten die Werte des SET-LOGON-PARAMETERS-Kommando des laufenden Auftrags.

kennwort
Kennwort für die Benutzerkennung `userid1`.
`kennwort` ist eine Zeichenkette der Länge 8 Byte (c-string) bzw. der Länge 16 Byte (x-string). Da `kennwort` eine Zeichenkette innerhalb einer Zeichenkette ist, müssen die Apostrophe verdoppelt werden. Das Kennwort wird nicht auf SYSOUT protokolliert, d.h. es erscheint nicht im Druckerprotokoll des ENTER-Auftrages.
Fehlen die Angaben `userid1`, `abrechnr` und `kennwort`, so werden sie bei FROM-LOGON=YES aus dem SET-LOGON-PARAMETERS-Kommando in der ENTER-Datei übernommen, sonst aus dem SET-LOGON-PARAMETERS-Kommando des einleitenden Auftrags.

FROM-LOGON=
gibt an, ob die Operanden des Kommandos SET-LOGON-PARAMETERS, mit dem die ENTER-Datei beginnt, ausgewertet werden sollen (so wie bei einem ENTER-JOB-Kommando an der Konsole) oder nicht.

NO
Die Operanden des SET-LOGON-PARAMETERS-Kommando in der ENTER-Datei werden nicht ausgewertet.

YES
Dieser Wert darf nur von einem Aufrufer mit Privileg OPERATING angegeben werden.
Die Operanden des SET-LOGON-PARAMETERS-Kommando in der ENTER-Datei werden ausgewertet. Angaben im ENTER-Makro haben allerdings Vorrang, d.h. ein im SET-LOGON-PARAMETERS-Kommando angegebener Wert wird nur wirksam, wenn der entsprechende Operand im ENTER-Makro **nicht** angegeben ist.

FPASS=
bezeichnet das Ausführ- oder Schreib-Kennwort für die ENTER-Datei: Schreibkennwort, wenn ERASE=YES angegeben wird, sonst das Ausführkennwort. Der Operand erlaubt den Zugriff auf die ENTER-Datei, wenn das anzugebende Kennwort mit dem Kennwort übereinstimmt, das diese Datei schützt.

kennwort
Kennwort für den Zugriff.
`kennwort` ist eine Zeichenkette der Länge 4 Byte (c-string) bzw. der Länge 8 Byte (x-string). Da kennwort eine Zeichenkette innerhalb einer Zeichenkette ist, müssen die Apostrophe verdoppelt werden. Das Kennwort wird nicht auf SYSOUT protokolliert, d.h. es erscheint nicht im Druckprotokoll des ENTER-Auftrages.

CRPASS=
bezeichnet das Kennwort, mit dem die ENTER-Datei verschlüsselt ist. Der Operand erlaubt den Zugriff auf die verschlüsselte ENTER-Datei, wenn das anzugebende Kennwort mit dem Kennwort übereinstimmt, das diese Datei schützt.

kennwort
Kennwort für den Zugriff.
`kennwort` ist eine Zeichenkette der Länge 8 Byte (c-string) bzw. der Länge 16 Byte (x-string). Da kennwort eine Zeichenkette innerhalb einer Zeichenkette ist, müssen die Apostrophe verdoppelt werden. Das Kennwort wird nicht auf SYSOUT protokolliert, d.h. es erscheint nicht im Druckprotokoll des ENTER-Auftrages.

ERASE=
gibt an, ob die ENTER-Datei bei Auftragsende gelöscht werden soll.

NO
Die ENTER-Datei soll nicht gelöscht werden.

YES
Die ENTER-Datei soll am Ende des ENTER-Auftrages gelöscht werden.

Trotz der Angabe von ERASE=YES wird die Datei nicht gelöscht, wenn

a) die Datei ein Bibliothekselement ist

b) der Auftraggeber nicht (Mit-)Eigentümer der Datei ist

c) der Auftrag abnormal beendet wird

d) der Auftrag durch ein EXIT-JOB(MODE=*ABNORMAL)-, CANCEL-JOB- oder SHUTDOWN-Kommando abgebrochen wird

Die Fälle c) und d) gelten nicht für eine Datei auf privater Platte, für eine temporäre Datei und für eine Datei, die nicht unter der Benutzerkennung katalogisiert ist, unter der auch der ENTER-Auftrag ablaufen soll. In diesen Fällen wird die Datei nach dem Erstellen der S.IN.-Datei gelöscht.

HOST=
bestimmt das Ziel-System, auf dem der Auftrag ablaufen soll.
Nur für Anwender der Software-Produkte „HIPLEX MSCF“ [26] und „JV“ [22].

***ANY**
Der Auftrag kann auf einem beliebigen Ziel-System ablaufen.

"hostid"
hostid = Bezeichnung des Ziel-Systems.

jvname1
Jobvariable, die hostid enthält. Die Syntax für jvname1 muss den Regeln einer **GETJV**-Operation (siehe Handbuch „JV“ [22]) genügen.

CAT=
bestimmt das Ziel-System über die angegebene Katalogkennung. Der Auftrag wird dem System zugeleitet, dem der angegebene Katalog zugeordnet ist.
Nur für Anwender der Software-Produkte „HIPLEX MSCF“ [26] und „JV“ [22].

"catid"
catid = Katalogkennung.

jvname2
Jobvariable, die catid enthält. Die Syntax für jvname2 muss den Regeln einer **GETJV**-Operation (siehe Handbuch „JV“ [22]) genügen.

JOB-CLASS=
bezeichnet eine Jobklasse, in die der Auftrag eingereiht werden soll.
Wird der Operand nicht angegeben, erhält er bei FROM-LOGON=YES den Wert aus dem Kommando SET-LOGON-PARAMETERS in der ENTER-Datei, sonst den Wert *STD.
Die Berechtigung zu den verschiedenen Jobklassen kann mit dem Kommando SHOW-USER-ATTRIBUTES oder SHOW-JOB-CLASS abgefragt werden.

***STD**
Die Jobklasse ist die für den Anwender oder das System voreingestellte (Standard-) Jobklasse.

jobklasse
Name der Jobklasse.

MONJV=
bezeichnet eine Jobvariable, die den Auftrag überwacht.
Wird der Operand nicht angegeben, dann gilt bei FROM-LOGON=YES der Wert aus dem Kommando SET-LOGON-PARAMETERS in der ENTER-Datei, sonst wird der ENTER-Job ohne MONJV gestartet.

Über diese Jobvariable kann der Anwender seinen Auftrag ansprechen. Während des Ablaufs des Auftrages ordnet das Betriebssystem der Jobvariablen folgende Werte zu:
$S Auftrag in Auftragswarteschlange
$R Auftrag in Verarbeitung
$T Auftrag normal beendet
$A Auftrag vorzeitig abgebrochen
$M Auftrag mit /MOVE-JOBS exportiert
Nur für Anwender des Software-Produktes „JV“ [22].

jvname
Name der Jobvariablen.

JVPASS=
bezeichnet ein Passwort, das zum Zugriff auf die überwachende Jobvariable berechtigt.
JVPASS wird ignoriert, wenn MONJV nicht angegeben wurde.

kennwort
Kennwort für die Jobvariable jvname.
kennwort = Zeichenkette in der Länge 4 Byte (c-string) bzw. in der Länge 8 Byte (x-string). Da kennwort eine Zeichenkette innerhalb einer Zeichenkette ist, müssen die Apostrophe verdoppelt werden. Das Kennwort wird nicht auf SYSOUT protokolliert, d.h. es erscheint nicht im Druckerprotokoll des ENTER-Auftrages.

JOB-PRIO=
bestimmt die Dringlichkeit (relativ zu den anderen Jobs) für den Start eines Batchjobs. Auf den weiteren Jobablauf hat die Angabe keinen Einfluss.
Wird der Operand nicht angegeben, erhält er bei FROM-LOGON=YES den Wert aus dem Kommando SET-LOGON-PARAMETERS in der ENTER-Datei, sonst den Wert STD.

STD
Der Standardwert für die Jobklasse wird angenommen.

jprio
Jobpriorität. MAXIMUM ≤ jprio ≤ 9. Je niedriger der Wert, desto höher (größer) die Jobpriorität (Dringlichkeit). Der Wert für MAXIMUM ist in der Jobklassendefinition festgelegt und kann mit dem Kommando SHOW-JOB-CLASS abgefragt werden.

RERUN=
bestimmt, ob der Auftrag im nächsten BS2000-Systemlauf neu eingeleitet werden soll, wenn die Ausführung durch schwere Systemfehler oder Systemlaufende unterbrochen wurde.
Wird der Operand nicht angegeben, erhält er bei FROM-LOGON=YES den Wert aus dem Kommando SET-LOGON-PARAMETERS in der ENTER-Datei, sonst den Wert NO.

NO
Keine Neueinleitung des Auftrags.

YES
Der Auftrag wird nochmals eingeleitet.

FLUSH=
gibt an, ob der Auftrag aus der Auftragswarteschlange entfernt wird, wenn er bis Ende des Systemlaufs (Shutdown) nicht bearbeitet wurde.
Wird der Operand nicht angegeben, erhält er bei FROM-LOGON=YES den Wert aus dem Kommando SET-LOGON-PARAMETERS in der ENTER-Datei, sonst den Wert NO.

NO
Der Auftrag verbleibt in der Warteschlange.
Der nächste Systemlauf muss mit einem Warm- oder Selektivstart eingeleitet werden.

YES
Der Auftrag wird aus der Warteschlange entfernt.

Auftragssteuerung mit RERUN/FLUSH:

- wurde FLUSH=YES und RERUN=YES angegeben und der Auftrag während des vorherigen Systemlaufs unterbrochen, wird im nächsten Systemlauf FLUSH=NO angenommen. Damit ist garantiert, dass der Auftrag in der Auftragswarteschlange verbleibt, auch wenn er in diesem Systemlauf nicht gestartet wird.
- eine überwachende Jobvariable wird bei Wiederholung eines Auftrages auf „$S“ gesetzt.

– RERUN und FLUSH werden bei Wiederholungsaufträgen nicht ausgewertet.
– FLUSH=YES wird für Kalenderjobs mit der Warnung JMS0056 ignoriert.

START=
bezeichnet einen Zeitpunkt (Zeitraum) für den Start des Auftrages.
Wird der Operand nicht angegeben, erhält er bei FROM-LOGON=YES den Wert aus dem Kommando SET-LOGON-PARAMETERS in der ENTER-Datei, sonst den Wert STD.

STD
Der Standardwert für die gewählte Jobklasse wird angenommen.

SOON
Der Auftrag soll unter Berücksichtigung seiner Priorität so bald wie möglich gestartet werden.

IMMEDIATELY
Der Auftrag soll unmittelbar gestartet werden.

WITHIN(...)
Der Auftrag soll innerhalb der angegebenen Zeit (in Stunden und Minuten) gestartet werden. 0 ≤ stunde ≤ 23; 0 ≤ minute ≤ 59.

AT(...)
Der Auftrag sollte exakt zu dem angegebenen Zeitpunkt (Datum, Tageszeit) gestartet werden.

```
DATE=yy-mm-dd    Datum (yy = Jahr, mm = Monat, dd = Tag)
TIME=hh:mm       Tageszeit (hh = Tagesstunde, mm = Minute)
```

– Bindestriche bzw. Doppelpunkt in DATE bzw. TIME müssen angegeben werden. Beispiel: 31.Mai 2012 um 15.08 Uhr ergibt:
AT (DATE=12-05-31, TIME= 15:08).
– Für TIME gilt: 00 ≤ hh ≤ 23; 00 ≤ mm ≤ 59.
– Jahreszahlen < 80 werden als Jahr 20yy interpretiert, Angaben ≥ 80 werden als Jahr 19yy interpretiert.

EARLIEST(...)
Der Auftrag soll frühestens zu dem angegebenen Zeitpunkt (Datum, Tageszeit) gestartet werden.

```
DATE=yy-mm-dd    Datum (yy = Jahr, mm = Monat, dd = Tag).
TIME=hh:mm       Tageszeit (hh = Tagesstunde, mm = Minute).
```

Siehe Operand START=AT(...)

LATEST(...)
Der Auftrag sollte spätestens bis zu dem angegebenen Zeitpunkt (Datum, Tageszeit) gestartet werden.

```
DATE=yy-mm-dd     Datum (yy = Jahr, mm = Monat, dd = Tag).
TIME=hh:mm        Tageszeit (hh = Tagesstunde, mm = Minute).
```

Siehe Operand START=AT(...)

AT-STREAM-STARTUP
Der Auftrag soll nach dem Startup des Jobschedulers gestartet werden.

Die Startwerte SOON, IMMEDIATELY, WITHIN, AT, EARLIEST, LATEST und AT-STREAM-STARTUP sind nur dann zulässig, wenn sie auch in der Jobklassendefinition zugelassen sind, siehe auch Kommando SHOW-JOB-CLASS.

REPEAT=
bezeichnet einen Zeitabschnitt, nach dessen Ablauf der Auftrag periodisch gestartet werden soll. Die Wiederholung wird als Auftragsfolge (Jobfolge) betrachtet. J(0) bezeichnet den ersten Auftraglauf, J(1) die erste Wiederholung, ..., J(n) die n-te Wiederholung des Auftrages (Job). Mit dem Start des Auftrages J(i) wird auch die Wiederholung J(i+1) kreiert, ($i \geq 0$).
Wird der Operand nicht angegeben, erhält er bei FROM-LOGON=YES den Wert aus dem Kommando SET-LOGON-PARAMETERS in der ENTER-Datei, sonst den Wert STD.

STD
Es wird der Standardwert in der gewählten Jobklasse angenommen.

NO
Der Auftrag wird nicht wiederholt.

DAILY
Tägliche Wiederholung zu der mit START angegebenen Tageszeit.

WEEKLY
Wöchentliche Wiederholung zu der mit START angegebenen Tageszeit.

PERIOD(...)
Wiederholung nach dem angegebenen Zeitintervall (in Stunden und Minuten).
$0 \leq$ stunde ≤ 23; $0 \leq$ minute ≤ 59.

AT-STREAM-STARTUP
Wiederholung nach jedem Startup des Jobschedulers.

Hinweise

- Die Angabe der Repeatwerte NO, DAILY, WEEKLY, PERIOD und AT-STREAM-STARTUP ist nur dann zulässig, wenn sie auch in der Jobklassendefinition zugelassen sind; (siehe auch Kommando SHOW-JOB-CLASS).
- Bei einem Wiederholungsauftrag werden die Operanden FLUSH und RERUN nicht ausgewertet. Der Auftrag wird zum nächsten Wiederholungszeitpunkt gestartet.
- Die i-te Wiederholung, ($i \geq 1$), eines Auftrages (Jobs) wird nur dann gestartet, wenn die (i-1)-te Ausführung beendet ist.
- Abbrechen des gerade laufenden Auftrages J(i) hat keine Auswirkung auf den Start von J(i+1); ($i \geq 0$).

– Abbruch des gesamten Auftrages: es muss sowohl der gerade laufende Auftrag J(i) als auch der Folgeauftrag J(i+1) abgebrochen werden, (i≥0); (Kommando CANCEL-JOB oder mit Kommando MODIFY-JOB tsn, REPEAT=NO den Auftrag J(i) zum letzten Auftrag der Repeat-Folge machen).

CALENDAR=
Der Startzeitpunkt des Auftrages und mögliche Wiederholungen werden durch ein symbolisches Datum, das in einer Kalenderdatei definiert ist, festgelegt (Kalenderjob).
Die Operanden CALENDAR und SYMDATE müssen zusammen angegeben werden.
Bei Angabe von CALENDAR und SYMDATE dürfen die Operanden START und REPEAT nicht angegeben werden.

"pfadname"
pfadname (siehe Seite 493) = Name der Kalenderdatei.

SYMDATE=
Siehe CALENDAR.

symdatname
Symbolisches Datum, das den Startzeitpunkt und ggf. Wiederholungszyklen innerhalb der Kalenderdatei bezeichnet.

LIMIT=
bestimmt die Lebensdauer eines Kalenderjobs. Diese Begrenzung gilt zusätzlich zu den Grenzen, die durch den Kalender gesetzt sind.

STD
Die Lebensdauer des Kalenderjobs ergibt sich allein aus dem Eintrag des symbolischen Datums im Kalender.

anzahl
Die Angabe ist nur für Kalenderjobs zulässig.
1 ≤ anzahl ≤ 32767
Maximale Anzahl der Wiederholungen des Kalenderjobs. Nach Beendigung eines einzelnen Joblaufs wird der Ablaufzähler um 1 erhöht. Danach wird geprüft, ob der Ablaufzähler die maximale Anzahl erreicht bzw. überschritten hat. Trifft dies zu, wird der gesamte Kalenderjob beendet.

(DATE=yy-mm-dd,TIME=hh:mm)
Die Angabe ist nur für Kalenderjobs zulässig.
Einträge in der Kalenderdatei werden nur bis zum angegebenen Limit berücksichtigt. Für Kalendereinträge nach dem Limit wird kein Wiederholungsauftrag mehr erzeugt; der Kalenderjob beendet sich.
Die Limitierung bezieht sich ausschließlich auf die Termineinträge in der Datei, nicht auf die reale Laufzeit der Aufträge. Wiederholungsaufträge mit „zulässigem“ Starttermin unterliegen keinen weiteren Beschränkungen und werden z.B. auch nach dem angegebenen Datum noch gestartet, wenn dies vorher wegen Verzögerungen im Job-Scheduler nicht möglich war.

Das Datum wird bestimmt durch Angabe des Tages und der Uhrzeit:

Siehe Operand START=AT(...)

RUN-PRIO=
bestimmt die Dringlichkeit (relativ zu anderen Tasks) für die Abarbeitung des Auftrages (Jobs).
Wird der Operand nicht angegeben, erhält er bei FROM-LOGON=YES den Wert aus dem Kommando SET-LOGON-PARAMETERS in der ENTER-Datei, sonst den Wert STD.

STD
Standardwert der gewählten Jobklasse. Der Standardwert wird auch angenommen, wenn unzulässige Werte für rprio angegeben werden.

rprio
Runpriorität; MAXIMUM ≤ rprio ≤ 255
Je niedriger der Wert, desto höher (größer) die Dringlichkeit. Der Wert für MAXIMUM ist sowohl in der Jobklassendefinition als auch im Benutzerkatalog festgelegt und kann mit dem Kommando SHOW-JOB-CLASS bzw. SHOW-USER-ATTRIBUTES abgefragt werden. Stimmen die Werte nicht überein, wird der für den Anwender günstigere Grenzwert zugelassen.

TIME=
bezeichnet die CPU-Zeit (in Sekunden), die die Task höchstens verbrauchen darf. Die maximal angebbare CPU-Zeit wird durch die gewählte Jobklasse festgelegt.
Wird der Operand nicht angegeben, erhält er bei FROM-LOGON=YES den Wert aus dem Kommando SET-LOGON-PARAMETERS in der ENTER-Datei, sonst den Wert STD.

STD
Standardwert der gewählten Jobklasse.

t
CPU-Zeit in Sekunden. 0 ≤ t ≤ maximale CPU-Zeit.

NTL
NTL = No Time Limit. Die Task läuft ohne Begrenzung der CPU-Zeit.

PROTECTION=
bestimmt, ob der Auftrag gegen eine versehentliche Beendigung durch das Kommando CANCEL-JOB geschützt werden soll.

NONE
Der Auftrag wird nicht geschützt.

CANCEL
Der Auftrag wird geschützt.

PRINT=
bezeichnet die maximale Anzahl von Sätzen, die von der Task (summarisch) in die Systemdateien SYSLST, SYSLST01, SYSLST02,...,SYSLST99 ausgegeben werden. Datensätze in die Systemdatei SYSOUT, die gleichzeitig nach SYSLST geschrieben werden (Angabe LOG=(LISTING=YES) oder MSG=FH) zählen nicht mit.
Wird der Operand nicht angegeben, erhält er bei FROM-LOGON=YES den Wert aus dem Kommando SET-LOGON-PARAMETERS in der ENTER-Datei, sonst den Wert STD.

STD
Standardwert der gewählten Jobklasse.

anzahl
Anzahl der Sätze. 0 ≤ anzahl ≤ MAXIMUM. Der Wert für MAXIMUM ist in der Jobklassendefinition festgelegt und kann mit dem Kommando SHOW-JOB-CLASS abgefragt werden.

NO-LIMIT
Anzahl der Sätze ist nicht begrenzt.

LOG=(...)
gibt an, ob das Protokoll des Jobablaufs zusätzlich nach SYSLST (LISTING=YES) ausgegeben wird.
Wird der Operand nicht angegeben, erhält er bei FROM-LOGON=YES den Wert aus dem Kommando SET-LOGON-PARAMETERS in der ENTER-Datei, sonst den Wert LISTING=NO.

JOB-PAR=
ermöglicht die Angabe zusätzlicher Attribute für die gewählte Jobklasse, sofern die Systemverwaltung solche definiert und bekannt gegeben hat.
Wird der Operand nicht angegeben, erhält er bei FROM-LOGON=YES den Wert aus dem Kommando SET-LOGON-PARAMETERS in der ENTER-Datei, sonst den Wert *NO.

***NO**
Keine zusätzlichen Attribute.

"attribute"
attribute = Folge beliebiger Zeichen; wird von der Systemverwaltung zur Kennzeichnung weiterer Jobklassenattribute vergeben.

PRIORITY=
bestimmt die Dringlichkeit (relativ zu anderen Tasks) für die Abarbeitung des Auftrages (Jobs).

p
Runpriorität. MAXIMUM ≤ p ≤ 255. Je niedriger der Wert, desto höher (größer) die Dringlichkeit. Der Wert für MAXIMUM ist sowohl in der Jobklassendefinition als auch im Benutzerkatalog festgelegt und kann mit dem Kommando SHOW-JOB-CLASS bzw. SHOW-USER-ATTRIBUTES abgefragt werden. Stimmen die Werte nicht überein, wird der für den Anwender günstigere Grenzwert zugelassen.
Voreinstellung: Standardwert der gewählten Jobklasse.

Dieser Standardwert wird auch angenommen, wenn unzulässige Werte für p angegeben werden.

([p],EXP[RESS])
Die Angabe EXPRESS hat zur Folge, dass der ENTER-Auftrag sofort gestartet wird. Auf die weitere Auftragsabarbeitung hat die Angabe keinen Einfluss. Die Berechtigung für die Angabe EXPRESS ist im Benutzerkatalog und/oder in der Jobklassendefinition festgelegt.

Der Operand wird nur noch aus Kompatibilitätsgründen unterstützt. Die Angabe PRIORITY=p sollte durch RUN-PRIO=rprio, die Angabe PRIORITY= (p,EXPRESS) durch RUN-PRIO=rprio, START=IMMEDIATELY ersetzt werden.

Der Operand PRIORITY wird ignoriert, wenn der Operand RUN-PRIO spezifiziert wurde; die Angabe EXPRESS wird ignoriert, wenn der Operand START angegeben ist.

MSG=
Mit diesem Operanden wird gesteuert, in welcher Weise Systemmeldungen ausgegeben oder der Jobablauf protokolliert werden soll.

F
(Full message) Die Systemmeldungen werden ungekürzt auf die Systemdatei SYSOUT ausgegeben.

C
(Code) Die kodierte Kurzform der Systemmeldungen wird auf SYSOUT ausgegeben.

L
(Log) Konsolmeldungen und Operator-Antworten für diesen Auftrag werden protokolliert. Gibt der Benutzer MSG=LH, so werden die in SYSLST protokollierten Meldungen zusätzlich mit der Uhrzeit versehen, zu der sie gegeben wurden.

H
(Hold message) Der Ablauf wird zusätzlich auf SYSLST protokoliert.

Der Operand MSG wird nur noch aus Kompatibilitätsgründen unterstützt. Er wird vollständig ignoriert, wenn LOG spezifiziert wurde.

adr
Adresse des Empfangsfeldes für das SYSOUT-Protokoll. Wenn adr nicht angegeben ist, geht die Ausgabe nur nach SYSOUT. Das ist auch der Fall, wenn die Länge des Empfangsfeldes Null ist. Das Empfangsfeld muss auf Wortgrenze ausgerichtet sein. Aufbau:

Byte 0 - 1: Länge des Empfangsfeldes (≤ 32767 Byte)
Byte 2 - 3: kein Eintrag
Byte 4 - n: Anfang des SYSOUT-Protokolls.

Jeder Satz des SYSOUT-Protokolls, der in das Empfangsfeld übertragen wird, enthält in den ersten vier Byte sein Satzlängenfeld (Byte 0-1 enthält die Satzlänge, Byte 2-3 sind der reservierte Teil).
Der eigentliche Ausgabetext beginnt ab Byte 4 jedes Satzes. Die Ausgabesätze werden fortlaufend in den Bereich geschrieben, bis die Bereichsgrenze erreicht ist. Weitere Ausgabesätze, die in dem Bereich nicht mehr Platz haben, werden nur noch nach SYSOUT ausgegeben. Bei Überschreiten der Bereichsgrenze kann ein Satz abgeschnitten werden (Fehleranzeige X'0C').

(r)
r = Register mit dem Adresswert von adr.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

Kombinationen der Operanden START und REPEAT

START	REPEAT		
	AT-STREAM-STARTUP	**DAILY bzw. WEEKLY**	**PERIOD**
IMMEDIATELY / SOON	a)	c)	c)
AT / EARLIEST	a)	d)	f)
LATEST / WITHIN	a)	c)	g)
AT-STREAM-STARTUP	b)	e)	e)

a) Der erste Start und alle weiteren Starts des Auftrages erfolgen wie angegeben.

b) Der erste Start des Auftrages erfolgt mit START=AT-STREAM-STARTUP. Alle weiteren Starts erfolgen nach dem Startup des Jobschedulers mit START=SOON.

c) Zeitbasis für den Wiederholungszyklus ist der Zeitpunkt der Jobannahme.

d) Der angegebene Zeitpunkt (START=...., TIME=....) ist die Zeitbasis für den Wiederholungszyklus.

e) Der erste Start des Auftrages erfolgt nach dem Startup des Jobschedulers. Diese Startzeit ist die Zeitbasis für den Wiederholungszyklus. Die weiteren Starts erfolgen mit START=SOON.

f) Der angegebene Zeitpunkt (START=...., TIME=....) ist die Zeitbasis für den Wiederholungszyklus. Der zweite und alle weiteren Starts erfolgen mit START=SOON.

g) Zeitbasis für den Wiederholungszyklus ist der Zeitpunkt der Jobannahme. Alle weiteren Starts erfolgen mit START=SOON.

Allgemeine Hinweise

- Die Liste der Kommando-Operanden muss in Apostrophe eingeschlossen werden.
- Eine Kopie der Datei wird in folgenden Fällen unter dem Namen S.IN.tsn.datum.hhmmss erstellt:
  - wenn die Datei auf privater Platte steht
  - wenn der ENTER-Auftrag unter einer anderen Benutzerkennung ablaufen soll, als die Datei katalogisiert ist
  - wenn die Datei eine temporäre Datei ist
  - wenn die Datei verschlüsselt ist

  Wenn die ENTER-Datei ein Bibliotkekselement ist, wird eine Kopie unter dem Namen S.IN.bibliotheksname.elementname.tsn.hhmmss erstellt.
  Die S.IN.-Datei wird bei Auftragsende (EXIT-JOB) automatisch gelöscht, außer, während des Joblaufs wurden Fixpunkte (Makro **WRCPT**) gesetzt. In diesem Fall muss für einen problemlosen Wiederanlauf (Kommando RESTART-PROGRAM) die S.IN.-Datei vorhanden sein.
- Obwohl S.IN.-Dateien mit einem Kennwort (EXEC-PASSWORD) geschützt sind, ist es möglich, sie durch /DELETE-FILE zu löschen, ohne das Kennwort vorher anzugeben. Auf diese Weise ist es möglich, nicht mehr benötigte, bzw. nicht systemseitig gelöschte S.IN.-Dateien aus dem System zu entfernen.
- ENTER-Dateien können mit Kennwörtern gegen Lesen (READ-PASSWORD), Überschreiben (WRITE-PASSWORD) und Ausführen (EXEC-PASSWORD) geschützt werden (Kommando CREATE-FILE). Das EXEC-PASSWORD oder ein höherwertiges Kennwort muss im Operanden FPASS angegeben werden, wenn ein ENTER-Makro ausgeführt wird. Das WRITE-PASSWORD muss angegeben werden, wenn die Datei nach der Ausführung gelöscht werden soll (Operand ERASE=YES).
  Die Kennwörter werden auf Richtigkeit geprüft, sobald der ENTER-Aufruf bearbeitet wird. Ändert danach ein Benutzer die Kennwörter, so gilt noch die erfolgreiche Prüfung, und die Datei wird ausgeführt.
- ENTER-Dateien können SAM- oder ISAM-Dateien sein, und zwar mit variabler Satzlänge (RECFORM=V). Pro Datensatz werden 72 Zeichen interpretiert. Bei ISAM-Dateien kann das Schlüsselfeld an beliebiger Stelle im Datensatz stehen, da es ausgeblendet wird.

Hinweise zur Auftragsüberwachung (siehe auch Handbuch „JV“ [22])

- Zum ENTER-Zeitpunkt wird die Zustandsanzeige von „jvname“ auf „$S“, die TSN-Anzeige auf die zum Auftrag gehörende Auftragsnummer und die Prozessor-Anzeige auf die Katalogkennung des Prozessors gesetzt, der den Auftrag ausführt.
- Wenn auf `jvname` zum Zeitpunkt der Makroverarbeitung nicht zugegriffen werden kann, dann wird der Aufruf abgewiesen. Wenn erst später (der ENTER-Job will einen Wert eintragen) nicht auf die Jobvariable zugegriffen werden kann, dann gibt der Job eine Fehlermeldung nach SYSOUT aus und läuft normal weiter.
- Sowohl die Benutzerkennung, von der die überwachende Jobvariable abgesetzt wird, als auch die Benutzerkennung, für die der Auftrag bearbeitet wird, müssen Zugriff zu `jvname` haben.
- JVPASS bezeichnet das Kennwort, entsprechend der Kennwort-Hierarchie, für den Zugriff auf die überwachende Jobvariable. Das Kennwort muss im **ENTER**-Makro angegeben werden, wenn Auftragsverteilung (siehe Handbuch „HIPLEX MSCF“ [26]) gefordert wird. Ohne Auftragsverteilung kann das Kennwort auch über ein separates ADD-PASSWORD-Kommando gegeben werden.
- Für den Zugriff auf die überwachende Jobvariable gelten die gleichen Regeln wie für den Zugriff auf die ENTER-Datei.

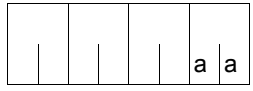
Hinweise zur Auftragsverteilung (siehe auch Handbuch „HIPLEX MSCF“ [26])

- „hostid“ muss ein aktives System des MRS-Netzes bezeichnen, andernfalls wird der **ENTER**-Aufruf abgewiesen.
- „jvname1“ muss die „hostid“ eines aktiven Systems des MRS-Netzes enthalten, andernfalls wird der **ENTER**-Aufruf abgewiesen.
- „catid“ muss einen (im MRS-Netz) bekannten und zugreifbaren Katalog bezeichnen, andernfalls wird der **ENTER**-Aufruf abgewiesen.
- „jvname2“ muss die „catid“ eines (im MRS-Netz) bekannten und zugreifbaren Katalogs enthalten, andernfalls wird der **ENTER**-Aufruf abgewiesen.
- Werden die Operanden HOST und CAT spezifiziert, so wird der Wert des HOST-Operanden zur Bestimmung des Ziel-Systems benutzt.
- Alle Kennwörter (sowohl FPASS und CRPASS für die ENTER-Datei als auch JVPASS für die MONJV) müssen im **ENTER**-Makro angegeben werden, wenn Auftragsverteilung gefordert wird; ohne Auftragsverteilung kann das Kennwort auch über ein separates ADD-(CRYPTO-)PASSWORD-Kommando gegeben werden.

Rückinformation und Fehleranzeigen

Meldungen bezüglich der Kommando-Bearbeitung sind Bestandteil des SYSOUT-Protokolls, das auf Wunsch dem Programm übergeben wird.

R15:



Über die Ausführung des Makros ENTER wird im rechtsbündigen Byte des Registers R15 ein Returncode übergeben.

X'aa'	**Erläuterung**
X'00'	Normale Beendigung
X'04'	Ungenügender Speicherbereich ist vorhanden, die Anforderung wurde nicht bearbeitet.
X'08'	Fehler im Datenbereich (Adressbereich)
X'0C'	Der letzte Ausgabesatz, der in den Benutzerbereich gebracht wurde, ist abgeschnitten.
X'10'	Makroaufruf/Kommando-Fehler (das Kommando gab dem MCLP einen Fehler zurück), z.B. Datei nicht katalogisiert (DMS0D33) oder fehlerhafter Operand (JMS0021).

ETABIT – Eintrag für Symboltabelle erzeugen oder ändern

Allgemeines

Anwendungsgebiet: Binden und Laden; siehe Seite 47
Makrotyp: S-Typ, MF-Format **2**: Standardform/C-/D-/L-/M-Form; siehe Seite 29

Zum dynamischen Bindelader DBL siehe auch Handbuch „BLSSERV“ [4].

Makrobeschreibung

Der Makroaufruf **ETABIT** erzeugt einen Eintrag für eine Symboltabelle, die beim Makroaufruf **ETABLE** an den DBL übergeben wird.

Makroaufrufformat und Operandenbeschreibung

ETABIT
MF=D / C / L / M
,AMODE=*NOT-SPECIFIED / *31 / *24 / *ANY
,HSI_CODE=*BY-SYSTEM / *390 / *RISC / *SPARC / *X86E [1]
,INVISIBLE=*NOT-SPECIFIED / *NO / *YES
,LEN=0 / <integer 0..2147473647>
,LOAD_ADDR=NULL-1 / <var: pointer>
,PAGE_ALIGNED=*NOT-SPECIFIED / *NO / *YES
,PRIVILEGED=*NOT-SPECIFIED / *NO / *YES
,PUBLIC=*NOT-SPECIFIED / *NO / *YES
,READ_ONLY=*NOT-SPECIFIED / *NO / *YES
,RESIDENT=*NOT-SPECIFIED / *NO / *YES
,SYMBOL_NAME='␣' / <c-string 1..32> / <var: char 1..32>
,SYMBOL_TYPE=*NOT-SPECIFIED / *CSECT / *ENTRY / *COMMON
,PARAM=<var: pointer> / (reg: pointer>)
,PREFIX=P / p
,MACID=BET / macid

[1] Die Operandenwerte *RISC und *SPARC sind seit BS2000/OSD-BC V9.0 bedeutungslos

Der Wert *NOT-SPECIFIED führt beim Erzeugen eines Eintrags dazu, dass die Attribute des Symbols mit 'FALSE' initialisiert werden.

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

AMODE=*NOT-SPECIFIED / *31 / *24 / *ANY
Wert des Attributes AMODE

HSI_CODE=*BY-SYSTEM / *390 / *X86E
Anzugeben ist der Code-Typ, auf den das Symbol verweist (*390, *X86E). Default-Wert ist der Typ des Servers, auf der das Benutzerprogramm abläuft. Es sollte jedoch immer der richtige HSI-Code eingetragen werden, damit es nicht zur unerwünschten und eventuell falschen Aktualisierung beim Makroaufruf `ETABLE ACTION=*UPDATE` kommt.

INVISIBLE=*NOT-SPECIFIED / *NO / *YES
Wert des Attributes INVISIBLE

LEN=0 / <integer 0..2147483647>
Länge des Elements.
Für CSECTs und COMMON-Bereiche muss LEN größer als 0 sein.
Für ENTRYs muss LEN gleich 0 sein.

LOAD_ADDR=NULL-1 / <var: pointer>
Ladeadresse des Symbols.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. angegebenen Operanden PARAM, PREFIX und MACID siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

PAGE_ALIGNED=*NOT-SPECIFIED / *NO / *YES
Wert des Attributes PAGE_ALIGNED

PRIVILEGED=*NOT-SPECIFIED / *NO / *YES
Wert des Attributes PRIVILEGED

PUBLIC=*NOT-SPECIFIED / *NO / *YES
Wert des Attributes PUBLIC

READ_ONLY=*NOT-SPECIFIED / *NO / *YES
Wert des Attributes READ_ONLY

RESIDENT=*NOT-SPECIFIED / *NO / *YES
Wert des Attributes RESIDENT

SYMBOL_NAME=' ' / <c-string 1..32> / <var: char 1..32>
Name des Symbols

SYMBOL_TYPE=*NOT-SPECIFIED / *CSECT / *ENTRY / *COMMON
Symboltyp, der in die Symboltabelle eingetragen wird.

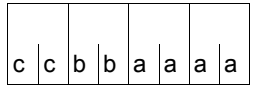
Hinweise zum Makroaufruf

– Wird der Parameter HSI_CODE nicht angegeben, so gilt für ihn der Default-Wert *BY-SYSTEM. Wenn dann der Makro **ETABLE** mit ACTION=*UPDATE aufgerufen wird, kann es in manchen Fällen zu einer unerwünschten Aktualisierung des HSI-Codes kommen. Um das zu verhindern, sollte der HSI-Code des Symbols immer angegeben werden.

– Für den Makroaufruf **ETABIT** gelten zudem die Hinweise, die beim Makro **ETABLE** beschrieben sind. Das Layout eines Tabelleneintrags ist ebenfalls dort beschrieben.

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros ETABIT wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Der Makro wurde normal ausgeführt.
X'01'	X'00'	X'0000'	Dasselbe ETABLE-Symbol existiert bereits.
X'60'	X'01'	X'0006'	Fehler beim Adressierungsmodus.
X'60'	X'01'	X'0007'	Ungültige Längenangabe.
X'60'	X'01'	X'0009'	Ungültiger Attributwert.
X'60'	X'01'	X'000B'	Ungültiger Wert bei HSI.
X'60'	X'01'	X'000C'	Der Symboltabelleneintrag für ETABLE ist ungültig.
X'60'	X'01'	X'000D'	SYMBOL-TYPE wurde beim Erzeugen des Eintrags nicht angegeben.
X'60'	X'01'	X'002C'	Ungültiger Symbolname.
X'60'	X'01'	X'0060'	Symbol nicht gefunden.
X'60'	X'01'	X'0130'	Ungültiger Operand LOAD_ADDR.
X'60'	X'01'	X'0150'	Fehler bei Symboltyp.
X'60'	X'01'	X'0151'	Ein mit ETABLE generiertes Symbol mit diesem Namen existiert bereits.
X'60'	X'01'	X'0152'	Ein nicht mit ETABLE generiertes Symbol mit diesem Namen existiert bereits, und die gewünschte Aktion ist für ein solches Symbol nicht erlaubt (siehe Seite 518).
X'00'	X'01'	X'FFFF'	Die Funktion wird nicht mehr oder noch nicht unterstützt.
X'00'	X'03'	X'FFFF'	Die Version der Schnittstelle wird nicht unterstützt.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

ETABLE – Ladeinformation übergeben

Allgemeines

Anwendungsgebiet: Binden und Laden; siehe Seite 47
Makrotyp: S-Typ, MF-Format **2**: Standardform/C-/D-/L-/E-/M-Form siehe Seite 29

Zum dynamischen Bindelader DBL siehe auch Handbuch „BLSSERV“ [4].

Makrobeschreibung

Mit dem Makroaufruf **ETABLE** übergibt das Benutzerprogramm dem DBL eine Symboltabelle, die in die Symboltabelle des angegebenen Kontextes eingemischt wird. Die übergebene Tabelle informiert den DBL über Namen und Attribute von CSECTs, ENTRYs und COMMON-Bereichen des Benutzerprogramms.

Der Makroaufruf ist nicht kompatibel zu dem früheren Makroaufruf TABLE.

Makroaufrufformat und Operandenbeschreibung

```
ETABLE

MF=S / D / C / E / L / M
,ACTION=*CREATE / *UPDATE / *DELETE
,CONTEXT_NAME='␣'  / <char 1..32>
,CONTEXT_STATE=*DBL-OPTIONS / *ANY / *NEW / *OLD
,TABLE_ADDRESS=NULL-1 / <var: pointer>
,TABLE_LENGTH=0 / <integer 0..2147483647>
,PARAM=<var: pointer> / (reg: pointer>)
,PREFIX=P / p
,MACID=BEI / macid
```

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

ACTION=
Aktion, die für die einzelnen Symbole in der übergebenen Tabelle ausgeführt werden soll.

***CREATE**
Die Symbole werden in die Symboltabelle des Kontextes eingetragen. Sie dürfen darin noch nicht enthalten sein.

***UPDATE**
Die Symbole werden mit den neuen Eigenschaften in die Symboltabelle des Kontextes eingetragen. Sie müssen bereits darin enthalten sein. Hierbei wird auch die Sichtbarkeit von Symbolen geändert, die nicht mit einem vorangegangenen Aufruf des **ETABLE**-Makros eingetragen wurden.

***DELETE**
Die Symbole sind aus der Symboltabelle des Kontextes zu löschen.

CONTEXT_NAME='␣' / <char 1..32>
Name des Kontextes, dem die Symbole zugeordnet werden. Das erste Zeichen muss ein Buchstabe sein.

CONTEXT_STATE=
Status des bei CONTEXT_NAME angegebenen Kontextes

***DBL-OPTIONS**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt der Wert, der in der Makro-Syntaxbeschreibung auf *DBL-OPTIONS folgt.

***ANY**
Wenn der Kontext bereits existiert, wird dieser verwendet; ansonsten wird ein neuer Kontext erzeugt.

***NEW**
Der Kontext wird angelegt. Er darf noch nicht existieren.

***OLD**
Der Kontext existiert bereits.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. angegebenen Operanden PARAM, PREFIX und MACID siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

TABLE_ADDRESS=NULL-1 / <var: pointer>
Adresse einer Tabelle von Symbolen, die an DBL übergeben werden soll.

TABLE_LENGTH=0 / <integer 0..2147483647>
Tatsächliche Länge der Tabelle (in Byte)

Hinweise zum Makroaufruf

- Typ und Name eines Symbols können mit **ETABLE** nicht geändert werden.
- Die Adresse im Feld LOAD_ADDR muss eine gültige Klasse-6-Speicher-Adresse sein.
- Ein Symbol, das mit **ETABLE** in die Symboltabelle des Kontextes eingetragen wurde, kann nicht mit dem Makro UNBIND aus der Symboltabelle gelöscht werden. Symbole, die mit den Kommandos LOAD-/START-EXECUTABLE-PROGRAM (oder LOAD-/START-PROGRAM) oder mit den Makros BIND und TABLE eingeführt wurden, können dagegen nicht mit **ETABLE** ACTION=*DELETE gelöscht werden.
- Bei ACTION=*DELETE wird der Operand CONTEXT_STATE ignoriert. Nur die Symbolnamen werden beachtet.
- Symbole, die mit **ETABLE** in die Symboltabelle eingetragen wurden, liegen in der Verantwortung des Benutzers und sind nur für die aktuelle Task sichtbar.
- Wenn ein COMMON-Bereich mit **ETABLE** eingeführt wird, geht der DBL davon aus, dass der COMMON-Bereich bereits geladen ist und das Feld LOAD_ADDR auf einen zugewiesenen Speicherbereich verweist.
- Wenn ACTION=UPDATE auf ein Symbol angewandt wird, das nicht von ETABLE erzeugt wurde, werden bei der Bearbeitung nur Symbolname, Symboltyp und das Attribut INVISIBLE berücksichtigt.
- Für ein Symbol, das nicht von **ETABLE** erzeugt wurde, kann nur die Sichtbarkeit aktualisiert werden.
- Da die Sichtbarkeit auch für Symbole aktualisiert werden kann, die nicht von **ETABLE** erzeugt wurden, wird grundsätzlich die Sichtbarkeit des ersten gefundenen Symbols aktualisiert.

Format der Symboltabelle

Die bei TABLE_ADDRESS angegebene Adresse zeigt auf eine Symboltabelle, die mehrere Einträge haben kann. Ein einzelner Eintrag hat folgendes Format:

Byte	Länge	Feldname	Bedeutung und/oder Wert
0	8	HDR	Standardheader
8	1	TYPE	/ CSECT (X'F0') oder / ENTRY (X'F1') oder / COMMON (X'F3')
9	7	ATTRIBUTE	Attribute wie bei CSECTs (je 1 Byte pro Attribut mit folgender Bedeutung): 1. INVISIBLE: X'01' ≙ NO, X'02' ≙ YES 2. AMODE: X'01' ≙ 31, X'02' ≙ 24, X'03' ≙ ANY 3. RESIDENT: X'01' ≙ NO, X'02' ≙ YES 4. PAGE_ALIGNED X'01' ≙ NO, X'02' ≙ YES 5. READ_ONLY: X'01' ≙ NO, X'02' ≙ YES 6. PUBLIC: X'01' ≙ NO, X'02' ≙ YES 7. PRIVILEGED: X'01' ≙ NO, X'02' ≙ YES
16	4	LOAD_ADDR	Ladeadresse des Symbols
20	2	LEN	Länge der CSECT / des COMMON-Bereiches
22	32	SYMBOL_NAME	Name des Symbols
54	1	HSI_CODE	390 (X'01') oder X86E (X'09')
55	1	MMODE	TU_4K_DEPENDENT (X'02') oder COMPATIBLE (X'03) oder NATIVE (X'04')

Die DSECT für einen solchen Symboleintrag wird mit `ETABIT MF=D` erzeugt.

Behandlung von Namenskonflikten

Die folgende Tabelle zeigt, wie Namenskonflikte zwischen Symbolen behandelt werden, die durch die Makros **ETABLE**, **TABLE** und **BIND** eingeführt wurden:

	Existierendes Symbol wurde eingeführt durch		
ACTION	ETABLE	TABLE	BIND
*CREATE	(1)	(2)	(2)
*UPDATE	(3)	(4)	(4)
*DELETE	(5)	(6)	(6)

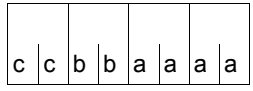
1. Ein vorhandenes **ETABLE**-Symbol hat den gleichen Namen.
   Folgender Returncode wird übergeben: ETABLE_SYMB_DUPLICATE
2. Ein Namenskonflikt tritt auf. Folgender Returncode wird übergeben:
   ETABLE_NAME_COLLISION
3. Das Symbol wird geändert (kein Namenskonflikt).
4. Nur die Sichtbarkeit (visibility) des Symbols ist aktualisierbar.
5. Das Symbol wird gelöscht (kein Namenskonflikt).
6. Unzulässige Aktion.

Rückinformation und Fehleranzeigen

Wenn bei der Verarbeitung der Tabelleneinträge ein Fehler auftritt, wird im HDR-Feld des fehlerhaften Tabelleneintrags ein entsprechender Returncode eingetragen. Im Standardheader der Parameterliste wird der Returncode FUNCTION_PARTIALLY_PROCESSED übergeben und die Verarbeitung der Tabelleneinträge wird fortgesetzt.

Das Feld PROCESSED_ITEMS der Parameterliste enthält die Anzahl korrekt bearbeiteter Einträge.

Standard-header:



Über die Ausführung des Makros ETABLE wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Der Makro wurde normal ausgeführt.
X'01'	X'00'	X'0000'	Die Funktion wurde bereits ausgeführt.
X'02'	X'00'	X'0001'	Die Funktion wurde teilweise ausgeführt.
X'60'	X'01'	X'0001'	Ungültiger Operand TABLE_LENGTH
X'60'	X'01'	X'0002'	Ungültige Angaben bei TABLE_ADDRESS oder TABLE_LENGTH.
X'60'	X'01'	X'0008'	Ungültiger Operand ACTION
X'60'	X'01'	X'000A'	Interner ETABLE-Fehler.
X'60'	X'01'	X'0019'	Ein reserviertes Feld enthält keine binären Nullen.
X'60'	X'01'	X'0040'	Der Kontext ist noch nicht vorhanden und es wurde CONTEXT_STATE=*OLD angegeben.
X'60'	X'01'	X'0048'	Der Kontext ist bereits vorhanden und es wurde CONTEXT_STATE=*NEW angegeben.
X'60'	X'01'	X'0134'	Ungültiger Parameter CONTEXT_STATE
X'60'	X'01'	X'0148'	Ungültiger Kontextname
X'60'	X'40'	X'0158'	Maximale Anzahl von Benutzerkontexten ist erreicht.
X'60'	X'20'	X'0200'	Interner DBL-Fehler
X'60'	X'20'	X'0300'	Systemfehler
X'00'	X'01'	X'FFFF'	Die Funktion wird nicht mehr oder noch nicht unterstützt.
X'00'	X'03'	X'FFFF'	Die Version der Schnittstelle wird nicht unterstützt.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Beispiel

Dieses Beispiel soll die Anwendung der ETABLE/ETABIT-Schnittstellen verdeutlichen:

```
* DSECT of ETABLE item
*
         ETABIT MF=D
         ...
* Inititialisierung der ETABLE Parameterliste
*
         MVC   ETAPL(PBEI#),ETAPLI
         LA    1,ETATAB
         LA    2,ETATABL
         ETABLE MF=M,TABLE_ADDRESS=(1),TABLE_LENGTH=(2)
*
* Initialisierung der Eintraege
*
         USING PBETDS,1                reg 1 = Adresse des ersten Eintrags
*                                      PBETDS = DSECT fuer den Eintrag
*
* Erster Eintrag
*
         MVC   0(PBET#,1),ETAITEMI                Eintrag initialisieren
         ETABIT MF=M,SYMBOL_NAME=CS1NAM,LEN=CS1L,
               LOAD_ADDR=CS1@,SYMBOL_TYPE=*COMMON
*
* Zweiter Eintrag
*
         LA    1,PBET#(1)              reg 1 = Adresse des zweiten Eintrags
         L     2,CS2@
         MVC   0(PBET#,1),ETAITEMI               Eintrag initialisieren
         ETABIT MF=M,SYMBOL_NAME='CS2',LEN=200,
               LOAD_ADDR=(2),SYMBOL_TYPE=*CSECT
*
* ETABLE aufrufen
*
         ETABLE MF=E,PARAM=ETAPL
*
* Returncode-Auswertung
*
         ...
*
* ETABLE-Tabelle
*
ETATAB   DS    XL(2*PBET#)    Tabelle fuer zwei Eintraege
ETATABL  EQU   *-ETATAB       Tabellenlaenge
         DS    0F
CS1@     DS    F              Adresse des ersten eingefuegten Symbols
```

```
CS1NAM   DS    CL32           Name des ersten eingefuegten Symbols
CS1L     DS    Y              Laenge des ersten eingefuegten Symbols
CS2@     DS    F              Adresse des zweiten eingefuegten Symbols
*
ETAPL    ETABLE MF=C          Parameterliste
*
* Struktur zur Initialisierung der ETABLE-Parameterliste
*
ETAPLI   ETABLE MF=L,ACTION=*CREATE,CONTEXT_STATE=*NEW,
              CONTEXT_NAME='ETACTX'
*
* Struktur zur Initialisierung eines ETABLE-Eintrags
*
 ETAITEMI ETABIT MF=L
```

EXIT – STXIT-Prozess (-Routine) beenden

Allgemeines

Anwendungsgebiete: Starten, Unterbrechen und Beenden; Seite 72
STXIT-Verfahren; siehe Seite 133
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

Programmunterbrechungen können mit STXIT-Routinen behandelt werden. Diese laufen als eigenständige Prozesse (eigener PCB, eigene Verarbeitungsebene) ab. Über die Ausführung des Makros **EXIT** wird kein Returncode übergeben.

Makrobeschreibung

Der Makro **EXIT** beendet einen STXIT-Prozess.
Es kann angegeben werden, ob eine weitere, der gleichen Ereignisklasse zugeordnete STXIT-Routine gestartet werden soll. Ist das nicht der Fall, wird der unterbrochene Prozess fortgesetzt oder das Programm beendet.

Hinweise

– Ist die STXIT-Routine der Ereignisklasse „ABEND“ oder „Normale Programmbeendigung“ zugeordnet, wird nach dem **EXIT**-Aufruf das Programm durch das System beendet, wenn alle STXIT-Routinen für diese STXIT-Ereignisklassen abgelaufen sind (bei CONTINU=YES) und das Ereignis (z.B. LOGOFF(ABEND)) vom System beendet wurde.
– Eine der Ereignisklasse „Ende der Programmlaufzeit“ zugeordnete STXIT-Routine sollte mit dem Makro **TERM** beendet werden.

Makroaufrufformat und Operandenbeschreibung

```
EXIT

CONTINU=YES / NO / STD

,TERM= { NO                          }
       { (PRGR[,DUMP]  ,NORMAL])     }
       { (STEP[,DUMP] [,NORMAL])     }

[,MF=L / (E,..)]
```

CONTINU=
beschreibt, ob eine weitere STXIT-Routine derselben Ereignisklasse gestartet werden soll.

YES
Eine weitere STXIT-Routine wird gestartet.

NO
Es wird keine weitere STXIT-Routine gestartet.

STD
Diese Option entspricht normalerweise CONTINU=YES. Eine abweichende Bedeutung hat sie nur, wenn alle der folgenden drei Bedingungen erfüllt sind:

– Die STXIT-Klasse ist 'PROGRAM CHECK' oder 'UNRECOVERABLE PROGRAM ERROR',

– alle bisherigen STXIT-Routinen dieser STXIT-Klasse haben sich mit CONTINU=STD beendet und

– es existieren keine weiteren STXIT-Routinen derselben STXIT-Klasse.

In diesem Fall wird das Programm so fortgesetzt, als hätte es diese STXIT-Routinen nicht gegeben, d.h. die für den aufgetretenen Fehler vorgesehene Fehlerbehandlung wird gestartet.

TERM=
beschreibt, ob das Programm beendet werden soll, wenn keine weiteren STXIT-Routinen derselben STXIT-Ereignisklasse in dem Programm aktiviert sind.

NO
Das Programm soll nicht beendet werden.

(PRGR[,DUMP][,NORMAL])
Das Programm wird beendet.
Bei Angabe von DUMP wird zusätzlich ein Speicherabzug ausgegeben (User Dump).
Bei Angabe von NORMAL wird das Programm normal beendet.
Voreinstellung ist ABNORMAL, d.h. das Programm wird abnormal beendet.

(STEP[,DUMP][,NORMAL])
In einer Dialog-Task wird das Programm beendet.
In einer Batch-Task wird außerdem zum nächsten SET-JOB-STEP- oder EXIT-JOB-Kommando verzweigt.
Bei Angabe von DUMP wird zusätzlich ein Speicherabzug ausgegeben (User Dump).
Bei Angabe von NORMAL wird das Programm normal beendet.
Voreinstellung ist ABNORMAL, d.h. das Programm wird abnormal beendet.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Beispiel siehe Abschnitt „STXIT-Verfahren mit Contingency-Verarbeitung“ auf Seite 133.

GCCSN – CCS-Namen für Kommando- und Dateneingabe anzeigen

Allgemeines

Anwendungsgebiete:	Ein-/Ausgabe von Dateien und Sätzen; siehe Seite 159 Abfragen und Zugriff zu Tabellen und Listen; siehe Seite 158
Makrotyp:	S-Typ, MF-Format **3**: D-/C-/S-/E-/L-Form; siehe Seite 29

Makrobeschreibung

Mit dem Makro **GCCSN** kann sich der Anwender den Namen des aktuell gültigen Zeichensatzes (Coded Character Set Name, Name der Codiertabelle) für die Ein-/Ausgabe von Kommandos oder Daten anzeigen lassen.
Der Name des aktuell gültigen Zeichensatzes (Codiertabelle) ist abhängig von der Eingabequelle oder dem Ausgabeziel:

– SYSDTA/SYSCMD/SYSOUT ist einer Datensichtstation zugewiesen:
Der Codiertabellen-Name wird von VTSU bestimmt. Der Makro **GCCSN** erhält den Namen (intern) aus dem Feld ACTCH des Datenbereichs vom **TSTAT**-Makro.

– SYSDTA/SYSCMD/SYSOUT/SYSLST ist einer S-Variablen zugeordnet:
Dialogbetrieb: Entspricht der Zuweisung zu einer Datensichtstation.
Batch-Betrieb: Es wird der Codiertabellen-Name „EDF03IRV“ angezeigt.

– SYSDTA/SYSCMD/SYSOUT/SYSLST ist einem Element einer PLAM-Bibliothek zugeordnet:
Es wird der Codiertabellen-Name des Bibliothekselements angezeigt.
Wenn dem Bibliothekselement keine Codiertabelle zugeordnet ist, dann wird der Codiertabellen-Name „EDF03IRV“ angezeigt.

– SYSCMD ist einer Datei zugewiesen:
Es wird der im Katalogeintrag der Datei stehende Codiertabellen-Name angezeigt. Als Dateien gelten katalogisierte Dateien, S-Prozeduren und Nicht-S-Prozeduren.

– SYSDTA/SYSOUT/SYSLST ist einer Datei zugewiesen:
Es wird der im Katalogeintrag der Datei stehende Codiertabellen-Name angezeigt. Als Dateien gelten katalogisierte Dateien, S-Prozeduren und Nicht-S-Prozeduren (bei SYSDTA=(SYSCMD)).
Der Name der Codiertabelle für SYSOUT/SYSLST kann mit den Kommandos `/ASSIGN-SYSOUT/SYSLST CODED-CHARACTER-SET=` zugeordnet werden.
Wenn keine Codiertabelle zugeordnet ist, dann wird der Standard-Name des Benutzer-Katalogeintrags angezeigt.

Nach der Initialisierung des Datenbereichs (MF=L) wird der Funktionsaufruf (MF=E) abgesetzt. Wird der Aufruf ohne Fehler ausgeführt, steht im Feld <PREFIX><MACID>CCSN des Datenbereichs der Codiertabellen-Name der abgefragten Systemdatei.

Makroaufrufformat und Operandenbeschreibung

GCCSN
STREAM=SYSDTA / SYSCMD / SYSOUT / SYSLST ,MF=D / C / S / E / L [,PARAM=adr / (r)] ,PREFIX=C / p ,MACID=CSN / macid

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form oder D-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).
Es ist außerdem möglich, diesen Makro mit MF=S aufzurufen.

STREAM=
bestimmt die Systemdatei, deren Codiertabellen-Name ausgegeben werden soll.

SYSDTA
Der Name der Codiertabelle von SYSDTA soll angezeigt werden.

SYSCMD
der Name der Codiertabelle von SYSCMD soll angezeigt werden.

SYSOUT
der Name der Codiertabelle von SYSOUT soll angezeigt werden.

SYSLST
der Name der Codiertabelle von SYSLST soll angezeigt werden.

Hinweise

– Ist SYSDTA einer Datensichtstation zugeordnet, kann der Anwender den Codiertabellen-Namen in einem Programm (im VTSU-Control-Block des **RDATA**-Makros) dynamisch ändern.

Dabei ist die Änderung nur während dieser **RDATA**-Ein-/Ausgabe gültig und hat keinen Einfluss auf die Ausgabe des **GCCSN**-Makros, der in diesem Fall immer den im Benutzerkatalog des Anwenders voreingestellten Codiertabellen-Namen anzeigt.

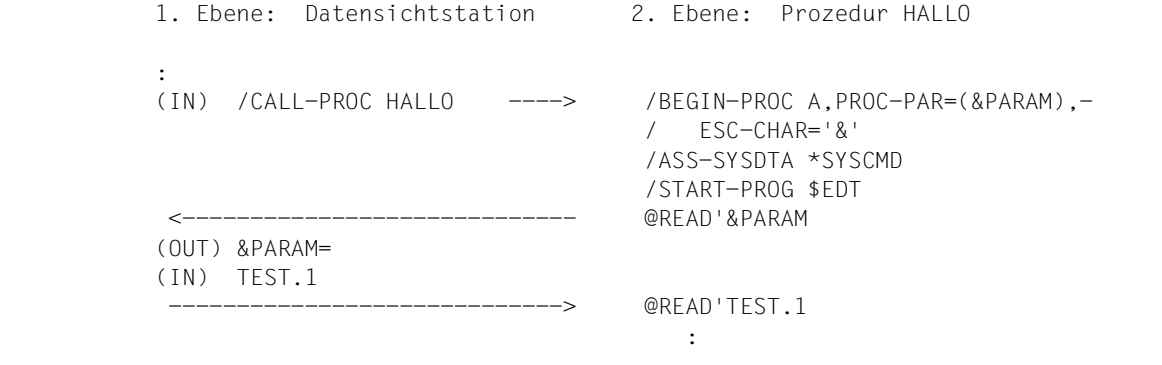
– Ist SYSCMD einer Datensichtstation zugeordnet, kann der *privilegierte* Anwender den Codiertabellen-Namen dynamisch ändern (im VTSU-Control-Block des **WRTRD**-Makros).
Dabei ist die Änderung nur während dieser **WRTRD**-Ein-/Ausgabe gültig und hat keinen Einfluss auf die Ausgabe des **GCCSN**-Makros, der in diesem Fall immer den im Benutzerkatalog des Anwenders voreingestellten Codiertabellen-Namen anzeigt.

– Ist SYSDTA/SYSCMD/SYSOUT/SYSLST einer Datei oder einem Bibliothekselement zugeordnet, ändert sich der Codiertabellen-Name zwischen dem Öffnen (OPEN) und Schließen (CLOSE) der Datei bzw. des Bibliothekselements nicht.

– Bei der Unterbrechung von Prozeduren mit K2 ist der aktuelle SYSDTA-Codiertabellen-Name nicht der der Datensichtstation, sondern der der soeben unterbrochenen Prozedur.

– Wird während der Abarbeitung einer Prozedur ein Prozedurparameter an der Datensichtstation durch Prompting angefordert, ist für den festen (anfordernden) Bestandteil der Kommando-/Anweisungszeile der für diese Prozedur aktuelle Codiertabellen-Name gültig. Für den eingegebenen Operandenwert ist der Codiertabellen-Name der Datensichtstation gültig.

Beispiel



Die Zeichenfolge `@READ'` wird mit dem Codiertabellen-Namen der Prozedur HALLO gelesen, die Zeichenfolge `TEST.1` mit dem der Datensichtstation.

– Mit dem Makroaufruf **RDATA** ..,A kann sich der Anwender über jede Änderung der SYSDTA-Zuweisung informieren.

Layout der CSECT

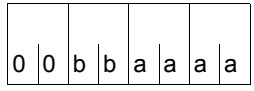
```
           GCCSN MF=C
1          #INTF REFTYPE=REQUEST,INTNAME=GCCSN,INTCOMP=002
1          MFCHK MF=C,SUPPORT=(C,D,S,L,E),PREFIX=C,                      C
1                MACID=CSN,DMACID=CSN,                                   C
1                PARAM=,ALIGN=F,SVC=39
2          DS    0F
2                *,##### PREFIX=C, MACID=CSN #####
1 CCSNCS   FHDR  MF=(C,CCSN)
2 CCSNCS   DS    0A
2 CCSNFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
2 CCSNIFID DS    0A              0   INTERFACE IDENTIFIER
2 CCSNFCTU DS    AL2             0   FUNCTION UNIT NUMBER
2 *                                  BIT 15    HEADER FLAG BIT,
2 *                                  MUST BE RESET UNTIL FURTHER NOTICE
2 *                                  BIT 14-12 UNUSED, MUST BE RESET
2 *                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
2 CCSNFCT  DS    AL1             2   FUNCTION NUMBER
2 CCSNFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 CCSNRET  DS    0A              4   GENERAL RETURN CODE
2 *
2 * GENERAL_RETURN_CODE CLEARED (X'00000000') MEANS
2 * REQUEST SUCCESSFUL PROCESSED AND NO ADDITIONAL INFORMATION
2 *
2 CCSNSRET DS    0AL2            4   SUB RETURN CODE
2 CCSNSR2  DS    AL1             4   SUB RETURN CODE 2
2 * ALWAYS CLEARED (X'00') IF MAIN_RETURN_CODE IS X'FFFF'
2 * Standard subcode2 values as defined by convention:
2 CCSNR2OK EQU   X'00'               All correct, no additional info
2 CCSNR2NA EQU   X'01'               Successful, no action was necessary
2 CCSNR2WA EQU   X'02'               Warning, particular situation
2 CCSNSR1  DS    AL1             5   SUB RETURN CODE 1
2 *
2 * GENERAL INDICATION OF ERROR CLASSES
2 *
2 * CLASS A    X'00'           FUNCTION WAS SUCCESSFULLY PROCESSED
2 * CLASS B    X'01' - X'1F'   PARAMETER SYNTAX ERROR
2 * CLASS C    X'20'           INTERNAL ERROR IN CALLED FUNCTION
2 * CLASS D    X'40' - X'7F'   NO CLASS SPECIFIC REACTION POSSIBLE
2 * CLASS E    X'80' - X'82'   WAIT AND RETRY
2 *
2 CCSNRFSP EQU   X'00'               FUNCTION SUCCESSFULLY PROCESSED
2 CCSNRPER EQU   X'01'               PARAMETER SYNTAX ERROR
2 *  3 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'01' - X'1F'
2 CCSNRFNS EQU   X'01'               CALLED FUNCTION NOT SUPPORTED
```

```
2 CCSNRFNA EQU   X'02'               CALLED FUNCTION NOT AVAILABLE
2 CCSNRVNA EQU   X'03'               INTERFACE VERSION NOT SUPPORTED
2 *
2 CCSNRAER EQU   X'04'               ALIGNMENT ERROR
2 CCSNRIER EQU   X'20'               INTERNAL ERROR
2 CCSNRCAR EQU   X'40'               CORRECT AND RETRY
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'40' - X'7F'
2 CCSNRECR EQU   X'41'               SUBSYSTEM (SS) MUST BE CREATED
2 *                                  EXPLICITELY BY CREATE-SS
2 CCSNRECN EQU   X'42'               SS MUST BE EXPLICITELY CONNECTED
2 *
2 CCSNRWAR EQU   X'80'               WAIT FOR A SHORT TIME AND RETRY
2 CCSNRWLR EQU   X'81'                    "     LONG       "
2 CCSNRWUR EQU   X'82'               WAIT TIME IS UNCALCULABLY LONG
2 *                                  BUT RETRY IS POSSIBLE
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'80' - X'82'
2 CCSNRTNA EQU   X'81'               SS TEMPORARILY NOT AVAILABLE
2 CCSNRDH  EQU   X'82'               SS IN DELETE / HOLD
2 *
2 CCSNMRET DS    0AL2            6   MAIN RETURN CODE
2 CCSNMR2  DS    AL1             6   MAIN RETURN CODE 2
2 CCSNMR1  DS    AL1             7   MAIN RETURN CODE 1
2 *
2 * SPECIAL LAYOUT OF LINKAGE_MAIN_RETURN_CODE (YYYY IN X'00XXYYYY')
2 *
2 CCSNRLNK EQU   X'FFFF'              LINKAGE ERROR / REQ. NOT PROCESSED
2 CCSNFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 CCSNFLAG   DS    X                 STREAM IDENTIFIER
1 CCSNDTA    EQU   X'01'             SYSDTA
1 CCSNCMD    EQU   X'02'             SYSCMD
1 CCSNOUT    EQU   X'03'             SYSOUT
1 CCSNLST    EQU   X'04'             SYSLST
1 CCSNRES1   DS    CL3               RESERVED
1 CCSNRES2   DS    A                 RESERVED
1 CCSNRES3   DS    A                 RESERVED
1 CCSNCCSN   DS    CL8               CODED CHARACTER SET NAME
1 CCSNPAR    EQU   X'01'             SC1 RC : PARAMETER ERROR
1 CCSNERR    EQU   X'20'             SC1 RC : INTERNAL ERROR
1 CCSN#      EQU   *-CCSNCS          PARAMETER LIST LENGTH
1          SPACE 2
```

Rückinformation und Fehleranzeigen

Während der Makrobearbeitung erhält Register R1 die Adresse des Datenbereichs. Im Feld `<PREFIX><MACID>CCSN` des Datenbereichs wird der Name der Codiertabelle übergeben.

Standard-header:



Über die Ausführung des Makros GCCSN wird ein gegliederter Returncode übergeben (bb=Subcode1, aaaa=Maincode):

X'bb'	X'aaaa'	Erläuterung
X'00'	X'0000'	Normale Ausführung.
X'01'	X'0008'	Operandenfehler.
X'20'	X'0004'	Interner Fehler.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

GEPRT – Programmzeit lesen

Allgemeines

Anwendungsgebiet: Abfragen und Zugriff zu Listen und Tabellen;
siehe Seite 158
Makrotyp: O-Typ/R-Typ; siehe Seite 28
S-Typ, MF-Format **1** (31-Bit-Schnittstelle):
Standardform/E-/L-/D-Form; siehe Seite 29

- Die Makroauflösung als O-Typ benutzt keine Register.
- Je nach Aufrufform werden unterschiedliche SVCs benutzt.

Makrobeschreibung

Der Makro **GEPRT** informiert über die durch den Auftrag seit SET-LOGON-PARAMETERS verbrauchte CPU-Zeit und/oder über die für das laufende Programm noch verfügbare CPU-Zeit.
Die noch verfügbare CPU-Zeit bezieht sich auf das für den Programmlauf vorgegebene Zeitlimit (Kommando START-PROGRAM, Operand CPU-LIMIT); wenn dort nicht spezifiziert, auf das Zeitlimit für den Auftrag (Angabe oder Voreinstellung im Kommando SET-LOGON-PARAMETERS).
Die Zeitangaben (gezonte Dezimalzahlen) werden in einzurichtende Felder übertragen.

Der Makroaufruf ohne Operandenangabe (R-Typ) liefert die seit SET-LOGON-PARAMETERS verbrauchte CPU-Zeit. Diese Zeitangabe (im TODR-Format) wird dem Anwender in den Registern R0 und R1 übergeben. Die Performance der Funktionsausführung ist wesentlich höher als beim Makroaufruf mit Operandenangabe (220 Befehle beim R-Typ gegenüber 550 Befehlen beim S-Typ/O-Typ).

Makroaufrufformat und Operandenbeschreibung

```
GEPRT

[ { [adr1] [,adr2] } [,FORMAT=B8]]
  { [(r1)] [,(r2)] }

[,PARMOD=31]

[,MF=L / (E,..) / D]

,PREF=G / p
```

adr1
symbolische Adresse eines 6 Byte langen Feldes
Die von dem Auftrag bereits verbrauchte CPU-Zeit wird in dieses Feld eingetragen.
Feldlänge = 8 Byte, wenn der Operand FORMAT=B8 angegeben wird.

(r1)
r1 = Register mit dem Adresswert adr1
Bei PARMOD=31 darf der Adresswert nicht im Register R1 übergeben werden.

adr2
symbolische Adresse eines 6 Byte langen Feldes
Die restliche (maximale) CPU-Zeit für das Programm bzw. den Auftrag wird in dieses Feld eingetragen.
Feldlänge = 8 Byte, wenn der Operand FORMAT=B8 angegeben wird.

(r2)
r2 = Register mit dem Adresswert adr2. Bei PARMOD=31 darf der Adresswert nicht im Register R1 übergeben werden.

FORMAT=B8
Die Zeitangaben werden in der Form hhhhmmss (hhhh=Stunden, mm=Minuten, ss=Sekunden) übertragen.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der D-Form des Makroaufrufs kann ein Präfix PREF (1 Buchstabe, Voreinstellung: G) angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

PARMOD=
steuert die Makroauflösung. Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (24-Bit-Schnittstelle).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

Funktionsweise

Die CPU-Zeit wird als Zeichenfolge (gezonte Dezimalzahl) 6-stellig (hhmmss) oder bei Angabe von FORMAT=B8 8-stellig (hhhhmmss) übergeben.

Angaben für Aufträge ohne CPU-Zeitbeschränkung (NTL):

Feld für verbrauchte CPU-Zeit:

	aktueller Wert < 100h	aktueller Wert ≥ 100h
6-Byte-Feld	aktueller Wert	99h 59min 59sec
8-Byte-Feld	aktueller Wert	aktueller Wert

Feld für restliche CPU-Zeit:

6-Byte-Feld	99h 59min 59sec
8-Byte-Feld	9999h 59min 59sec

Hinweis

Der Makro **GEPRT** übergibt keinen Returncode. Bei Adressierungsfehler durch den Anwender generiert das System das Ereignis „Adressenfehler“ (STXIT-Ereignisklasse „Nicht behebbarer Programmfehler“). Das Programm wird mit der Anzeige „Adressenfehler“ beendet, es sei denn, im Programm ist für dieses Ereignis eine STXIT-Routine vorgesehen.

Beispiel

```
GEPRT    START
         PRINT NOGEN
         BALR  3,0
         USING *,3
         GEPRT FIELD1,FIELD2,FORMAT=B8 ———————————————————————————   (1)
         WROUT OUTB,ERROR
ERROR    TERM
****     DEFINITIONS     *******
OUTB     DC    Y(OUTBE-OUTB)
         DS    CL3
         DC    C'CPU TIME USED: '
FIELD1   DS    CL8
         DC    C'  AVAILABLE CPU TIME: '
FIELD2   DS    CL8
OUTBE    EQU   *
         END
```

Ablaufprotokoll:

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,geprt), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,geprt))
%  ASS6011 ASSEMBLY TIME: 337 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 80 MSEC
//end
%  ASS6012 END OF ASSEMBH
/start-executable-program library=macexmp.lib,element-or-symbol=geprt
%  BLS0523 ELEMENT 'GEPRT', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'GEPRT', VERSION ' ' OF '<date> <time>' LOADED
CPU TIME USED: 00000059  AVAILABLE CPU TIME: 00022901   ————————————————  (2)
```

(1) Sowohl die seit SET-LOGON-PARAMETERS verbrauchte CPU-Zeit, als auch die für den Programmlauf noch zur Verfügung stehende CPU-Zeit werden abgefragt.

(2) Die seit SET-LOGON-PARAMETERS verbrauchte CPU-Zeit und die für den Programmlauf noch zur Verfügung stehende CPU-Zeit werden ausgegeben.

GETPRGV – Programmversion abfragen

Allgemeines

Anwendungsgebiet: Binden und Laden; siehe Seite 47
Makrotyp: S-Typ, MF-Format **2**: Standardform/C-/D-/E-/L-/M-Form siehe Seite 29

Zum dynamischen Bindelader DBL siehe auch Handbuch „BLSSERV“ [4].

Makrobeschreibung

Der Makroaufruf **GETPRGV** gibt die Programmversion aus, die vom Benutzer vorher mit dem Makroaufruf **SELPRGV** oder mit dem Kommando SELECT-PROGRAM-VERSION ausgewählt wurde.

Makroaufrufformat und Operandenbeschreibung

GETPRGV
MF=S / C / D / E / L / M
,PRGNAME=<name 1..32>
,PRGNAM@=<var: name 32..32> / (<reg: pointer>)
,PRGVER@=<var: structure> / <var: pointer> / (<reg: pointer>)
,PARAM=<var: pointer> / (<reg: pointer>)
,PREFIX=P / p
,MACID=BGT / macid

PRGNAME=<name 1..32>
Name des Programmes. Der Name darf nur alphanumerische Zeichen enthalten. Angabe nur mit MF=L oder MF=S.

PRGNAM@=<var: name 32..32> / (<reg: pointer>)
Symbolische Adresse oder Register mit der Adresse eines 32 Zeichen langen Feldes, in dem der Programmname steht. Kürzere Namen müssen mit Leerzeichen aufgefüllt werden. Angabe nur mit MF=M.

PRGVER@=
Adresse einer Struktur, in die der DBL die Programmversion einträgt, falls eine Version ausgewählt wurde. Die DSECT <prefix><macid>VRDS für die Struktur wird bei MF=D erzeugt und hat folgendes Layout:

Byte	Länge	Feldname	Beschreibung
0	1	VERL	Länge der Programmversion
1	24	VERS	Programmversion

<var: structure>
Symbolische Adresse der Struktur. Angabe nur mit MF=L oder MF=S.

<var: pointer>
Symbolische Adresse eines Hilfsfeldes, das die Strukturadresse enthält.
Angabe nur mit MF=M.

(<reg: pointer>)
Register mit der Adresse der Struktur. Angabe nur mit MF=M.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. angegebenen Operanden PARAM, PREFIX und MACID siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

Hinweise zum Makroaufruf

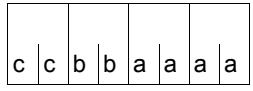
– DBL sucht zuerst unter den Programmversionen, für die SCOPE=PROGRAM festgelegt wurde (siehe Makro **SELPRGV**). Findet der DBL das angegebene Programm darunter, so übergibt er die Programmversion und beendet die Suche.

– Bei PRGNAM@ und PRGVER@ müssen gültige Klasse-6-Speicher-Adressen angegeben werden.

Rückinformation und Fehleranzeigen

Die Programmversion und deren Länge werden in der Struktur übergeben, die mit dem Parameter PRGVER@ angegeben wurde.

Standard-header:



Über die Ausführung des Makros GETPRGV wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	**X'bb'**	**X'aaaa'**	**Erläuterung**
X'00'	X'00'	X'0000'	Der Makro wurde normal ausgeführt.
X'00'	X'01'	X'0001'	PRGNAME nicht angegeben oder ungültig.
X'00'	X'01'	X'0002'	PRGVER@ nicht angegeben oder ungültig.
X'00'	X'00'	X'0004'	Für das Programm ist keine Version ausgewählt.
X'00'	X'00'	X'0007'	Programm mit dem angegebenen Namen nicht gefunden.
X'00'	X'20'	X'0300'	Systemfehler.
X'00'	X'01'	X'FFFF'	Die Funktion wird nicht mehr oder noch nicht unterstützt.
X'00'	X'03'	X'FFFF'	Die Version der Schnittstelle wird nicht unterstützt.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standardheader“ auf Seite 43 entnommen werden.

GPARMOD – Makroauflösung steuern

Allgemeines

Anwendungsgebiet: XS-Programmierung; siehe Seite 168
Makrotyp: O - Typ; siehe Seite 28

- Es kann ein 24-Bit-Adressierungsmodus oder ein 31-Bit-Adressierungsmodus (AMODE=24/31) vereinbart werden. Oberhalb der 16MB-Grenze muss ein Programm im 31-Bit-Adressierungsmodus ablaufen.
- Die Makros, deren Schnittstellen noch kein 31-Bit-Format hatten, wurden um die 31-Bit-Schnittstelle erweitert. Die alte Schnittstelle wird als 24-Bit-Schnittstelle bezeichnet.

  Datenbereiche und Befehle für die 24-Bit-Schnittstelle benutzen 24-Bit-Adressen und 31-Bit-Adressen für die 31-Bit-Schnittstelle. Die 31-Bit-Schnittstelle kann auch im 24-Bit-Adressierungsmodus benutzt werden. Die Adressen im Datenbereich werden in diesem Fall als 24-Bit-Adressen interpretiert (linksbündiges Byte wird nicht ausgewertet).

Makrobeschreibung

Mit dem Makro **GPARMOD** wird der globalen Assemblervariablen &SYSMOD bei der Programmübersetzung der Wert 24 oder 31 zugewiesen. &SYSMOD wird bei der Generierung der Makros ausgewertet, die sowohl eine 24-Bit- als auch eine 31-Bit-Schnittstelle generieren können und in deren Makroaufruf der Operand PARMOD nicht spezifiziert wurde. Je nach Angabe für &SYSMOD wird für die nachfolgenden Makros die 24-Bit- oder die 31-Bit-Schnittstelle generiert. Die Angabe für den Operanden PARMOD im jeweiligen Makroaufruf hat bei der Makrogenerierung Vorrang vor &SYSMOD.

Makroaufrufformat und Operandenbeschreibung

GPARMOD
[24 / 31]

24
Für die nachfolgenden Makros wird die 24-Bit-Schnittstelle generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Für die nachfolgenden Makros wird die 31-Bit-Schnittstelle generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB).

Beispiele

MAK1, MAK2 und MAK3 bezeichnen Makros, die sowohl eine 24-Bit- als auch eine 31-Bit-Schnittstelle generieren können.

```
   :
   :
GPARMOD 31 ------------------------------------------------------------  (1)
MAK1    op1, ..., op4
MAK2    op1, op2, op3
MAK3    op1, ..., op7
   :
   :

   :
   :
GPARMOD 24 ------------------------------------------------------------  (2)
MAK1    op1, ..., op4
MAK2    op1, ..., PARMOD=31
MAK3    op1, ..., op7
   :
   :

   :
   :
GPARMOD 31 ------------------------------------------------------------  (3)
MAK1    op1, ..., PARMOD=24
MAK2    op1, ..., PARMOD=24
MAK3    op1, ..., op7
   :
   :
```

(1) &SYSMOD wird auf 31 gesetzt. Da die nachfolgenden Makros den Operanden PARMOD nicht spezifiziert haben, wird für jeden Makro die 31-Bit-Schnittstelle generiert.

(2) &SYSMOD wird auf 24 gesetzt. Für MAK1 und MAK3 wird die 24-Bit-Schnittstelle generiert; für MAK2 die 31-Bit-Schnittstelle.

(3) &SYSMOD wird auf 31 gesetzt. Für MAK1 und MAK2 wird entsprechend der Angabe für PARMOD die 24-Bit-Schnittstelle generiert; für MAK3 entsprechend &SYSMOD die 31-Bit-Schnittstelle.

GTIME – Datum und Uhrzeit anfordern

Allgemeines

Anwendungsgebiet: Abfragen und Zugriff zu Listen und Tabellen; siehe Seite 158
Makrotyp: S-Typ, MF-Format **3**: C-/D-/E-/L-/M-Form; siehe Seite 29

Makrobeschreibung

Der Makro **GTIME** informiert über

- das aktuelle Datum und die aktuelle Uhrzeit, wahlweise in Form der universellen Weltzeit UTC (Universal Time Coordinate, entspricht der Greenwich-Zeit) oder der gesetzlichen, lokalen Landeszeit LT (Local Time)
- dabei kann ein Monotonieverhalten für Aufrufer auf unterschiedlichen XCS-Knotenrechnern gefordert werden
- den aktuellen Wochentag
- die Zeitzone des Systems in Stunden und Minuten (entspricht der Zeitverschiebung gegenüber UTC)
- die Größe der Zeitverschiebung bei Sommerzeit in Stunden und Minuten
- die aktuelle Zeitverschiebung gegenüber der Normalzeit auf Grund der Sommerzeit
- die Umstellzeitpunkte, an denen von Sommer- auf Winterzeit oder umgekehrt geschaltet wird (sie werden mit dem Makro **CTIME** verwaltet)
- ob die Systemzeit mit einer externen Referenz synchronisiert ist, und wenn ja, mit welcher Referenz-Zeitquelle (z.B. Funkuhr)
- ob in der nächsten Stunde eine Zeitumstellung ansteht
- die aktuelle Epoche für das TODR (siehe Handbuch „Systembetreuung“ [10])

GTIME stellt diese Informationen im Datenbereich wahlweise in abdruckbarer, binärer oder TODR/TODX-Form (nur für die Ausgabe der lokalen Zeit (LT) oder der universellen Weltzeit (UTC)) zur Verfügung.

Hinweise

- Da die Funktionen des Makros **GTIME** nicht über einen SVC, sondern über eine Unterprogrammschnittstelle ausgelöst werden, muss ein Programm, das **GTIME** aufruft, einen 18 Worte langen Sicherstellungsbereich zur Verfügung stellen. Vor dem Makroaufruf ist die Adresse dieses Sicherstellungsbereiches in Register R13 zu laden.
- Abgesehen von einer Überprüfung des Standardheaders findet keine Validierung des Datenbereichs statt.

Makroaufrufformat und Operandenbeschreibung

```
GTIME

MODE=LT / UTC
,FORMAT=ISO4 / BIN / TODR / TODX
,RESOLVE=SEC / MICROSEC
,LINKADR=*NONE / linkadr
,DATE=NO / YES
,DAY=NO / YES
,TOD=NO / YES
,ZONE=NO / YES
,EXTREF=NO / YES
,CHDATE=NONE / NEXT / PREV
,CHD_ANNOUNCMNT=*NO / *YES
,XCS_MODE=*NO / *YES
,MF=D / E / L / C / M
[,PARAM=adr / (r)]
,PREFIX=N / p
,MACID=TIG / macid
```

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

CHD_ANNOUNCMNT=
kündigt einen Umstellungszeitpunkt durch einen Indikator an.

***NO**
Es soll kein (weiterer) Indikator gesetzt werden.

***YES**
Es wird ein Indikator für eine Zeitumstellung im Zeitintervall von einer Stunde vor der Umstellung gesetzt.

Der Datenbereich zur Aufnahme des Indikators hat folgenden Aufbau (Makroauflösung mit MF=D und Standardwert für PREFIX):

```
NTIGGINF            DS    AL1     general_info
NTIGICNH            EQU   X'80'   chdate is expected in next
*                                   hour
NTIGRESERVED_7BITS  EQU   X'7F'   not yet used
NTIGFRES            DS    XL2     reserved
```

CHDATE=
gibt die Richtung an, in welcher die Umstellungszeitpunkte (CHDATEs) gesucht werden sollen.
Für weitere Informationen zu Umstellungszeitpunkten siehe Hinweise auf Seite 550.

NONE
Es soll kein (weiterer) Umstellungszeitpunkt gesucht werden.

NEXT
Es soll der nächste Umstellungszeitpunkt gesucht werden.
Basis für den Suchbeginn ist der Inhalt des Feldes `<prefix><macid>CHD`.

PREV
Es soll der vorhergehende Umstellungszeitpunkt gesucht werden.
Basis für den Suchbeginn ist der Inhalt des Feldes `<prefix><macid>CHD`.

DATE=
entscheidet, ob das aktuelle Datum ausgegeben wird.

NO
Es werden keine Informationen über das aktuelle Datum übergeben.

YES
Das aktuelle Datum (Kalendertag und julianisches Datum) wird in einen Datenbereich übertragen. In welchem Bereich und in welchem Format die Information übergeben wird, hängt vom Wert des Operanden FORMAT ab:

FORMAT=ISO4
Das aktuelle Datum wird in der Form `yyyy-mm-ddjjj` übergeben. Die beiden Bindestriche sind Bestandteil der Ausgabe. Dabei bedeuten:

`yyyy`	Jahr (vierstellig)
`mm`	Monat (zweistellig, ggf. mit führender Null)
`dd`	Tag (zweistellig, ggf. mit führender Null)
`jjj`	Julianisches Datum: Laufender Tag des Jahres (dreistellig, ggf. mit führenden Nullen)

Der Datenbereich zur Aufnahme des Datums hat folgenden Aufbau (Makroauflösung mit MF=D und Standardwert für PREFIX):

```
* DATE IN ISO4 FORMAT (EXAMPLE :2012-01-20020)
NTIGDTI      DS    0XL16       date_iso4
NTIGDATE_UN  DS    0XL10       date union
NTIGDATE_1   DS    0XL10       date struct
NTIGDTIY     DS    CL4         year
NTIGDTI1     DS    CL1         hyphen1
NTIGDTIM     DS    CL2         month
NTIGDTI2     DS    CL1         hyphen2
NTIGDTID     DS    CL2         day
             ORG   NTIGDATE_UN
```

```
NTIGDTIC     DS    CL10        date_char
             ORG   NTIGDATE_UN+10
NTIGDTIJ     DS    CL3         julian date
NTIGDTIB     DS    CL1         blank
NTIGDYID     DS    CL2         weekday in ISO4
```

FORMAT=BIN

Jahr, Monat, Tag und julianisches Datum werden jeweils als ganze Zahl (in Binärform) übergeben. Der Datenbereich zur Aufnahme des Datums hat folgenden Aufbau (Makroauflösung mit MF=D und Standardwert für PREFIX):

```
* DATE IN BINARY FORMAT
NTIGDTB      DS    0XL16       date_bin
*
NTIGDATE_2   DS    0XL6        date
NTIGDTBY     DS    H           year
NTIGDTBM     DS    H           month
NTIGDTBD     DS    H           day
NTIGDTBJ     DS    H           Julian date
NTIGFILL_6   DS    XL6         fill for weekday
NTIGDYBD     DS    H           weekday bin.: MO=0, DI=1, ...
*                                          SO=6
```

DAY=

entscheidet, ob der aktuelle Wochentag ausgegeben wird.

NO

Es werden keine Informationen über den aktuellen Wochentag übergeben.

YES

Der aktuelle Wochentag wird in ein Feld des Datenbereichs übertragen. In welchem Feld und in welchem Format die Information übergeben wird, hängt vom Wert des Operanden FORMAT ab:

FORMAT=ISO4

Die ersten beiden Buchstaben des aktuellen Wochentages (MO, DI, MI, DO, FR, SA, SO) werden in das folgende Feld des Datenbereichs übertragen (Makroauflösung mit MF=D und Standardwert für PREFIX):

```
* DAY OF WEEK IN ISO4 FORMAT (EXAMPLE : MO)
NTIGDYI      DS    0XL2
```

FORMAT=BIN

Der aktuelle Wochentag wird als ganze Zahl (in Binärform) (0 für Montag, 1 für Dienstag,..., 6 für Sonntag) in folgendem Halbwort des Datenbereichs übergeben (Makroauflösung mit MF=D und Standardwert für PREFIX):

```
* DAY OF WEEK IN BINARY FORMAT
NTIGDYB      DS    0XL2
```

EXTREF=
bestimmt, ob die Information über die externe Referenz der Systemzeit geliefert werden soll, sofern sie vorhanden ist.

NO
Es soll keine Information geliefert werden.

YES
Die Information über eine evtl. vorhandene externe Referenz der Systemzeit soll geliefert werden.

Der Datenbereich zur Ausgabe der Information hat folgenden Aufbau (für Detailinformationen siehe die DSECT von GTIME; Standardwerte für PREFIX und MACID):

```
* FLAGS OF GTIME: TIME REFERENCE
NTIGFLG             DS    0XL4    flags of GTIME
NTIGTREF            DS    FL1     time_reference
*   _time_reference_s
NTIGNONE            EQU   0       no external time reference
NTIGSVPF            EQU   1       SVP radio clock reference
NTIGSVCE            EQU   1       Server connected ext.reference
NTIGCHNF            EQU   2       Channel rad. cl. reference
NTIGBSCE            EQU   2       BS2 connected ext. reference
NTIGDCET            EQU   3       DCE reference
NTIGXCST            EQU   4       XCS reference
NTIGSKPX            EQU   5       SKP-X reference
NTIGX2K             EQU   5       X2000 reference
```

FORMAT=
legt fest, in welcher Form **GTIME** die angeforderten Informationen zur Verfügung stellt.

ISO4
Die Informationen werden in abdruckbarer Form übergeben.

BIN
Die Informationen werden in Binärform übergeben.

TODR / TODX
Die Informationen werden dem Inhalt des TOD-Registers und der Einstellung für die Epoche des TOD-Registers entnommen.
Die Operandenwerte für DATE, DAY, TOD und ZONE müssen „NO“ sein. Der Operand MODE wird ausgewertet, d.h. für die Ausgabe der lokalen Zeit im TODR- oder TODX-Format muss MODE=LT eingestellt werden, sonst erhält man die UTC.
Die Unterschiede zwischen TODR und TODX sind beschrieben im Abschnitt „Systemzeit-Verwaltung“ des Handbuchs „Systembetreuung“ [10] und auch beim CTIME-Makro auf Seite 370.

LINKADR=
gibt an, auf welche Weise dem Anwenderprogramm die Adresse des Einsprungpunktes I@GTIME für die **GTIME**-Routine im Subsystem GET-TIME zur Verfügung gestellt wird. Bei MF=E muss LINKADR angegeben werden, in allen anderen Fällen ist die Angabe von LINKADR wirkungslos.

***NONE**
Bei der Übersetzung generiert der Assembler einen Externverweis für den Einsprungpunkt I@GTIME, der beim Binden über die Autolink-Funktion des BLS aufgelöst wird.

Dieser Wert kann verwendet werden, wenn das Modul, das den **GTIME**-Aufruf enthält,

– immer mit dem Dynamischen Bindelader DBL gebunden und geladen wird (in diesem Fall lässt man den **GTIME** in der E-Form eine V-Konstante absetzen, welche beim Ladevorgang durch das BLS versorgt wird)
  oder
– mit dem BINDER des neuen BLS (siehe Handbuch „BINDER“ [5]) unter der BINDER-Anweisung SET-EXTERN-RESOLUTION RESOLUTION=STD gebunden wird.

linkadr
symbolische Adresse (Name) eines Wortes, in dem der Anwender vor dem **GTIME**-Aufruf die Adresse des Einsprungpunktes I@GTIME bereitgestellt hat.

Das folgende Beispiel zeigt, wie dem Programm die Adresse des Einsprungpunktes I@GTIME zunächst durch einen geeigneten **BIND**-Aufruf im Register R1 zur Verfügung gestellt und anschließend für den **GTIME**-Aufruf in das mit linkadr bezeichnete Wort übertragen werden kann:

```
         BIND  MF=E,PARAM=BINDPL
         :
         GTIME MF=E,PARAM=OPLIST,LINKADR=AENTRY
         :
AENTRY   DS    F
OPLIST   GTIME MF=L,...
BINDPL   BIND  MF=L,SYMBOL=I@GTIME,SYMBLAD=AENTRY
```

Auf diese Weise muss dem Anwenderprogramm die Einsprungadresse der **GTIME**-Routine immer dann mitgeteilt werden, wenn keiner der unter LINKADR=*NONE erwähnten Fälle vorliegt, z.B. also insbesondere dann, wenn das Modul mit dem **GTIME**-Aufruf z.B. mit dem BINDER unter der BINDER-Anweisung SET-EXTERN-RESOLUTION RESOLUTION=MANDATORY gebunden wird.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

Bei der E-Form des Makroaufrufs wird das Label des Datenbereichs im Operanden PARAM angegeben. Voreinstellung: NTIG$PL
Beim Aufruf des Makros mit MF=L muss der Anwender dieses Label explizit angeben, sonst wird eine `MNOTE` ausgegeben.

MODE=
legt fest, in welchem Zeitsystem Datum und Uhrzeit ausgegeben werden sollen.

LT
Datum und Uhrzeit werden im lokalen Zeitsystem (gesetzliche Landeszeit) ausgegeben.

UTC
Datum und Uhrzeit werden in Form der universellen Weltzeit UTC (Universal Time Coordinate ≙ Greenwich-Zeit) ausgegeben.

RESOLVE=
gibt die gewünschte Auflösung der **GTIME**-Zeitangabe für die Formate ISO4 und BIN an. Zum Aufbau der Ausgabebereiche siehe die beim Operanden TOD=YES angegebenen Bereiche.

SEC
Die Zeit wird auf Sekunden genau angegeben.

MICROSEC
Die Zeit wird auf Mikrosekunden genau angegeben.

TOD=
entscheidet, ob die aktuelle Uhrzeit ausgegeben wird.

NO
Es werden keine Informationen über die aktuelle Uhrzeit übergeben.

YES
Die aktuelle Uhrzeit wird in einen Datenbereich übertragen. In welchem Bereich und in welchem Format die Information übergeben wird, hängt vom Wert des Operanden FORMAT ab:

FORMAT=ISO4

Die aktuelle Uhrzeit wird in der Form `hh:mm:ss` übergeben (die beiden Doppelpunkte sind Bestandteil der Ausgabe). Dabei bedeuten:

`hh` Stunde (zweistellig, ggf. mit führender Null)
`mm` Minute (zweistellig, ggf. mit führender Null)
`ss` Sekunde (zweistellig, ggf. mit führender Null)

Der Datenbereich zur Aufnahme der Uhrzeit hat folgenden Aufbau (Makroauflösung mit MF=D und Standardwert für PREFIX):

```
* TIME OF DAY IN ISO4 FORMAT (EXAMPLE : 08:31:09)
NTIGTDI      DS    0XL8        _tod_iso4_mdl
NTIGTDIH     DS    CL2         hour
NTIGTDI1     DS    CL1         colon1
NTIGTDIM     DS    CL2         minute
NTIGTDI2     DS    CL1         colon2
NTIGTDIS     DS    CL2         second

* SECOND FRACTION OF TIME OF DAY IN ISO4 FORMAT
* (EXAMPLE : .123456)
NTIGTFI      DS    0XL6        _ftod_iso4_mdl
NTIGTFIM     DS    CL3         millisecond
NTIGTFIN     DS    CL3         microsecond
```

FORMAT=BIN

Stunde, Minute und Sekunde werden jeweils als ganze Zahl (in Binärform) übergeben. Der Datenbereich zur Aufnahme der Uhrzeit hat folgenden Aufbau (Makroauflösung mit MF=D und Standardwert für PREFIX):

```
* TOD IN BINARY FORMAT
NTIGTDB      DS    0XL6        _tod_bin_mdl
NTIGTDBH     DS    H           hour
NTIGTDBM     DS    H           minute
NTIGTDBS     DS    H           second

* SECOND FRACTION OF TOD IN BINARY FORMAT
NTIGTFB      DS    0XL4        _ftod_bin_mdl
NTIGTFBM     DS    H           millisecond
NTIGTFBN     DS    H           microsecond
```

XCS_MODE=
liefert einen Zeitwert, der innerhalb des XCS-Verbundes bzgl. eines DLM-Locks für eine gemeinsame Ressource streng monoton ist.

***NO**
Die oben beschriebene Monotonie wird nicht gefordert.

***YES**
gibt einen Zeitwert aus, der unter der folgenden Bedingung innerhalb eines XCS-Verbundes monoton steigt:
Der Aufrufer muss zum Aufrufzeitpunkt einen DLM-Lock (siehe Abschnitt „Distributed-Lock-Manager (DLM)“ auf Seite 142) halten. Die Monotoniebeziehung gilt nur für die Zeitstempel, die unter dem gleichen Lock geholt wurden.
Der Zeitwert kann für Logging-Einträge genutzt werden.
Der Einsatz ist auf das Ausgabeformat FORMAT=*TODR und MODE=*UTC beschränkt. Nur im TODR-Format wird die benötigte Auflösung erreicht. Mit der Einschränkung auf MODE=*UTC ist sichergestellt, dass die Monotonie auch im Moment einer Zeitumstellung für alle Knotenrechner erhalten bleibt.

ZONE=
entscheidet, ob Kenngrößen der lokalen Zeitzone ausgegeben werden.

NO
Es werden keine Informationen über die lokale Zeitzone übergeben.

YES
Kenngrößen der lokalen Zeitzone werden in einen Datenbereich übertragen. In welchem Bereich und in welchem Format die Informationen übergeben werden, hängt vom Wert des Operanden FORMAT ab:

FORMAT=ISO4
Die Kenngrößen der lokalen Zeitzone werden in der Form `shh1:mm1-hh2:mm2-z` übergeben (die Doppelpunkte und Bindestriche sind Bestandteil der Ausgabe). Dabei bedeuten:

`s`	„+“ oder „-“: Vorzeichen des Zeitunterschiedes der lokalen Zeitzone zur UTC (Universal Time Coordinate ≙ Greenwich-Zeit)
`hh1:mm1`	Zeitunterschied der lokalen Zeitzone zur UTC in Stunden (hh) und Minuten (mm) (hh und mm jeweils zweistellig, ggf. mit führenden Nullen)
`hh2:mm2`	Zeitverschiebung in der lokalen Zeitzone zwischen Sommer- und Normalzeit in Stunden (hh) und Minuten (mm) (hh und mm jeweils zweistellig, ggf. mit führenden Nullen)
`z`	„W“ oder „S“: Aktuelle Zeitzählung in der lokalen Zeitzone (W für Winterzeit ≙ Normalzeit, S für Sommerzeit)

Der Datenbereich zur Aufnahme dieser Informationen hat folgenden Aufbau (Makroauflösung mit MF=D und Standardwert für PREFIX):

```
* ZONE IN ISO4 FORMAT (EXAMPLE :+08:00-01:00-S)
NTIGZOI      DS    0XL14       _zone_iso4_mdl
*
NTIGZOIC     DS    0XL6        time_zone
NTIGZOIS     DS    CL1         sign
NTIGZOIH     DS    CL2         hour
NTIGZOI1     DS    CL1         colon1
NTIGZOIM     DS    CL2         minute
*
NTIGZOI2     DS    CL1         hyphen1
NTIGZSIC     DS    0XL5        seasonal_difference
NTIGZSIH     DS    CL2         hours
NTIGZSI1     DS    CL1         colon1
NTIGZSIM     DS    CL2         minutes
*
NTIGZSI2     DS    CL1         hyphen2
NTIGZSIA     DS    FL1         actual season
*   _season_iso4_s
NTIGZSIW     EQU   230         'W': Wintertime
NTIGZSIS     EQU   226         'S': Daylight Savings Time
```

FORMAT=BIN

Die Kenngrößen der lokalen Zeitzone werden jeweils als ganze Zahlen (in Binärform) übergeben. Der Datenbereich zur Aufnahme dieser Informationen hat folgenden Aufbau (Makroauflösung mit MF=D und Standardwert für PREFIX):

```
* ZONE IN BINARY FORMAT
NTIGZOB                 DS    0XL10   _zone_bin_mdl
*
NTIGTIMEZONE            DS    0XL4    time_zone
NTIGZOBH                DS    H       hour
NTIGZOBM                DS    H       minute
*
*
NTIGSEASONAL_DIFFERENCE DS    0XL4    seasonal_difference
NTIGZSBH                DS    H       hours
NTIGZSBM                DS    H       minutes
*
NTIGZSBA                DS    FL1     actual season
*   _season_bin_s
NTIGZSBW                EQU   0       Wintertime
NTIGZSBS                EQU   1       Daylight Savings Time
*
NTIGZONE_BIN_FILL       DS    XL1     to fill the gap
```

Hinweise zu Umstellungszeitpunkten

Um die **CTIME**-Funktionalität mit nicht-systemeigenen Umstellungszeitpunkten zu nutzen, muss man die Umstellungszeitpunkte vom System, auf dem die Zeitdaten entstehen, erfragen können. BS2000 kann max. 400 Umstellungszeitpunkte verwalten. Dabei kann pro **GTIME**-Aufruf ein Umstellungszeitpunkt erfragt werden.
Die Informationen über die Umstellungszeitpunkte werden in einem Feld mit der Länge 8 Byte (<prefix><macid>CHD) zur Verfügung gestellt. Mit den beim Operanden CHDATE angegebenen Werten wird die Richtung der Abfrage der Umstellungszeitpunkte vorgegeben. Bezugspunkt ist dabei die Information, die zur Aufrufzeit von **GTIME** in diesem Feld steht:
Sind alle acht Byte des Feldes mit X'00' belegt, beginnt die Suche nach dem nächsten oder vorhergehenden Umstellungszeitpunkt bei der aktuellen Systemzeit. Ist das Feld dagegen mit einem Datum vorbelegt, wird dieses als bereits von **GTIME** gelieferte CHDATE-Information interpretiert und die Suche beginnt ab diesem Datum.
Die CHDATE-Informationen liegen als STCK-Werte auf UTC-Basis vor, die um 8 Bits logisch nach rechts verschoben wurden (SRDL-Befehl). Das niederwertigste Bit zeigt dabei die Art der Umstellung an: Ist das Bit 0, wird von Winter- auf Sommerzeit umgestellt, bzw. umgekehrt, wenn das Bit 1 ist.
Diese Daten müssen für die **CTIME**-Funktion in einer Tabelle angeordnet werden. Der Aufbau der Tabelle ist aus der Beschreibung des **CTIME**-Makros, Operand CHDLxIN bzw. CHDLOUT, zu entnehmen. Die Arbeitsweise mit Umstellungszeitpunkten ist ebenfalls beim Makro **CTIME** anhand eines Beispiels beschrieben (siehe Seite 365).

Auf die Rückgabedaten im Datenbereich kann mit folgenden Namen zugegriffen werden (Standardwerte für PREFIX und MACID):

```
NTIGCHD             DS    0XL8    change date
*
NTIGTODC            DS    0XL8    s_todr
NTIGCHDATE_UN       DS    0XL8    chdate_un
NTIGCHDATE_VALUE    DS    XL8     chdate_bit64
                    ORG   NTIGCHDATE_UN
*
NTIGCHDATE_MDL      DS    0XL8    chdate_mdl
NTIGCHD1            DS    F       most significant word
NTIGCHD2            DS    F       least significant word
```

Registerverwendung

Beim Aufruf des Makros **GTIME** werden folgende Register benötigt:

R1 wird vom Makro mit der Adresse des Datenbereichs geladen.

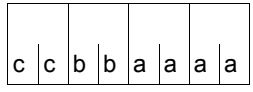
R13 ist vor dem Makroaufruf mit der Adresse eines 18 Worte langen Sicherstellungsbereiches zu laden, den das aufrufende Programm zur Verfügung stellen muss.

R14 wird vom Makro mit der Rückkehradresse des Anwenderprogramms geladen.

R15 wird von der (über **GTIME**) gerufenen Routine überschrieben.

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros GTIME wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

cc	bb	aaaa	Erläuterung
00	00	0000	Funktion erfolgreich ausgeführt.
02		0010	Die Funktion wurde ausgeführt, aber dem System ist kein früheres CHDATE als das im Datenbereich angegebene bekannt. Ein weiterer Aufruf mit CHDATE=PREV liefert das gleiche Ergebnis und ist deshalb nicht sinnvoll. Die 8 Byte der CHDATE-Information enthalten X'00..0001' .
02		0011	Die Funktion wurde ausgeführt, aber dem System ist kein späteres CHDATE als das im Datenbereich angegebene bekannt. Ein weiterer Aufruf mit CHDATE=NEXT liefert das gleiche Ergebnis und ist deshalb nicht sinnvoll. Die 8 Byte der CHDATE-Information enthalten X'00FFFFFFFFFFFFss', wobei 'ss' für die SEASON nach dem vorhergehenden CHDATE steht (also X'00' oder X'01' ist).
02		0012	Die Funktion wurde ausgeführt, aber dem System ist kein CHDATE bekannt. Es gibt also im System keine Zeitumstellung. Ein weiterer Aufruf des Makros GTIME ist nicht sinnvoll.
	20	0008	Der Returncode kann nur bei Angabe von XCS_MODE=*YES auftreten. Die Funktion konnte nicht ausgeführt werden: Bei der Ermittlung der XCS-verbundweit monotonen Zeit ist ein interner Fehler aufgetreten. cc weist auf die Art des Fehlers hin, ist aber nur für die Diagnose beim Hersteller von Bedeutung. Ein weiterer Aufruf des Makros GTIME mit XCS_MODE=*YES ist nicht sinnvoll.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Das aufrufende Programm wird beendet, wenn folgende Fehler auftreten:
– Der Datenbereich ist dem Aufrufer nicht zugewiesen.
– Der Datenbereich ist nicht auf Wortgrenze ausgerichtet.
– Der Datenbereich ist gegen Schreibzugriff geschützt.

Im Datenfeld <prefix><macid>EPD (ein Byte) wird bei jedem GTIME-Aufruf die aktuelle Epoche des TODR geliefert, siehe Abschnitt „Systemzeit-Verwaltung“ im Handbuch „Systembetreuung“ [10].

Beispiele siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29 und beim Makro **CTIME** auf Seite 365.

ILEMGT – Verwaltung von Indirect Linkage Entries (ILEs)

Allgemeines

Anwendungsgebiet: Binden und Laden; siehe Seite 47
Makrotyp: S-Typ, MF-Format **2**: Standardform/C-/D-/E-/L-/M-Form; siehe Seite 29

Zum dynamischen Bindelader DBL siehe auch Handbuch „BLSSERV“ [4].

Makrobeschreibung

Mit dem Makroaufruf **ILEMGT** kann der Benutzer eine Liste von ILEs (Indirect Linkage Entries) verwalten. Die ILEs in der Liste können erzeugt, geändert oder gelöscht werden.

Informationen über ILEs können mit dem Makroaufruf **VSVI1** angefordert werden.

Makroaufrufformat und Operandenbeschreibung

```
ILEMGT

MF=S / D / C / E / L / M
,ACTION=*CREATE / *UPDATE / *DELETE
,CONTEXT_NAME= '␣' / <c-string 1..32> / <var: char 1..32>
,CONTEXT_STATE=*DBL-OPTIONS / *ANY / *NEW / *OLD
,ILE_LIST_ADDR=NULL-1 / <var: pointer>
,ILE_LIST_LEN=0 / <integer 0..2147483647>
,MPID_ADDR=NULL-1 / <var: pointer>
,PARAM=<var: pointer> / (reg: pointer>)
,PREFIX=P / p
,MACID=ILE / macid
```

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

ACTION=
Aktion, die für die ILEs in der Liste ausgeführt werden soll.

***CREATE**
Die ILEs sind zu erzeugen.

***UPDATE**
Die ILEs sind zu ändern.

***DELETE**
Die ILEs sind zu löschen.

CONTEXT_NAME=

'␣' / <c-string 1..32> / <var: char 1..32>
Name des Kontextes, zu dem die ILEs gehören.

CONTEXT_STATE=
Status des bei CONTEXT_NAME angegebenen Kontextes

***DBL-OPTIONS**
Der Parameterwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Parameter mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt der Wert, der in der Makro-Syntaxbeschreibung auf *DBL-OPTIONS folgt.

***ANY**
Wenn der Kontext bereits existiert, wird dieser verwendet; ansonsten wird ein neuer Kontext erzeugt.

***NEW**
Der Kontext wird angelegt. Er darf noch nicht existieren.

***OLD**
Der Kontext muss bereits existieren.

ILE_LIST_ADDR=NULL-1 / <var: pointer>
Adresse einer Liste von ILEs, die an DBL übergeben werden soll.

ILE_LIST_LEN=0 / <integer 0..2147483647>
Länge der ILE-Liste (in Byte)

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. angegebenen Operanden PARAM, PREFIX und MACID siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29.Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

MPID_ADDR=NULL-1 / <var: pointer>
Adresse eines Feldes mit der Kurzkennung des Memory Pools, das den angegebenen Kontext enthält. Diese Kurzkennung wird dem Benutzer durch den Makroaufruf **ENAMP** zur Verfügung gestellt.

Format der ILE-Liste

Die bei ILE_LIST_ADDR angegebene Adresse zeigt auf eine ILE-Liste, die mehrere Einträge haben kann. Ein einzelner ILE-Eintrag hat folgendes Format:

Byte	Länge	Feldname	Bedeutung und/oder Wert
0	8	HDR	Standardheader
8	1	STATE	ACTIVE (X'01') oder NOT_ACTIVE (X'02')
9	1	CONTROL	SYSTEM (X'01') oder USER (X'02')
10	1	HSI_CODE	390 (X'01') oder X86E (X'09')
11	1	RESERVED1	reserviert (muss mit X'00' belegt sein)
12	4	LOAD_ADDR	Adresse der IL-Routine
16	4	SERVER_ADDR	Adresse des ILE-Servers
20	2	REF_DISPL	Distanz des Externverweises auf den Server innerhalb der IL-Routine
22	32	NAME	Name des ILE-Symbols
54	2	RESERVED2	reserviert (muss mit X'0000' belegt sein)

Die DSECT für einen solchen ILE-Listeneintrag wird mit `ILEMIT MF=D` erzeugt.

Hinweise zum Makroaufruf

– Wenn bei MPID_ADDR eine Adresse angegeben wurde, entscheidet das erste Zeichen des Kontextnamens (Pflichtparameter) darüber, ob der Kontext gemeinsam benutzbar ist oder nicht.

  Beginnt der Kontextname mit „#“, so sucht DBL den Kontext in Common Memory Pools und erzeugt den Kontext gegebenenfalls (**ASHARE**-Makro). Die MPID muss in diesem Fall einen Memory Pool identifizieren, der entweder bereits beim **ASHARE**-Makro verwendet wird oder der leer ist (d.h. in dem noch kein Speicherplatz belegt wurde).

  Beginnt der Kontextname nicht mit „#“,dann sucht der DBL in den normalen Benutzerkontexten. Ein Kontext, dessen Name mit einem Buchstaben beginnt, darf nicht gemeinsam benutzbar sein. DBL erkennt dies als einen Konflikt zwischen der Nutzungsart des Memory Pools und der Namenskonvention und weist den Makroaufruf mit dem passenden Returncode ab.
– Die Anzahl von Memory Pools, in denen der Benutzer Shared Code ablegen kann, ist für eine Benutzerkennung auf 16 pro Geltungsbereich begrenzt, siehe Seite 223.

- Bei ACTION = *CREATE / *UPDATE ist das Feld STATE eines ILE-Eintrages mit ACTIVE belegt und sowohl LOAD_ADDR als auch SERVER_ADDR müssen auf zugeteilte Speicherbereiche verweisen.
- Ist das Feld CONTROL mit SYSTEM belegt, so führt DBL folgende Aktionen aus:

  a) Beim Laden des ILE-Servers
  wird das Feld STATE auf ACTIVE gesetzt und die Adresse des ILE-Servers wird in die IL-Routine eingetragen.

  b) Beim Entladen des ILE-Servers
  wird das Feld STATE auf NOT_ACTIVE gesetzt und als Adresse des ILE-Servers X'FFFFFFFF' in die IL-Routine eingetragen.

  Die Kombination der Feldbelegung CONTROL=SYSTEM mit dem Makroparameter ACTION=*UPDATE ist nicht erlaubt.

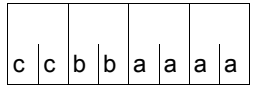
- Bei ACTION=*CREATE muss das Feld LOAD_ADDR auf einen zugeteilten Speicherbereich verweisen, in dem die benutzerdefinierte (CONTROL=USER) IL-Routine steht. Wenn das nicht zutrifft (CONTROL=SYSTEM), dann erzeugt der DBL eine Standard-IL-Routine im Kontext. Die Standard-IL-Routine wird in den HSI-Code, der beim Makroaufruf **ILEMIT** angegeben wurde oder ansonsten im HSI-Code des Servers, auf dem das Programm abläuft.
  Falls eine benutzerdefinierte IL-Routine vorhanden ist, muss das Feld REF_DISPL die Distanz des Wortes in der IL-Routine enthalten, das auf den ILE-Server verweist. Dieses Wort muss schreibbar sein.
  Die Angabe des HSI-Codes ist für benutzerdefinierte IL-Routinen Pflicht.
- Bei ACTION=*DELETE wird, sofern das zu löschende ILE mit CONTROL= *BY-SYSTEM erzeugt wurde, auch die IL-Routine aus dem Kontext gelöscht (vorausgesetzt, die IL-Routine wurde vom DBL erzeugt).
- Alle nicht maskierten CSECTs und ENTRYs im Zielkontext können als ILE-Server verwendet werden - auch die, die mit dem Makroaufruf **ETABLE** eingeführt wurden. Um daraus resultierende Probleme bei der ILE-Verwaltung zu vermeiden, sollte ein Kontext für ILEs und deren Server und ein anderer Kontext für ETABLE-Symbole eingerichtet werden.
- Wenn LLMs, die aus PUBLIC und PRIVATE Slices gebildet sind, für das indirekte Binden verwendet werden sollen, dann müssen diese LLMs im Format 1 vorliegen (siehe Benutzerhandbuch „BINDER“ [5]).
- Die IL-Routine sollte die Attribute AMODE und RMODE=ANY haben und unterhalb 16 Mbyte geladen werden, damit sie von allen Programmen aus erreichbar ist.
- Wenn ein ILE geändert wird, wird beim Laden seines Servers geprüft, ob HSI-Code und AMODE von ILE und Server übereinstimmen. Ist das nicht der Fall, so wird das Laden des Servers abgewiesen.

Rückinformation und Fehleranzeigen

Wenn bei der Verarbeitung der Listeneinträge ein Fehler auftritt, wird im HDR-Feld des fehlerhaften Listeneintrages ein entsprechender Returncode eingetragen. Im Standardheader der Parameterliste wird der Returncode FUNCTION_PARTIALLY_PROCESSED übergeben und die Verarbeitung der Listeneinträge wird fortgesetzt.

Das Feld PROCESSED_ITEMS der Parameterliste enthält die Anzahl korrekt bearbeiteter Einträge.

Standard-header:



Über die Ausführung des Makros ILEMGT wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Der Makro wurde normal ausgeführt.
X'02'	X'00'	X'0001'	Die Funktion wurde teilweise ausgeführt.
X'61'	X'01'	X'0001'	Ungültiger Wert bei ILE_LIST_LEN.
X'61'	X'01'	X'0002'	Unzulässige Parameterkombination in der Parameterliste.
X'61'	X'01'	X'0003'	Der Parameter MESSAGE ist ungültig.
X'61'	X'01'	X'0004'	Der Kontextname ist ungültig.
X'61'	X'01'	X'0008'	Der Parameter ACTION ist ungültig.
X'61'	X'40'	X'0011'	Die Task ist nicht an den Memory Pool angeschlossen.
X'61'	X'01'	X'0014'	Der neue USER-Kontext existiert bereits in einem anderen Memory Pool.
X'61'	X'40'	X'0015'	Maximale Anzahl von Kontexten im Memory Pool ist bereits erreicht.
X'61'	X'40'	X'0016'	Die maximal nutzbare Anzahl von Memory Pools mit diesem Geltungsbereich ist für die Benutzerkennung erreicht. Die Nutzung weiterer Memory Pools mit diesem Geltungsbereich ist nicht zulässig.
X'61'	X'40'	X'0017'	Der Systemspeicherbereich für Benutzerkontexte ist belegt.
X'61'	X'01'	X'0018'	Ein reserviertes Feld enthält keine binären Nullen.
X'61'	X'01'	X'0019'	Der aktuelle Memory Pool ist nicht gemeinsam benutzbar.
X'61'	X'20'	X'0100'	Systemfehler (z.B.: $REQM, $RELM, $CSTAT, RDTFT)
X'61'	X'20'	X'0101'	Interner DBL-Fehler.
X'61'	x'80'	X'0103'	Gemeinsam benutzbare Ressourcen sind nicht verfügbar.
X'61'	X'01'	X'0114'	CONTEXT_STATE=*OLD wurde gegeben und der angegebene Kontext ist nicht vorhanden.
X'61'	X'01'	X'0118'	CONTEXT_STATE=*NEW wurde gegeben und der angegebene Kontext ist bereits vorhanden.

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'61'	X'40'	X'0120'	MPID_ADDR ist nicht auf Wortgrenze ausgerichtet.
X'61'	X'01'	X'0134'	Der Parameter CONTEXT_STATE ist ungültig.
X'61'	X'40'	X'0148'	Es wurde versucht, einen Kontext zu benutzen, der durch einen vorhergehenden Fehler verfälscht worden ist.
X'61'	X'40'	X'0158'	Maximale Anzahl von 16 Benutzerkontexten erreicht. Ein neuer Kontext kann nicht mehr erzeugt werden.
X'61'	X'01'	X'0184'	Der angegebene Memory Pool ist ungültig.
X'61'	X'20'	X'0198'	Für die geforderte Aktion steht nicht genügend Speicherplatz zur Verfügung.
X'61'	X'20'	X'0204'	Inkonsistenzen in den Memory Management Tabellen des DBL (Systemfehler).
X'00'	X'01'	X'FFFF'	Die Funktion wird nicht mehr oder noch nicht unterstützt.
X'00'	X'03'	X'FFFF'	Die Version der Schnittstelle wird nicht unterstützt.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

ILEMIT – Listeneintrag für ILE-Liste erzeugen oder ändern

Allgemeines

Anwendungsgebiet: Binden und Laden; siehe Seite 47
Makrotyp: S-Typ, MF-Format **2**: Standardform/C-/D-/L-/M-Form siehe Seite 29

Zum dynamischen Bindelader DBL siehe auch Handbuch „BLSSERV“ [4].

Makrobeschreibung

Der Makroaufruf **ILEMIT** erzeugt oder ändert einen Listeneintrag für eine ILE-Liste, die beim Makroaufruf **ILEMGT** verwendet wird.

Makroaufrufformat und Operandenbeschreibung

ILEMIT
MF=D / C / L / M
,CONTROL=*NOT-SPECIFIED / *BY-SYSTEM / *BY-USER
,HSI_CODE=*BY-SYSTEM / *390 / *RISC [1]
,ILE_NAME=<c-string 1..32> / <var: char 1..32>
,LOAD_ADDR=NULL-1 / <var: pointer>
,REF_DISPL=0 / <integer 0..65535>
,SERVER_ADDR=NULL-1 / <var: pointer>
,STATE=*NOT-SPECIFIED / *ACTIVE / *NOT-ACTIVE
,PARAM=<var: pointer> / (reg: pointer>)
,PREFIX=P / p]
,MACID=ILT / macid

[1] Der Operandenwert *RISC ist seit BS2000/OSD-BC V9.0 bedeutungslos

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

CONTROL=
Legt fest, ob das ILE vom DBL oder vom Benutzer gesteuert wird, d.h. wer die IL-Routine bereitstellt.

***NOT-SPECIFIED**
Dieser Wert ist beim Erzeugen eines Listeneintrages nicht erlaubt.

***BY-SYSTEM**
Der DBL erzeugt die Standard-IL-Routine.

***BY-USER**
Es gibt eine benutzerdefinierte IL-Routine.

HSI_CODE=
Zeigt an, in welchem Maschinencode des Servers die IL-Routine vorliegt. Bei Angabe von LOAD_ADDR ist dieser Operand Pflicht.

***BY-SYSTEM / *390**
Voreinstellung ist der Typ des Servers, auf der die IL-Routine abläuft.

ILE_NAME=<c-string 1..32> / <var: char 1..32>
Name des ILE-Symbols.

LOAD_ADDR=NULL-1 / <var: pointer>
Adresse der IL-Routine.
Wird dieser Operand angegeben, muss zusätzlich der Operand HSI_CODE angegeben werden.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. angegebenen Operanden PARAM, PREFIX und MACID siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

REF_DISPL=0 / <integer 0..65535>
Distanz des Server-Adresswortes innerhalb der IL-Routine.

SERVER_ADDR=NULL-1 / <var: pointer>
Adresse des Servers (nur bei STATE=ACTIVE).

STATE=

***NOT-SPECIFIED**
Zeigt an, ob der ILE-Server geladen ist. Dieser Wert ist beim Erzeugen eines Listeneintrages nicht erlaubt.

***ACTIVE**
Der Server ist geladen.

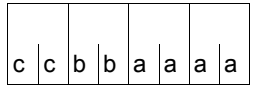
***NOT-ACTIVE**
Der Server ist nicht geladen.

Hinweis zum Makroaufruf

Für den Makroaufruf **ILEMIT** gelten die Hinweise, die bei **ILEMGT** beschrieben sind. Das Layout eines Listeneintrags ist ebenfalls dort beschrieben.

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros ILEMIT wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Der Makro wurde normal ausgeführt.
X'61'	X'01'	X'000A'	Der ILE-Eintrag ist ungültig.
X'61'	X'01'	X'0012'	ILE nicht gefunden (bei ACTION=*DELETE oder ACTION=*UPDATE).
X'61'	X'01'	X'0013'	ILE existiert bereits (bei ACTION=*CREATE).
X'61'	X'01'	X'0018'	Ein reserviertes Feld enthält keine binären Nullen.
X'61'	X'01'	X'0020'	Ein ungültiger Wert für STATE, CONTROL oder HSI_CODE wurde festgestellt.
X'61'	X'01'	X'0021'	Geforderte Aktion kann bei dieser CONTROL-Einstellung nicht ausgeführt werden.
X'61'	X'01'	X'0022'	Server-Adresswort (REF_DISPL) in der IL-Routine ist nicht schreibbar.
X'61'	X'01'	X'0023'	Die benutzerspezifische IL-Routine ist nicht geladen.
X'61'	X'01'	X'0024'	Fehlende Operanden (STATE oder CONTROL) beim Erzeugen des Listeneintrags.
X'61'	X'01'	X'002C'	Der ILE-Name ist ungültig. Er darf nicht mit einem Leerzeichen beginnen.
X'61'	X'40'	X'0050'	HSI-Codes von ILE und ILE-Server passen nicht zusammen: ILE wurde nicht erzeugt.
X'61'	X'01'	X'0130'	Ungültige Adresse (LOAD_ADDR, SERVER_ADDR) im aktuellen Listeneintrag.
X'61'	X'40'	X'0194'	Fehler bei Validierung eines Klasse-6-Speicherbereichs.
X'00'	X'01'	X'FFFF'	Die Funktion wird nicht mehr oder noch nicht unterstützt.
X'00'	X'03'	X'FFFF'	Die Version der Schnittstelle wird nicht unterstützt.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

IOSID – Betriebssystem und -Version abfragen

Allgemeines

Anwendungsgebiet: Abfragen und Zugriff zu Listen und Tabellen; siehe Seite 158
Makrotyp: O-Typ; siehe Seite 28

Makrobeschreibung

Der Makro **IOSID** informiert über Kennung (Bezeichnung) und Versionsnummer des Betriebssystems. Die Information wird in die globale Textvariable &IOSID oder in das Register R1 eingetragen. Der Eintrag erfolgt in der Form:

`C'x'` in &IOSID

`C'xvvv'` in Register R1

Es bedeuten:

```
x=2:  BS2000
vvv:  Bezeichnung der Version, z.B. 190 ≙ V19.0
```

Der Eintrag in &IOSID erfolgt während der Assemblierung des Makros und ist von der verwendeten Makrobibliothek abhängig.

Zu globalen Textvariablen (variable Parameter) siehe Handbuch „ASSEMBH“ [2].

Makroaufrufformat und Operandenbeschreibung

IOSID
[GBLC]

GBLC
Die Bezeichnung des Betriebssystems wird in die globale Textvariable &IOSID eingetragen.

Angabe ohne Operand
Die Bezeichnung des Betriebssystems und die Versionsnummer werden in das Register R1 eingetragen.

Beispiel

```
IOSID    START
         PRINT NOGEN
*
*    Definition of macro QUERY
*
         MACRO ------------------------------------------------------------  (1)
&NAME    QUERY
         GBLC  &IOSID
&NAME    NOP   &NAME
         AIF   ('&IOSID'(1,1) EQ '2').BS2 ---------------------------------  (2)
         MNOTE 9,'Only BS2000 possible'
.BS2     WROUT MESS,ERROR
ERROR    B     END
MESS     DC    Y(ENDM-MESS)
         DC    X'404001'
         DC    C'Operating system BS2000'
ENDM     EQU   *
END      NOP   &NAME
         MEND
*
*        Program start
*
         BALR  3,0
         USING *,3
         IOSID ------------------------------------------------------------  (3)
         ST    1,WORD
         MVC   GR1CONT,WORD
         WROUT GR1OUT,END1
         IOSID GBLC -------------------------------------------------------  (4)
CALLMAC  QUERY ------------------------------------------------------------  (5)
END1     TERM
GR1OUT   DC    Y(GR1END-GR1OUT)
         DC    X'404001'
         DC    C'Reg1: '
GR1CONT  DS    CL4
GR1END   EQU   *
WORD     DS    F
         END
```

(1) Zur Auswertung des Inhalts von &IOSID wird der Makro QUERY erstellt:
- Meldung ausgeben (WROUT), wenn &IOSID das Zeichen „2“ enthält (BS2000)
- MNOTE-Meldung für alle übrigen Fälle

(2) Die globale Textvariable &IOSID wird abgefragt.

(3) Kennung und Version des Betriebssystems sind in Register R1 auszugeben.

(4) Die Kennung des Betriebssystems ist in die globale Textvariable &IOSID auszugeben.

(5) Aufruf des Makros QUERY.

Ablaufprotokoll:

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,iosid), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,iosid))
%  ASS6011 ASSEMBLY TIME: 336 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 87 MSEC
//end
%  ASS6012 END OF ASSEMBH
/start-executable-program library=macexmp.lib,element-or-symbol=iosid
%  BLS0523 ELEMENT 'IOSID', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'IOSID', VERSION ' ' OF '<date> <time>' LOADED
Reg1: 2190 ————————————————————————————————————————————————————  (6)
Operating system BS2000 ———————————————————————————————————————  (7)
```

(6) Ausgabe des aufbereiteten Registers R1 nach Aufruf des Makros **IOSID** ohne Operanden. Es enthält folgende Informationen:

Betriebssystem: BS2000
Version: 19.0

(7) Nach Aufruf des Makros **IOSID** mit dem Operanden GBLC wird das Ergebnis der Auswertung durch den Makro QUERY ausgegeben.

JINF – Jobdaten anfordern

Allgemeines

Anwendungsgebiete: Abfragen und Zugriffe zu Listen und Tabellen; siehe Seite 158
Kommunikation; siehe Seite 167
Makrotyp: S-Typ, MF-Format **1**:
24-Bit-Schnittstelle: Standardform/E-Form/L-Form
31-Bit-Schnittstelle: Standardformsiehe Seite 29

Ein Job (Auftrag) beginnt mit dem Kommando SET-LOGON-PARAMETERS und endet mit dem Kommando EXIT-JOB. Ein Job kann auch dadurch erzeugt werden, dass mit dem Kommando ENTER-PROCEDURE eine Prozedurdatei an das System übergeben wird. Ein Job wird vom Betriebssystem je nach seinen Jobdaten (Jobattributen) in eine Jobklasse eingereiht. Zu den Jobdaten zählen sowohl benutzerspezifische (Abrechnungs-) Daten als auch jobspezifische Daten (Jobname, Jobklasse, Jobpriorität, Startzeit,...). Die letztgenannten Angaben übergibt der Anwender dem Betriebssystem im Kommando SET-LOGON-PARAMETERS bzw. im Kommando ENTER-JOB oder ENTER-PROCEDURE.

Makrobeschreibung

Der Makro **JINF** überträgt eine Liste von Jobdaten in einen anzugebenden Bereich. Die Liste ist unterteilt in Daten, die im Kommando SET-LOGON-PARAMETERS bzw. ENTER-JOB spezifiziert wurden und in aktuelle Jobdaten (Job-Startzeit, Anzahl Jobwiederholungen). Der Makro **DJINF** erzeugt eine Beschreibung (DSECT/ Datenabschnitt) zu der Ausgabeliste. Die Längen der einzurichtenden Bereiche sind dieser Beschreibung zu entnehmen.
Im 31-Bit-Adressierungsmodus muss der einzurichtende Bereich (Operand PARLIST) mit dem Standardheader beginnen.

Makroaufrufformat und Operandenbeschreibung

```
JINF

{ [PARMOD=24] [,JATTR=adr / (r)] [,RUNTIME=adr / (r)]
                  [,MF=L / (E,..)]
  [PARMOD=31],PARLIST=adr / (r)                        }
```

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen. (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen. (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

JATTR=
beschreibt die Adresse eines Bereichs zur Aufnahme der Jobdaten bei Auftragserzeugung. Der Bereich ist auf Wortgrenze auszurichten.

adr
symbolische Adresse (Name) des Bereichs

(r)
r = Register mit dem Wert der Adresse adr

RUNTIME=
beschreibt die Adresse eines Bereichs zur Aufnahme aktueller Jobdaten (Job-Startzeit, Anzahl Jobwiederholungen). Der Bereich ist auf Wortgrenze auszurichten.

adr
symbolische Adresse (Name) des Bereichs

(r)
r = Register mit dem Wert der Adresse adr

PARLIST=
beschreibt die Adresse eines Bereichs zur Aufnahme der Jobdaten. Der Bereich ist auf Wortgrenze auszurichten und muss mit dem Standardheader beginnen. Bei Verwendung des durch **DJINF** erzeugten Datenabschnitts ist der Standardheader initialisiert; in allen anderen Fällen muss der Standardheader durch den Anwender initialisiert werden.

adr
symbolische Adresse (Name) des Bereichs

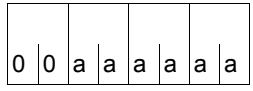
(r)
r = Register mit dem Adresswert adr

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Rückinformation und Fehleranzeige

Während der Makrobearbeitung enthält Register R1 die Adresse der Operandenliste.

R15:



Über die Ausführung des Makros JINF wird im Register R15 ein Returncode übergeben.

X'aaaaaa'	**Erläuterung**
X'000000'	Normale Ausführung.
X'000008'	Operandenfehler oder unzureichende Länge oder PARLIST mit PARMOD=24 angegeben.
X'00000C'	Systemfehler

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Beispiel

```
JINF    START
JINF    AMODE ANY
        PRINT NOGEN
        BALR  3,0
        USING *,3
        JINF  PARLIST=JIJOBDPL,PARMOD=31 ------------------------------  (1)
        MVC   EMPF1,JIJOUID ------------------------------------------  (2)
        MVC   EMPF2,JIJOACC
        MVC   EMPF3,JIJOJCLA
        MVC   EMPF4,JIJOTSN
        WROUT AUS1,ERROR,MODE=LINE,PARMOD=31
TERM    TERM
ERROR   TERM  DUMP=Y
        DS    0F
AUS1    DC    Y(AUS1END-AUS1)
        DS    CL3
        DC    C'USER ID:  '
EMPF1   DS    CL8
F1      DC    X'15'
        DC    C'ACCOUNT NUMBER: '
EMPF2   DS    CL8
F2      DC    X'15'
        DC    C'JOB CLASS:  '
EMPF3   DS    CL8
F3      DC    X'15'
        DC    C'TSN:  '
EMPF4   DS    CL4
AUS1END EQU   *
        DJINF DSECT=NO,PARLIST=YES ------------------------------------  (3)
        END
```

Mit diesem Programm werden unter Benutzung der symbolischen Namen in der Datenliste (Makro **DJINF**) einige Jobdaten (Benutzerkennung, Abrechnungsnummer, Jobklasse, Task Sequence Number = TSN) ausgegeben. Die Programmausführung erfolgt im 31-Bit-Adressierungsmodus.

Ablaufprotokoll:

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,jinf), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,jinf))
%  ASS6011 ASSEMBLY TIME: 344 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 81 MSEC
//end
%  ASS6012 END OF ASSEMBH
/start-executable-program library=macexmp.lib,element-or-symbol=jinf
%  BLS0523 ELEMENT 'JINF', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'JINF', VERSION ' ' OF '<date> <time>' LOADED
USER ID:  QM212    ------------------------------------------------------  (4)
ACCOUNT NUMBER: 89002
JOB CLASS:  JCDSTD
TSN:  2QSE
```

(1) Aufruf des Makros **JINF**.

(2) Jobdaten in das Ausgabefeld übertragen und mit dem Makro **WROUT** zeilenweise ausgeben.

(3) Aufruf des Makros **DJINF** zur Generierung der Datenliste (Ausgabebereich). Die Initialisierungswerte für den Standardheader sind eingetragen.

(4) Es folgt die Ausgabe (zeilenweise) der ausgewählten Jobdaten nach SYSOUT.

JMGDJP – DSECT oder Datenbereich für Makro JMGJPAR erstellen

Allgemeines

Anwendungsgebiete: Abfragen und Zugriffe zu Listen und Tabellen; siehe Seite 158
Makrotyp: Definitionsmakro; siehe Seite 28

Makrobeschreibung

Der Makro **JMGDJP** erzeugt eine Beschreibung (DSECT/Datenbereich) der Operandenliste des Makros **JMGJPAR**. DSECT und Datenliste beginnen mit dem Standardheader, die Initialisierungswerte sind eingetragen.

Makroaufrufformat und Operandenbeschreibung

JMGDJP
DSECT=YES / NO [,PREFIX=p]

DSECT=
gibt an, ob eine Dummy Section (DSECT) zum Datenbereich generiert wird oder ob Datenbereich und Definitionen direkt im Anwenderprogramm angelegt werden.

YES
Eine DSECT wird generiert.

NO
Datenbereich und Definitionen werden direkt im Anwenderprogramm angelegt.

PREFIX=
bezeichnet eine Zeichenfolge, mit der die symbolischen Namen der DSECT bzw. des Datenbereichs beginnen.

p
Präfix für die symbolischen Namen. Länge ≤ 2 Zeichen.
Voreinstellung: p=JP.

JMGJPAR – Jobparameter anfordern

Allgemeines

Anwendungsgebiet: Abfragen und Zugriffe zu Listen und Tabellen; siehe Seite 158
Makrotyp: R-Typ; siehe Seite 28

In den Kommandos SET-LOGON-PARAMETERS, ENTER-JOB und ENTER-PROCEDURE können für die Jobklasse zusätzliche Attribute (Jobparameter) angegeben werden, sofern die Systemverwaltung solche definiert und bekannt gegeben hat (siehe Kommandos ENTER-JOB, ENTER-PROCEDURE und SET-LOGON-PARAMETERS, Handbuch „Kommandos“ [19]).

Makrobeschreibung

Der Makro **JMGJPAR** überträgt die im SET-LOGON-PARAMETERS-, ENTER-JOB- oder ENTER-PROCEDURE-Kommando angegebenen Jobparameter in einen anzugebenden Bereich.
Der Makro **JMGDJP** erzeugt eine Beschreibung (DSECT/Datenliste) des Ausgabebereichs.

Makroaufrufformat und Operandenbeschreibung

JMGJPAR
PARLIST=adr / (r)

PARLIST=
beschreibt die Adresse eines Bereichs zur Aufnahme der Jobparameter. Der Bereich ist auf Wortgrenze auszurichten und muss mit dem Standardheader beginnen. Der Standardheader ist initialisiert, wenn der Bereich mit dem Makro **JMGDJP** erzeugt wurde.

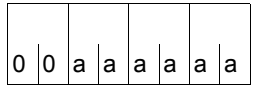
adr
symbolische Adresse (Name) des Bereichs

(r)
r = Register mit dem Adresswert adr

Rückinformation und Fehleranzeige

Während der Makrobearbeitung enthält Register R1 die Adresse der Operandenliste.

R15:



Über die Ausführung des Makros JMGJPAR wird im Register R15 ein Returncode übergeben.

X'aaaaaa'	**Erläuterung**
X'000000'	normale Ausführung
X'000004'	keine Jobparameter angegeben
X'000008'	Operandenfehler
X'00000C'	Systemfehler

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

JOBINFO – Jobdaten ausgewählter Jobs anfordern

Allgemeines

Anwendungsgebiet: Abfragen und Zugriffe zu Listen und Tabellen; siehe Seite 158
Makrotyp: S-Typ, MF-Format **2**: Standardform/C-/D-/L-/E-Form; siehe Seite 29

- In Erweiterung zu **JINF** liefert **JOBINFO** auch Jobdaten ausgewählter Jobs (Angabe der TSN). Informationen über Spoolout-Aufträge können nicht abgefragt werden.

Ein Job (Auftrag) beginnt mit dem Kommando SET-LOGON-PARAMETERS und endet mit dem Kommando EXIT-JOB. Ein Job kann auch dadurch erzeugt werden, dass mit dem Kommando ENTER-PROCEDURE eine Prozedurdatei an das System übergeben wird. Ein Job wird vom Betriebssystem je nach seinen Jobdaten (Jobattributen) in eine Jobklasse eingereiht. Zu den Jobdaten zählen sowohl benutzerspezifische (Abrechnungs-) Daten als auch jobspezifische Daten (Jobname, Jobklasse, Jobpriorität, Startzeit, ...). Die letztgenannten Angaben übergibt der Anwender dem Betriebssystem im Kommando SET-LOGON-PARAMETERS bzw. ENTER-JOB/ENTER-PROCEDURE.

Makrobeschreibung

Der Makro **JOBINFO** überträgt eine Liste von Jobdaten in einen Output-Bereich. Der Job muss mit dem Kommando SET-LOGON-PARAMETERS, ENTER-JOB oder ENTER-PROCEDURE beim Jobmanagement angemeldet sein, d. h. die Abfrage kann sich auf einen Dialog- oder Batch-Job oder einen noch nicht gestarteten Job im Jobpool beziehen. Der Job wird über die angegebene TSN identifiziert. Der nichtprivilegierte Anwender erhält nur Informationen über Jobs unter seiner Benutzerkennung oder über solche, die von seiner Benutzerkennung aus gestartet wurden. Der Umfang der auszugebenden Informationen kann mit dem Operanden FORM gesteuert werden.

Als Kurzinformation werden ausgegeben:
TSN des Auftrags (Jobs), Benutzerkennung, Abrechnungsnummer, Jobklasse, Jobname, Jobtyp (Dialog-, Batch-Job, Job im Jobpool), angeforderte CPU-Zeit, Startattribute (SOON, EARLIEST,...,DAILY, WEEKLY,..), Startdatum und Startzeit (wenn angefordert).

In der erweiterten Ausgabe werden zusätzlich ausgegeben:
Wiederholungsintervall und Anzahl der Wiederholungen eines Repeatjobs, Jobpriorität, RERUN- und FLUSH-Angabe, Startdatum (real), Startzeit (real), Spoolin-Datum, Spoolin-Zeit, Name der überwachenden Jobvariable, Name der SYSCMD-(Enter-)Datei, Länge des Jobparameter, Jobparameter, BCAM-Name (wenn der Job nicht auf dem Home-Rechner läuft), TSN der den Makro aufrufenden Task.

Der Input-/Output-Bereich wird im Anschluss an den Standardheader angelegt. Der Bereich beginnt mit dem (Input-)Feld für die TSN des Auftrags. Dieses Feld muss bei Aufrufen unter Verwendung der C-/D-Form dynamisch versorgt werden. Anschließend folgt der Output-Bereich. Seine Länge ist einer mit MF=C/D erzeugten Liste zu entnehmen. Die Initialisierungswerte sind im Standardheader eingetragen.

Makroaufrufformat und Operandenbeschreibung

```
JOBINFO
[TSN='tsn' / adr / (r)]
,FORM=LONG / SHORT
,CONST=YES / NO
,MF=S / E / L / C / D
[,PARAM=adr / (r)]
,PREFIX=J / p
,MACID=OBI / macid
```

TSN=
bezeichnet die Auftragsnummer des Auftrags (Jobs), dessen Jobdaten ausgegeben werden sollen. Der nichtprivilegierte Anwender kann nur Aufträge unter seiner Benutzerkennung oder solche, die von seiner Benutzerkennung aus gestartet wurden, angeben. Unter der Kennung der Systemverwaltung können beliebige Aufträge angegeben werden.

'tsn'
tsn = Auftragsnummer (TSN). Voreinstellung: tsn='␣␣␣␣' (4 Leerzeichen): das bedeutet, die Jobdaten des aufrufenden Programm werden ausgegeben.

adr
symbolische Adresse (Name) des Feldes mit der Auftragsnummer; Länge = 4 Byte.

(r)
r = Register, das rechtsbündig die TSN enthält.

FORM=
steuert den Umfang der auszugebenden Informationen, siehe Makrobeschreibung.

LONG
Erweiterte Ausgabe.

SHORT
Ausgabe der Kurzinformationen.

CONST=
legt fest, ob in die Ausgabeliste zur besseren Interpretation der Feldinhalte Equates eingetragen werden. CONST kann nur in Verbindung mit MF=C/D angegeben werden.

YES
Equates werden eingetragen.

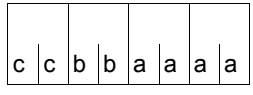
NO
Equates werden nicht eingetragen.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form oder D-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros JOBINFO wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Funktion ausgeführt.
X'00'	X'01'	X'0001'	Jobdaten des aufrufenden Auftrages beim Jobmanagement nicht gefunden.
X'00'	X'02'	X'0001'	Job ist beim Jobmanagement nicht angemeldet; keine Jobdaten gefunden. (TSN kann keinem Auftrag zugeordnet werden).
X'00'	X'03'	X'0001'	Job ist beim Jobmanagement nicht angemeldet; keine Jobdaten gefunden. (Spezifizierter Auftrag ist SPOOL-Task).
X'00'	X'01'	X'0003'	Systemfehler: Interner Schnittstellenfehler
X'00'	X'02'	X'0003'	Systemfehler: Jobbeschreibende Daten nicht zugreifbar
X'00'	X'03'	X'0003'	Systemfehler: kein Speicherplatz verfügbar.
X'00'	X'00'	X'0004'	Berechtigungsfehler: z.B., die angegebene TSN ist einer anderen Benutzerkennung zugeordnet.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Das aufrufende Programm wird beendet, wenn folgende Fehler auftreten:
- Der Datenbereich ist dem Aufrufer nicht zugewiesen.
- Der Datenbereich ist nicht auf Wortgrenze ausgerichtet.
- Der Datenbereich ist gegen Schreibzugriff geschützt.

JSATTCH – Jobscheduler mit dem Job Management System verbinden

Allgemeines

Anwendungsgebiet: Jobscheduler (Systemverwaltermakro); siehe Seite 169
Makrotyp: S-Typ, MF-Format **1**:
24-Bit-Schnittstelle: Standardform/E-Form/L-Form
31-Bit-Schnittstelle: Standardform; siehe Seite 29

- Der Makro **JSATTCH** kann nur unter der Kennung TSOS (Systemverwaltung) aufgerufen werden.
- Der Makro **DJSIPL** generiert für die 31-Bit-Schnittstelle eine Beschreibung (DSECT/Datenabschnitt) des Datenbereichs; der Makro **DJSI** für die 24-Bit-Schnittstelle.
- JMS = Job Management System; JSS = Job Scheduling Supports.
  JSS ist Bestandteil des Job Management Systems.

Makrobeschreibung

Mit dem Makroaufruf **JSATTCH** wird dem JMS angezeigt, dass der Jobscheduler bereit ist, JSS-Ereignisse zu verarbeiten.
JMS bestätigt durch Rückgabe einer Datumsangabe in Form der seit dem 01.01.1980 (0.00 Uhr) vergangenen Minuten.

Hinweis

Der Aufruf ist zwingend notwendig, damit JSS den Jobscheduler unterstützen kann (Jobstart-Anforderungen entgegennehmen, JSS-Ereignisse übermitteln).

Makroaufrufformat und Operandenbeschreibung

```
JSATTCH

{ CLOCK=adr / (r) [,PARMOD=24] [,MF=L / (E,..)] }
{ [PARMOD=31] ,PARLIST=adr / (r)                }
```

CLOCK=
beschreibt die Adresse eines Bereiches, in den die Datumsangabe eingetragen wird. Der Bereich muss auf Wortgrenze ausgerichtet sein. Der Operand kann nur für die 24-Bit-Schnittstelle angegeben werden.

adr
symbolische Adresse (Name) des Bereiches

(r)
r = Register mit dem Wert der Adresse adr

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29.
Die gültigen MF-Werte und Voreinstellungen für diesen Makro sind zu Beginn der Makrobeschreibung dargestellt.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

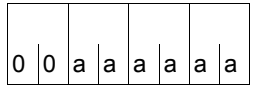
PARLIST=
bezeichnet die Adresse eines Bereichs mit der Datumsangabe. Der Bereich ist auf Wortgrenze auszurichten und muss mit dem Standardheader beginnen. Der Operand kann nur bei Verwendung der 31-Bit-Schnittstelle angegeben werden. Der Makro **DJSIPL** erzeugt eine Beschreibung (DSECT/Datenabschnitt) des Bereichs; die Initialisierungswerte für den Standardheader sind eingetragen.

adr
symbolische Adresse (Name) des Bereichs

(r)
r = Register mit dem Adresswert adr

Rückinformation und Fehleranzeigen

R15:



Über die Ausführung des Makros JSATTCH wird im Register R15 ein Returncode übergeben.

X'aaaaaa'	**Erläuterung**
X'000000'	Normale Ausführung
X'000008'	Operandenfehler
X'00000C'	Systemfehler
X'000010'	Unberechtigter Aufrufer (nicht TSOS)
X'000018'	Aufruf erfolgte nach einem JSATTCH-Aufruf oder der Job-Stream ist bereits beendet worden.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

JSDETCH – Jobscheduler vom Job Management System lösen

Allgemeines

Anwendungsgebiet: Jobscheduler (Systemverwaltermakro); siehe Seite 169
Makrotyp: S-Typ, MF-Format **1**:
24-Bit-Schnittstelle: Standardform/E-Form/L-Form
31-Bit-Schnittstelle: Standardform; siehe Seite 29

- Der Makro **JSDETCH** kann nur unter der Kennung TSOS (Systemverwaltung) aufgerufen werden.
- Der Makro **DJSIPL** erzeugt für die 31-Bit-Schnittstelle eine Beschreibung (DSECT/ Datenabschnitt) des Datenbereichs.
- JMS = Job Management System; JSS = Job Scheduling Supports.
  JSS ist Bestandteil des Job Management Systems.

Makrobeschreibung

Mit dem Makroaufruf **JSDETCH** wird dem JMS angezeigt, dass der Jobscheduler keine Systemunterstützung mehr benötigt. Der Jobscheduler wird von JSS getrennt.
Makroaufruf ohne Operandenangabe generiert die 24-Bit-Schnittstelle.

Makroaufrufformat und Operandenbeschreibung

```
JSDETCH

{ PARMOD=24 [,MF=L / (E,..)]          }
{ [PARMOD=31] ,PARLIST=adr / (r)      }
```

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

PARLIST=
bezeichnet die Adresse des Datenbereichs. Die Liste ist auf Wortgrenze auszurichten und muss mit dem Standardheader beginnen. Der Makro **DJSIPL** erzeugt eine Beschreibung (DSECT/ Datenabschnitt) des Datenbereichs mit initialisiertem Standardheader.

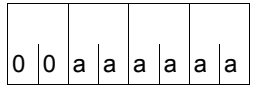
adr
symbolische Adresse (Name) des Datenbereichs

(r)
r = Register mit dem Adresswert adr

Rückinformation und Fehleranzeigen

Bei Verwendung der 31-Bit-Schnittstelle wird Register R1 überschrieben.

R15:



Über die Ausführung des Makros JSDETCH wird im Register R15 ein Returncode übergeben.

X'aaaaaa'	Erläuterung
X'000000'	Normale Ausführung
X'000008'	Operandenfehler

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

JSEXPCT – JSS-Ereignisse anfordern

Allgemeines

Anwendungsgebiet: Jobscheduler (Systemverwaltermakro); siehe Seite 169
Makrotyp: 24-Bit-Schnittstelle, S-Typ:
MF-Format **1** (Standardform/E-Form/L-Form)
31-Bit-Schnittstelle, S-Typ:
MF-Format **2** (Standardform); siehe Seite 29

- Der Makro **JSEXPCT** kann nur unter der Kennung TSOS (Systemverwaltung) aufgerufen werden.
- Der Makro **DJSIPL** erzeugt für die 31-Bit-Schnittstelle eine Beschreibung (DSECT/ Datenabschnitt) des Datenbereichs und aller JSS-Ereignisse, sowie Equates für den Returncode; der Makro **DJSI** für die 24-Bit-Schnittstelle.
- JMS = Job Management System; JSS = Job Scheduling Supports.
  JSS ist Bestandteil des Job Management Systems.

Makrobeschreibung

Mit dem Makroaufruf **JSEXPCT** wird von JMS das nächste vorliegende Ereignis für den Jobscheduler angefordert. Die Ereignisse beziehen sich auf einen von dem Jobscheduler verwalteten Job (Jobannahme, Jobbeendigung, Job-RELEASE, Job-CANCEL, ...), auf die Jobklasse (Jobklasse im HOLD-Status, Rücksetzen, Wiederannahme von Jobs) oder auf den Jobscheduler (Jobscheduler im HOLD-Status, Rücksetzen, geänderte STREAM-PARAMETER, ...).

Hinweis

Der Jobscheduler wird in einen Wartezustand versetzt, wenn kein Ereignis zur Verarbeitung vorliegt. Beim Eintreffen eines Ereignisses wird er aus dem Wartezustand herausgenommen.

Makroaufrufformat und Operandenbeschreibung

```
JSEXPCT

{ EVENT=adr / (r) [,PARMOD=24] [,MF=L / (E,..)] }
{ [PARMOD=31] ,PARLIST=adr / (r)                }
```

EVENT=
beschreibt die Adresse eines Bereiches, in den das Ereignis eingetragen wird. Der Bereich muss auf Wortgrenze ausgerichtet sein.

adr
symbolische Adresse (Name) des Bereiches

(r)
r = Register mit dem Wert der Adresse adr

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29.
Die gültigen MF-Werte und Voreinstellungen für diesen Makro sind zu Beginn der Makrobeschreibung dargestellt.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

PARLIST=
bezeichnet die Adresse des Datenbereichs. Die Liste ist auf Wortgrenze auszurichten und muss mit dem Standardheader beginnen. Der Makro **DJSIPL** erzeugt eine Beschreibung (DSECT/ Datenabschnitt) des Datenbereichs mit initialisiertem Standardheader.

adr
symbolische Adresse (Name) des Datenbereichs

(r)
r = Register mit dem Adresswert adr

Beschreibung der Ereignisse (24-Bit-Schnittstelle/31-Bit-Schnittstelle)

JSIEJINT/JSEXJINT: Annahme eines Jobs
Der Job Scheduler ist aufgefordert, den Job in die Menge der von ihm verwalteten Jobs aufzunehmen. Folgende Informationen über den Job werden dem Scheduler zur Verfügung gestellt:
- Auftragsnummer (TSN)
- Name der Jobklasse
- CPU-Zeit
- Job Priorität
- Ankunftzeit
- Starttyp (SOON, IMMEDIATE, ...)
- Startzeit
- Hold Status
- Job Parameter (J-PAR)

JSIEJTER/JSEXJTER: Beendigung des Laufs eines vom Job Scheduler gestarteten Jobs.

JSIEJHLD/JSEXJHLD: Setzen eines auf den Start wartenden Jobs in den Hold Status.
Der Job Scheduler ist aufgefordert, den Job beim Scheduling nicht mehr zu berücksichtigen. Ein im Hold Status befindlicher Job ist auch im Job Pool (Datei EQUISAMQ) als solcher gekennzeichnet.

JSIEJREL/JSEXJREL: Rücksetzen eines im Hold Status befindlichen Jobs. Der Job Scheduler ist aufgefordert, den Job beim Scheduling wieder zu berücksichtigen.

JSIEJCAN/JSEXJCAN: Löschen eines vom Job Scheduler noch nicht gestarteten Jobs.
Der Job Scheduler ist aufgefordert, den Job zu streichen.

JSIEJEXP/JSEXJEXP: Annahme eines Jobs, der sobald wie möglich gestartet werden soll.

JSIEJRES/JSEXJRES: Änderung von Attributen für einen noch nicht gestarteten Job auf Grund des Kommandos MODIFY-JOB.

JSIECHLD/JSEXCHLD: Setzen der angegebenen Jobklasse in den Hold Status.
Wie bei JSIEJHLD/JSEXJHLD ist der Job Scheduler aufgefordert, Jobs dieser Klasse nicht weiter zu berücksichtigen. Alle Jobs einer im Hold Status befindlichen Jobklasse sind auch im Job Pool (Datei EQUISAMQ) als solche gekennzeichnet.

JSIECREL/JSEXCREL: Rücksetzen der im Hold Status befindlichen Jobklasse.
Der Job Scheduler ist aufgefordert, Jobs dieser Klasse beim Scheduling wieder zu berücksichtigen.

JSIECAVA/JSEXCAVA: Für die angegebene Jobklasse ist CLASS-LIMIT unterschritten worden. Dies erfolgt entweder durch Beendigung eines in der Klasse laufenden Jobs oder durch Erhöhen von CLASS-LIMIT mit dem Kommando MODIFY-JOB-CLASS.

JSIESHLD/JSEXSHLD: Der angegebene Stream ist in den Hold Status versetzt worden. Der Job Scheduler ist aufgefordert, keine weiteren Jobs zu starten.

JSIESREL/JSEXSREL: Rücksetzen des Hold Status für den Stream. Der Job Scheduler ist aufgefordert, wieder Jobs zu starten.

JSIESCLQ/JSEXSCLQ: Anzeige, dass sich das Betriebssystem im Zustand 'Shutdown' befindet.
Dieses Ereignis hat die gleiche Wirkung wie JSIESHLD/JSEXSHLD, d.h. es werden keine weiteren Jobs gestartet.

JSIESCLI/JSEXSCLI: soll den Scheduler veranlassen, sich sofort zu beenden (Programmbeendigung). Dieses Ereignis wird mit dem STOP-JOB-STREAM-Kommando ausgelöst. Nach diesem Ereignis werden keine weiteren Ereignisse an den Stream gesendet.

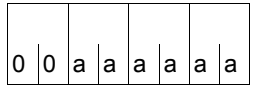
JSIESCHA/JSEXSCHA: Information über geänderte STREAM-PARAMETER.

JSIETIM/JSEXTIM: Periodisches, jede Minute eintretendes Ereignis, welches dem Scheduler die Ausführung von Zeitfunktionen (z.B. Unterstützung des Operanden START im SET-LOGON-PARAMETERS-Kommando und des Operanden REPEAT-JOB im ENTER-JOB-Kommando) ermöglicht.

Rückinformation und Fehleranzeigen

Während der Makrobearbeitung enthält Register R1 die Adresse der Operandenliste.

R15:



Über die Ausführung des Makros JSEXPCT wird im Register R15 ein Returncode übergeben.

X'aaaaaa'	**Erläuterung**
X'000000'	Normale Ausführung
X'000008'	Operandenfehler
X'00000C'	Systemfehler
X'000010'	Unberechtigter Aufrufer (nicht TSOS)
X'000018'	Aufruf erfolgte vor einem JSATTCH-Aufruf oder nach einem JSDETCH-Aufruf

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

JSINFO – STREAM-PARAMETER abfragen

Allgemeines

Anwendungsgebiet: Jobscheduler (Systemverwaltermakro); siehe Seite 169
Makrotyp: S-Typ, MF-Format **1**:
24-Bit-Schnittstelle: Standardform/E-Form/L-Form
31-Bit-Schnittstelle: Standardform; siehe Seite 29

- Der Makro **JSINFO** kann nur unter der Kennung TSOS (Systemverwaltung) aufgerufen werden.
- Der Makro **DJSIPL** generiert für die 31-Bit-Schnittstelle eine Beschreibung (DSECT/Datenabschnitt) des Datenbereichs; der Makro **DJSI** für die 24-Bit-Schnittstelle.
- JMS = Job Management System; JSS = Job Scheduling Supports.
  JSS ist Bestandteil des Job Management Systems.

Makrobeschreibung

Mit dem Makroaufruf **JSINFO** werden die STREAM-PARAMETER der Stream-Definition abgefragt.
Der Makro **JSINFO** sollte vor dem Makro **JSATTCH** aufgerufen werden.

Makroaufrufformat und Operandenbeschreibung

```
JSINFO

{ JSINF=adr / (r) [,PARMOD=24] [,MF=L / (E,..)] }
{ [PARMOD=31] ,PARLIST=adr / (r)                }
```

JSINF=
beschreibt die Adresse eines Bereiches, in den die STREAM-PARAMETER eingetragen werden. Der Bereich muss auf Wortgrenze ausgerichtet sein.

adr
symbolische Adresse (Name) des Bereiches

(r)
r = Register mit dem Wert der Adresse adr

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert. Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

PARLIST=
bezeichnet die Adresse des Datenbereichs. Die Liste ist auf Wortgrenze auszurichten und muss mit dem Standardheader beginnen. Der Makro **DJSIPL** erzeugt eine Beschreibung (DSECT/ Datenabschnitt) des Datenbereichs mit initialisiertem Standardheader.

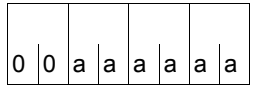
adr
symbolische Adresse (Name) des Datenbereichs

(r)
r = Register mit dem Adresswert adr

Rückinformation und Fehleranzeigen

Während der Makrobearbeitung enthält Register R1 die Adresse der Operandenliste.

R15:



Über die Ausführung des Makros JSINFO wird im Register R15 ein Returncode übergeben.

X'aaaaaa'	Erläuterung
X'000000'	Normale Ausführung
X'000008'	Operandenfehler
X'000010'	Unberechtigter Aufrufer (nicht TSOS)

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

JSRUNJB – Job zum Start übergeben

Allgemeines

Anwendungsgebiet: Jobscheduler (Systemverwaltermakro); siehe Seite 169
Makrotyp: S-Typ, MF-Format **1**:
24-Bit-Schnittstelle: Standardform/E-Form/L-Form
31-Bit-Schnittstelle: Standardform; siehe Seite 29

- Der Makro **JSRUNJB** kann nur unter der Kennung TSOS (Systemverwaltung) aufgerufen werden.
- Der Makro **DJSIPL** generiert für die 31-Bit-Schnittstelle eine Beschreibung (DSECT/Datenabschnitt) des Datenbereichs; der Makro **DJSI** für die 24-Bit-Schnittstelle.
- JMS = Job Management System; JSS = Job Scheduling Supports.
  JSS ist Bestandteil des Job Management Systems.

Makrobeschreibung

Mit dem Makroaufruf **JSRUNJB** fordert der Jobscheduler den Klassenscheduler auf, den angegebenen Job zu starten.
Der Job wird beschrieben durch
– seine TSN,
– Name der Jobklasse, in die der Job eingereiht wurde und
– Startangaben für den Job (Startindikator).

Hinweis

Falls das CLASS-LIMIT für die Jobklasse erreicht ist, wird der Job nicht gestartet. Ein entsprechender Returncode wird übergeben.

Makroaufrufformat und Operandenbeschreibung

```
JSRUNJB

{ PARMOD=24 ,STRTINF=adr / (r) [,MF=L / (E,..)] }
{ [PARMOD=31] ,PARLIST=adr / (r)                }
```

STRTINF=
beschreibt die Adresse eines Bereichs mit den Daten des zu startenden Jobs. Der Bereich muss auf Wortgrenze ausgerichtet sein.

adr
symbolische Adresse (Name) des Bereichs

(r)
r = Register mit dem Wert der Adresse adr

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

PARLIST=
bezeichnet die Adresse des Datenbereichs. Die Liste ist auf Wortgrenze auszurichten und muss mit dem Standardheader beginnen. Der Makro **DJSIPL** erzeugt eine Beschreibung (DSECT/ Datenabschnitt) des Datenbereichs mit initialisiertem Standardheader.

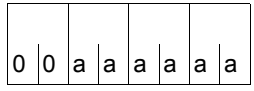
adr
symbolische Adresse (Name) des Datenbereichs

(r)
r = Register mit dem Adresswert adr

Rückinformation und Fehleranzeigen

Während der Makrobearbeitung enthält Register R1 die Adresse der Operandenliste.

R15:



Über die Ausführung des Makros JSRUNJB wird im Register R15 ein Returncode übergeben.

X'aaaaaa'	Erläuterung
X'000000'	Normale Ausführung
X'000004'	TSN des zu startenden Jobs ist nicht bekannt
X'000008'	Operandenfehler
X'0000OC'	Systemfehler
X'000010'	Unberechtigter Aufrufer (nicht TSOS)
X'000018'	Aufruf erfolgte vor einem JSATTCH-Aufruf oder nach einem JSDETCH-Aufruf
X'00001C'	Der Job befindet sich nicht in Q1 oder er ist vom Job Management System nicht als „für das Scheduling angenommen“ gekennzeichnet.
X'000020'	Jobklassenlimit überschritten.
X'000024'	Jobstream ist im HOLD-Status
X'000028'	Job ist im HOLD-Status

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

JSWAKE – Zeitereignis für Jobscheduler initiieren

Allgemeines

Anwendungsgebiet: Jobscheduler (Systemverwaltermakro); siehe Seite 169
Makrotyp: S-Typ, MF-Format **1**:
24-Bit-Schnittstelle: Standardform/E-Form/L-Form
31-Bit-Schnittstelle: Standardform; siehe Seite 29

- Der Makro **JSWAKE** kann nur unter der Kennung TSOS (Systemverwaltung) aufgerufen werden.
- Der Makro **DJSIPL** generiert für die 31-Bit-Schnittstelle eine Beschreibung (DSECT/Datenabschnitt) des Datenbereichs; der Makro **DJSI** für die 24-Bit-Schnittstelle.
- JMS = Job Management System; JSS = Job Scheduling Supports.
  JSS ist Bestandteil des Job Management Systems.

Makrobeschreibung

Mit dem Makro **JSWAKE** kann das nächste Zeitereignis (Timer Event) für den Jobscheduler initiiert werden. Die Zeitangabe für das Zeitereignis ist in einem Feld zu übergeben.

Makroaufrufformat und Operandenbeschreibung

```
JSWAKE

{ PARMOD=24 ,JSWAKE= adr / (r) [,MF=L / (E,..)] }
{ [PARMOD=31] ,PARLIST=adr / (r)                 }
```

JSWAKE=
beschreibt die Adresse des Feldes mit der Zeitangabe für das Zeitereignis. Das Feld muss auf Wortgrenze ausgerichtet sein.

adr
symbolische Adresse (Name) des Bereiches

(r)
r = Register mit dem Wert der Adresse adr

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert. Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

PARLIST=
bezeichnet die Adresse des Datenbereichs. Die Liste ist auf Wortgrenze auszurichten und muss mit dem Standardheader beginnen. Der Makro **DJSIPL** erzeugt eine Beschreibung (DSECT/ Datenabschnitt) des Datenbereichs mit initialisiertem Standardheader.

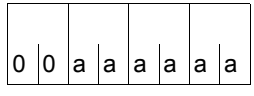
adr
symbolische Adresse (Name) des Datenbereichs

(r)
r = Register mit dem Adresswert adr

Rückinformation und Fehleranzeigen

Während der Makrobearbeitung enthält Register R1 die Adresse der Operandenliste.

R15:



Über die Ausführung des Makros JSWAKE wird im Register R15 ein Returncode übergeben.

X'aaaaaa'	Erläuterung
X'000000'	Normale Ausführung
X'000008'	Operandenfehler
X'000010'	Unberechtigter Aufrufer (nicht TSOS)

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

LDSLICE – Slice laden

Allgemeines

Anwendungsgebiet: Binden und Laden; siehe Seite 47
Makrotyp: S-Typ, MF-Format **2**: Standardform/C-/D-/L-/E-/M-Form; siehe Seite 29

Zum dynamischen Bindelader DBL siehe auch Handbuch „BLSSERV“ [4].

Makrobeschreibung

Der Makro **LDSLICE** lädt eine Slice, die in einem LLM vom Benutzer definiert wurde, in den Hauptspeicher. Welche Slices das LLM aufbauen, definiert der Benutzer mit Anweisungen SET-USER-SLICE-POSITION (siehe Handbuch „BINDER“ [5]).

Makroaufrufformat und Operandenbeschreibung

```
LDSLICE

[ { MODULE=name       } ]
  { MODULE@=adr / (r) }
[,MSG=*DBLOPT / INFORMATION / WARNING / ERROR / NONE ]

, { NAME=name       }
  { NAME@=adr / (r) }
,PATH=NO / YES
,RELOAD=NO / YES
[,SLICE@=adr / (r) / label]
,MF=S / C / D / E / L / M
[,PARAM=adr / (r)]
,PREFIX=P / p
[,LABEL=name]
```

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

LABEL=name
Angabe nur mit MF=M.
Name der Struktur, d.h. der DSECT, die den Datenbereich des **LDSLICE**-Makros beschreibt. Der Operand muss angegeben werden, wenn keine gültige USING-Anweisung für die Definition des Basisadressregisters für die DSECT des Datenbereichs angegeben ist. Der Operand LABEL muss zusammen mit dem Operanden PARAM angegeben werden. Beide Operanden werden benutzt, um eine gültige USING-Anweisung zu gewinnen.
Als Name kann angegeben werden:

1. Der Name der im Namensfeld eines vorhergehenden Makroaufrufs `name LDSLICE MF=D` angegeben wurde.
2. Der Name „xSLICDS“, wenn bisher noch kein Name „name“ angegeben wurde. Dabei ist „x“ der Wert des Operanden PREFIX eines vorhergehenden Makroaufrufs
   `LDSLICE MF=D,PREFIX=x`
   Der Standardwert von „x“ ist „P“.
3. Der Name der längeren DSECT, die den Datenbereich des **LDSLICE**-Makros enthält, wenn zuvor der Makroaufruf `LDSLICE MF=C` angegeben wurde.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. angegebenen Operanden PARAM und PREFIX siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29.Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

MODULE=name
Gibt den internen Namen des (PAM-)LLM an, der beim Erzeugen des LLM festgelegt wurde (maximal 32 Zeichen). Der interne Name sollte immer dann angegeben werden, wenn mehrere LLMs mit benutzerdefinierten Slices im Hauptspeicher geladen sind. Fehlt der Operand MODULE, wählt der DBL das erste LLM aus, das die Slice enthält.

MODULE@=
Gibt die Adresse eines Feldes an, das den internen Namen des LLM enthält. Angabe nur mit MF=M.

=adr
Symbolische Adresse des Feldes, das den Namen enthält.

=(r)
r = Register mit dem Adresswert adr.

MSG=
Legt die niedrigste Meldungsklasse fest, ab der Meldungen ausgegeben werden. Der Default-Wert wird aus dem Ladeaufruf mit LOAD- oder START-EXECUTABLE-PROGRAM (bzw. LOAD- oder START-PROGRAM) übernommen.

***DBLOPT**
Der Operandenwert wird aus dem letzten Aufruf des Kommandos /MODIFY-DBL-PARAMETERS übernommen. Falls für den betreffenden Operanden mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt MSG=INFORMATION.

INFORMATION
Die Meldungen aller Meldungsklassen werden ausgegeben.

WARNING
Nur Meldungen der Meldungsklasse WARNING und ERROR werden ausgegeben. Nicht ausgegeben werden Meldungen der Meldungsklasse INFORMATION.

ERROR
Nur Meldungen der Meldungsklasse ERROR werden ausgegeben.

NONE
Keine Meldungen werden ausgegeben.

NAME=name
Gibt den Namen der Slice an, die geladen wird. Für „name“ ist der Name anzugeben, den der Benutzer beim Definieren der Slice in der Anweisung SET-USER-SLICE-POSITION angegeben hat (maximal 32 Zeichen lang).
Die angegebene Slice kann nur geladen werden, wenn die Root-Slice (%ROOT) geladen ist.

NAME@=
Gibt die Adresse eines Feldes an, das den Namen der Slice enthält, die geladen wird. Angabe nur mit MF=M.

adr
Symbolische Adresse des Feldes, das den Namen enthält.

(r)
r = Register mit dem Adresswert adr.

PATH=
Legt fest, ob nur die mit NAME angegebene Slice geladen wird oder ob zusätzlich zur Slice NAME alle „höheren“ Slices im selben Ast (zwischen Root-Slice und der Slice NAME) geladen werden.

NO
Die mit NAME angegebene Slice wird geladen.

YES
Alle höheren Slices im selben Ast werden zusätzlich zur Slice NAME geladen.

RELOAD=
Legt fest, ob eine Slice, die bereits im Hauptspeicher geladen ist, wiedergeladen wird.

NO
Eine bereits geladene Slice wird nicht wiedergeladen.

YES
Eine bereits geladene Slice wird wiedergeladen und überschreibt die vorhergehende Slice im Hauptspeicher.

SLICE@=
Gibt die Adresse eines 4 Byte langen Feldes an, in das der DBL die Ladeadresse der Slice übergibt. Die übergebene Adresse bezieht sich auf das erste Byte der Slice, die mit NAME festgelegt wurde. Das Feld muss auf Wortgrenze ausgerichtet sein und Schreibzugriff haben.
Fehlt der Operand SLICE@ wird die Ladeadresse der Slice nicht übergeben.

adr
Symbolische Adresse des Feldes. Angabe nur mit MF=M.

(r)
r = Register mit dem Adresswert adr. Angabe nur mit MF=M.

label
Direkte Angabe der symbolischen Adresse des Feldes. Angabe nur mit MF=S oder MF=L.

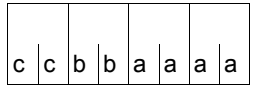
Regeln für das Laden von Slices

- Die Slice, die geladen werden soll, muss Teil der physikalischen Struktur sein, für die die Root-Slice geladen wurde.
- Die Root-Slice muss bereits mit dem Kommando START-PROGRAM bzw. LOAD-PROGRAM oder mit dem Makroaufruf **BIND** geladen worden sein.

Rückinformation und Fehleranzeigen

Falls ein Feld mit dem Operanden SLICE@ angegeben wurde, übergibt der DBL die Ladeadresse der Slice.

Standard-header:



Über die Ausführung des Makros LDSLICE wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Der Makro wurde normal ausgeführt.
X'0C'	X'01'	X'0018'	Ein reserviertes Feld des Datenbereichs ist nicht mit Nullen vorbelegt.
X'0C'	X'01'	X'0100'	Ungültige Parameterliste: Pflichtoperand NAME fehlt.
X'0C'	X'20'	X'0102'	Das angegebene Modul wurde nicht gefunden.
X'0C'	X'01'	X'0104'	Die angegebene Slice wurde nicht gefunden.
X'00'	X'00'	X'0108'	Die angegebene Slice ist bereits geladen.
X'0C'	X'01'	X'010C'	Das mit SLICE@ festgelegte Feld ist nicht auf Wortgrenze ausgerichtet.
X'0C'	X'01'	X'0110'	Das mit SLICE@ festgelegte Feld hat nur Lesezugriff oder ist nicht zugewiesen.
X'0C'	X'20'	X'0198'	Nicht genügend Speicherplatz vorhanden, um das Objekt zu laden.
X'0C'	X'40'	X'0204'	Interner Fehler im Memory Management.
X'0C'	X'40'	X'0208'	Interner Fehler im Data Manager.
X'0C'	X'20'	X'0300'	Fehler während eines Systemaufrufs.
X'00'	X'01'	X'FFFF'	Die Funktion wird nicht mehr oder noch nicht unterstützt.
X'00'	X'03'	X'FFFF'	Die Version der Schnittstelle wird nicht unterstützt.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

LEVCO – Priorität des Contingency-Prozesses ändern

Allgemeines

Anwendungsgebiete: Contingency-Verfahren; siehe Seite 111
STXIT-verfahren; siehe Seite 133
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

Makrobeschreibung

Mit dem Makroaufruf **LEVCO** kann der Anwender die Verarbeitungsebene eines laufenden Basis- oder Contingency-Prozesses ändern. Zugelassener Bereich:

Basisprozess: Verarbeitungsebene 0 - 127 (Standardwert: 0)
Contingency-Prozess: Verarbeitungsebene 1 - 127 (Standardwert: 1)

Makroaufrufformat und Operandenbeschreibung

```
LEVCO

{ NEWLV=level / (r)    }
{ NEWLVAD=adr / (r)    }

[,OLDLVAD=adr / (r)]
,QUEUE=FIFO / LIFO
[,PARMOD=24 / 31]
[,MF=L / (E,..)]
```

NEWLV=
bezeichnet die neue Verarbeitungsebene, mit der der Prozess fortgesetzt werden soll.

level
Verarbeitungsebene (ganze Zahl)

(r)
r = Register mit der Angabe für level

NEWLVAD=
bezeichnet die neue Verarbeitungsebene, mit der der Prozess fortgesetzt werden soll.

adr
symbolische Adresse eines 1 Byte langen Feldes, mit der Angabe für die Verarbeitungsebene

(r)
r = Register mit dem Adresswert für adr

OLDLVAD=
bezeichnet die alte Verarbeitungsebene des Prozesses, der **LEVCO** aufgerufen hat.

adr
symbolische Adresse eines 1 Byte langen Feldes, in das die alte Verarbeitungsebene eingetragen werden soll

(r)
r = Register mit dem Adresswert für adr

QUEUE=
gibt an, nach welchem Prinzip der aufrufende Prozess mit der geänderten Verarbeitungsebene unter Prozessen mit (dann) gleicher Verarbeitungsebene eingereiht werden soll.

FIFO
Es gilt das Prinzip First In First Out. Das bedeutet, dass der aufrufende Prozess nach Ausführung des Makros **LEVCO** nicht nur durch Prozesse mit höherer Verarbeitungsebene unterbrochen werden kann, sondern dass auch Prozesse mit gleicher Verarbeitungsebene, die bereits aktiviert sind, vor ihm fortgesetzt werden. Neue LIFO-Prozesse mit gleicher Verarbeitungsebene werden vor ihm gestartet, da der aufrufende Prozess am Ende der Warteschlange eingereiht wird.

LIFO
Es gilt das Prinzip Last In First Out. Der aufrufende Prozess kommt auf Grund seiner neuen Verarbeitungsebene an die Spitze der Warteschlange mit Prozessen gleicher Verarbeitungsebene; er kann also nur von Prozessen mit höherer Verarbeitungsebene oder von LIFO-Prozessen mit gleicher Verarbeitungsebene unterbrochen werden.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen. (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen. (Adressraum ≤ 2 GB).

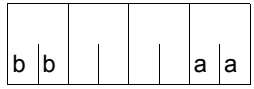
Hinweis zum Makroaufruf

Ein Prozess darf seine Verarbeitungsebene nicht derartig herabsetzen, dass sie unter der eines bereits gestarteten, aber unterbrochenen Prozesses liegt (siehe Abschnitt „Contingency-Prozesse“ auf Seite 111). Sind die Verarbeitungsebenen gleich, so darf nicht QUEUE=FIFO angegeben werden (nur QUEUE=LIFO ist erlaubt).

Rückinformation und Fehleranzeigen

Während der Makrobearbeitung enthält Register R1 die Adresse der Operandenliste.

R15:



Über die Ausführung des Makros LEVCO wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'aa'	X'bb'	Erläuterung
X'00'	X'00'	Der Prozess wird mit der neuen Priorität fortgesetzt.
X'04'	X'04'	Keine Aktion: Die neue Priorität ist niedriger (oder gleich bei FIFO) als die eines bereits aktivierten Prozesses.
X'10'	X'04'	Es wurden ungültige Operanden angegeben. Keine Aktion.

LGOFF – Auftrag beenden

Allgemeines

Anwendungsgebiet: Starten, Unterbrechen und Beenden; siehe Seite 72
Makrotyp: S-Typ, MF-Format **1**:
31-Bit-Schnittstelle: Standardform/L-/E-/C-/D-Form; siehe Seite 29

Makrobeschreibung

Durch den Aufruf des Makros **LGOFF** wird über den Makroanschluss des Kommando-Sprachübersetzers MCLP das Kommando EXIT-JOB gegeben, ohne dass der Programm-Modus verlassen wird (siehe Abschnitt „Makroaufrufe an den Kommandosprachübersetzer“ auf Seite 45). Meldungen bezüglich der Kommandobearbeitung werden auf SYSOUT ausgegeben.
Mit dem EXIT-JOB-Kommando beendet der Benutzer einen Auftrag. Daraufhin gibt das Betriebssystem die von dem Auftrag belegten virtuellen Speicherseiten und Geräte frei und stellt die Ausgabe-Systemdateien zur Ausgabe auf Schnelldrucker bzw. Band bereit. Wenn während der Abarbeitung des Auftrags neue Dateigenerationen erzeugt wurden, so wird der Basiswert der Gruppe aktualisiert (siehe DVS-Handbücher [7] und [8]). Auf SYSOUT wird in einer Meldung der Name der betroffenen Dateigenerationsgruppe, die erste und die aktuelle Generation und der Basiswert ausgegeben.

Makroaufrufformat und Operandenbeschreibung

```
LGOFF

[ { 'BUT[,TAPE]'      } ]
  { 'TAPE[,BUT]'      }
  { 'BUT[,NOSPOOL]'   }
  { 'NOSPOOL[,BUT]'   }

[,MF=C / D / L / (E,..)]
```

BUT=
Dieser Operand gilt nur für Dialogbenutzer und wird im Batch-Betrieb ignoriert. Der Benutzer zeigt damit an, dass er nach Beendigung des laufenden Auftrags erneut einen Auftrag beginnen will und daher die Verbindung zum Server nicht abgebaut werden soll. Fehlt die Angabe BUT, so wird die Verbindung zum Server abgebaut.

TAPE=
Dieser Eintrag bewirkt, dass die Systemdateien nicht auf Schnelldrucker, sondern auf Band ausgespult werden. Die Dateien SYSLST und/oder SYSOUT werden auf das gleiche Band in eine Datei mit dem Namen „TAPE.TSNnnnn“ geschrieben.
nnnn = TSN des zu beendenden Auftrags.
Die Datei SYSOPT wird auf ein gesondertes Band geschrieben und erhält ebenfalls einen Dateinamen „TAPE.TSNnnnn“, wobei aber nnnn eine neue Auftragsnummer ist. Sie wird dem Benutzer über die Systemdatei SYSOUT mitgeteilt.

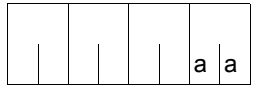
NOSPOOL=
Dieser Operand verhindert die Ausgabe der Systemdatei SYSLST und SYSOUT (für LOGGING=PARAMETERS (LISTING=YES) im Kommando SET-LOGON-PARAMETERS oder MODIFY-JOB-OPTIONS) und SYSOPT auf Drucker.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Hinweise zum Makroaufruf

- Die Kommando-Operanden müssen in Apostrophen eingeschlossen sein.
- Eine in dem aufrufenden Programm definierte STXIT-Routine für die Ereignisklasse „Programmbeendigung“ wird aktiviert.
- Überwachende Jobvariablen (Programm und Auftrag) werden auf „$T“ gesetzt.

Rückinformation und Fehleranzeigen

R15: 

Über die Ausführung des Makros LGOFF wird im rechtsbündigen Byte des Registers R15 ein Returncode übergeben.

X'aa'	Erläuterung
X'00'	Normale Beendigung.
X'04'	Es ist nicht genügend Speicherplatz vorhanden, die Anforderung wurde nicht bearbeitet.
X'08'	Fehler im Datenbereich (Adressbereich).
X'0C'	Der letzte Ausgabesatz, der in den Benutzerbereich gebracht wurde, ist abgeschnitten.
X'10'	Makroaufruf/Kommando-Fehler (das Kommando gab dem MCLP einen Fehler zurück).

Beispiel

```
LGOFF    START
         PRINT NOGEN
         BALR  3,0
         USING *,3
         LGOFF 'NOSPOOL,BUT'
         TERM
         END
```

Ablaufprotokoll:

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,lgoff), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,lgoff))
%  ASS6011 ASSEMBLY TIME: 408 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 77 MSEC
//end
%  ASS6012 END OF ASSEMBH
/start-executable-program library=macexmp.lib,element-or-symbol=lgoff
%  BLS0523 ELEMENT 'LGOFF', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'LGOFF', VERSION ' ' OF '<date> <time>' LOADED
/LOGOFF NOSPOOL,BUT
%  EXC0419 /LOGOFF AT 1505 ON <date> FOR TSN '52IS'
%  EXC0421 CPU TIME USED: 1.5734
%  JMS0150 INSTALLATION ' S200-40', BS2000 VERSION 'V190', HOST 'D016ZE04':
   PLEASE ENTER '/SET-LOGON-PARAMETERS' OR '?'
```

Die Ausgabe der Systemdateien wurde unterdrückt. Die Verbindung zum Server bleibt erhalten. Es kann sofort ein neues SET-LOGON-PARAMETERS-Kommando eingegeben werden; (siehe auch Handbuch „Kommandos“ [19], Kommando EXIT-JOB).

LKCAN – Lock-Anforderung löschen

Allgemeines

Anwendungsgebiet: Distributed-Lock-Manager (DLM); siehe Seite 142
Makrotyp: S-Typ, MF-Format **3**: C-/D-/L-/M-/E-Form; siehe Seite 29

Makrobeschreibung

Der Makro **LKCAN** löscht Lock-Anforderungen, die noch nicht vom DLM zugeteilt wurden.

Die Lock-Anforderungen, die vom **LKENQ**-Makro erstellt wurden und sich in der WAITING- oder CONVERTING-Warteschlange des Locks befinden, werden vollständig gelöscht. Befindet sich die Lock-Anforderung in der GRANTED-Warteschlange, wird mit einem Fehler-Code abgebrochen.

Die Lock-Anforderungen, die durch den **LKCVT**-Makro verändert werden sollen, werden nicht gelöscht. Es wird nur der Änderungsauftrag gelöscht.

Der Aufruf kann synchron oder asynchron erfolgen.

Makroaufrufformat und Operandenbeschreibung

```
LKCAN

MF=C / D / L / M / E
,ACKEVTT=*SYNCH / *TUCONTI / *TUEVENT / <var: enum-of _evttype_s:1>
,ACKNID=0 / <var: int:4>
,LOCKID=0 / <var: int:4>
,LSBADR=<var: pointer>
,USERPAR=0 / <var: int:4>
,PARAM=<var: pointer> / (reg: pointer>)
,PREFIX=N / p
,MACID=LDA / mac
```

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

ACKEVTT=
Beschreibt in welcher Art Informationen über die Löschung zurückgeliefert werden sollen. Es gibt drei Methoden zur Steuerung. Die angegebene Kurzkennung (Contingency-Kurzkennung oder Ereigniskennung) ist für die aktuelle Lösch-Anforderung gültig. Andere Lösch-Anforderungen von anderen Tasks können andere Kurzkennungen angeben.

***SYNCH**
Synchrone Lösch-Anforderung. Rücksprung aus dem Makro, wenn die Lock-Anforderung gelöscht oder eine Fehlerbedingung erkannt wurde. Die Lösch-Information wird im Returncode zurückgeliefert.

***TUCONTI**
Contingency-Prozess. Dieser Wert muss angegeben werden, um eine Bestätigung der Lösch-Anforderung während der Contingency-Verarbeitung zu erhalten.

***TUEVENT**
Ereignissteuerung (Eventing). Um die Lösch-Information zu erhalten, kann eine Ereignisvariable benutzt werden. Durch Aufruf des **SOLSIG**-Makros wird die Lösch-Information geliefert.

<var:enum-of _evttype_s:1>
Name des Feldes mit dem Wert der Lösch-Methode.

ACKNID=
Gibt bei einer asynchronen Lock-Anforderung die Contingency-Kurzkennung oder die Ereigniskennung an, die die Information erhält, dass der Lock jetzt gelöscht wurde.

<var: int:4>
Voreinstellung ist 0.
Contingency-Kurzkennung oder Ereigniskennung.

LOCKID=
Lock-Kurzkennung der Lock-Anforderung, die gelöscht werden soll.

<var: int:4>
Voreinstellung ist 0.
Lock-Kurzkennung, die vom **LKENQ**-Makro zurückgeliefert wurde.

LSBADR=
Feld mit der Adresse des Lock-Status-Blocks. Der Lock-Status-Block enthält den Returncode des asynchronen Aufrufs.

<var: pointer>
Name des Feldes mit der Adresse des Lock-Status-Blocks.

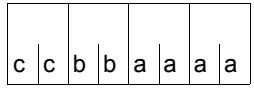
MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

USERPAR=
Die Parameter für asynchrone Mitteilungen, die an den Contingency-Prozess oder an die Ereignisvariable übergeben werden sollen.

<var: int:4>
Voreinstellung ist 0.
Benutzerdefinierte Werte.

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros LKCAN wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Der Makro wurde normal ausgeführt.
X'00'	X'00'	X'0001'	Der Makro wurde normal ausgeführt. Die Lösch-Anforderung wurde in die Warteschlange eingereiht. Die Lock-Anforderung wird asynchron gelöscht.
X'00'	X'00'	X'0004'	Löschen der Lock-Anforderung wurde eingeleitet, aber der Lock ist noch zugeteilt.
X'00'	X'01'	X'1005'	Der Lock-Status-Block ist nicht zugreifbar.
X'00'	X'01'	X'1006'	Der Typ von ACKNID ist nicht derselbe wie in ACKEVTT vereinbart.
X'00'	X'01'	X'100C'	Die Angabe im Operanden LOCKID ist ungültig.
X'00'	X'01'	X'1014'	Die ausgewählte Funktion ist für den Benutzer nicht erlaubt.
X'00'	X'01'	X'10FF'	Es wurde ein falscher Parameter angegeben, der keinen spezifischen Returncode hat.
X'00'	X'20'	X'2001'	Es trat ein interner Fehler auf.
X'00'	X'20'	X'2003'	Es trat ein interner Fehler im Zusammenhang mit dem Ressourcen-Block auf.
X'00'	X'20'	X'2004'	Es trat ein interner Fehler im Zusammenhang mit Zeitlimit-überschreitung auf.
X'00'	X'20'	X'2005'	Es trat ein interner Fehler im Zusammenhang mit der Lock-Anforderung auf.
X'00'	X'20'	X'2006'	Es trat ein interner Fehler im Zusammenhang mit XCS auf.
X'00'	X'82'	X'8004'	Der Lock wurde bereits freigegeben.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

LKCVT – Lock-Anforderung konvertieren

Allgemeines

Anwendungsgebiet: Distributed-Lock-Manager (DLM); siehe Seite 142
Makrotyp: S-Typ, MF-Format **3**: C-/D-/L-/M-/E-Form; siehe Seite 29

Makrobeschreibung

Der Makro **LKCVT** konvertiert eine existierende Lock-Anforderung. Der Lock muss mit dem Makro **LKENQ** erstellt und eine Lock-Kurzkennung muss zurückgeliefert worden sein.

Diese Lock-Kurzkennung (LOCKID) muss beim Aufruf des Makros **LKCVT** mitangegeben werden. Die Lock-Anforderung des Benutzers wird über die Lock-Kurzkennung identifiziert.

Die Lock-Anforderung wird in die CONVERTING-Warteschlange des Locks eingereiht. Kann der Lock zugeteilt werden, wird er in die GRANTED-Warteschlange eingereiht. Der Makro **LKCVT** wurde erfolgreich ausgeführt.

Die Lock-Anforderung kann synchron oder asynchron erfolgen.
Bei einer asynchronen Lock-Anforderung wird der Lock sobald wie möglich zugeteilt. Die ausgewählte Ereignissteuerung wird gestartet und die Zuteilung wird dem aufrufenden Programm mitgeteilt.

Während der Zuteilung (Konvertierung von einem schwächeren in einen stärkeren Lock-Schutz), kann der Benutzer angeben, dass der Lock-Value-Block gelesen werden soll (LVBCTL=*MOVE).
Während der Freigabe (Konvertierung von einem PW- oder EX-Modus in einen schwächeren Lock-Schutz), kann der Benutzer angeben, dass der Lock-Value-Block geändert werden soll (LVBCTL=*MOVE).

Makroaufrufformat und Operandenbeschreibung

```
LKCVT

MF=C / D / L / M / E
,ASYNCTL=*SYNCH / *ASYNCH / <var: enum-of _asyncctl_s:1>
,CONTROL=*STD / *EXPRESS / <var: enum-of _ctltype_s:1>
,GRANTID=0 / <var: int:4>
,GRTEVTT=*SYNCH / *TUCONTI / *TUEVENT / <var: enum-of _evttype_s:1>
,HOLDTIM=*INFINITE / *SYSTEM / <var: int:2> / <integer 0..32767>
,LCKMODE=*NU / *CR / *CW / *PR / *PW / *EX / <var: enum-of _lckmode_s:1>
```

LKCVT (Fortsetzung)
,LOCKID=0 / <var: int:4>
,LSBADR=<var: pointer>
,LVBCTL=*IGNORE / *MOVE / <var: enum-of _lvbctl_s:1>
,RELEVTT=*NO / *TUCONTI / *TUEVENT / <var: enum-of _evttype_s:1>
,RELID=0 / <var: int:4>
,STATUS=*UNCHANGE / *RESET / *INVALIDATE / <var: enum-of _status_s:1>
,USERPAR=0 / <var: int:4>
,WAITTIM=*STD / *INFINITE / <var: int:2> / <integer 0..32767>
,WAITTYP=*TIME / *IMMEDIATE / <var: enum-of _timetype_s:1>
,PARAM=<var: pointer> / (reg: pointer>)
,PREFIX=N / p
,MACID=LDC / mac

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

ASYNCTL=
Zuteilungssteuerung einer asynchronen Lock-Anforderung. Kann die Lock-Anforderung sofort zugeteilt werden, kann ein Contingency-Prozess oder ein Ereignis-Signal unterdrückt werden.

***SYNCH**
Kann die Lock-Anforderung sofort zugeteilt werden, wird ein Contingency-Prozess oder ein Ereignis-Signal unterdrückt.

***ASYNCH**
Auch wenn die Lock-Anforderung sofort zugeteilt werden kann, wird ein Contingency-Prozess oder ein Ereignis-Signal erstellt.

<var: enum-of _asyncctl_s:1>
Name des Feldes mit der Zuteilungssteuerung der asynchronen Lock-Anforderung zum Ausführungszeitpunkt.

CONTROL=
Gibt die Steuerungs-Optionen für die Konvertierungs-Anforderung an.

***STD**
Standard-Steuerung der Konvertierungs-Anforderung.

***EXPRESS**
Die Konvertierungs-Anforderung soll ausgeführt werden, bevor andere Konvertierungs-Anforderungen ausgeführt werden. Dieser Operanden-Wert sollte nur dann angegeben werden, wenn der konvertierte Lock-Modus sehr schnell wieder freigegeben wird.

<var: enum-of _ctltype_s:1>
Name des Feldes mit dem Wert, den der Benutzer festgelegt hat.

GRANTID=
Kurzkennung für die Lock-Zuteilung. Gibt an, welche Contingency-Kurzkennung oder Ereigniskennung bei asynchronen Lock-Anforderungen die Information erhält, dass der Lock jetzt zugeteilt wurde.

<var: int:4>
Voreinstellung ist 0.
Die Contingency-Kurzkennung oder Ereigniskennung.

GRTEVTT=
Beschreibt in welcher Art Informationen über die Lock-Zuteilung zurückgeliefert werden sollen. Es gibt es eine synchrone und zwei asynchrone Methoden zur Steuerung. Die angegebene Kurzkennung (Contingency-Kurzkennung oder Ereigniskennung) ist für die aktuelle Lock-Anforderung gültig. Andere Lock-Anforderungen von anderen Tasks können andere Kurzkennungen angeben.

***SYNCH**
Synchrone Lock-Anforderung. Rücksprung aus dem Makro, wenn der Lock zugeteilt oder eine Fehlerbedingung erkannt wurde. Die Zuteilungs-Information wird im Returncode zurückgeliefert.

***TUCONTI**
Contingency-Prozess. Dieser Wert muss angegeben werden, um eine Bestätigung der Zuteilungs-Anforderung während der Contingency-Verarbeitung zu erhalten. Der Benutzer wird über inkompatible Zuteilungs-Anforderungen informiert, die von anderen Benutzern in die Warteschlange eingereiht wurden. Die Freigabe-Contingency wird gestartet, wenn sich der Lock in einem inkompatiblen Lock-Modus befindet.

***TUEVENT**
Ereignissteuerung (Eventing). Um die Zuteilungs-Information zu erhalten, kann eine Ereignisvariable benutzt werden. Durch Aufruf des **SOLSIG**-Makros wird die Zuteilungs-Information geliefert. Die inkompatiblen Zuteilungs-Anforderungen von anderen Tasks werden über die selbe Ereignisvariable gemeldet.

<var:enum-of _evttype_s:1>
Name des Feldes mit dem Zuteilungs-Ereignis-Typ zum Ausführungszeitpunkt.

HOLDTIM=
Die Haltezeit ist die Zeit, die der Lock-Halter den Lock halten will. Nach Ablauf der Haltezeit wird ein Freigabe-Ereignis vom DLM erzeugt, damit der Lock-Halter seine Lock-Anforderung abschwächen bzw. freigeben kann.
Darf nur zusammen mit dem Operanden RELEVTT angegeben werden.

***INFINITE**
Kein Haltezeitlimit.

***SYSTEM**
Vom Betriebssystem definierte Haltezeit.

<var: int:2>
Vom Benutzer festgelegte Haltezeit in Sekunden.

<integer 0..32767>
Direkte Angabe der Haltezeit in Sekunden.

LCKMODE=
Gibt den angeforderten Lock-Modus an.

***NU**
Der Lock befindet sich im Null-Modus und ist immer zugeteilt. Er ist mit allen anderen Lock-Anforderungen kompatibel. Ein Zugriff auf die Ressource ist aber nicht erlaubt.

***CR**
Der Lock befindet sich im Concurrent-Read-Modus. Erlaubt sind andere Locks nur im Null-Modus, im Concurrent-Read-Modus, im Concurrent-Write-Modus, im Protected-Read-Modus und im Protected-Write-Modus. Dem Lock-Halter wird ein ungeschützter Lesezugriff auf die Ressource gewährt.

***CW**
Der Lock befindet sich im Concurrent-Write-Modus. Erlaubt sind andere Locks nur im Null-Modus, im Concurrent-Write-Modus oder im Concurrent-Read-Modus. Dem Lock-Halter wird ein ungeschützter Schreibzugriff auf die Ressource gewährt.

***PR**
Der Lock befindet sich im Protected-Read-Modus. Erlaubt sind andere Locks nur im Null-Modus, im Concurrent-Read-Modus oder im Protected-Read-Modus. Dem Lock-Halter wird ein geschützter Lesezugriff auf die Ressource gewährt.

***PW**
Der Lock befindet sich im Protected-Write-Modus. Erlaubt sind andere Locks nur im Null-Modus und im Concurrent-Read-Modus. Dem Lock-Halter wird ein geschützter Schreibzugriff auf die Ressource gewährt.

***EX**
Der Lock befindet sich im Exklusiv-Modus. Erlaubt sind andere Locks nur im Null-Modus. Nur der Lock-Halter darf auf die Ressource zugreifen.

<var: enum-of _lckmode_s:1>
Name des Feldes mit dem Lock-Modus.

LOCKID=
Lock-Kurzkennung des Locks, der konvertiert werden soll.

<var: int:4>
Voreinstellung ist 0.
Lock-Kurzkennung, die vom **LKENQ**-Makro zurückgeliefert wurde.

LSBADR=
Feld mit der Adresse des Lock-Status-Blocks. Der Lock-Status-Block enthält den Lock-Value-Block.

<var: pointer>
Name des Feldes mit der Adresse des Lock-Status-Blocks.

LVBCTL=
Steuerung des Lock-Value-Blocks.

***IGNORE**
Der Lock-Value-Block wird nicht benutzt.

***MOVE**
Den Lock-Value-Block lesen oder schreiben. Der Lock-Value-Block kann gelesen werden, wenn der Lock zugeteilt wird. Er kann geschrieben werden, wenn der Lock im PW- oder EX-Modus freigegeben wird.

<var: enum-of _lvbctl_s:1>
Name des Feldes mit der Steuerung des Lock-Value-Blocks zum Ausführungszeitpunkt.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden.

RELEVTT=
Beschreibt in welcher Art Informationen über die Lock-Freigabe zurückgeliefert werden sollen. Es gibt zwei asynchrone Methoden zur Steuerung. Die angegebene Kurzkennung (Contingency-Kurzkennung oder Ereigniskennung) ist für die aktuelle Lock-Anforderung gültig.

***NO**
Es werden keine Informationen über andere inkompatible Lock-Anforderungen die diesen Lock betreffen ausgegeben. Dies kann zu Deadlock-Problemen führen, wenn der

Lock nicht freigegeben ist. Ist die angegebene Haltezeit für diesen Lock abgelaufen, wird der Lock-Halter dieser Lock-Anforderung ohne weitere Meldung beendet.

***TUCONTI**
Contingency-Prozess. Dieser Wert muss angegeben werden, um eine Bestätigung der Freigabe-Anforderung während der Contingency-Verarbeitung zu erhalten. Der Benutzer wird über inkompatible Zuteilungs-Anforderungen informiert, die von anderen Benutzern in die Warteschlange eingereiht wurden. Die Freigabe-Contingency wird gestartet, wenn sich der Lock in einem inkompatiblen Lock-Modus befindet.

***TUEVENT**
Ereignissteuerungs (Eventing). Um die Freigabe-Information zu erhalten, kann eine Ereignisvariable benutzt werden. Durch Aufruf des **SOLSIG**-Makros wird die Zuteilungs-Information geliefert. Die inkompatiblen Zuteilungs-Anforderungen von anderen Tasks werden über die selbe Ereignisvariable gemeldet.

<var:enum-of _evttype_s:1>
Name des Feldes mit dem Freigabe-Ereignis-Typ zum Ausführungszeitpunkt.

RELID=
Kurzkennung für die Lock-Freigabe. Gibt an, welche Contingency-Kurzkennung oder Ereigniskennung die Information erhält, dass die Lock-Anforderung eine andere Lock-Anforderung eines anderen Benutzers blockiert.

<var: int:4>
Voreinstellung ist 0.
Die Contingency-Kurzkennung oder Ereigniskennung.

STATUS=
Lock-Status-Änderungsparameter.
Gibt an, ob der Lock-Status zurückgesetzt (= validated) wird, wenn ein Lock im PW- oder EX-Modus freigegeben wird.

***UNCHANGE**
Während der Freigabe eines Locks im PW- oder EX-Modus wird der Lock-Status nicht verändert.

***RESET**
Während der Freigabe eines Locks im PW- oder EX-Modus wird der Lock-Status wieder auf VALID gesetzt.

***INVALIDATE**
Während der Freigabe eines Locks im PW- oder EX-Modus wird der Lock-Status auf INVALID gesetzt.

<var: enum-of _status_s:1>
Name des Feldes mit dem Lock-Status-Änderungsparameter zum Ausführungszeitpunkt.

USERPAR=
Die Parameter für asynchrone Mitteilungen, die an den Contingency-Prozess oder an die Ereignissteuerung übergeben werden sollen.

<var: int:4>
Voreinstellung ist 0.
Benutzerdefinierte Werte.

WAITTIM=
Die Wartezeit ist die Zeit, die gewartet werden soll, bis der Lock zugeteilt ist.

***STD**
Standard-Wartezeit. Bei Angabe von WAITTYP=*IMMEDIATE wird der Lock sofort zugeteilt oder der Auftrag beendet. Falls die Bearbeitung des Auftrages verzögert wird (Cluster, Netz) wird 600 Sekunden gewartet.

***INFINITE**
Die Lock-Anforderung überschreitet niemals das Zeitlimit. Die Lock-Anforderung wartet solange bis der Lock zugeteilt oder eine Deadlock-Situation erkannt wird.

<var: int:2>
Vom Benutzer festgelegte Wartezeit in Sekunden.

<integer 0..32767>
Direkte Angabe der Wartezeit in Sekunden.

WAITTYP
Gibt die Wartezeit-Art für wartende Lock-Anforderungen an.

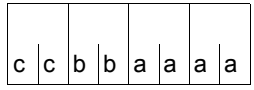
***TIME**
Der Wert der Wartezeit wird über den Operanden WAITTIM angegeben.

***IMMEDIATE**
Sofort-Lock-Anforderung. Die Lock-Anforderung wird nicht in die Warteschlange eingereiht, wenn sie nicht sofort zugeteilt werden kann. Der Wert für die Zeitüberschreitung kann über den Operanden WAITTIM angegeben werden. Dieser Wert wird verwendet, wenn die Behandlung der Lock-Anforderung zu einem DLM-internen Warte-Zustand führt. Dies kann z.B. auftreten, wenn eine Netzwerk-Verbindung nicht mehr zur Verfügung steht, nachdem die Lock-Anforderung zur Verarbeitung an einen anderen Knoten gesendet wurde. Die notwendige Antwort wird dadurch von diesem anderen Knoten niemals übertragen. Um dies zu verhindern, kann der Benutzer eine Wartezeit über den Operanden WAITTIM angeben.

<var: enum-of _timetype_s:1>
Benutzer-definierte Werte.

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros LKCVT wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Der Makro wurde normal ausgeführt. Die Lock-Anforderung wurde in einen neuen Lock-Modus konvertiert.
X'00'	X'00'	X'0001'	Der Makro wurde normal ausgeführt. Die Lock-Anforderung wurde in die Warteschlange eingereiht, der Lock wird asynchron zugeteilt.
X'00'	X'00'	X'0002'	Der Makro wurde ausgeführt. Der Lock wurde im angeforderten Lock-Modus zugeteilt, aber der Lock-Status ist ungültig. Diese Information wird nur ausgegeben, wenn der Lock-Modus größer oder gleich CW ist.
X'00'	X'01'	X'1005'	Der Lock-Status-Block ist nicht verfügbar.
X'00'	X'01'	X'1006'	Der Typ von GRANTID ist nicht derselbe wie in GRTTYPE vereinbart.
X'00'	X'01'	X'1007'	Der Typ von RELID ist nicht derselbe wie in RELTYPE vereinbart.
X'00'	X'01'	X'1008'	Ungültige Kombination von GRTEVTT und RELEVTT.
X'00'	X'01'	X'1009'	Die Angabe im Operanden WAITTIM ist ungültig.
X'00'	X'01'	X'100A'	Die Angabe im Operanden HOLDTIM ist ungültig.
X'00'	X'01'	X'100C'	Die Angabe im Operanden LOCKID ist ungültig.
X'00'	X'01'	X'1015'	Die Angabe im Operanden LCKMODE ist ungültig.
X'00'	X'01'	X'10FF'	Es wurde ein falscher Parameter angegeben, der keinen spezifischen Returncode hat.
X'00'	X'20'	X'2001'	Es trat ein interner Fehler auf.
X'00'	X'20'	X'2003'	Es trat ein interner Fehler im Zusammenhang mit dem Ressourcen-Block auf.
X'00'	X'20'	X'2004'	Es trat ein interner Fehler im Zusammenhang mit Zeitlimit-überschreitung auf.
X'00'	X'20'	X'2005'	Es trat ein interner Fehler im Zusammenhang mit der Lock-Anforderung auf.
X'00'	X'20'	X'2006'	Es trat ein interner Fehler im Zusammenhang mit XCS auf.
X'00'	X'40'	X'4001'	Ein Deadlock wurde entdeckt.
X'00'	X'40'	X'4003'	Die vorherige Lock-Anforderung ist noch nicht beendet.
X'00'	X'82'	X'8002'	Das Zeitlimit wurde überschritten.
X'00'	X'82'	X'8004'	Der Lock wurde bereits aus der Warteschlange freigegeben.
X'00'	X'82'	X'8005'	Der Lock wurde bereits gelöscht.

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'82'	X'8006'	Sofort-Lock-Anforderungen sind nicht möglich. Es existieren inkompatible Lock-Anforderungen, die bereits zugeteilt sind.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

LKDEQ – Lock-Anforderung freigeben

Allgemeines

Anwendungsgebiet: Distributed-Lock-Manager (DLM); siehe Seite 142
Makrotyp: S-Typ, MF-Format **3**: C-/D-/L-/M-/E-Form; siehe Seite 29

Makrobeschreibung

Der Makro **LKDEQ** gibt eine Lock-Anforderung aus der Warteschlange eines existierenden Locks frei. Wird die letzte (einzige) Lock-Anforderung freigegeben, wird der Lock selbst gelöscht. Der Makro **LKDEQ** kann synchron oder asynchron aufgerufen werden. Der Lock wird über seine Lock-Kurzkennung (LOCKID) identifiziert, die vom **LKENQ**-Makro zurückgeliefert wird. Diese Lock-Kurzkennung (LOCKID) muss beim Aufruf des Makros **LKDEQ** mitangegeben werden.
Der Benutzer kann angeben, dass während der Freigabe eines Locks im PW- oder EX-Modus, der Lock-Value-Block verändert wird (LVBCTL=*MOVE).

Makroaufrufformat und Operandenbeschreibung

```
LKDEQ

MF=C / D / L / M / E
,ASYNCTL=*SYNCH / *ASYNCH / <var: enum-of _asyncctl_s:1>
,DEQEVTT=*SYNCH / *TUCONTI / *TUEVENT / <var: enum-of _evttype_s:1>
,DEQID=0 / <var: int:4>
,LOCKID=0 / <var: int:4>
,LSBADR=<var: pointer>
,LVBCTL=*IGNORE / *MOVE / <var: enum-of _lvbctl_s:l>
,STATUS=*UNCHANGE / *RESET / *INVALIDATE / <var: enum-of _status_s:1>
,USERPAR=0 / <var: int:4>
,PARAM=<var: pointer> / (reg: pointer>)
,PREFIX=N / p
,MACID=LDD / mac
```

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

ASYNCTL=
Steuerung einer asynchronen Lock-Anforderung. Kann die Lock-Anforderung sofort aus der Warteschlange freigegeben werden, kann ein Contingency-Prozess oder ein Ereignis-Signal unterdrückt werden.

***SYNCH**
Kann die Lock-Anforderung sofort freigegeben werden, wird ein Contingency-Prozess oder ein Ereignis-Signal unterdrückt.

***ASYNCH**
Auch wenn die Lock-Anforderung sofort freigegeben werden kann, wird ein Contingency-Prozess oder ein Ereignis-Signal erstellt.

<var: enum-of _asyncctl_s:1>
Name des Feldes mit der asynchronen Lock-Anforderung zum Ausführungszeitpunkt.

DEQEVTT=
Beschreibt in welcher Art Freigabe-Informationen zurückgeliefert werden sollen. Es gibt drei Methoden zur Steuerung. Die angegebene Kurzkennung (Contingency-Kurzkennung oder Ereigniskennung) ist gültig für die aktuelle Freigabe-Anforderung. Andere Freigabe-Anforderungen von anderen Tasks können andere Kurzkennungen angeben.

***SYNCH**
Synchrone Lock-Anforderung den Lock aus der Warteschlange freizugeben. Rücksprung aus dem Makro, wenn der Lock freigegeben oder eine Fehlerbedingung erkannt wurde. Die Freigabe-Information wird im Returncode zurückgeliefert.

***TUCONTI**
Contingency-Prozess. Dieser Wert muss angegeben werden, um eine Bestätigung der Freigabe-Anforderung während der Contingency-Verarbeitung zu erhalten.

***TUEVENT**
Ereignissteuerung (Eventing). Um die Freigabe-Information zu erhalten, kann eine Ereignisvariable benutzt werden. Durch Aufruf des **SOLSIG**-Makros wird die Freigabe-Information geliefert.

<var:enum-of _evttype_s:1>
Name des Feldes mit dem Typ der Freigabe-Information zum Ausführungszeitpunkt.

DEQID=
Kurzkennung für die Lock-Freigabe. Gibt an, welche Contingency-Kurzkennung oder Ereigniskennung bei asynchronen Lock-Anforderungen die Information erhält, dass der Lock jetzt freigegeben wurde.

<var: int:4>
Voreinstellung ist 0. Die Contingency-Kurzkennung oder Ereigniskennung.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden.

LOCKID=
Lock-Kurzkennung des Locks, der aus der Warteschlange freigegeben werden soll.

<var: int:4>
Voreinstellung ist 0.
Lock-Kurzkennung eines früheren **LKENQ**-Makro-Aufruf, die aus der Warteschlange freigegeben werden soll.

LSBADR=
Feld mit der Adresse des Lock-Status-Blocks. Der Lock-Status-Block enthält den Lock-Value-Block.

<var: pointer>
Name des Feldes mit der Adresse des Lock-Status-Blocks.

LVBCTL=
Steuerung des Lock-Value-Blocks.

***IGNORE**
Der Lock-Value-Block wird nicht benutzt.

***MOVE**
Den Lock-Value-Block lesen oder schreiben. Der Lock-Value-Block kann gelesen werden, wenn der Lock zugeteilt wurde. Er kann geschrieben werden, wenn der Lock im PW- oder EX-Modus freigegeben wurde.

<var: enum-of _lvbctl_s:1>
Name des Feldes mit der Steuerung des Lock-Value-Blocks zum Ausführungszeitpunkt.

STATUS=
Gibt an, ob der Lock-Status zurückgesetzt (= validated) wird, wenn ein Lock im PW- oder EX-Modus freigegeben wird.

***UNCHANGE**
Während der Freigabe eines Locks im PW- oder EX-Modus wird der Lock-Status nicht verändert.

***RESET**
Während der Freigabe eines Locks im PW- oder EX-Modus wird der Lock-Status wieder auf VALID gesetzt.

***INVALIDATE**
Während der Freigabe eines Locks im PW- oder EX-Modus wird der Lock-Status auf INVALID gesetzt.

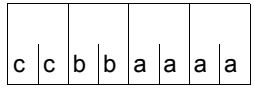
<var: enum-of _status_s:1>
Name des Feldes mit dem Lock-Status-Änderungsparameter zum Ausführungszeitpunkt.

USERPAR=
Die Parameter für asynchrone Mitteilungen, die an den Contingency-Prozess oder die Ereignissteuerung übergeben werden sollen.

<var: int:4>
Voreinstellung ist 0.
Benutzerdefinierte Werte.

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros LKDEQ wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Der Makro wurde normal ausgeführt. Der Lock wurde aus der Warteschlange freigegeben.
X'00'	X'00'	X'0001'	Der Makro wurde normal ausgeführt. Die Freigabe-Anforderung wurde in die Warteschlange eingereiht. Der Lock wird asynchron aus der Warteschlange freigegeben. Die Freigabe wird der ausgewählten Ereignissteuerungs-Methode mitgeteilt.
X'00'	X'01'	X'1005'	Der Lock-Status-Block ist nicht verfügbar.
X'00'	X'01'	X'100C'	Die Angabe im Operanden LOCKID ist ungültig.
X'00'	X'01'	X'1006'	Der Typ aus DEQID ist nicht derselbe wie in DEQEVTT vereinbart.
X'00'	X'01'	X'10FF'	Es wurde ein falscher Parameter angegeben, der keinen spezifischen Returncode hat.
X'00'	X'20'	X'2001'	Es trat ein interner Fehler auf.
X'00'	X'20'	X'2003'	Es trat ein interner Fehler im Zusammenhang mit dem Ressourcen-Block auf.
X'00'	X'20'	X'2004'	Es trat ein interner Fehler im Zusammenhang mit Zeitlimitüberschreitung auf.
X'00'	X'20'	X'2005'	Es trat ein interner Fehler im Zusammenhang mit der Lock-Anforderung auf.
X'00'	X'20'	X'2006'	Es trat ein interner Fehler im Zusammenhang mit XCS auf.
X'00'	X'40'	X'4003'	Die vorherige Lock-Anforderung ist noch nicht beendet.
X'00'	X'82'	X'8004'	Der Lock wurde bereits freigegeben.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

LKENQ – Lock generieren

Allgemeines

Anwendungsgebiet: Distributed-Lock-Manager (DLM); siehe Seite 142
Makrotyp: S-Typ, MF-Format **3**: C-/D-/L-/M-/E-Form; siehe Seite 29

Makrobeschreibung

Der Makro **LKENQ** generiert einen Lock. Existiert der Lock bereits, wird er nicht neu erstellt, aber die aufrufende Task wird ihm zugeordnet. Die Lock-Anforderung wird im angegebenen Lock-Modus in die Warteschlange eingereiht.

Ein Lock wird identifiziert mittels eines Lock-Namens und eines Geltungsbereiches. Der Lock-Name wird vom Benutzer an den DLM geliefert.

An die aufrufende Task wird eine Lock-Kurzkennung (LOCKID) zurückgeliefert, die in weiteren Aufrufen (**LKCVT**, **LKCAN**, **LKDEQ**) zur Bearbeitung angegeben werden muss. Die Lock-Kurzkennung ist eine task-spezifische Steuerung für den Lock.

Wurde der Lock bereits für diese Task in die Warteschlange eingereiht, wird die Lock-Kurzkennung der vorherigen **LKENQ**-Anforderung zurückgeliefert, aber der Makro wurde nicht ausgeführt. Der Lock-Modus wurde nicht verändert!

Makroaufrufformat und Operandenbeschreibung

```
LKENQ
```

```
MF=C / D / L / M / E
,ASYNCTL=*SYNCH / *ASYNCH / <var: enum-of _asyncctl_s:1>
,GRANTID=0 / <var: int:4>
,GRTEVTT=*SYNCH / *TUCONTI / *TUEVENT / <var: enum-of _evttype_s:1>
,HOLDTIM=*INFINITE / *SYSTEM / <var: int:2>  / <integer 0..32767>
,LCKMODE=*NU / *CR / *CW / *PR / *PW / *EX / <var: enum-of _lckmode_s:1>
,LSBADR=<var: pointer>
,LVBCTL=*IGNORE / *MOVE / <var: enum-of _lvbctl_s:1>
,MULTENQ=*NO / *YES / <var: enum-of _multiple_s:1>
,NAMEADR=<var: pointer>
,NAMELEN=0 / <integer 8..48>  / <var: int:2>
,NAMRNGE=*OWNSYSTEM / *CLUSTER / <var: enum-of _namerange_s:1>
,SCOPE=*NAMESPACEID / *USERID / *GROUPID / <var: emun-of_scope_s:1>
```

LKENQ (Fortsetzung)

```
,RELEVTT=*NO / *TUCONTI / *TUEVENT / <var: enum-of _evttype_s:1>
,RELID=0 / <var: int:4>
,TERMNTE=*STD / *FIRST / *SECOND / *THIRD / <var: enum-of _terminate_s:1>
,USERPAR=0 / <var: int:4>
,WAITTIM=*STD / *INFINITE / <var: int:2>  / <integer 0..32767>
,WAITTYP=*TIME / *IMMEDIATE / <var: enum-of _timetype_s:1>
,PARAM=<var: pointer> / (reg: pointer>)
,PREFIX=N / p
,MACID=LDE / mac
```

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

ASYNCTL=
Zuteilungssteuerung einer asynchronen Lock-Anforderung. Kann die Lock-Anforderung sofort zugeteilt werden, kann ein Contingency-Prozess oder ein Ereignis-Signal unterdrückt werden.

***SYNCH**
Kann die Lock-Anforderung sofort zugeteilt werden, wird eine Zuteilungs-Contingency oder ein Ereignis-Signal unterdrückt.

***ASYNCH**
Es wird immer eine Zuteilungs-Contingency oder ein Ereignis-Signal vom DLM erstellt, auch wenn die Lock-Anforderung sofort zugeteilt werden kann.

<var: enum-of _asyncctl_s:1>
Name des Feldes mit der Zuteilungs-Steuerung der asynchronen Lock-Anforderung zum Ausführungszeitpunkt.

GRANTID=
Kurzkennung für die Lock-Zuteilung. Gibt bei asynchronen Anforderungen an, welche Contingency-Kurzkennung oder Ereigniskennung die Information erhält, dass der Lock jetzt zugeteilt wurde.

<var: int:4>
Voreinstellung ist 0.
Die Contingency-Kurzkennung oder Ereigniskennung.

GRTEVTT=
Beschreibt in welcher Art Informationen über die Lock-Zuteilung zurückgeliefert werden sollen. Es gibt eine synchrone und zwei asynchrone Methoden zur Steuerung. Die angegebene Kurzkennung (Contingency-Kurzkennung oder Ereigniskennung) ist für die aktuelle Lock-Anforderung gültig. Andere Lock-Anforderungen von anderen Tasks können andere Kurskennungen angeben.

***SYNCH**
Synchrone Lock-Anforderung. Rücksprung aus dem Makro, wenn der Lock zugeteilt oder eine Fehlerbedingung erkannt wurde. Die Information wird im Returncode zurückgeliefert.

***TUCONTI**
Contingency-Prozess. Dieser Wert muss angegeben werden, um eine Bestätigung der Zuteilungs-Anforderung während der Contingency-Verarbeitung zu erhalten. Der Benutzer wird über inkompatible Lock-Anforderungen informiert, die von anderen Benutzern in die Warteschlange eingereiht wurden. Die Freigabe-Contingency wird gestartet, wenn der Lock sich in einem inkompatiblen Lock-Modus befindet.

***TUEVENT**
Ereignissteuerung (Eventing). Um die Zuteilungs-Information zu erhalten, kann eine Ereignisvariable benutzt werden. Durch Aufruf des **SOLSIG**-Makros wird die Zuteilungs-Information geliefert. Die inkompatiblen Lock-Anforderungen von anderen Tasks werden über die selbe Ereignisvariable gemeldet.

<var:enum-of _evttype_s:1>
Name des Feldes mit dem Zuteilungs-Ereignis-Typ zum Ausführungszeitpunkt.

HOLDTIM=
Die Haltezeit ist die Zeit, die der Lock-Halter den Lock halten will. Nach Ablauf der Haltezeit wird ein Freigabe-Ereignis vom DLM erzeugt, damit der Lock-Halter seine Lock-Anforderung abschwächen bzw. freigeben kann.
Darf nur zusammen mit dem Operanden RELEVTT angegeben werden.

***INFINITE**
Keine Haltzeitlimit.

***SYSTEM**
Vom Betriebssystem definierte Haltezeit.

<var: int:2>
Vom Benutzer festgelegte Haltezeit in Sekunden.

<integer 0..32767>
Direkte Angabe der Haltezeit in Sekunden.

LCKMODE=
Gibt den Lock-Modus des angeforderten Locks an.

***NU**
Der Lock befindet sich im Null-Modus und wird immer zugeteilt. Er ist mit allen anderen Lock-Anforderungen kompatibel. Ein Zugriff auf die Ressource ist nicht erlaubt.

***CR**
Der Lock befindet sich im Concurrent-Read-Modus. Erlaubt sind andere Locks nur im Null-Modus, im Concurrent-Read-Modus, im Concurrent-Write-Modus, im Protected-Read-Modus und im Protected-Write-Modus. Dem Lock-Halter wird ein ungeschützter Lesezugriff auf die Ressource gewährt.

***CW**
Der Lock befindet sich im Concurrent-Write-Modus. Erlaubt sind andere Locks nur im Null-Modus, im Concurrent-Write-Modus oder im Concurrent-Read-Modus. Dem Lock-Halter wird ein ungeschützter Schreibzugriff auf die Ressource gewährt.

***PR**
Der Lock befindet sich im Protected-Read-Modus. Erlaubt sind andere Locks nur im Null-Modus, im Concurrent-Read-Modus oder im Protected-Read-Modus. Dem Lock-Halter wird ein geschützter Lesezugriff auf die Ressource gewährt.

***PW**
Der Lock befindet sich im Protected-Write-Modus. Erlaubt sind andere Locks nur im Null-Modus und im Concurrent-Read-Modus. Dem Lock-Halter wird ein geschützter Schreibzugriff auf die Ressource gewährt.

***EX**
Der Lock befindet sich im Exklusiv-Modus. Erlaubt sind andere Locks nur im Null-Modus. Nur der Lock-Halter darf auf die Ressource zugreifen.

<var: enum-of _lckmode_s:1>
Name des Feldes mit dem Lock-Modus.

LSBADR=
Feld mit der Adresse des Lock-Status-Blocks. Ein Teil des Lock-Status-Blocks ist der Lock-Value-Block.

<var: pointer>
Name des Feldes mit der Adresse des Lock-Status-Blocks.

LVBCTL=
Gibt die Behandlung des Lock-Value-Blocks an.

***IGNORE**
Der Lock-Value-Block wird nicht benutzt.

***MOVE**
Den Lock-Value-Block lesen oder schreiben. Der Lock-Value-Block kann gelesen werden, wenn der Lock zugeteilt wurde. Er kann geschrieben werden, wenn der Lock freigegeben wurde.

<var: enum-of _lvbctl_s:1>
Name des Feldes mit der Behandlung des Lock-Value-Blocks zum Ausführungszeitpunkt.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden.

MULTENQ=
Mehrfache **LKENQ**-Anforderungen sind erlaubt.
Gibt an, ob der Benutzer mehr als eine Lock-Anforderung für denselben Lock-Namen (=Lock) in die Warteschlange einreihen darf. Jede dieser Lock-Anforderungen erhält ihre eigene Lock-Kurzkennung (LOCKID) und jede muss für sich wieder aus der Warteschlange freigegeben werden.

***NO**
Existiert bereits eine Lock-Anforderung dieser Task für den angegebenen Lock, wird diese (aktuelle) Lock-Anforderung abgewiesen.

***YES**
Diese (aktuelle) Lock-Anforderung wird unabhängig von bereits existierenden Lock-Anforderungen dieser Task für diesen Lock in die Warteschlange eingereiht. Diese Lock-Anforderung erhält ihre eigene eindeutige Lock-Kurzkennung (LOCKID).

<var: enum-of _multiple_s:1>
Name des Feldes mit dem Wert von MULTENQ zum Ausführungszeitpunkt.

NAMEADR=
Feld mit der Adresse des Lock-Namens.

<var: pointer>
Name des Feldes mit der Adresse des Lock-Namens.

NAMELEN=
Voreinstellung ist 0.
Gibt die Länge des Lock-Namens an.

<integer 8..48>
Direkte Angabe der Länge des Lock-Namens zum Übersetzungszeitpunkt.

<var: int:2>
Name des Feldes mit der Länge des Lock-Namens zum Ausführungszeitpunkt.

NAMRNGE=
Gibt an, in welchem Bereich der Lock-Name gültig ist. Wird eine **LKENQ**-Anforderung auf einem Single-System ausgeführt (das nicht Teil eines Clusters ist und auch nie Teil eines Clusters wird), wird der Lock behandelt, als sei das Single-System Teil eines Clusters. Diese Eigenschaft erlaubt das Portieren von Programmen von Single-Non-Clustered-Systemen auf Cluster-Systeme.

***OWNSYSTEM**
Der angegebene Lock-Name ist nur auf dem lokalen System gültig.

***CLUSTER**
Der angegebene Lock-Name ist clusterweit gültig.

<var: enum-of _namerange_s:1>
Name des Feldes mit dem Bereich, in dem der Lock-Name gültig ist, zum Ausführungszeitpunkt.

RELEVTT=
Beschreibt in welcher Art Informationen über die Lock-Freigabe zurückgeliefert werden sollen. Es gibt eine synchrone und zwei asynchrone Methoden zur Steuerung. Die angegebene Kurzkennung (Contingency-Kurzkennung oder Ereigniskennung) ist für die aktuelle Lock-Anforderung gültig. Andere Lock-Anforderungen von anderen Tasks können andere Kurzkennungen angeben.

***NO**
Es werden keine Informationen über andere inkompatible Lock-Anforderungen, die diesen Lock betreffen ausgegeben. Dies kann zu Deadlock-Problemen führen, wenn der Lock nicht freigegeben wird. Ist die angegebene Haltezeit für diesen Lock abgelaufen, wird der Lock-Halter dieser Lock-Anforderung ohne weitere Meldung beendet.

***TUCONTI**
Contingency-Prozess. Dieser Wert muss angegeben werden, um das Freigabe-Ereignis während der Contingency-Verarbeitung zu erhalten. Der Benutzer wird über inkompatible Lock-Anforderungen informiert, die von anderen Benutzern in die Warteschlange eingereiht wurden. Die Freigabe-Contingency wird gestartet, wenn der Lock sich in einem inkompatiblen Modus befindet.

***TUEVENT**
Ereignissteuerung (Eventing). Um das Freigabe-Ereignis zu erhalten, kann eine Ereignisvariable benutzt werden. Durch Aufruf des Makros **SOLSIG** wird die Information geliefert. Die inkompatiblen Lock-Anforderungen von anderen Tasks werden über die selbe Ereignisvariable gemeldet.

<var:enum-of _evttype_s:1>
Name des Feldes mit dem Freigabe-Ereignis-Typ zum Ausführungszeitpunkt.

RELID=
Kurzkennung für die Lock-Freigabe. Gibt an, welche Contingency-Kurzkennung oder Ereigniskennung die Information erhält, dass die Lock-Anforderung eine andere Lock-Anforderung eines anderen Benutzers blockiert.

<var: int:4>
Voreinstellung ist 0.
Die Contingency-Kurzkennung oder Ereigniskennung.

SCOPE=
Bestimmt den lokalen Geltungsbereich des Lock-Namens.

***NAMESPACEID**
ist Voreinstellung: Der angegebene Lock-Name wird als interner Lock-Name verwendet. Der erste Teil (8 Byte) des angegebenen Lock-Namens bildet dabei implizit den lokalen Geltungsbereich. Der lokale Geltungsbereich muss eine Zeichenfolge sein. Die gültigen Zeichen sind die Buchstaben „A..Z", „a..z"; die Ziffern „0..9" und die Sonderzeichen „@" und „#". Die maximale Länge des angegebenen Lock-Namens ist 48 Zeichen.

***USERID**
Die Benutzerkennung, zu der die Aufrufer-Task gehört, wird für die Bildung des internen Lock-Namens verwendet. Der DLM bestimmt die Benutzerkennung und setzt sie an den Anfang des Lock-Namens. Der erste Teil des angegebenen Lock-Namens wird nicht als lokaler Geltungsbereich betrachtet. Die maximale Länge des angegebenen Lock-Namens verringert sich auf 40 Zeichen.
Durch die Angabe des Operanden SCOPE=*USERID können die Locks einer Anwendung auf einfache Weise gegen den Zugriff einer anderen Anwendung geschützt werden. Die Anwendungen müssen nur unter verschiedenen Benutzerkennungen gestartet werden.

***GROUPID**
Die Benutzergruppe, zu der die Aufrufer-Task gehört, wird für die Bildung des internen Lock-Namens verwendet. Der DLM bestimmt die Benutzergruppe und setzt sie an den Anfang des Lock-Namens. Der erste Teil des angegebenen Lock-Namens wird nicht als lokaler Geltungsbereich betrachtet. Die maximale Länge des angegebenen Lock-Namens verringert sich auf 40 Zeichen.
Der Operand SCOPE=*GROUPID darf nur angegeben werden, wenn das Software-Produkt SECOS im Einsatz ist, sonst führt der **LKENQ**-Aufruf zu einem Fehler.

<var: enum-of _scope_s:1>
Name des Feldes mit dem lokalen Geltungsbereich des Lock-Namens zum Ausführungszeitpunkt.

TERMNTE=
Gibt an, in welcher Reihenfolge dieser Lock während der abnormalen Beendigung des Lock-Halter-Prozesses (Task oder Programm) freigegeben wird. Die Beendigungs-Sequenz ist in drei Klassen aufgeteilt. Die Locks des sich beendenden Prozesses werden freigegeben, unter Beachtung der angegebenen Sequenz oder sie werden alle zur gleichen Zeit freigegeben (von den anderen Prozessen aus gesehen).

<u>*STD</u>
Dieser Lock wird in der vom DLM festgelegten Beendigungs-Sequenz freigegeben.

***FIRST**
Dieser Lock wird freigegeben, bevor oder zur gleichen Zeit, wie die Locks der folgenden Klasse freigegeben werden.

***SECOND**
Dieser Lock wird freigegeben, nachdem oder zur gleichen Zeit, wie die Locks der vorherigen Klasse freigegeben werden. Und bevor oder zur gleichen Zeit, wie die Locks der folgenden Klasse freigegeben werden.

***THIRD**
Dieser Lock wird freigegeben, nachdem oder zur gleichen Zeit, wie die Locks der vorherigen Klasse freigegeben werden.

<var: enum-of _terminate_s:1>
Name des Feldes mit der Beendigungs-Klasse zum Ausführungszeitpunkt.

USERPAR=
Enthält die Parameter für die asynchrone Mitteilungen. Sie werden an den Contingency-Prozess oder an die Ereignissteuerung übergeben.

<var: int:4>
Benutzerdefinierte Werte. Voreinstellung ist 0.

WAITTIM=
Die Wartezeit ist die Zeit, die gewartet werden soll, bis der Lock zugeteilt ist.

<u>*STD</u>
Standard-Wartezeit. Bei WAITTYP=*TIME entspricht dies *INFINITE. Bei Angabe von WAITTYP=*IMMEDIATE wird der Lock sofort zugeteilt oder der Auftrag beendet. Falls die Bearbeitung des Auftrages verzögert wird (Cluster, Netz), wird 600 Sekunden gewartet.

***INFINITE**
Endlose Wartezeit. Die Anforderung überschreitet niemals das Zeitlimit. Die Lock-Anforderung wartet solange bis der Lock zugeteilt oder eine Deadlock-Situation erkannt wird.

<var: int:2>
Vom Benutzer festgelegte Wartezeit in Sekunden.

<integer 0..32767>
Direkte Angabe der Wartezeit in Sekunden.

WAITTYP
Gibt die Wartezeit-Art für wartende Lock-Anforderungen an.

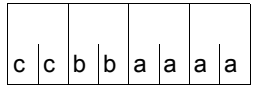
***TIME**
Der Wert der Zeitüberschreitung wird über den Operanden WAITTIM angegeben.

***IMMEDIATE**
Sofort-Lock-Anforderung. Die Lock-Anforderung wird nicht in die Warteschlange eingereiht, wenn sie nicht sofort zugeteilt werden kann. Der Wert für die Zeitüberschreitung kann über den Operanden WAITTIM angegeben werden. Dieser Wert wird verwendet, wenn die Behandlung der Lock-Anforderung zu einem DLM-internen Warte-Zustand führt. Dies kann z.B. auftreten, wenn eine Netzwerk-Verbindung nicht mehr zur Verfügung steht, nachdem die Lock-Anforderung zur Verarbeitung an einen anderen Knoten gesendet wurde. Die notwendige Antwort wird dadurch von diesem anderen Knoten niemals übertragen. Um dies zu verhindern, kann der Benutzer eine Wartezeit über den Operanden WAITTIM angeben.

<var: enum-of _timetype_s:1>
Benutzer-definierte Werte.

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros LKENQ wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Der Makro wurde normal ausgeführt. Der Lock wurde zugeteilt oder die Lock-Anforderung wurde in die Warteschlange eingereiht.
X'00'	X'00'	X'0001'	Der Makro wurde normal ausgeführt. Die Lock-Anforderung wurde in die Warteschlange eingereiht, der Lock wird asynchron zugeteilt.
X'00'	X'00'	X'0002'	Der Makro wurde ausgeführt, aber der Lock-Status ist ungültig. Diese Information wird nur ausgegeben, wenn der Lock-Modus größer oder gleich CW ist.

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'01'	X'1001'	Die Länge des Lock-Namens ist zu groß.
X'00'	X'01'	X'1002'	Die Adresse des Lock-Namens ist nicht verfügbar.
X'00'	X'01'	X'1003'	Der Lock-Name ist nicht erlaubt.
X'00'	X'01'	X'1004'	Der angegebene Namensraum ist ungültig.
X'00'	X'01'	X'1005'	Der Lock-Status-Block ist nicht verfügbar.
X'00'	X'01'	X'1006'	Der Typ des Operanden GRANTID ist nicht derselbe wie im Operanden GRTEVTT vereinbart.
X'00'	X'01'	X'1007'	Der Typ des Operanden RELID ist nicht derselbe wie im Operanden RELEVTT vereinbart.
X'00'	X'01'	X'1008'	Ungültige Kombination der Operanden GRTEVTT und RELEVTT.
X'00'	X'01'	X'1009'	Ungültige Angabe im Operanden WAITTIM.
X'00'	X'01'	X'100A'	Ungültige Angabe im Operanden HOLDTIM.
X'00'	X'01'	X'100B'	Ungültige Angabe im Operanden LCKTYPE.
X'00'	X'01'	X'1010'	Der Lock ist bereits für diese Task in die Warteschlange eingereiht.
X'00'	X'01'	X'1015'	Ungültige Angabe im Operanden LCKMODE.
X'00'	X'01'	X'10FF'	Ein falscher Parameter wurde angegeben, der keinen spezifischen Returncode hat.
X'00'	X'20'	X'2001'	Es trat ein interner Fehler auf.
X'00'	X'20'	X'2003'	Es trat ein interner Fehler im Zusammenhang mit dem Ressourcen-Block auf.
X'00'	X'20'	X'2004'	Es trat ein interner Fehler im Zusammenhang mit der Zeitlimitüber-schreitung auf.
X'00'	X'20'	X'2005'	Es trat ein interner Fehler im Zusammenhang mit der Lock-Anforderung auf.
X'00'	X'20'	X'2006'	Es trat ein interner Fehler im Zusammenhang mit XCS auf.
X'00'	X'40'	X'4003'	Die vorherige Lock-Anforderung ist noch nicht beendet.
X'00'	X'82'	X'8001'	Es können keine weiteren Locks angelegt werden.
X'00'	X'82'	X'8002'	Das Zeitlimit wurde überschritten.
X'00'	X'82'	X'8003'	Es ist im Moment nicht möglich, Locks mit NAMRNGE=*CLUSTER anzugeben.
X'00'	X'82'	X'8005'	Die Lock-Anforderung wurde bereits gelöscht.
X'00'	X'82'	X'8006'	Sofort-Lock-Anforderungen sind nicht möglich, weil bereits zugeteilte inkompatible Lock-Anforderungen existieren.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

LKEQU – DLM-eigene Layouts generieren

Allgemeines

Anwendungsgebiet: Distributed-Lock-Manager (DLM); siehe Seite 142]
Makrotyp: Definitionsmakro, siehe Seite 28

Makrobeschreibung

Der Makro **LKEQU** generiert die DLM-eigenen Layouts und Werte für die Ereignis-Typ-Codes und die globalen Returncodes, die von den verschiedenen Makros des DLMs gesetzt werden.

Der Makro **LKEQU** darf nur mit MF=D aufgerufen werden.

Makroaufrufformat und Operandenbeschreibung

LKEQU
MF=D ,PARAM=<var: pointer> / (reg: pointer) ,PREFIX=N / p ,MACID=LDQ / mac

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Layout des Datenbereichs für LKEQU MF=D

```
           LKEQU MF=D
1          MFTST MF=D,PREFIX=N,MACID=LDQ,ALIGN=F,                        C
1                DMACID=LDQ,SUPPORT=(D),DNAME=LDQMDL
2 NLDQMDL  DSECT ,
2                *,##### PREFIX=N, MACID=LDQ #####
1 NLDQNMM# EQU   48                        maximum length of a lock name
1 *                                        including the name space id
1 *                                        which always must be the
1 *                                        first part of the lock name
1 *                                        string
1 *
1 NLDQNMP# EQU   48                        reserved
1 *
1 NLDQNSP# EQU   8                         length ot the name space id
1 *
1 NLDQTIMI EQU   -1                        value for infinite wait- or
1 *                                        holdtime
1 *
1 NLDQTIMS EQU   -2                        value for system defined
1 *                                        holdtime
1 *
1 NLDQUNIT EQU   247                       DLM unit number in standard
1 *                                        header
1 *
1 *   layout of the event data for asynchronous calls
1 *
1 NLDQREG3 DS    0XL4                      word 1 or register 3 of event
1 *                                        data
1 NLDQETC  DS    FL1                       unit which issued the event
1 *                                        (here always the event type
1 *                                        code of DLM)
1 *   event type code for event notification
1 NLDQVENT EQU   22                        DLM event
1 *
1 NLDQEVNT DS    FL1                       notified event
1 *   type of the event that occurred
1 NLDQGRT  EQU   1                         grant event after LKENQ or
1 *                                        LKCVT call
1 NLDQWTOT EQU   2                         waittimeout event after LKENQ
1 *                                        or LKCVT call
1 NLDQREL  EQU   3                         release event
1 NLDQHTOT EQU   4                         holdtimeout event
1 NLDQERGT EQU   5                         error event after failing
1 *                                        LKENQ or LKCVT call
1 NLDQCANC EQU   6                         cancelled event after LKENQ
1 *                                        or LKCVT call when a LKCAN
```

```
1 *                                     was issued meanwhile
1 NLDQLSBR EQU   7                      general error event when lock
1 *                                     state block was not
1 *                                     accessible
1 NLDQOKDQ EQU   8                      dequeue event after LKDEQ
1 *                                     call
1 NLDQERDQ EQU   9                      error event after failing
1 *                                     LKDEQ call
1 NLDQOKCN EQU   10                     cancel done event after LKCAN
1 *                                     call
1 NLDQERCN EQU   11                     error event after failing
1 *                                     LKCAN call
1 *
1 NLDQBMOD DS    X                      lock mode of the lock request
1 *                                     which is blocked by the own
1 *                                     granted lock request
1 NLDQFILL DS    X                      reserved
1 *
1 *
1 NLDQREG4 DS    0XL4                   word 2 or register 4 of event
1 *                                     data
1 NLDQUPAR DS    F                      last given value of user
1 *                                     parameter
1 *
1 NLDQEQUATES# EQU  *-NLDQETC
```

LKINF – Informationen über Locks ausgeben

Allgemeines

Anwendungsgebiet: Distributed-Lock-Manager (DLM); siehe Seite 142
Makrotyp: S-Typ, MF-Format **3**: C-/D-/L-/M-/E-Form; siehe „Seite 29

Makrobeschreibung

Der Makro **LKINF** informiert darüber, welche Locks bereits benutzt werden. Zur besseren Auswahl können einige Such-Filter aktiviert werden. Die Locks werden über ihre Lock-Namen identifiziert. Die Lock-Namen können voll- oder teilqualifizierte Namen sein. Bei Angabe von teilqualifizierten Lock-Namen dauert der Zugriff ggf. sehr lange, da die gesamte DLM-Datenbasis nach Treffern durchsucht werden muss.

Makroaufrufformat und Operandenbeschreibung

```
LKINF

MF=C / D / L / M / E
,CONTROL=list-poss(3): *LOCKED / *LOCKED_BY_ME / *LCKMODE
,LCKMODE=*NU / *CR / *CW / *PR / *PW / *EX / <var: enum-of _lckmode_s:1>
,NAMEADR=<var: pointer>
,NAMELEN=0 / <integer 0..48>  / <var: int:2>
,NAMRNGE=*OWNSYSTEM / *CLUSTER / <var: enum-of _namerange_s:1>
,SCOPE=*NAMESPACEID / *USERID / *GROUPID / <var: emun-of_scope_s:1>
,NAMTYPE=*FULL / *PARTIAL / <var: enum-of _nametype_s:1>
,PARAM=<var: pointer> / (reg: pointer>)
,PREFIX=N / p
,MACID=LDI / mac
```

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

CONTROL=
Der Such-Filter steuert, welche Daten zurückgeliefert werden.

list-poss(3):
Aus den nachfolgenden Operandenwerten kann eine Liste gebildet werden, die maximal 3 Elemente enthält.

***LOCKED**
Es werden nur Informationen über Locks zurückgeliefert, die in einem Lock-Modus ungleich Null gehalten sind.

***LOCKED_BY_ME**
Es werden nur Informationen über Locks zurückgeliefert, die durch den Benutzer in einem Lock-Modus ungleich Null gehalten sind. Der Benutzer wird über seine Lock-Kurzkennung (LOCKID) identifiziert.

***LCKMODE**
Es werden nur die Informationen zurückgeliefert, die zu dem angegebenen Lock-Modus gehören.

LCKMODE=
Gibt den angeforderten Lock-Modus an. Wird nur ausgewertet, wenn auch der Operand CONTROL=*LCKMODE angegeben wurde.

***NU**
Der Lock befindet sich im Null-Modus und ist immer zugeteilt. Er ist mit allen anderen Lock-Anforderungen kompatibel. Ein Zugriff auf die Ressource ist nicht erlaubt.

***CR**
Der Lock befindet sich im Concurrent-Read-Modus. Erlaubt sind andere Locks nur im Null-Modus, im Concurrent-Read-Modus, im Concurrent-Write-Modus, im Protected-Read-Modus und im Protected-Write-Modus. Dem Lock-Halter wird ein ungeschützter Lesezugriff auf die Ressource gewährt.

***CW**
Der Lock befindet sich im Concurrent-Write-Modus. Erlaubt sind andere Locks nur im Null-Modus, im Concurrent-Write-Modus oder im Concurrent-Read-Modus. Dem Lock-Halter wird ein ungeschützter Schreibzugriff auf die Ressource gewährt.

***PR**
Der Lock befindet sich im Protected-Read-Modus. Erlaubt sind andere Locks nur im Null-Modus, im Concurrent-Read-Modus oder im Protected-Read-Modus. Dem Lock-Halter wird ein geschützter Lesezugriff auf die Ressource gewährt.

***PW**
Der Lock befindet sich im Protected-Write-Modus. Erlaubt sind andere Locks nur im Null-Modus und im Concurrent-Read-Modus. Dem Lock-Halter wird ein geschützter Schreibzugriff auf die Ressource gewährt.

***EX**
Der Lock befindet sich im Exklusiv-Modus. Erlaubt sind andere Locks nur im Null-Modus. Nur der Lock-Halter darf auf die Ressource zugreifen.

<var: enum-of _lckmode_s:1>
Name des Feldes mit dem Lock-Modus.

Hinweis
Die Operanden LCKMODE und CONTROL dürfen nicht zusammen angegeben werden. Die Angabe des Lock-Modus darf nur direkt über den Operanden LCKMODE oder indirekt über den Operanden CONTROL angegeben werden.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden.

NAMEADR=
Feld mit der Adresse des Lock-Namens.

<var: pointer>
Name des Feldes mit der Adresse des Lock-Namens.

NAMELEN=
Voreinstellung ist 0.
Gibt die Länge des Lock-Namens an.

<integer 0..48>
Direkte Angabe der Länge des Lock-Namens zum Übersetzungszeitpunkt.

<var: int:2>
Name des Feldes mit der Länge des Lock-Namens zum Ausführungszeitpunkt.

NAMRNGE=
Gibt an, in welchem Bereich der Lock-Name gültig ist.

***OWNSYSTEM**
Der angegebene Lockname ist nur auf dem lokalen System gültig.

***CLUSTER**
Der angegebene Lockname ist cluster-weit gültig.

<var: enum-of _namerange_s:1>
Name des Feldes mit dem Bereich, in dem der Lock-Name gültig ist.

NAMTYPE=
Gibt an, ob die Namens-Zeichenfolge einen voll-qualifizierten Lock-Namen oder einen teil-qualifizierten Lock-Namen bezeichnet.

***FULL**
Der angegebene Lock-Name ist ein voll-qualifizierter Lock-Name.

***PARTIAL**
Der angegebene Lock-Name ist ein teil-qualifizierter Lock-Name.

<var: enum-of _nametype_s:1>
Name des Feldes mit dem Typ des Lock-Namens.

SCOPE=
Bestimmt den lokalen Geltungsbereich des Lock-Namens.

***NAMESPACEID**
ist Voreinstellung: Der angegebene Lock-Name wird als interner Lock-Name verwendet. Der erste Teil (8 Byte) des angegebenen Lock-Namens bildet dabei implizit den lokalen Geltungsbereich. Der lokale Geltungsbereich muss eine Zeichenfolge sein. Die gültigen Zeichen sind die Buchstaben „A..Z", „a..z"; die Ziffern „0..9" und die Sonderzeichen „@" und „#". Die maximale Länge des angegebenen Lock-Namens ist 48 Zeichen.

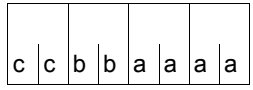
***USERID**
Die Benutzerkennung zu der die Aufrufer-Task gehört, wird für die Bildung des internen Lock-Namens verwendet. Der DLM bestimmt die Benutzerkennung und setzt sie an den Anfang des Lock-Namens. Der erste Teil des angegebenen Lock-Namens wird nicht als lokaler Geltungsbereich betrachtet. Die maximale Länge des angegebenen Lock-Namens verringert sich auf 40 Zeichen.
Durch die Angabe des Operanden SCOPE=*USERID können die Locks einer Anwendung auf einfache Weise gegen den Zugriff einer anderen Anwendung geschützt werden. Die Anwendungen müssen nur unter verschiedenen Benutzerkennungen gestartet werden.

***GROUPID**
Die Benutzergruppe zu der die Aufrufer-Task gehört, wird für die Bildung des internen Lock-Namens verwendet. Der DLM bestimmt die Benutzergruppe und setzt sie an den Anfang des Lock-Namens. Der erste Teil des angegebenen Lock-Namens wird nicht als lokaler Geltungsbereich betrachtet. Die maximale Länge des angegebenen Lock-Namens verringert sich auf 40 Zeichen.
Der Operand SCOPE=*GROUPID darf nur angegeben werden, wenn das kostenpflichtige Software-Produkt SECOS im Einsatz ist, sonst führt der **LKENQ**-Aufruf zu einem Fehler.

<var: enum-of _scope_s:1>
Name des Feldes mit dem lokalen Geltungsbereich des Lock-Namens zum Ausführungszeitpunkt.

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros LKINF wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Der Makro wurde normal ausgeführt. Mindestens ein entsprechender Lock wurde gefunden.
X'01'	X'00'	X'0000'	Der Makro wurde ausgeführt, aber es wurde kein entsprechender Lock gefunden.
X'02'	X'00'	X'0000'	Der Makro wurde ausgeführt, entsprechende Locks wurden gefunden, aber nicht alle Knoten haben wegen eines Netzwerksproblems Informationen geliefert.
X'00'	X'01'	X'1001'	Die Länge des Lock-Namens ist zu groß.
X'00'	X'01'	X'1002'	Die Adresse des Lock-Namens ist nicht verfügbar.
X'00'	X'01'	X'1003'	Der angegebene Lock-Name ist unzulässig.
X'00'	X'01'	X'1004'	Der angegebene Namensraum ist ungültig.
X'00'	X'01'	X'100F'	Der Such-Filter ist unzulässig.
X'00'	X'01'	X'1015'	Die Angabe im Operanden LCKMODE ist ungültig.
X'00'	X'01'	X'1016'	Die Angabe im Operanden NAMTYPE ist ungültig.
X'00'	X'01'	X'1017'	Die Angabe im Operanden NAMRNGE ist ungültig.
X'00'	X'01'	X'10FF'	Es wurde ein falscher Parameter angegeben, der keinen spezifischen Returncode hat.
X'00'	X'20'	X'2001'	Es trat ein interner Fehler auf.
X'00'	X'20'	X'2003'	Es trat ein interner Fehler im Zusammenhang mit dem Ressourcen-Block auf.
X'00'	X'20'	X'2004'	Es trat ein interner Fehler im Zusammenhang mit Zeitlimit-überschreitung auf.
X'00'	X'20'	X'2005'	Es trat ein interner Fehler im Zusammenhang mit der Lock-Anforderung auf.
X'00'	X'20'	X'2006'	Es trat ein interner Fehler im Zusammenhang mit XCS auf.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

LKLSB – Layout des Lock-Status-Blocks generieren

Allgemeines

Anwendungsgebiet: Distributed-Lock-Manager (DLM); siehe Seite 142
Makrotyp: S-Typ, MF-Format **3**: C-/D-/L-/M-/E-Form; siehe Seite 29

Makrobeschreibung

Der Makro **LKLSB** generiert das Layout des Lock-Status-Blocks (LSB). Der LSB ist ein Teil des Benutzer-Adressraums, der vom Benutzer für eine asynchrone Lock-Anforderung generiert werden muss. Dieser Benutzer-Adressraum muss für den DLM zugreifbar sein, bis die erzeugte Lock-Anforderung beendet wird.

Die Beendigung der asynchronen Lock-Anforderung veranlasst, dass ein entsprechender Returncode in den LSB geschrieben wird. Es wird die Benutzer-definierte Ereignis-Methode für diese Lock-Anforderung gestartet.

Um entscheiden zu können, ob der Returncode im LSB vom DLM geliefert wurde, muss der Benutzer den LSB initialisieren, indem er den Returncode mit dem Wert des Returncodes beim Aufruf `LKLSB MF=L` initialisiert.

Makroaufrufformat und Operandenbeschreibung

LKLSB
MF=C / D / L / M ,PARAM=<var: pointer> / (reg: pointer) ,PREFIX=N / p ,MACID=LDL / mac

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden.

Initialisierung des Lock-Status-Blocks

```
           LKLSB MF=L
1          MFTST MF=L,PREFIX=N,MACID=LDL,ALIGN=F,                        C
1                DMACID=LDL,SUPPORT=(D,C,M,L)
2          DS    OF
1          FHDR  MF=L,UNIT=247,FUNCT=6,VERS=1,RC=-1
2          DS    0A
2          DS    0XL8               GENERAL OPERAND LIST HEADER
2          DC    AL2(247)           FUNCTION UNIT NUMBER
2          DC    AL1(6)             FUNCTION NUMBER
2          DC    AL1(1)             FUNCTION INTERFACE VERSION NUMBER
2          DC    A(-1)               Returncode
1 *                                      FHDR
1          DC    A(0)                    LOID
1          DC    A(0)                    OWID
1          DC    CL16' '                 VAL
```

LPOV – Segment laden

Allgemeines

Anwendungsgebiet: Binden und Laden; siehe Seite 47
Makrotyp: O-Typ; siehe Seite 28

Makrobeschreibung

Mit dem Makroaufruf **LPOV** wird nur das eine angegebene Segment in den Speicher geladen, unabhängig davon, ob es schon im Speicher steht.
Der Makro **LPOV** wird verwendet, wenn das Laden von Segmenten auf nicht-automatische Weise veranlasst werden soll. Beim automatischen Laden von Segmenten können mit einem Makroaufruf mehrere Segmente geladen werden (siehe Handbuch „Dienstprogramme“ [27]).

Makroaufrufformat und Operandenbeschreibung

LPOV
,modulname [,adresse]

modulname
Symbolischer Name des zu ladenden Segmentes, der aus 6 alphanumerischen Zeichen bestehen kann.

adresse
Symbolische Adresse im aufrufenden oder einem anderen Modul, bei der nach dem Laden fortgesetzt wird. Wird dieser Operand nicht angegeben, so wird mit dem auf **LPOV** folgenden Befehl fortgesetzt. Das aufrufende Programm wartet, bis das Modul geladen ist.

Funktionsweise

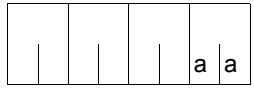
Die Ausführung des Makros **LPOV** bewirkt einen Aufruf des statischen Laders, der das mit „modulname“ angegebene Segment in den Speicher lädt. Nach Abschluss des Ladevorgangs wird die Steuerung an den mit „adresse“ bezeichneten Befehl übergeben. Diese Adresse kann im aufrufenden Modul liegen oder ein Externverweis in einem anderen Modul sein. Ist der Operand „adresse“ nicht angegeben, wird das aufrufende Modul mit dem auf **LPOV** folgenden Befehl fortgesetzt.
Wenn eine externe Referenz als Übergangspunkt verwendet wird, liegt es in der Verantwortung des Benutzers, dafür zu sorgen, dass das Modul mit der entsprechenden Einsprungadresse nach Abschluss des Ladevorgangs im Speicher ist. Da mit der Ausführung des Makros **LPOV** nur ein Segment geladen wird, muss jedes Segment explizit geladen werden. Wenn der Makro **LPOV** ausgeführt wird, wird das angegebene Segment geladen, unabhängig davon, ob es schon im Speicher steht oder nicht. Der Makro **LPOV** hat keine Verbindung zum Überlagerungssteuermodul, und es wird keine Liste der augenblicklich im Speicher vorhandenen Überlagerungssegmente geführt.

Hinweise zum Makroaufruf

– Nach dem Laden eines Segments durch den Makro **LPOV** können mit der Testhilfe AID mit `%ON %LPOV` noch Korrekturen vorgenommen werden (siehe Handbuch „AID“ [3]).
– Der Makro **LPOV**, mit dem das nicht-automatische Laden von Segmenten durchgeführt wird, verkehrt direkt mit dem Betriebssystem. Wenn **LPOV** zusammen mit den Makros **CALL** oder **SEGLD** (automatisches Laden von Segmenten) verwendet wird, kann die Statusliste der Programmüberlagerungsstruktur falsch sein; (siehe Handbuch „Dienstprogramme“ [27]). Ein solcher Fall kann während des Programmlaufes zu Fehlern führen.

Rückinformation und Fehleranzeigen

Wenn Fehler während des Ladens eines Überlagerungssegments auftreten, setzt der Ablaufteil das Programm mit dem **LPOV** folgenden Befehl fort. Tritt kein Fehler auf, so gilt die im Makroaufruf angegebene Fortsetzungsadresse.

R15: 

Über die Ausführung des Makros LPOV wird im rechtsbündigen Byte des Registers R15 ein Returncode übergeben.

X'aa'	**Erläuterung**
X'00'	Das Segment wurde geladen.
X'04'	Falscher Nur-Lese-Modifikationssatz.
X'08'	Falscher Satzcode im Text/Modifikationsblock oder es kommt vor dem 1. Textsatz ein Modifikationssatz.
X'0C'	Es sind zu viele Nur-Lese-Modifikationssätze vorhanden oder Segmentname kann in keinem der Indexsätze für Lademoduln (= Segment) gefunden werden.
X'10'	Es gibt keinen Indexsatz für Lademoduln mit dem Segmentnamen %ROOT.
X'14'	Es ist nicht genügend Speicherplatz vorhanden, um das Segment zu laden.
X'18'	Schlüsselinformation falsch oder Fehler beim Lesen von PAM-Lademoduldatei oder Fehler bei der Ausgabe von Meldungen.
X'1C'	Falscher Segmentname.
X'20'	Fehler beim Laden von AID.
X'24'	Fehler beim Lesen eines C-Elements einer PLAM-Bibliothek.

MINF – Speicherbelegung für Klasse-6-Speicher oder Memory Pool ausgeben

Allgemeines

Anwendungsgebiet:	Arbeiten mit virtuellem Speicher; siehe Seite 55 Memory-Pool-Technik; siehe Seite 55 Abfragen und Zugriff zu Listen und Tabellen; siehe Seite 158
Makrotyp:	S-Typ, MF-Format **1** (31-Bit-Schnittstelle): Standardform/L-/D-/E-Form; siehe Seite 29

Die Größe des Klasse-6-Speichers kann sich während des Programmlaufs verändern, abhängig von den Anforderungen an den Klasse-5- und Klasse-6-Speicher.

Makrobeschreibung

Der Makro **MINF** informiert den Anwender über Größe und Belegung seines

– Klasse-6-Speichers oder eines
– Memory Pools im Klasse-6-Speicher.

In Form einer Bittabelle wird ausgegeben, ob die Seite belegt ist oder nicht.
In der Bittabelle bedeuten:

Byte 0, Bit 2^7 repräsentiert die 1. (angeforderte) Speicherseite,
Byte 0, Bit 2^6 repräsentiert die 2. Speicherseite,
:
:
Byte 0, Bit 2^0 repräsentiert die 8. Speicherseite,
Byte 1, Bit 2^7 repräsentiert die 9. Speicherseite,
usw.

Für die Belegung gilt:

Bit 2^n = 0:	Speicherseite ist belegt.
Bit 2^n = 1:	Speicherseite ist nicht belegt.

Hinweis

Sollen Informationen über den Klasse-6-Speicher ausgegeben werden (Operand CL6), gilt jede Seite innerhalb eines Memory Pools als belegt. Dabei ist es unabhängig, ob die Seite bereits mittels **REQMP** angefordert wurde oder nicht.
Sollen Informationen über den Memory Pool selbst ausgegeben werden (Operand MP), gelten nur die Seiten als belegt, die bereits mittels **REQMP** angefordert wurden.

Makroaufrufformat und Operandenbeschreibung

```
MINF

{ CL6 }
{ MP  }
,INF={ SIZE              }
      { FREE,MAP=adr     }

,ADDR=adr
,MF=S / (E,...) / L / (D,pre) / D
```

CL6
Informationen über die Speicherseitenbelegung des Klasse-6-Speichers des Anwenders werden ausgegeben. Speicherseiten eines Memory Pools werden immer als belegt gekennzeichnet.

MP
Informationen über die Speicherseitenbelegung eines Memory Pools (Klasse-6-Speicher) werden ausgegeben. Es werden nur die Speicherseiten als belegt gekennzeichnet, die mit **REQMP** angefordert waren.

INF=
bezeichnet die Art der angeforderten Information.

SIZE
Die virtuelle Seitennummer (VPN) der ersten Speicherseite und die Größe (Anzahl Speicherseiten) des Klasse-6-Speichers/ Memory Pools werden ausgegeben. Die Ausgabe erfolgt in das beim Operanden ADDR angegebene Feld.

FREE,MAP=adr
Eine Bittabelle über die Belegung der Speicherseiten des beim Operanden ADDR angegebenen Speicherbereichs wird ausgegeben. Die Ausgabe erfolgt in das beim Operanden MAP angegebene Feld. Eine Zusatzinformation über die Länge der Bittabelle wird in das bei ADDR angegebene Feld eingetragen.

adr = symbolische Adresse (Name) eines Feldes
(Für je 8 Speicherseiten wird 1 Byte Feldlänge benötigt).

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der D-Form des Makroaufrufs kann ein Präfix (pre = 1..3 Buchstaben), wie im Aufruf-format dargestellt, angegeben werden.
Voreinstellung: pre = MNF

ADDR=adr
bezeichnet ein Feld, das sowohl zur Eingabe als auch zur Ausgabe dient (Operand INF). Feldlänge = 16 Byte (4 Worte). Das Feld ist auf Wortgrenze auszurichten.

adr
symbolische Adresse (Name) des Feldes

Der Inhalt ist wie folgt von der Angabe CL6/MP/SIZE abhängig:

CL6, INF=SIZE: (nur Ausgabefeld)

Byte	Output
0 - 3	VPN der ersten Speicherseite unterhalb 16 MB
4 - 7	Anzahl Speicherseiten unterhalb 16 MB
8 - 11	VPN der ersten Speicherseite oberhalb 16 MB oder X' 00000000'
12 - 15	Anzahl Speicherseiten oberhalb 16 MB od. X' 00000000'

X'00000000' wird ausgegeben, wenn oberhalb 16 MB kein Klasse-6-Speicher existiert.

MP,INF=SIZE:

Byte	Input	Output
0 - 3	VPN einer beliebigen Poolseite	VPN der ersten Speicherseite des Memory Pools
4 - 7	nicht benutzt	Größe (Anzahl der Seiten) des Memory Pools
8 - 11	nicht benutzt	nicht verändert [1)]
12 - 15	nicht benutzt	nicht verändert [1)]

[1)] nicht verändert bedeutet: Bei mehreren aufeinander folgenden **MINF**-Aufrufen steht der beim vorherigen Aufruf angegebene Inhalt in dem Feld.

CL6/MP,INF=FREE:

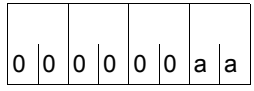
Byte	Input	Output
0 - 3	VPN der ersten Seite des gewünschten Bereichs.	nicht verändert
4 - 7	Anzahl Speicherseiten über die eine Bittabelle erstellt werden soll.	Anzahl Speicherseiten, die durch die Bittabelle real beschrieben werden.
8 - 11	nicht benutzt	nicht verändert
12 - 15	nicht benutzt	nicht verändert

Die angegebene VPN muss ein n-faches von 16 sein (n=0,1,...).

Rückinformation und Fehleranzeigen

Nach der Makrobearbeitung enthält Register R1 die Adresse der Operandenliste.

R15:



Über die Ausführung des Makros MINF wird ein Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben. Dieser Returncode wird auch im Standardheader der Operandenliste übergeben.

X'aa'	Erläuterung
X'00'	Funktion ausgeführt.
X'04'	Operandenfehler (ungültige Adresse bei MF=(E,...) angegeben).
X'08'	Ungültige VPN (die angegebene Seite befindet sich nicht im Klasse-6-Speicher / Memory Pool oder der Wert bezeichnet kein n-faches von 16 (n = 0, 1, ...)).
X'0C'	Adressenfehler (Datenbereich/ADDR-Feld/MAP).

Beispiel

In dem Beispiel soll die Größe, Lage und Seitenbelegung eines Memory Pools abgefragt werden, der oberhalb der 16MB-Grenze eingerichtet ist.

```
  MINF     START
           PRINT NOGEN
  MINF     AMODE 31
  MINF     RMODE ANY
           GPARMOD 31
1                *,MACRO: GPARMOD, VERSION: VER121
           BALR  3,0
           USING *,3
           ENAMP MPNAME=MEMP,SCOPE=GLOBAL,MPIDRET=PID,BSIZE=48 ————————  (1)
1                *,ENAMP: 144/951025
  REQMP    REQMP MPID=PID,BSIZE=5 ——————————————————————————————————  (2)
1                *,REQMP: 141 / 950210
  *
  DTH1     LR    4,1 ———————————————————————————————————————————————  (3)
           LA    4,4095(1)
           LA    4,1(4)
           LR    5,4
           SRL   5,12
           ST    5,MPINF1
  MINF1    MINF  MP,ADDR=MPINF,INF=SIZE ———————————————————————————————  (4)
1                *,MACRO: MINF, VERSION: VER174
  DTH2     MVC   MPINF2,=F'16'
           MVC   0(27,4),TEXT
  MINF2    MINF  MP,ADDR=MPINF,INF=FREE,MAP=BITTAB ————————————————————  (5)
```

```
1                *,MACRO: MINF, VERSION: VER174
  TERM     TERM
  *
  ****  Definitions  ****
  *
  MPINF    DS    OF
  MPINF1   DS    F
  MPINF2   DS    F
  MPINF3   DS    F
  MPINF4   DS    F
  PID      DS    F
  BITTAB   DS    CL4
  TEXT     DC    C'2ND PAGE OF MEMORY POOL'
           END
                 =F'16'
```

Ablaufprotokoll:

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,minf), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,minf)), -
//        test-support=*aid
%  ASS6011 ASSEMBLY TIME: 358 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 82 MSEC
//end
%  ASS6012 END OF ASSEMBH
/load-executable-program library=macexmp.lib,element-or-symbol=minf, -
/     test-options=*aid,prog-mode=*any
%  BLS0523 ELEMENT 'MINF', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'MINF', VERSION ' ' OF '<date> <time>' LOADED
/%in reqmp <%d c' ',c'- label reqmp:',%1,%15>
/%in dth1 <%d c' ',c'- label dth1:',%1,%15>
/%in minf1 <%d c' ',c'- label minf1:',mpinf1, mpinf1 %x>
/%in dth2 <%d c' ',c'- label dth2:',mpinf1, mpinf1 %x,mpinf2, mpinf2 %x>
/%in term <%d c' ',c'- label term:',%@(bittab) -> %xl4;%d %4 -> %xl30>
/%r
```

```
- LABEL REQMP: ------------------------------------------------------------ (6)
*** TID: 005000D8 *** TSN: 2QSE ***************************************
CURRENT PC: 01000040    CSECT: MINF  ***************************************
**
%1              = 01100000
%15             = 04000000

- LABEL DTH1: ------------------------------------------------------------- (7)
CURRENT PC: 0100006A    CSECT: MINF  ***************************************
%1              = 01100000
%15             = 00000000

- LABEL MINF1: ------------------------------------------------------------ (8)
SRC_REF:   100 SOURCE: MINF  PROC: MINF  ***********************************
MPINF1          =        4353
CURRENT PC: 01000080    CSECT: MINF  ***************************************
V'010000E0' = MPINF1   + #'00000000'
010000E0 (00000000) 00001101                              ....

- LABEL DTH2: ------------------------------------------------------------- (9)
SRC_REF:   117 SOURCE: MINF  PROC: MINF  ***********************************
MPINF1          =        4352
CURRENT PC: 0100009A    CSECT: MINF  ***************************************
V'010000E0' = MPINF1   + #'00000000'
010000E0 (00000000) 00001100                              ....
SRC_REF:   117 SOURCE: MINF  PROC: MINF  ***********************************
MPINF2          =         256
CURRENT PC: 0100009A    CSECT: MINF  ***************************************
V'010000E4' = MPINF2   + #'00000000'
010000E4 (00000000) 00000100                              ....

- LABEL TERM: ------------------------------------------------------------ (10)
CURRENT PC: 010000C2    CSECT: MINF  ***************************************
V'010000F4' = MINF     + #'000000F4'
010000F4 (000000F4) 07FF0000                              .~..
V'01101000' = ABSOLUT  + #'01101000'
01101000 (01101000) F2D5C440 D7C1C7C5 40D6C640 D4C5D4D6    2ND PAGE OF MEMO
01101010 (01101010) D9E840D7 D6D6D300 00000000 0000        RY POOL.......
```

(1) Einrichten eines Memory Pools oberhalb der 16MB-Grenze. Größe des Memory Pools = 48 Speicherseiten (wird auf 1 MB gerundet).

(2) Es werden 5 Speicherseiten ab der Anfangsadresse angefordert.

(3) REQMP liefert in Register R1 die Anfangsadresse des Memory Pools. Das Feld `MPINF1` soll als Eingabefeld für den Makro **MINF** eine (beliebige) virtuelle Seitennummer (VPN) innerhalb des Memory Pools enthalten. Der Inhalt von Register R4 ist die Adresse des ersten Bytes auf der zweiten Memory Pool-Seite, Register R5 enthält die dazugehörige VPN. Diese VPN wird im Feld `MPINF1` abgespeichert.

(4) Erster **MINF**-Aufruf:
Größe und Lage des Memory Pools sollen ausgegeben werden.
Die Ausgabe erfolgt in das Feld `MPINF`: Die VPN der ersten Seite des Memory Pools steht im Feld `MPINF1`, die Anzahl der Seiten des Memory Pools im Feld `MPINF2`. Die Felder `MPINF3` und `MPINF4` werden nicht verändert.

(5) Zweiter **MINF**-Aufruf:
Eine Bittabelle über die ersten 16 Seiten des Memory Pools soll ausgegeben werden (der vorhergegangene MVC überträgt die Zahl 16 in das zweite Eingabefeld `MPINF2`).
Im Inputfeld für **MINF** stehen jetzt die vom ersten **MINF**-Aufruf gelieferte VPN der ersten Seite (`MPINF1`) und die Anzahl der Poolseiten, deren Seitenbelegung ausgegeben werden soll (`MPINF2`). Die Ausgabe erfolgt in das Feld `BITTAB`.

(6) Nach Ausführung des **ENAMP** werden die Anfangsadresse und der Returncode abgefragt. Der Memory Pool beginnt auf Adresse X'01100000'.
RC = X'04000000' bedeutet: ein neuer Memory Pool wurde eingerichtet.

(7) Nach Ausführung des **REQMP** werden die Adresse des reservierten Bereichs und der Returncode abgefragt. Der reservierte Bereich beginnt ab Adresse X'01100000'; RC = X'00000000'.

(8) Im **MINF**-Inputfeld steht X'00001101' = 4353 als VPN einer Memory Pool-Seite.

(9) Nach dem ersten **MINF**:
Die VPN der ersten Seite und die Größe des Memory Pools werden ausgegeben: VPN = 4352 (≙ X'1100'); Größe = 256 Seiten (≙ X'100'). Die VPN der ersten Seite bleibt als Input für den zweiten **MINF**-Aufruf.

(10) Nach dem zweiten **MINF**:

– Die Bittabelle steht auf Adresse X'000000EC'; Länge = 2 Byte (≙ 16 Seiten). Inhalt: X'07FF'; Bitmuster: 0000/0111/1111/1111 . Die mit **REQMP** reservierten Speicherseiten sind als belegt und die restlichen der 16 Seiten als frei gekennzeichnet.

– In die zweite Seite des Memory Pool wurde der Text `2ND PAGE OF ...` eingetragen.

MSG7X – Meldung ausgeben

Allgemeines

Anwendungsgebiet: Meldungswesen; siehe Seite 165
Makrotyp: S-Typ, MF-Format **3**: C-/D-/L-/M-/E-Form; siehe Seite 29

Die Angabe von Registerinhalten ist nur beim M-Form-Makroaufruf möglich.

Makrobeschreibung

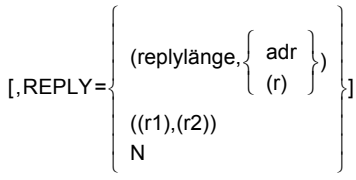
Der **MSG7X**-Makro gibt eine Systemmeldung auf SYSOUT, SYSLST, den Bedienplatz, in einen Bereich des Benutzerprogramms oder in eine S-Variable aus. Der angegebene Meldungsschlüssel muss 7-stellig sein.
MSG7X verwendet das neue Layout der Operandenliste (mit Standardheader; siehe Seite 43).

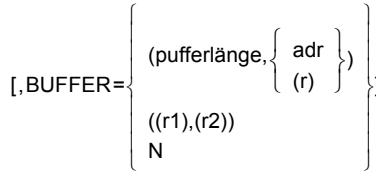
Jede Systemmeldung trägt einen 7-stelligen Meldungsschlüssel. Die ersten 3 Zeichen des Schlüssels bezeichnen die Meldungsklasse; die restlichen 4 Zeichen dienen der laufenden Nummerierung innerhalb einer Klasse. Systemmeldungen können variable Teile „(&nn)“ enthalten, die durch „Inserts“ ersetzt werden können.

Makroaufrufformat und Operandenbeschreibung

```
MSG7X

ID=msgid / (r1) / (class,(r)) / (,(r)) / (class,(adr)) / (,adr)

                  ⎧ (insertlänge,⎧ adr    ⎫)                    ⎫
                  ⎪              ⎪ (r)    ⎪                     ⎪
                  ⎪              ⎩ 'insert' ⎭                    ⎪
[,INSERT=         ⎨ ((insertlänge,⎧ adr    ⎫),...)               ⎬]
                  ⎪               ⎪ (r)    ⎪                     ⎪
                  ⎪               ⎩ 'insert' ⎭                    ⎪
                  ⎪ ((r1),(r2))                                  ⎪
                  ⎪ (((r1),(r2)),...)                            ⎪
                  ⎪ anzahl                                       ⎪
                  ⎩ NONE                                         ⎭

[,LAN='sprache']

,DEST=SYSOUT / SYSLST / CONSOLE / (ausgabeort,...) / NONE

[,UCDEST='destcod' / destadr / (r) / N]
```

MSG7X (Fortsetzung)





,MAP=NO / YES

,DMS=APPL / NOTAPPL / NA

,BUFFUSE=INTERNAL / EXTERNAL

,DEFTEXT=NONE / (textlänge,adr) / (textlänge,(r)) / ((r1),(r2))

,TIMER=UNLIMITED / (r) / wert

,TIMESTAMP=NO / YES

,DATESTAMP=NO / YES

,MF=D / C / L / E / M

[,PARAM=adr / (r)]

,PREFIX=X / p

,MACID=MSG / macid

ID=
dient der Angabe des Meldungsschlüssels der auszugebenden Systemmeldung.

msgid
7-stelliger Meldungsschlüssel

(class,adr)
Ändert den Meldungsschlüssel im Datenbereich.

(class,(r))
Die Angabe dieses Operandenwertes ist nur mit MF=M erlaubt.

`class`	Meldungsklasse
`adr`	symbolische Adresse (Name) eines Feldes, das die neue Meldungsnummer enthält
`(r)`	Register, das die neue Meldungsnummer enthält

(,adr)
Ändert die Meldungsnummer im Datenbereich.

(,(r))
Die Angabe dieses Operandenwertes ist nur mit MF=M erlaubt.

`adr` symbolische Adresse (Name) eines Feldes, das die neue Meldungsnummer enthält

`(r)` Register, das die neue Meldungsnummer enthält

INSERT=
gibt max. 30 Längen und Adressen von Einfügungen an. Für jede Eintragung wird im Datenbereich der Meldungsverarbeitung ein entsprechender Adressverweis aufgebaut. Werden mehr Einfügungen angegeben, als in der Meldung vorgesehen sind, so werden die weiteren Einfügungen ignoriert. Eine Einfügung, die nur aus Leerzeichen besteht, wird auf 1 Leerzeichen gekürzt.
Leerzeichen am Ende einer Einfügung werden unterdrückt. Um das Abschneiden der Leerzeichen zu verhindern, muss am Ende der Einfügung das Zeichen X'01' angegeben werden. Findet die Meldungsverarbeitung das Zeichen X'01', werden die bis dahin eingetragenen Leerzeichen nicht unterdrückt.
Wenn insertlänge=0 und adr=0 angegeben werden, übernimmt das System für die Einfügung den Standardwert aus der Meldungsdatei. Ist dort kein Standardwert angegeben, wird die Einfügung ausgelassen.
Enthält ein Meldungstext mehr Einfügungen als im Makroaufruf angegeben, wird für jede übrige Einfügung der Standardwert übernommen. Ist kein Standardwert vorhanden, so wird die Ersatzeinfügung (&nn) eingesetzt.
Bei Angabe von MF=M wird für alle bei INSERT angegebenen Einfügungen ein Adressverweis aufgebaut. Z.B. ändert der Aufruf MF=M,INSERT=(((R1),(R2)),((R3),(R4))) die Einfügungen 0 und 1 im Datenbereich, aber nicht 2 und 3.
Alle vorher definierten Einfügungen im Datenbereich werden ignoriert. Die Anzahl der bei MF=M definierten Einfügungen muss kleiner oder gleich einer max. Anzahl sein, die durch MF=C/D vorgegeben wird.

(insertlänge,...)

`insertlänge` Länge des Einfügung

`adr1` symbolische Adresse (Name) des Bereichs mit der Einfügung

`(r)` Register mit der Adresse der Einfügung
(Angabe nur möglich bei MF=M)

`'insert'` direkte Angabe der Einfügung
(Länge der Zeichenkette: max. 4 Zeichen)

Bei insertlänge = 0 muss die Einfügung mit einem Satzlängenfeld (4 Byte) beginnen und ist somit immer mindestens 4 Byte lang:

`Byte 0-1` Länge der Einfügung

`Byte 2-3` reserviert.

Einfügungen können übersprungen werden, indem man die ausgelassenen Stellen mit einem Komma belegt, z.B. INSERT=(,(insertlänge2,adr2),,(insertlänge4,adr4)). Eine solche ausgelassene Einfügung wird durch ihren Standardwert oder durch eine leere Zeichenkette ersetzt.
Die Summe aller Einfügungen muss ≤ 4079 Byte sein. Eine Liste von Einfügungen muss in ein zusätzliches Klammerpaar eingeschlossen werden.
Bei Ausgabe auf Konsole muss die Summe aller Einfügungen ≤ 218 Byte sein.

((r1),(r2))
`r1` Register mit der Länge der Einfügung
`r2` Register mit der Adresse der Einfügung.
Diese Angabe ist nur mit MF=M möglich.
Eine Liste von Einfügungen muss in ein zusätzliches Klammerpaar eingeschlossen werden.

anzahl
Anzahl der INSERT-Verweise, für die Platz im Datenbereich der Meldungsverarbeitung reserviert werden soll.
Mit MF=C/D ist anzahl standardmäßig 15.
Bei MF=M kann die Anzahl der Einfügungen im Datenbereich geändert werden.

NONE
Alle zuvor definierten Einfügungen werden ignoriert.
Die Angabe dieses Operandenwertes ist nur mit MF=M erlaubt.

LAN='sprache'
bezeichnet die Sprache, in der die Meldungstexte ausgegeben werden sollen. Der Operand wird nicht ausgewertet, wenn DEST=CONSOLE angegeben ist.

'sprache'
sprache = 1 Buchstabe zur Kennzeichnung der Sprache; D = Deutsch, E = Englisch. Weitere Möglichkeiten sind bei der Systemverwaltung zu erfragen. Der Standardwert wird durch den Systemparameter MSGLPRI festgelegt und wird auch dann eingesetzt, wenn eine ungültige Angabe erfolgte.

REPLY=
beschreibt einen Antwortbereich. Der Bereich muss auf Halbwortgrenze ausgerichtet sein und mit einem Satzlängenfeld (4 Byte; Byte 1-2: Länge, Byte 3-4: reserviert) beginnen. Vor der Makroausführung muss in Byte 1-2 die Länge des Antwortbereichs (≤ 4095 Byte) stehen. Bei Ausführung des Makros wird dann die aktuelle Länge der Antwort in Byte 1-2 eingetragen.

Bei Eingabe über den Operanden REPLY werden Kleinbuchstaben in Großbuchstaben umgesetzt. REPLY nur in Verbindung mit DEST=SYSOUT/CONSOLE angeben.

Wird als Antwort ein „?“ eingegeben, das in MIP als Schlüsselwort reserviert ist, zeigt MIP Bedeutung und Maßnahme der betreffenden Meldung an, bevor die Meldung erneut zur Beantwortung ausgegeben wird.

(replylänge,...)

`replylänge`	Länge des Antwortbereichs > 4 Byte
`adr`	symbolische Adresse (Name) des Bereichs
`(r)`	Register mit der Adresse des Bereichs (Angabe nur möglich bei MF=M)

((r1),(r2))

`r1` Register mit der Länge des Antwortbereichs
`r2` Register mit der Adresse des Bereichs
Diese Angabe ist nur mit MF=M möglich.

N

Ein Bereich für die Antwort wird in der CSECT/DSECT nicht generiert.
Die Angabe wird zum Aufbau der CSECT/DSECT benötigt und kann nur in Verbindung mit MF=C/D angegeben werden.

DEST=

der Operand beschreibt Ausgabeorte für die konvertierte Systemmeldung. Zusammen mit REPLY darf nur DEST=SYSOUT/CONSOLE angegeben werden.

SYSOUT

Die Ausgabe erfolgt nach SYSOUT. Diese Angabe ist Voreinstellung, wenn nicht mit BUFFER ein weiterer Ausgabeort angegeben wird. Wird mit BUFFER ein Ausgabebereich bestimmt und wird eine Ausgabe nach SYSOUT gewünscht, muss DEST=SYSOUT explizit angegeben werden.
Wurde eine S-Variable deklariert und der Variablenstrom in sie umgelenkt und handelt es sich um eine garantierte Meldung, erfolgt die Ausgabe auch in die S-Variable.

SYSLST

Ausgabe nach SYSLST

CONSOLE

Ausgabe an den Bedienplatz

(ausgabeort,...)

Kombination der genannten Ausgabeorte; Angabe in runden Klammern, getrennt durch Kommas.

NONE

Löscht die Ausgabeorte SYSOUT, SYSLST und CONSOLE aus dem Datenbereich. Die Angabe dieses Operandenwertes ist nur mit MF=M erlaubt. Zu beachten ist, dass immer mindestens ein Ausgabeort im Datenbereich definiert sein muss, z.B. ein speziell dafür vorgesehener Bereich (siehe Operand BUFFER).

BUFFER=
beschreibt einen Bereich, in den die konvertierte Systemmeldung übertragen werden soll. Wird zusätzlich eine Ausgabe nach SYSOUT gewünscht, muss DEST=SYSOUT explizit angegeben werden.
BUFFER muss auf Halbwortgrenze ausgerichtet sein. In die ersten 2 Byte wird ein Satzlängenfeld (WROUT-Format) eingetragen:

`Byte 0-1`	Länge des Bereichs
`Byte 2-3`	reserviert
`Byte 4`	Output-Steuerzeichen
`Byte 5-n`	Meldungstext

(pufferlänge,...)

`pufferlänge`	Länge der Bereichs > 16 Byte
`adr`	symbolische Adresse (Name) des Bereichs
`(r)`	Register mit der Adresse des Bereichs (Angabe nur möglich bei MF=M)

Die Summe der Bereichslängen muss ≤ 4095 Byte sein.

((r1),(r2))

`r1`	Register mit der Länge des Meldungsbereichs
`r2`	Register mit der Adresse des Bereichs

Diese Angabe ist nur mit MF=M möglich.

N
Es wird kein Bereich für die konvertierte Systemmeldung in der CSECT/DSECT generiert.
Die Angabe wird zum Aufbau der CSECT/DSECT benötigt und kann nur in Verbindung mit MF=C/D angegeben werden.

MAP=
beschreibt, ob BUFFER eine andere Struktur (Mapping-Format) erhält.

NO
BUFFER wird im **WROUT**-Format eingerichtet.

YES
BUFFER wird im Mapping-Format eingerichtet (Aufbau siehe unten).

UCDEST=
UCON-Ausgabeort (UCON=**U**niversal **Con**sole); (siehe „Systembetreuung“ [10]). Das Ziel (Ausgabeort) einer Meldung kann wie folgt angegeben werden:
– mnemotechnischer Gerätename für einen bestimmten Bedienplatz
– Berechtigungsschlüssel (routing code) für Bedienplätze und berechtigte Benutzerprogramme, denen ein bestimmtes Aufgabengebiet zugeordnet ist.
– Berechtigungsname für einen berechtigten Benutzerprogramm.

UCDEST wird nur ausgeführt, wenn DEST=CONSOLE spezifiziert wurde.
UCDEST besitzt Vorrang vor dem Routing Code der Meldung in der Meldungsdatei.

'destcod'
folgende Angaben sind für 'destcode' möglich:
– '(mn)'
mn: 2 Zeichen langer mnemotechnischer Gerätename.
– '< x'
x: Berechtigungsschlüssel; das Zeichen < muss angegeben werden.
– 'name'
name: 4 Zeichen langer Name des Benutzerprogramms.

destadr
symbolische Adresse (Name) eines Bereiches (4 Byte) mit der Angabe für destcod.
Angaben linksbündig; Ausrichtung auf Wortgrenze.

(r)
r: Register, das die Adresse des Bereichs (4 Byte) enthält.
Angabe nur möglich bei MF=M.

N
Ein Feld für destcod wird in der CSECT/DSECT nicht generiert.
Die Angabe wird zum Aufbau der CSECT/DSECT benötigt und kann nur in Verbindung mit MF=C/D angegeben werden.

DMS=
spezifiziert den Meldungs-Suchmechanismus.

APPL
Die Meldung wird mit dem DMS gesucht (Meldungsdateien und DLAM-Bereich).

NOTAPPL / NA
die Meldung wird nur auf Systemebene im DLAM-Bereich (ohne DMS) gesucht. Rückmeldung, wenn die geforderte Meldung nicht gefunden wird:
`msgid,(DVS NICHT VERFUEGBAR),<befehlszähler>,<modul>,<inserts>`

BUFFUSE=
bestimmt, ob die Meldung über einen Variablenstrom (SYSMSG) in einer S-Variablen abgespeichert werden soll und ob sie mit dem Kommando HELP-MSG-INFORMATION MSG-ID=*LAST (also ohne explizite Angabe der Meldungsnummer) ausgegeben werden kann.
Die Angabe von BUFFUSE ist nur dann relevant, wenn die Ausgabe nicht nach SYSOUT geht (DEST ≠ SYSOUT) und mit dem Operanden BUFFER ein Übertragungsbereich angegeben wurde.
Wird die Meldung (auch) nach SYSOUT ausgegeben, wird sie - wenn sie eine garantierte Meldung ist - immer in die S-Variable ausgegeben (sofern eine solche deklariert und der Variablenstrom in sie umgelenkt wurde).
Der Variablenstrom nimmt nur „garantierte Meldungen“ auf. Siehe dazu die Handbücher „SDF-P“ [21] und „Dienstprogramme (MSGMAKER)“ [27].

INTERNAL
Die Ausgabe wird nicht in einen Variablenstrom gelenkt. Die Meldung kann nicht über das Kommando HELP-MSG-INFORMATION MSG-ID=*LAST (also ohne explizite Angabe der Meldungsnummer) ausgegeben werden.

EXTERNAL
Wurde eine S-Variable deklariert und der Variablenstrom in sie umgelenkt (mit dem Kommando ASSIGN-STREAM oder EXECUTE-COMMAND bzw. mit dem Makro CMD oder OPSGEN), erfolgt die Meldungsausgabe in diese S-Variable.
Es werden der Meldungsschlüssel, der Meldungstext und die Standardwerte der Einfügungen ausgegeben.
Voraussetzung ist, dass die Meldung eine garantierte Meldung ist, da sie sonst nicht in den Variablenstrom aufgenommen wird.
Ist die Meldung die zuletzt von der Task ausgegebene Meldung, kann sie mit dem Kommando HELP-MSG-INFORMATION MSG-ID=*LAST ausgegeben werden.

DEFTEXT=
verweist auf einen Standard-Meldungstext.
Dieser Standardmeldungstext wird ausgegeben, wenn die angeforderte Meldung nicht definiert ist bzw. wenn sie oder die MIP-Task z.Zt. nicht zur Verfügung stehen. Es wird ein entsprechender Returncode zurückgegeben.
Der Standard-Meldungstext muss ≤ 4079 Byte sein.

NONE
Es wird kein Standard-Meldungstext vorgegeben.

(textlänge,...)

`textlänge`	Länge des Bereichs
`adr1`	symbolische Adresse (Name) des Bereichs
`(r)`	Register mit der Adresse des Bereichs (Angabe nur möglich bei MF=M)

((r1),(r2))
`r1` Register mit der Länge des Standard-Meldungstextes
`r2` Register mit der Adresse des Bereichs
Diese Angabe ist nur mit MF=M möglich.

TIMER=
gibt die max. Wartezeit an, in der nach dem Senden einer Meldung nach SYSOUT die Antwort (Operand REPLY) erfolgen soll.
Wird in der angegebenen Zeit keine Antwort empfangen, wird der entsprechende Returncode zurückgegeben.

UNLIMITED
Die Wartezeit ist unbegrenzt.

(r)
r = Register, in dem die Wartezeit steht.
Angabe nur möglich bei MF=M.

wert
Wartezeit, die 10 bis 3600 Sekunden betragen kann

TIMESTAMP=
erlaubt die zusätzliche Ausgabe der lokalen Systemzeit.

NO
Die lokale Systemzeit wird nicht ausgegeben.

YES
Zusätzlich zur Meldungsausgabe wird die lokale Systemzeit im ISO4-Format ausgegeben. Die Ausgabe hat dann folgendes Format:
`%␣␣hh:mm:ss msgid text`

Es bedeuten:

`hh`	Stunde
`mm`	Minute
`ss`	Sekunde
`msgid`	Meldungsschlüssel
`text`	Meldungstext

Hinweis
Zur Darstellung von Zeitstempeln siehe auch Makro **CTIME** (Seite 365).

Kann MIP wegen eines internen Fehlers keine Zeit lesen, wird
`%␣␣HH:MM:SS msgid text` ausgegeben.

DATESTAMP=
erlaubt die zusätzliche Ausgabe des Datums.

NO
Es wird kein Datum ausgegeben.

YES
Zusätzlich zur Meldungsausgabe wird das Datum im ISO4-Format ausgegeben. Die Ausgabe hat dann folgendes Format:
`%␣␣yyyy-mm-dd msgid text`

Es bedeuten:

`yyyy`	Jahr
`mm`	Monat des Jahrs
`dd`	Tag des Monats
`msgid`	Meldungsschlüssel
`text`	Meldungstext

Hinweis

Zur Darstellung von Zeitstempeln siehe auch Makro **CTIME** (Seite 365).

Kann MIP wegen eines internen Fehlers kein Datum lesen, wird `%␣␣YYYY-MM-DD msgid text` ausgegeben.

Werden Datum- und Zeitausgabe angefordert, steht das Datum vor der lokalen Systemzeit.

MF=

Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

Hinweise

– Die Gesamtlänge einer Meldung muss einschließlich des Meldungsschlüssels ≤ 4079 Byte sein. Überzählige Zeichen werden automatisch abgeschnitten.

– Bei Ausgabe auf Konsole beträgt die maximale Länge einer Meldung einschließlich des Meldungsschlüssels nur 230 Byte.

– Leerzeichen am Ende einer Meldung werden automatisch unterdrückt.

– Die Meldungsausgabe nach SYSOUT, SYSLST oder in einen Bereich BUFFER beginnt immer mit dem Zeichen `%` (z.B. `%␣␣NMH1121 <text>`). Wird die Meldung dagegen auf die Konsole ausgegeben, beginnt sie mit einem dem `%` vorangestellten Leerzeichen (z.B. `␣%␣␣NMH1121 <text>`).

Mapping-Format

Das Mapping-Format beschreibt die Struktur des Bereichs BUFFER bei MAP=YES. Dabei werden der unter BUFFER beschriebenen Struktur verschiedene Einträge (Mapping-List) vorangestellt.

Struktur des Mapping-Formats:

`Byte 0 - 1`	Länge der Mapping-List
`Byte 2 - 3`	C'MP'
`Byte 4 - n`	Einträge
`Byte n+1 bis p`	weiterer Aufbau wie unter BUFFER beschrieben (WROUT-Format).

Die Einträge enthalten Informationen über:
- die Einfügungen (Inserts)
- den Meldungsschlüssel
- den Berechtigungsschlüssel (Routingcode)
- das Meldungsgewicht
- ggf. Füllbytes zum Ausrichten des **WROUT**-Puffers.

Sie werden in folgenden Formaten abgelegt:

Eintrag über	Aufbau des Eintrags
Insert	insert-länge | Distanz zur Einfügung im WROUT-Format 0H 2 4 ...(Für jedes Insert wird ein eigener Eintrag angelegt.)
Meldungs-schlüssel	X' 81' | 7stell. Meldungsschlüssel 0 1 8
Meldungs-gewicht	X' 20' | Meld.-gewicht 0 1 1
Berechti-gungs-schlüssel	X' 50' | Routingcode (linksbündig) 0 1 5
Füll-bytes	X' 00' | - - - | X' 00' | WROUT-Format 0 1 m H

Hinweise zum Makroaufruf

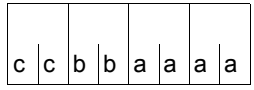
– Bei der Ausgabe im Mapping-Format steht ein Eintrag mit Länge=0 und Adresse=0 für einen Insert, der gar nicht existiert. Solche Einträge werden nur ausgegeben, wenn die Inserts einer Meldung nicht fortlaufend nummeriert sind.

– Inserts, für die kein aktueller Wert vorliegt, werden dagegen durch einen Eintrag mit der Länge=0 und der Distanz zur definierten Insert-Position beschrieben. Ab dieser Distanz steht der Default-Text des Inserts bzw. die Standard-Einfügung (&xx), falls kein Default-Text definiert ist.

– Soll aus der Mapping-Ausgabe wieder eine MSG7X-Parameterleiste für eine Eingabe aufgebaut werden, muss die Distanz nichtangegebener Inserts auf 0 zurückgesetzt werden, weil in Verbindung mit Länge=0 alle anderen Distanz-Werte als Definition eines Inserts angesehen werden, der mit einem 4-Byte-Satzlängenfeld beginnt.

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros MSG7X wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode); SUBCODE2 wird nur bei einem Ein-/Ausgabefehler versorgt und enthält dann den Main-Returncode des WROUT- bzw. WRTRD-Makros):

X'cc'	X'bb'	X'aaaa'	Erläuterung
	X'00'	X'0000'	MSG7X-Makro erfolgreich ausgeführt.
X'cc'	X'04'	X'0001'	Abnormale Beendigung der Ausgabe (SYSOUT, SYSLST oder CONSOLE). In diesem Fall enthält SUBCODE2 (X' cc') den MAIN-CODE des WROUT- bzw. WRTRD-Makros.
	X'08'	X'0001'	Operandenfehler: Unzulässige Angabe für Meldungsnummer, Bereichs-adresse, mnemotechnischen Gerätenamen oder Namen des berechtig-ten Benutzerprogramms oder ungültige Bereichslänge.
	X'0C'	X'0001'	Unzulässige Anforderung einer Antwort, z.B. im Batch-Betrieb oder bei Angabe unvereinbarer Ausgabeorte (SYSLST oder mehrere Ausgabe-orte) im Operanden DEST.
	X'10'	X'0001'	Es steht kein Speicherplatz zur Verfügung.
	X'14'	X'0001'	BREAK während der Ausführung des WROUT-Makros.
	X'18'	X'0001'	Meldungstext wurde bei der Übertragung in den Ausgabebereich abgeschnitten.
	X'20'	X'0001'	Meldungsausgabe wurde abgebrochen.
X'00'	X'41'	X'FFFF'	MIP-Subsystem ist nicht geladen.
X'02'	X'00'		Meldung nicht definiert.
X'03'	X'00'		MIP-Task nicht verfügbar.
X'05'	X'00'		DMS-Subsystem ist nicht geladen.
X'06'	X'00'		Meldung nicht verfügbar. Fehler in der Datei.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Das aufrufende Programm wird beendet, wenn folgende Fehler auftreten:
– Der Datenbereich ist dem Aufrufer nicht zugewiesen.
– Der Datenbereich ist nicht auf Wortgrenze ausgerichtet.
– Der Datenbereich ist gegen Schreibzugriff geschützt.

Beispiel 1

Dieses Beispiel zeigt verschiedene Möglichkeiten, Meldungen zu suchen und auszugeben.

```
MSG7X1   START
MSG7X1   AMODE ANY
MSG7X1   RMODE ANY
         PRINT NOGEN
         BALR  3,0
         USING *,3
*
         SYSFL 'SYSLST=LST.MSG7X'
MFE1     MSG7X MF=E,PARAM=MFL1 ------------------------------------------  (1)
MFE2     MSG7X MF=E,PARAM=MFL2 ------------------------------------------  (2)
         WROUT TEXT,ERROR,PARMOD=31
MFE3     MSG7X MF=E,PARAM=MFL3 ------------------------------------------  (3)
         WROUT TEXT,ERROR,PARMOD=31 -------------------------------------  (4)
         SYSFL 'SYSLST=(PRIMARY)'
*
ERROR    TERM
*
MFL1     MSG7X MF=L,ID=DMSOE27,DMS=NOTAPPL,LAN='D'
MFL2     MSG7X MF=L,ID=DMSOE27,DEST=(SYSLST,SYSOUT),LAN='D'
MFL3     MSG7X MF=L,ID=SCP0976,DEST=SYSLST,BUFFER=(75,TEXT)
*
TEXT     DC    Y(TEXTEND-TEXT)
         DS    3X
         DS    0CL75
         DC    C'message output via WROUT: '
TEXTEND  EQU   *
         END
```

Ablaufprotokoll

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,msg7x1), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,msg7x1))
%  ASS6011 ASSEMBLY TIME: 508 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 82 MSEC
//end
%  ASS6012 END OF ASSEMBH
```

```
/start-executable-program library=macexmp.lib,element-or-symbol=msg7x1
%  BLS0523 ELEMENT 'MSG7X1', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'MSG7X1', VERSION ' ' OF '<date> <time>' LOADED
%  DMS0E27   ( DVS NICHT VERFUEGBAR )  --PC=00000034 IN P1-
MODULE=********-- ------------------------------------------------------  (5)
%  DMS0E27 FEHLER BEIM SCHLIESSEN EINER DATEI. EIN- AUSGABE MIT
   HARDWARE-FEHLER BEENDET ------------------------------------------------  (6)
message output via WROUT:
%  SCP0976 LOGICAL VALIDATION PROBLEM DURING COMMAND PROCESSING --------  (7)
```

Ausgabe auf SYSLST:

```
%  DMS0E27 FEHLER BEIM SCHLIESSEN EINER DATEI. EIN- AUSGABE MIT
   HARDWARE-FEHLER BEENDET
%  SCP0976 LOGICAL VALIDATION PROBLEM DURING COMMAND PROCESSING
```

(1) Die Meldung `DMS0E27` soll nach SYSOUT ausgegeben werden. Dabei soll die Meldung nur im DLAM- Bereich (ohne DMS) gesucht werden. Im DLAM-Bereich wird sie nicht gefunden, deshalb findet die Ausgabe der Meldung nicht statt; es wird eine entsprechende Rückmeldung ausgegeben.

(2) Die Meldung `DMS0E27` soll nach SYSOUT und nach SYSLST ausgegeben werden. Die unter (1) angegebene Einschränkung entfällt.

(3) Die Meldung `SCP0976` soll nach SYSLST ausgegeben und in den spezifizierten Bereich `TEXT` geschrieben werden.

(4) Der Makro **WROUT** schreibt den Inhalt des Bereichs `TEXT` nach SYSOUT.

(5) Ausgabe der Meldung von (1).

(6) Ausgabe der Meldung von (2).

(7) Ausgabe von WROUT.

Beispiel 2

Das folgende Beispiel zeigt die Umlenkung der Meldungsausgabe in eine S-Variable. Die S-Variable muss vor Aufruf des Programms deklariert sein und der Variablenstrom SYSMSG muss ihr zugewiesen werden. Es werden nur garantierte Meldungen in den Variablenstrom SYSMSG aufgenommen.

```
MSG7X2   START
MSG7X2   AMODE ANY
MSG7X2   RMODE ANY
         PRINT NOGEN
         BALR  3,0
         USING *,3
*
MFE1     MSG7X MF=E,PARAM=MFL1
MFE2     MSG7X MF=E,PARAM=MFL2
MFE3     MSG7X MF=E,PARAM=MFL3
MFE4     MSG7X MF=E,PARAM=MFL4
*
ERROR    TERM
*
****  Definitions  ****
*
MFL1     MSG7X MF=L,ID=CMD0500,BUFFER=(250,BUF),MAP=YES,                *
               DEST=(SYSOUT,SYSLST) ------------------------------------  (1)
MFL2     MSG7X MF=L,ID=DMS0DF8,BUFFER=(250,BUF),MAP=YES,DEST=SYSOUT -   (2)
MFL3     MSG7X MF=L,ID=SCP0976,BUFFER=(250,BUF),DEST=SYSLST ----------   (3)
MFL4     MSG7X MF=L,ID=DMS0574,BUFFER=(250,BUF),DEST=SYSLST,            *
               BUFFUSE=EXTERNAL ----------------------------------------  (4)
*
BUF      DS    0CL250
         DC    Y(ENDBUF-BUF)
         DS    3X
         DS    CL245
ENDBUF   EQU   *
         END
```

Ablaufprotokoll:

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,msg7x2), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,msg7x2)), -
//        test-support=*aid
%  ASS6011 ASSEMBLY TIME: 405 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 80 MSEC
//end
%  ASS6012 END OF ASSEMBH
/declare-var msg(type=structure),mult-elem=*list ————————————————————— (5)
/assign-stream sysmsg,to=*var(msg)
/start-executable-program library=macexmp.lib,element-or-symbol=msg7x2
%  BLS0523 ELEMENT 'MSG7X2', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'MSG7X2', VERSION ' ' OF '<date> <time>' LOADED
%  CMD0500 INVALID DESCRIPTION OF COMMAND OR STATEMENT IN CURRENT SYNTAX FILE
%  DMSODF8 EXPECTED VSN '(&01)' FOR FILE '(&02)', VSEQ '(&03)' NOT MOUNTED ON
 DEVICE '(&00)'. VSN '(&04)' FOUND INSTEAD. REPLY (0=EXIT; 1=RETRY; 2=DISPLAY
 LABEL;  =ACCEPT)
/assign-stream sysmsg,to=*dummy ——————————————————————————————————— (6)
/show-var msg
MSG(*LIST).MSG-TEXT = %  BLS0523 ELEMENT 'MSG7X2', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
MSG(*LIST).MSG-ID = BLS0523
MSG(*LIST).I0 = MSG7X2
MSG(*LIST).I1 = @
MSG(*LIST).I2 = :2OSG:$QM212.MACEXMP.LIB
MSG(*LIST).MSG-TEXT = %  BLS0524 LLM 'MSG7X2', VERSION ' ' OF
   '<date> <time>' LOADED
MSG(*LIST).MSG-ID = BLS0524
MSG(*LIST).I0 = MSG7X2
MSG(*LIST).I1 =
MSG(*LIST).I2 = <date> <time>
MSG(*LIST).MSG-TEXT = %  BLS0551 COPYRIGHT (C) . . .
MSG(*LIST).MSG-ID = BLS0551
MSG(*LIST).I0 = FUJITSU TECHNOLOGY SOLUTIONS
MSG(*LIST).I1 = 2012
MSG(*LIST).MSG-TEXT = %  CMD0500 INVALID DESCRIPTION OF COMMAND OR STATEMENT
   IN CURRENT SYNTAX FILE
MSG(*LIST).MSG-ID = CMD0500
MSG(*LIST).MSG-TEXT = %  DMS0574 DMS ERROR CODE '(&00)' OCCURRED WHEN
```

```
   DELETING SYSTEM FILE. COMMAND NOT PROCESSED
MSG(*LIST).MSG-ID = DMS0574
MSG(*LIST).IO = (&00)
```

(1) Die Meldung mit dem Meldungsschlüssel `CMD0500` wird nach SYSOUT und SYSLST und in den Bereich `BUF` ausgegeben. Von dort wird sie - über den Variablenstrom SYSMSG - in die S-Variable ABC geschrieben. Der Ausgabebereich `BUF` ist im Mapping-Format strukturiert.

(2) Die Meldung mit dem Meldungsschlüssel `DMS0DF8` wird nach SYSOUT und in den Bereich `BUF` im Mapping-Format ausgegeben. Da sie keine garantierte Meldung ist, wird sie jedoch nicht in den Variablenstrom SYSMSG aufgenommen, also auch nicht nach ABC geschrieben.

(3) Die Meldung mit dem Meldungsschlüssel `CMD0800` wird nach SYSLST und in den Bereich `BUF` ausgegeben. BUFFUSE ist standardmäßig auf INTERNAL gesetzt; es erfolgt keine Aufnahme in den Variablenstrom SYSMSG und in die S-Variable ABC.

(4) Die Meldung mit dem Meldungsschlüssel `DMS0574` wird nach SYSLST und in den Bereich `BUF` ausgegeben. Von dort wird sie - über den Variablenstrom SYSMSG - in die S-Variable ABC geschrieben.

(5) Es wird eine zusammengesetzte S-Variable vom Typ „Liste“ mit dem Namen ABC deklariert. Jede garantierte Meldung wird in die strukturierte S-Variable ABC als Listenelement abgespeichert. Die S-Variable wird solange um Elemente erweitert, bis die Zuweisung nach SYSMSG beendet wird.

(6) Die Zuweisung des Variablenstroms SYSMSG zur S-Variablen ABC wird aufgehoben. Der Inhalt der S-Variablen ABC wird angezeigt.

Neben den vom Programm in die S-Variable ABC geschriebenenen Meldungen enthält die S-Variable auch alle garantierten Meldungen die beim Laden des Moduls MSG7X2 ausgegeben wurden. Auch diese wurden nach dem Umlenken des Variablenstroms SYSMSG in die S-Variable ABC beim Laden des Moduls MSG7X2 ausgegeben.

MSGRC – Returncodes für MSG-Makros ausgeben

Allgemeines

Anwendungsgebiet: Meldungswesen; siehe Seite 165
Makrotyp: Definitionsmakro; siehe Seite 28

Makrobeschreibung

Der Makro **MSGRC** gibt die Returncodes und Returncode-Erläuterungen in Form einer (Equate-) Liste für die folgenden, im Handbuch beschriebenen Makros aus:

MSG7 MSGSINIT MSGSHOW MSGSMOD

Die Makros können einzeln oder als Liste angegeben werden. Die vom Makro **MSGRC** ausgegebene Liste der Returncodes ist nicht zwingend in der gleichen Reihenfolge geordnet.

Makroaufrufformat und Operandenbeschreibung

MSGRC
P=I / p ,FUNCT=ALL / makro / (makro, ...,makro)

P=
Präfix aller symbolischen Namen der Liste.

I
Alle symbolischen Namen beginnen mit `I`.

p
1 Buchstabe, der als Präfix genutzt werden soll.

FUNCT=
dient der Angabe der Makros, deren Returncodes aufgelistet werden sollen.

ALL
Alle Makros der Funktionsgruppe Meldungswesen werden aufgelistet.

makro
Name des Makros, dessen Returncodes ausgegeben werden sollen.

(makro, ..., makro)
Liste von Makronamen, deren Returncodes ausgegeben werden sollen.

Liste für die Makros MSG7, MSGSHOW, MSGSINIT und MSGSMOD

```
          MSGRC  P=A,FUNCT=(MSGSMOD,MSG7,MSGSHOW,MSGSINIT)
1         #INTF INTCOMP=1,INTNAME=MIP-MSG7,REFTYPE=REQUEST
1 ********       ********
1 ********  MSG7  ********
1 ********       ********
1 *
1 AM7OK      EQU   X'00'      MSG7 PROCESSED SUCCESSFULLY
1 AM7IOERR   EQU   X'04'      I/O ERROR
1 AM7MBINC   EQU   X'08'      PARAMETER LIST ERROR
1 AM7REPBA   EQU   X'0C'      REPLY REQUIRED IN BATCH PROCESSING
1 AM7RQMER   EQU   X'10'      NO MEMORY AVAILABLE TO PROCESS THE FUNCTION
1 AM7BRKWR   EQU   X'14'      BREAK DURING THE WROUT MACRO
1 AM7TRUNC   EQU   X'18'      MESSAGE-TEXT TRUNCATED
1 AM7MOINT   EQU   X'20'      MESSAGE OUTPUT PROCESSING INTERRUPTED
1 AM7RCINC   EQU   X'24'      INCORRECT ROUTING-CODE
1 AM7RPLST   EQU   X'2C'      REPLY INCORRECT WHEN SYSLST REQUIRED
1 AM7RPMD    EQU   X'30'      REPLY INCORRECT WHEN MANY OUTPUT
1 *                           DESTINATIONS GIVEN
1         SPACE 3
1 ********           ********
1 ******** MSGSINIT  ********
1 ********           ********
1 *
1 ASIOK      EQU   X'00'      MSGSINIT PROCESSED SUCCESSFULLY
1 ASIUNRES   EQU   X'04'      FILE: NO FILE AVAILA&BLE.
1 *                          TRACE: REQUIRED STATUS ALREADY EXISTING
1 ASIPLERR   EQU   X'08'      PARAMETER LIST ERROR
1 ASINTSOS   EQU   X'0C'      USER NOT TSOS
1 ASINOTPR   EQU   X'10'      SYSTEM UNABLE TO PROCESS THE MACRO
1         SPACE 3
1 ********             ********
1 ********   MSGSMOD   ********
1 ********             ********
1 *
1 ASMOK      EQU   X'00'         MSGSMOD PROCESSED SUCCESSFULLY
1 ASMERRDP   EQU   X'04'         MSGSMOD ERROR DURING PROCESS
1 ASMPLERR   EQU   X'08'         MSGSMOD PARAMETER LIST ERROR
1 ASMNTSOS   EQU   X'0C'         MSGSMOD USER NOT TSOS
1 ASMRQMER   EQU   X'10'         NO MEMORY AVAILABLE ($REQM ERROR)
1 ASMNOFIL   EQU   X'24'         MSGSMOD NO MESSAGE FILE
1 ASMNOTPR   EQU   X'28'         MSGSMOD UNABLE TO PROCESS
1         SPACE 3
```

```
1 ********              ********
1 ********   MSGSHOW    ********
1 ********              ********
1 *
1 ASHOK      EQU   X'00'         MSGSHOW PROCESSED SUCCESSFULLY
1 ASHERRDP   EQU   X'04'         ERROR DURING PROCESS
1 ASHPLERR   EQU   X'08'         MSGSHOW PARAMETER LIST ERROR
1 ASHNTSOS   EQU   X'0C'         BUFFER TOO SHORT
1 ASHRQMER   EQU   X'10'         NO MEMORY AVAILABLE ($REQM ERROR)
1 ASHUNBLE   EQU   X'30'         UNABLE TO PROCESS (INTERNAL ERROR)
```

MSGSHOW – Informationen über System- oder taskspezifische Meldungsdateien ausgeben

Allgemeines

Anwendungsgebiet: Meldungswesen; siehe Seite 165
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-/C-/D-Form; siehe Seite 29

Für Meldungsdateien kann ein Geltungsbereich (System oder Task) vereinbart werden. Der nichtprivilegierte Anwender kann, begrenzt auf seine Task, eigene Meldungsdateien zur Meldungsausgabe benutzen. Zusätzlich kann eine Sprache vereinbart werden, in der die Meldungstexte bevorzugt ausgegeben werden sollen. Meldungsdateien und Sprachangabe werden mit dem Makro **MSGSMOD** oder dem Kommando MODIFY-MSG-FILE-ASSIGNMENT in das Meldungssystem eingebracht.

Dem Makro **MSGSHOW** entspricht das Kommando SHOW-MSG-FILE-ASSIGNMENT. Zu den Kommandos siehe Handbuch „Kommandos“ [19].

Makrobeschreibung

Der Makro **MSGSHOW** informiert über:

– Anzahl der Meldungsdateien (System, Task),
– Sprache, in der die Meldungstexte ausgegeben werden (System, Task),
– Namen der Meldungsdateien; vorangestellt ist jedem Namen ein Indikator zur Bezeichnung der Zugriffsmethode (DLAM, ISAM). Die System-Meldungsdateien werden zuerst aufgelistet, gefolgt von den taskspezifischen Meldungsdateien.

Das Layout der Ausgabe ist im Anschluss an die Operandenbeschreibung dargestellt.

Makroaufrufformat und Operandenbeschreibung

MSGSHOW

BUFFER= (länge,{ adr / (r) })

,SCOPE=<u>BOTH</u> / SYSTEM / TASK

,MF=<u>S</u> / C / (C,pre) / (E,...) / (D,pre) / D / L

BUFFER=
bezeichnet die Länge und Adresse eines Bereichs zur Aufnahme der Ausgabedaten. Der Bereich muss auf Wortgrenze ausgerichtet sein. Der Makro **MSGDSHL** generiert eine Beschreibung (DSECT/Datenabschnitt) des Ausgabebereichs.

länge
Länge des Bereichs in Byte; länge ≥ 16 Byte. Wenn der Bereich zu klein ist, wird nur die Anzahl der Meldungsdateien eingetragen; siehe Rückinformation RC = X'0C'.

adr
symbolische Adresse (Name) des Bereichs.

(r)
r = Register mit dem Adresswert adr.

SCOPE=
bestimmt, aus welchem Geltungsbereich die Meldungsdateien aufgelistet werden.

BOTH
Es werden sowohl die System- als auch die taskspezifischen Meldungsdateien aufgelistet.

SYSTEM
Es werden nur die System-Meldungsdateien aufgelistet.

TASK
Es werden nur die taskspezifischen Meldungsdateien aufgelistet.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Bei der C-Form oder D-Form kann ein Präfix (pre = 1..4 Buchstaben), wie im Aufrufformat dargestellt, angegeben werden.

Voreinstellung: pre = C für C-Form
pre = D für D-Form

Die Angabe von weniger als vier Buchstaben für das jeweilige Präfix führt zu der Zeichenfoge: SHOx (x ≙ 1. Buchstabe).

Layout der Ausgabe

MSGDSHL
[C/D][,p]

Der Makro **MSGDSHL** generiert eine Datenliste des Ausgabebereichs oder eine Dummy Section (DSECT) zu dem Ausgabebereich.

C/D
Ein Datenliste / DSECT wird generiert.

p
Präfix für die symbolischen Namen der DSECT/Datenliste. Als Präfix werden 1..3 Zeichen der angegebenen Zeichenfolge benutzt.
Voreinstellung: p = SHL

Registerverwendung

Register R1 enthält die Adresse des Datenbereichs.
Register R15 enthält den Returncode.

Rückinformation und Fehleranzeigen

R15:

0	0	0	0	0	0	a	a

Über die Ausführung des Makros MSGSHOW wird ein Returncode im rechtsbündigen Byte des Registers R15 übergeben.

X'aa'	**Erläuterung**
X'00'	Normale Ausführung.
X'04'	Fehler während der Makroausführung.
X'08'	Operandenfehler.
X'0C'	Ausgabebereich zu klein.
X'30'	Makro kann nicht ausgeführt werden.
X'41'	MIP-Subsystem ist nicht geladen.

DSECT zum Ausgabebereich

```
          MSGDSHL D
1         #INTF INTCOMP=1,INTNAME=MIP-SHOW,REFTYPE=REQUEST
1 SHLD    MFPRE MF=D,PREFIX=*NONE,DNAME=SHLD,MACID=SHL,DMACID=SHL,      C
1               ALIGN=D
2 SHLD    DSECT ,
2               *,##### PREFIX=, MACID=SHL #####
1 SHLDBUFL DC   Y(SHLDMLEN)           --LENGTH OF THE BUFFER
1 SHLDSYSN DC   X'00'                 --SYSTEM FILE NUMBER
1 SHLDTSKN DC   X'00'                 --TASK FILE NUMBER
1 SHLDSYSL DC   CL3' '                --SYSTEM LANGUAGE
1 SHLDTSKL DC   CL1' '                --TASK LANGUAGE
1 SHLDSEAR DC   X'00'                 --SEARCH VALUE
1 SHLDSALL EQU  X'80'                 --SEARCH = *ALL
1 SHLDSTSK EQU  X'20'                 --SEARCH = *TASK
1 SHLDRES  DS   CL7                   --RESERVED
1 SHLDFIXD EQU  *-SHLD                --FIXED-PART LENGTH
1 SHLDFLST DS   510CL55               --FILE NAMES LIST
1          ORG  SHLDFLST
1 SHLDFNAM DS   CL55                  --1ST FILE
1          ORG  SHLDFNAM
1 SHLDINDT DS   CL1                   --FILE TYPE
1 SHLDTIDL EQU  X'80'                 --DLAM + ISAM
1 SHLDTISA EQU  X'40'                 --ISAM
1 SHLDTDLA EQU  X'20'                 --DLAM
1 SHLDTLDL EQU  B'00010000'           --LOCAL DLAM + ...
1 SHLDNAME DS   CL54                  --FILE NAME
1          ORG
1 SHLDMLEN EQU  *-SHLD                --MAX LENGTH OF THE BUFFER
```

MSGSINIT – Meldungsdatei sperren oder dem Meldungssystem hinzufügen

Allgemeines

Anwendungsgebiet: Meldungswesen (Systemverwaltermakro); siehe Seite 165
Makrotyp: S-Typ, MF-Format **1** (31-Bit-Schnittstelle):
Standardform/L-/E-/C-/D-Form; siehe Seite 29

Die Zuordnung zwischen Systemmeldungen und den sie enthaltenden Meldungsdateien erfolgt mit der globalen Bereichszuordnungstabelle (Class List). Diese enthält alle Meldungsklassen (die ersten 3 Zeichen des Meldungsschlüssels) und die zugeordneten Namen der Meldungsdateien. Die globale Bereichszuordnungsliste wird beim Systemstart aufgebaut und kann von der Systemverwaltung im laufenden Betrieb mit den Makros **MSGSINIT** / **MSGSMOD** oder dem Kommando MODIFY-MSG-FILE-ASSIGNMENT modifiziert werden.
Eine Meldungsdatei besteht aus der Meldungs-Arbeitsdatei und der korrespondierenden HELP-Datei (reduzierte Meldungs-Primärdatei).

Makrobeschreibung

Der Makro **MSGSINIT** ermöglicht es der Systemverwaltung, dem Meldungssystem eine weitere Meldungsdatei hinzuzufügen oder den Zugriff auf eine Meldungsdatei zu verhindern. Meldungsklasse und Name der neuen Meldungsdatei werden am Anfang der globalen Bereichszuordnungsliste eingetragen. Für eine zu sperrende Meldungsdatei wird der Verweis in der globalen Bereichzuordnungsliste gelöscht. Die Modifizierung dieser Liste gilt nur für den aktuellen Systemlauf, die Generierungswerte werden nicht verändert. Mit **MSGSINIT** kann die Systemverwaltung die Trace-Funktion von MIP aktivieren oder deaktivieren.

Makroaufrufformat und Operandenbeschreibung

MSGSINIT
[FILE=ADD / DEL / STD] [,TRACE=ON / OFF] ,MF=S / C / (C,pre) / (E,...) / (D,pre) / D / L

FILE=
Der Operand bezieht sich auf die Meldungsdatei, die hinzugefügt oder gesperrt werden soll. Der betreffenden Meldungs-Arbeitsdatei muss vor dem Makroaufruf der Linkname SMSGFILE zugewiesen werden.

ADD
Meldungsklassen und Name der Meldungsdatei werden am Anfang der globalen Bereichszuordnungsliste eingefügt.

DEL
für die zu sperrende Meldungsdatei werden alle Verweise in der globalen Bereichszuordnungsliste gelöscht.

STD
der STARTUP-Zustand der globalen Bereichszuordnungsliste wird eingerichtet.

TRACE=
Der Operand schaltet die Trace-Funktion von MIP ein oder aus.

ON
Die Trace-Funktion ist aktiviert.

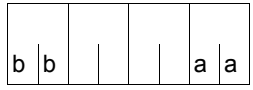
OFF
Die Trace-Funktion ist deaktiviert.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Bei der C-Form oder D-Form kann ein Präfix (pre = 1 Buchstabe), wie im Aufrufformat dargestellt, angegeben werden. Dieser Präfix wird allen symbolischen Namen der CSECT/DSECT an der 4. Stelle (nach der Silbe „INI“) eingefügt.

Voreinstellung: pre = C für C-Form
pre = D für D-Form

Rückinformation und Fehleranzeigen

R15:



Über die Ausführung des Makros MSGSINIT wird ein Returncode im rechtsbündigen Byte des Registers R15 übergeben.

Ret.Code	Erläuterung
X'00'	Normale Ausführung
X'04'	Fehler beim Operanden FILE: Die Meldungsdatei ist nicht verfügbar oder der Linkname ist nicht zugewiesen.
X'08'	Operandenfehler
X'0C'	Fehler beim Prüfen der Zugriffsberechtigung: Der Aufrufer ist kein Systembetreuer (hat keine TSOS-Kennung).
X'10'	Ressourcenbegrenzung: Makro kann nicht ausgeführt werden.
X'41'	MIP-Subsystem ist nicht geladen.

MSGSMOD – Meldungsdateien sperren oder dem Meldungssystem hinzufügen

Allgemeines

Anwendungsgebiet: Meldungswesen; siehe Seite 165
Makrotyp: S-Typ, MF-Format **1** (31-Bit-Schnittstelle):
Standardform/L-/E-/C-/D-Form; siehe Seite 29

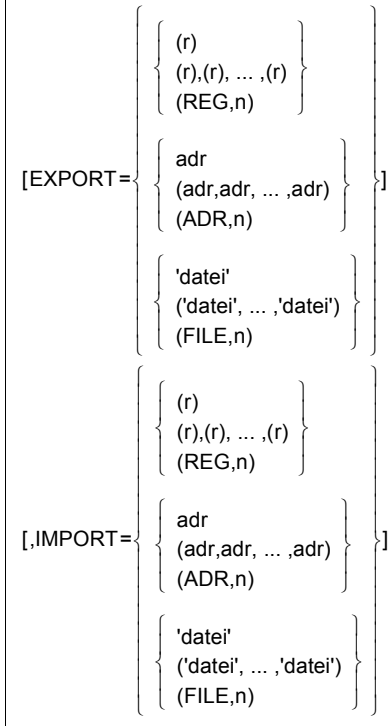
Die Zuordnung zwischen Systemmeldungen und den sie enthaltenden Meldungsdateien erfolgt mit der globalen Bereichszuordnungstabelle (Class List). Diese enthält alle Meldungsklassen (die ersten 3 Zeichen des Meldungsschlüssel) und die zugeordneten Namen der Meldungsdateien. Die globale Bereichszuordnungsliste wird beim Systemstart aufgebaut und kann von der Systemverwaltung im laufenden Betrieb mit den Makros **MSGSINIT**/**MSGSMOD** oder dem Kommando MODIFY-MSG-FILE-ASSIGNMENT modifiziert werden.
Der nichtprivilegierte Anwender kann, begrenzt auf seine Task, eigene Meldungsdateien zur Meldungsausgabe benutzen. Mit dem Makro **MSGSHOW** oder dem Kommando SHOW-MSG-FILE-ASSIGNMENT können die Namen der (System-) und/oder taskspezifischen Meldungsdateien abgefragt werden. Der Standardwert für die Sprache der Meldungsausgabe ist im Benutzerkatalog festgelegt, siehe Kommando SHOW-USER-ATTRIBUTES, Ausgabefeld DEFAULT-MSG-LANGUAGE. Wenn er dort nicht angegeben ist, wird der im Startup-Parameterservice eingestellte Wert benutzt. Zu den Kommandos siehe das Handbuch „Kommandos“ [19].

Makrobeschreibung

Die Systemverwaltung kann mit dem Makro **MSGSMOD** dem Meldungssystem weitere Meldungsdateien hinzuzufügen oder Zugriffe auf Meldungsdateien verhindern. Meldungsklassen und Namen neuer Meldungsdateien werden am Anfang der globalen Bereichszuordnungsliste eingetragen. Für zu sperrende Meldedateien werden alle Verweise in der globalen Bereichzuordnungsliste gelöscht. Die Modifizierung dieser Liste gilt nur für den aktuellen Systemlauf - die Generierungswerte werden nicht verändert.
Der nichtprivilegierte Anwender kann mit **MSGSMOD** für den aktuellen Tasklauf eigene Meldungsdateien in das Meldungssystem einbringen und eine Sprache für die Meldungsausgabe festlegen. Die taskspezifischen Meldungsdateien werden bei der Meldungssuche vor den (System-)Meldungsdateien durchsucht.
Mit einem Makroaufruf können maximal 8 Meldungsdateien dem Meldungssystem hinzugefügt (Operand IMPORT) und maximal 8 Meldungsdateien gesperrt werden (Operand EXPORT). Bei gleichzeitiger Angabe von EXPORT und IMPORT müssen die Dateinamen in derselben Art und Weise übergeben werden (entweder Register (r) oder Adressen (adr) oder Dateinamen (datei)).

Makroaufrufformat und Operandenbeschreibung

MSGSMOD



[,SCOPE=TASK / SYSTEM]

,SEARCH=*UNCHANGED / *ALL / *STD / *TASK [1]

,LAN=*UNCHANGED / *STD / 'sprache'

,MF=S / C / (C,pre) / (E,...) / (D,pre) / D / L

[1] Der Operand SEARCH wird nicht mehr ausgewertet. Er kann aus Kompatibilität noch angegeben werden.

EXPORT=
IMPORT=
bezeichnet die Meldungsdatei(en), die gesperrt (EXPORT) bzw. die dem Meldungssystem hinzugefügt (IMPORT) werden soll(en).

(r)
r = Register mit dem Adresswert des Feldes mit dem Namen der Meldungsdatei.

((r),..(r))
Angabe einer Liste mit maximal 8 Registern. Jedes Register beinhaltet den Adresswert eines Feldes mit dem Namen einer Meldungsdatei.

(REG,n)
Die Angabe der Register wird im Datenbereich erwartet. Der Operand kann nur in Verbindung mit MF=L/D/C angegeben werden.
n = Anzahl der Adressen; n ≤ 8. Voreinstellung: n=1.

adr
symbolische Adresse des Feldes mit dem Namen der Meldungsdatei

(adr,...)
Angabe einer Liste mit maximal 8 symbolischen Adressen. Die Adressen bezeichnen Felder mit den Namen der Meldungsdateien.

(ADR,n)
Die Angabe der Adressen wird im Datenbereich erwartet. Der Operand kann nur in Verbindung mit MF=L/D/C angegeben werden.
n = Anzahl der Adressen; n ≤ 8. Voreinstellung: n=1.

'datei'
datei = Name einer Meldungsdatei.

(datei,...)
Angabe einer Liste mit maximal 8 (Meldungs-)Dateinamen.

(FILE,n)
Die Angabe der Namen der Meldungsdateien wird in der Operandenliste erwartet. Der Operand kann nur in Verbindung mit MF=L/D/C angegeben werden.
n = Anzahl der Meldungsdateien; n ≤ 8. Voreinstellung: n=1.

SCOPE=
bestimmt den Geltungsbereich der bei IMPORT/EXPORT angegebenen Meldungsdateien. Geltungsbereich kann das System (alle Tasks im System) oder, eingeschränkt, die Task mit dem aufrufenden Programm sein.

TASK
Die Meldungssuche in den bei IMPORT/EXPORT angegebenen Meldungsdateien betrifft nur die Task mit dem den Makro **MSGSMOD** aufrufenden Programm. Bei Taskende wird die Zuordnung automatisch wieder gelöscht. EXPORT kann nur spezifiziert werden, wenn zuvor die Meldungsdatei(en) mit IMPORT der Task zugeordnet wurde(n). TASK ist Voreinstellung für den nichtprivilegierten Anwender.

SYSTEM
Die Meldungssuche in den bei IMPORT/EXPORT angegebenen Meldungsdateien betrifft alle Tasks im System. Die Zuordnung gilt nur für den aktuellen Systemlauf. Der Operand SYSTEM kann nur unter der Kennung der Systemverwaltung angegeben werden.
SYSTEM ist Voreinstellung für die Systemverwaltung.

LAN=
ermöglicht die Angabe einer Sprache für die Meldungsausgabe. Die Vereinbarung gilt nur für den Tasklauf.

***UNCHANGED**
Die für den Tasklauf vereinbarte Sprache wird nicht gewechselt.

***STD**
Sprache entsprechend Benutzerkatalog oder Systemparameter MSGLPRI.

'sprache'
sprache = 1 Buchstabe. Es gilt: D = Deutsch, E = Englisch.
Zeichen für andere Sprachen sind bei der Systemverwaltung zu erfragen.

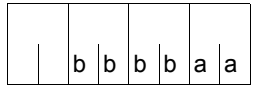
MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Die C-Form oder D-Form ist nur möglich in Verbindung mit IMPORT/EXPORT=REG/ADR/FILE. Es kann ein Präfix (pre = 1..4 Buchstaben), wie im Aufrufformat dargestellt, angegeben werden.

Voreinstellung: pre = MODC für C-Form
pre = MODD für D-Form

Die Angabe von weniger als vier Buchstaben für das jeweilige Präfix führt zu der Zeichenfoge: MODx (x ≙ 1. Buchstabe).

Rückinformation und Fehleranzeigen

R15:



Über die Ausführung von MSGSMOD wird im Register R15 ein Returncode übergeben. Die Werte bezeichnen Sedezimalkonstanten.
bbbb muss bitweise ausgewertet werden.
Bit $2^{n-1} = 1$: Die n-te Datei (in der angegebenen Reihenfolge, mit EXPORT=... beginnend) konnte nicht fehlerfrei bearbeitet werden.

X'bbbb'	**X'aa'**	**Erläuterung**
X'xxxx'	X'00'	Normale Ausführung.
X'xxxx'	X'04'	Fehler während der Makroausführung.
	X'08'	Operandenfehler.
	X'0C'	Zugriffsberechtigung: Anwender ist nicht TSOS (Systemverwaltung).
	X'10'	Ressourcenbegrenzung (REQM-Fehler)
X'xxxx'	X'30'	Makro kann nicht ausgeführt werden.
	X'41'	MIP-Subsystem ist nicht geladen.

NKDINF – Daten über (periphere) Konfiguration ausgeben

Allgemeines

Anwendungsgebiet: Abfragen und Zugriff zu Listen und Tabellen; siehe Seite 158
Makrotyp: S-Typ, MF-Format **2**: Standardform/E-/L-/C-/D-/M-Form; siehe Seite 29

Das NDM (Nucleus Device Management) stellt für den Anwender, den Operator und die Systemverwaltung Informationen über den Belegungs- und Verfügbarkeitszustand der Konfiguration und der montierten Datenträger zur Verfügung. Zur Konfiguration zählen CPUs, Kanäle, Steuerungen und Geräte.

Den Geräten (Plattengeräte, Bandgeräte, Drucker, ...) sind Gerätetypen (T-C), Gerätefamilien (F-C), Volumetypen (V-T) und Geräteklassen zugeordnet, siehe Handbuch „Systeminstallation“ [11]. Ein bestimmtes Gerät kann über seinen mnemotechnischen Gerätenamen (MN) angesprochen werden.

Der Makro **NKGTYPE** informiert über Namen, Gerätetypcode, Geräteeigenschaften, Pfadadressen und Pfadeigenschaften für einen Geräte- oder Volumetyp oder über Namen und Gerätetypcodes der Gerätetypen, die zu einer Gerätefamilie oder Geräteklasse gehören.

Makrobeschreibung

Der Makro **NKDINF** ermöglicht den Zugriff auf Daten, die von dem Informationsdienst (NKD) des NDM zur Verfügung gestellt werden. Die Daten informieren über

– die Belegungsmenge einer Task;
– den Belegungs- und Verfügbarkeitszustand von Geräten, Geräte eines Gerätetyps, Geräte einer Gerätefamilie, Platten oder Bändern;
– die Struktur der Konfiguration;
– die Warteschlange für die Gerätereservierung.

Der Makro **NKDINF** übergibt die angeforderten Informationen in geeignet strukturierten Ausgabesätzen. Zur Interpretation der Ausgabe stellt er dem Anwender die Layouts dieser Ausgabesätze als DSECTs zur Verfügung.

Die Ausgabesätze enthalten Informationen über den Belegungs- und Verfügbarkeitszustand der angegebenen Geräte, Hardwareeinheiten oder Betriebsmittel. Die Sätze werden in einen Bereich des Klasse-6-Speichers eingetragen, den der Makro zuvor angefordert hat.

In diesem Ausgabebereich wird den vom Anwender angeforderten Ausgabesätzen noch der Output-Control-Record vorangestellt, ein Satz mit allgemeinen Informationen über die Makroausführung und die Struktur des Ausgabebereiches. Die Anfangsadresse des Bereiches wird im Feld `NKDIOPTR` des Datenbereichs übergeben.

Für die Rückgabe des Speicherbereiches ist der Aufrufer zuständig (**RELM**-Makro). Die Längenangaben sind dem Output-Control-Record zu entnehmen. Bei der Rückgabe ist zu beachten:

– Die Anfangsadresse des Ausgabepuffers ist auf Seitengrenze ausgerichtet.
– Das Feld `NKDIOLEN` im Output-Control-Record enthält die Größe des Ausgabebereiches: Für den nichtprivilegierten Aufrufer ist sie in Hauptspeicherseiten (zu je 4K) angegeben.

Durch einen Makroaufruf mit MF=D und dem Operanden RECORD kann sich der Anwender das Layout eines jeden Ausgabesatzes als DSECT erzeugen lassen, der von **NKDINF** angeboten wird. Die DSECT ermöglicht eine symbolische Adressierung der einzelnen Felder eines Ausgabesatzes.

Der Anwender muss die Auswertung seines CONFIG-Records ändern, wenn er bestehende Programme (aus einer Version < BS2000/OSD-BC V3.0) mit BS2000/OSD-BC ≥ V3.0 übersetzt. Das Layout hat sich gegenüber BS2000/OSD-BC V2.0 inkompatibel geändert, da sich die Anzahl der Inner Connections des CONFIG von 4 auf 8 erhöht hat.
In Versionen < BS2000/OSD-BC V3.0 übersetzte Programme bekommen die Ausgabe im bisherigen Format übergeben, sind also weiterhin ablauffähig.

Für den nichtprivilegierten Anwender bestehen folgende Einschränkungen:

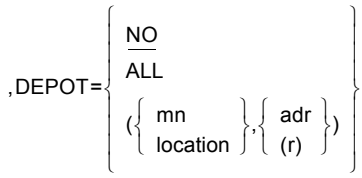
– Tasksätze werden nur für Tasks unter der eigenen Kennung ausgegeben.
– Der Aufrufer erhält keine Informationen über Gerätewarteschlangen (Operand DVQ), über Belegungen und Reservierungen angegebener Gerätetypen (Operand TYPTASK), über Platten, für die mit dem Kommando SET-DISK-PARAMETER Benutzungsvorgaben gemacht wurden (Operand DISC) und über Platten, für die das Produkt DRV im Einsatz ist (Operand DRV).
– Informationen über Belegungen anderer Benutzer werden ausgeblendet.
– Der Aufrufer erhält im LOC-Record nur die Ausgabe der SUMMARY-Information.

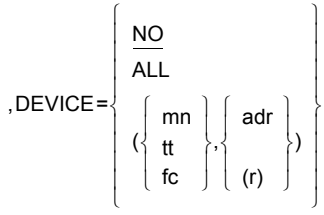
Makroaufrufformat und Operandenbeschreibung

NKDINF

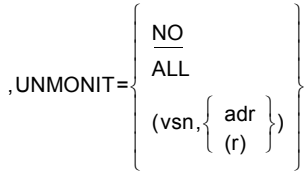
```
CONFIG={ NO
       | ALL
       | CHN
       | CPU
       | CTL
       | DVC
       | IOSIDE
       | SE
       | SIDE
       | ( { ctl-mn | chpid | chnrange | icuu | mn | ioside# | cpu# | se# | side# | dev# } ,{ adr | (r) } [,{ S | L }] ) }

,EXTMN=NO / YES

,DISC={ NO
      | MONITORED
      | SCHEDULED
      | ( { mn | vsn } ,{ adr | (r) } [,{ S | L[,TASK] }] ) }

,TAPE={ NO
      | ALL / (ALL,CAR)
      | ( { mn | vsn } ,{ adr | (r) } )
      | (mn,adr,CAR) }

,TASK={ NO
      | OWN
      | ( { tsn | tid } ,{ adr | (r) } ) }
```

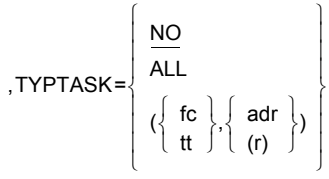
NKDINF (Fortsetzung)

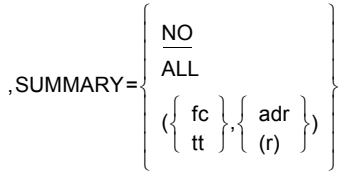


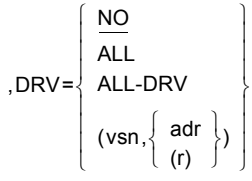


,GLOBAL=NO / YES



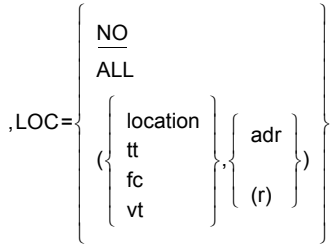






,DVQ=NO / YES

NKDINF (Fortsetzung)



[,RECORD=ALL / CONFIG / DEVICE / DEPOT / DISC / DRV / DVQ / GLOBAL / HEADER /
LOC / SUMMARY / TAPE / TASK / TYPTASK / UNMONIT]

,HWSTATE=NO / YES

,MF=S / E / L / C / D / M

[,PARAM=adr / (r)]

,PREFIX=N / p

,MACID=KDI / macid

Die Operanden `CONFIG=IOSIDE/SE/SIDE/(ioside#,...)/(se#,...)/(side#,...)` liefern keine Informationen mehr, da die zugehörige Hardware nicht mehr unterstützt wird. Die Operandenwerte können noch aus Kompatibilität angegeben werden.

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

CONFIG=
für jede angegebene Einheit der Konfiguration wird ein CONFIG-Ausgabesatz geschrieben.

NO
Diese Funktion wird nicht gewünscht.

ALL
Informationen über alle generierten Units (Hardwareeinheiten) werden angefordert. Bei großen Konfigurationen kann mit CONFIG=ALL ein sehr großer Ausgabebereich entstehen.

CHN
Informationen über alle Kanäle werden angefordert.

CPU
Informationen über alle Zentraleinheiten werden angefordert.

CTL
Informationen über alle Mehrgerätesteuerungen werden angefordert.

DVC
Informationen über alle Geräte werden angefordert.

(ctl-mn,...)
Der Anwender übergibt in einer Liste (siehe Seite 691) die mnemotechnischen Namen der Gerätesteuerungen, über die er Informationen anfordert.

(chpid,...)
Der Anwender übergibt in einer Liste (siehe Seite 691) die CHANNEL_PATH_ID (Kanalpfadbezeichnungen) der Kanäle, über die er Informationen anfordert.

(chnrange,...)
Der Anwender übergibt in einer Liste (siehe Seite 691) den Kanalbereich, über den er Informationen anfordert.

(icuu,...)
Der Anwender übergibt in einer Liste (siehe Seite 691) die Geräteadressen von Hardwareeinheiten, über die er Informationen anfordert.
Eine Geräteadresse beschreibt den Weg, auf dem ein Gerät angesprochen werden kann. Sie ist 2 Byte lang und setzt sich aus folgenden Bestandteilen zusammen:

1. Byte: 1. Halbbyte: Nummer des Ein-/Ausgabeprozessors
2. Halbbyte: Kanalnummer

2. Byte: Anschlussnummer Mehrgerätesteuerung und Anschlussnummer Gerät bzw. nur Anschlussnummer Gerät (falls direkt am Kanal angeschlossen).

(mn,...)
Der Anwender übergibt in einer Liste (s. unten) die mnemotechnischen Namen der Hardwareeinheiten, über die er Informationen anfordert.

(cpu#,...)
Der Anwender übergibt in einer Liste (s. unten) die Nummern der Zentraleinheiten, über die er Informationen anfordert.

(dev#,...)
Der Anwender übergibt in einer Liste (s. unten) die Gerätenummern der Hardwareeinheiten, über die er Informationen anfordert.

adr
symbolische Adresse (Name) eines Feldes; Das Feld enthält eine Liste von mnemotechnischen Gerätenamen, von Geräteadressen oder von Gerätenummern. Es ist auf Wortgrenze auszurichten.

(r)
r = Register mit dem Adresswert von adr. Die Angabe eines Registers ist nur in Verbindung mit MF=M erlaubt.

Aufbau der **Liste** zur Übergabe der mnemotechnischen Gerätenamen, der Geräteadressen oder -nummern:

– Das erste Wort enthält rechtsbündig hexadezimal die Anzahl der Einträge in der Liste.
– Ihm folgen die Einträge: Jeder Eintrag ist 4 Byte lang und enthält linksbündig den mnemotechnischen Gerätenamen, die Geräteadresse oder die Gerätenummer. Nicht benötigte Byte sind mit X'00' zu überschreiben.

S
Die Informationen werden im kurzen Format (Standardformat) ausgegeben.

L
Die Informationen werden im langen Format ausgegeben und enthalten zusätzlich Pfadbeschreibungen.

DEPOT=
gibt Informationen über die Zuordnung von physikalischen Bandgeräten (Mnemonics) zu Lagerorten (Locations).

NO
Diese Funktion wird nicht gewünscht.

ALL
Für alle bekannten Lagerorte werden Informationen ausgegeben.

(mn,...)
Der Anwender übergibt in einer Liste (s. unten) die mnemotechnischen Namen der Bandgeräte, für die er Informationen anfordert.

(location,...)
Der Anwender übergibt in einer Liste (s. unten) die Lagerorte, für die er Informationen anfordert.

adr
symbolische Adresse eines Feldes; Das Feld enthält eine Liste von Mnemonics bzw. Locations. Es ist auf Wortgrenze auszurichten.

(r)
r = Register mit dem Adresswert von adr. Die Angabe eines Registers ist nur zusammen mit MF=M erlaubt.

Aufbau der **Liste** zur Übergabe der (Band-)Mnemonics bzw. der Locations:

– Das erste Wort der Liste enthält rechtsbündig hexadezimal die Anzahl der Einträge.
– Es folgen die Einträge: Für die Mnemonics ist jeder Eintrag 4 Byte lang. Für die Locations ist jeder Eintrag 8 Byte lang. Die Einträge enthalten linksbündig die Mnemonics bzw. die Locations.

DEVICE=
für jedes angegebene Gerät wird ein DEVICE-Ausgabesatz (DEVICE-Record) geschrieben. Siehe Hinweis bei RECORD=DEVICE, Seite 697.

NO
Diese Funktion wird nicht gewünscht.

ALL
Informationen über alle Geräte werden angefordert.

(mn,...)
Der Anwender übergibt in einer Liste (s. unten) die mnemotechnischen Namen der Geräte, über die er Informationen anfordert.

(tt,...)
Der Anwender übergibt in einer Liste (s. unten) die Codes für die Typen der Geräte (Gerätetypcodes), über die er Informationen anfordert.

(fc,...)
Der Anwender übergibt in einer Liste (s. unten) die Codes für die Familien der Geräte (Familycodes), über die er Informationen anfordert.

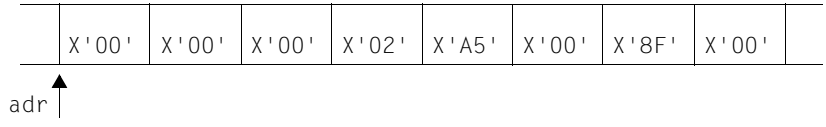
adr
symbolische Adresse (Name) eines Feldes; Das Feld enthält eine Liste von mnemotechnischen Gerätenamen, von Codes für Gerätefamilien oder von Gerätetypcodes. Es ist auf Wortgrenze auszurichten.

(r)
r = Register mit dem Adresswert von adr. Die Angabe eines Registers ist nur in Verbindung mit MF=M erlaubt.

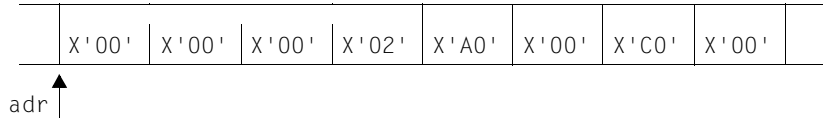
Aufbau der **Liste** zur Übergabe der Familycodes, Gerätetypcodes oder mnemotechnischen Gerätenamen:

- Das erste Wort enthält rechtsbündig hexadezimal die Anzahl der Einträge in der Liste.
- Ihm folgen die Einträge: Bei der Angabe von Family-Gerätetypcodes ist jeder Eintrag 2 Byte lang, bei der Angabe von mnemotechnischen Gerätenamen hängt die Länge eines Eintrags vom Wert des Operanden EXTMN ab: Bei EXTMN=NO ist jeder Eintrag 2 Byte lang, bei EXTMN=YES 4 Byte.
- Die Listen dürfen maximal 32 K Einträge enthalten.

Beispiel 1: 2 Gerätetypen mit Gerätetypcodes X'A5' (=D3435) und X'8F' (=D3475)



Beispiel 2: 2 Gerätefamilien mit Familycodes X'A0' (= Plattengeräte) und X'C0' (= MBK)



DISC=
fordert für jede(s) angegebene Platte (Plattengerät) einen DISC-Ausgabesatz (DISC-Record) an.

NO
Diese Funktion wird nicht gewünscht.

MONITORED
Es werden Informationen über alle Platten angefordert, die online verfügbar sind und von NDM überwacht werden.

SCHEDULED
Es werden Informationen über alle Platten angefordert, für die mit dem Kommando SET-DISK-PARAMETER explizit Benutzungsvorgaben gemacht wurden.
Dieser Wert kann nur unter der Kennung der Systemverwaltung (TSOS) angegeben werden.

(mn,...)
Der Anwender übergibt in einer Liste (s. unten) die mnemotechnischen Namen der Platten, über die er Informationen anfordert.

(vsn,...)
Der Anwender übergibt in einer Liste (s. unten) die VSNs (Volume Serial Numbers) der Platten, über die er Informationen anfordert.

adr
symbolische Adresse (Name) eines Feldes; Das Feld enthält entweder eine Liste von VSNs oder von mnemotechnischen Gerätenamen. Es ist auf Wortgrenze auszurichten.

(r)
r = Register mit dem Adresswert von adr. Die Angabe eines Registers ist nur in Verbindung mit MF=M erlaubt.

Aufbau der **Liste** zur Übergabe der mnemotechnischen Gerätenamen oder VSNs:
– Das erste Wort enthält rechtsbündig hexadezimal die Anzahl der Einträge in der Liste.
– Es folgen die Einträge: Jeder Eintrag ist 8 Byte lang und enthält linksbündig die VSN oder den mnemotechnischen Gerätenamen. Nicht benötigte Byte sind mit X'00' zu überschreiben.

S
Die Informationen werden im kurzen Format (Standardformat) ausgegeben.

L
Die Informationen werden im langen Format ausgegeben und enthalten zusätzlich Angaben über die SVL-Zustände und Plattenparameter.
Dieser Wert kann nur unter der Kennung der Systemverwaltung (TSOS) angegeben werden.

TASK
Ausgabe einer Liste mit den TSNs der Tasks, die momentan mit der Platte arbeiten. Die Liste besteht aus 4-Byte-Einträgen und wird nur für Privatplatten im Modus USE=DMS ausgegeben. Die Angabe von TASK ist nur für das lange Format (Angabe L) möglich.
Dieser Wert kann nur unter der Kennung der Systemverwaltung (TSOS) angegeben werden.

DRV=
fordert einen DRV-Ausgabesatz (DRV-Record) für jede angegebene Platte an, für die das Produkt DRV (Dual Recording by Volume; siehe Handbuch „DRV“ [25]) im Einsatz ist.

NO
Diese Funktion wird nicht gewünscht.

ALL
fordert für jede Platte einen DRV-Ausgabesatz an, die der Komponente DRV bekannt ist.
Dieser Wert kann nur unter der Kennung der Systemverwaltung (TSOS) angegeben werden.

ALL-DRV
fordert für jede Platte einen DRV-Ausgabesatz an, für die die Betriebsart DRV eingestellt ist.
Dieser Wert kann nur unter der Kennung der Systemverwaltung (TSOS) angegeben werden.

(vsn,...)
Der Anwender übergibt in einer Liste (s. unten) die VSN (Volume Serial Numbers) der Platten, über die er Informationen anfordert.
Dieser Wert kann nur unter der Kennung der Systemverwaltung (TSOS) angegeben werden.

adr
symbolische Adresse (Name) eines Feldes; Das Feld enthält eine Liste von VSN. Es ist auf Wortgrenze auszurichten.

(r)
r = Register mit dem Adresswert von adr. Die Angabe eines Registers ist nur in Verbindung mit MF=M erlaubt.

Aufbau der **Liste** zur Übergabe der VSN:
– Das erste Wort enthält rechtsbündig hexadezimal die Anzahl der Einträge in der Liste.
– Es folgen die Einträge: Jeder Eintrag ist 8 Byte lang und enthält linksbündig die VSN. Nicht benötigte Byte sind mit X'00' zu überschreiben.

DVQ=
gibt an, ob Informationen über die Gerätewarteschlange (Secure Queue) ausgegeben werden sollen.

NO
Die Funktion wird nicht gewünscht.

YES
DVQ-Ausgabesätze werden geschrieben.
Dieser Wert kann nur unter der Kennung der Systemverwaltung (TSOS) angegeben werden.

EXTMN=
gibt an, ob der Aufrufer mnemotechnische Gerätenamen im alten Format (2 Byte lang) oder im neuen Format (4 Byte lang) im Datenbereich übergibt und/oder im Ausgabebereich des Makros erwartet.
Zur Aufnahme der mnemotechnischen Namen stellt der Makro im Ausgabebereich jeweils Felder für das 2-Byte- und das 4-Byte-Format bereit. Ein Indikator zeigt für jeden Ausgabesatz an, ob das 2-Byte- oder das 4-Byte-Ausgabefeld versorgt ist.

NO
Der Anwender übergibt im Datenbereich nur 2 Byte lange mnemotechnische Gerätenamen und wertet auch nur 2 Byte lange Namen aus.
Werden für ein Gerät Informationen ausgegeben, dessen mnemotechnischer Name 4 Byte lang ist, so wird im Ausgabebereich das 2-Byte-Feld gelöscht, der Indikator für Ausgabe im 4-Byte-Format gesetzt und im Ausgabevorspann (OCR) sowie im Standardheader durch Returncodes die Unvollständigkeit der Information angezeigt.

YES
Der Anwender übergibt und erwartet mnemotechnische Gerätenamen immer im 4-Byte-Format.
Werden kürzere mnemotechnische Gerätenamen angegeben, so sind sie in die dafür vorgesehenen Felder des Datenbereichs linksbündig einzutragen und am rechten Ende mit Leerzeichen aufzufüllen.

GLOBAL=
gibt an, ob die Einstellung aller globalen NDM-Steuerparameter ausgegeben werden soll.

NO
Diese Funktion wird nicht gewünscht.

YES
Alle globalen NDM-Steuerparameter werden geschrieben.

HWSTATE=
gibt Informationen über den Hardwarestatus (ON/OFF) der Unit aus.

Anwender des CONFIG und CONFIG-L Records, die die Felder `SIDE_/GP_` etc. `HARDWARE_STATE` (in NKDCUTYP des CONFIG Records) nicht auswerten, sollten NKDINF mit dem Operanden HWSTATE=NO aufrufen, um die Bearbeitung zu beschleunigen.

NO
Diese Funktion wird nicht gewünscht.

YES
Der Hardwarestatus der Unit wird bestimmt (nur für spezielle Anwendungen von Bedeutung).

LOC=
gibt den Informationsumfang von SUMMARY und TYPTASK nach Lagerorten (Locations) geordnet aus.

NO
Diese Funktion wird nicht gewünscht.

ALL
Für jeden generierten Gerätetyp wird ein Ausgabesatz erzeugt.

(location,...)
Für jeden generierten Gerätetyp wird für die angegebenen Lagerorte ein Ausgabesatz erstellt. Zum Aufbau der Location-Liste siehe Operand DEPOT.

(tt,...)
Der Anwender übergibt in einer Liste (s. unten) die Codes der Gerätetypen, über die er Informationen anfordert.

(fc,...)
Der Anwender übergibt in einer Liste (s. unten) die Codes der Gerätefamilien (Familycodes), über die er Informationen anfordert.

(vt,...)
Der Anwender übergibt in einer Liste (s. unten) die Codes der Volumetypcodes, über die er Informationen anfordert.

adr
symbolische Adresse eines Feldes; Das Feld enthält eine Liste von Gerätetypcodes bzw. Familycodes. Es ist auf Wortgrenze auszurichten.

(r)
r = Register mit dem Adresswert von adr. Die Angabe eines Registers ist nur zusammen mit MF=M erlaubt.

Aufbau der **Liste** zur Übergabe der Gerätetypcodes bzw. Familycodes:
- Das erste Wort der Liste enthält rechtsbündig hexadezimal die Anzahl der Einträge in die Liste.
- Es folgen die Einträge: Jeder Eintrag ist 2 Byte lang und enthält linksbündig den Gerätetypcode bzw. den Familycode.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

RECORD
kann nur zusammen mit MF=C bzw. MF=D angegeben werden und legt fest, für welchen Ausgabesatz eine CSECT bzw. DSECT generiert wird.

ALL
erzeugt CSECTs/DSECTs für alle vom Makro angebotenen Ausgabesätze (HEADER bis DRV).

CONFIG
erzeugen eine CSECT/DSECT für den CONFIG-Ausgabesatz (Operand CONFIG).

DEPOT
erzeugt eine CSECT/DSECT für den DEPOT-Ausgabesatz (Operand DEPOT).

DEVICE
erzeugt eine CSECT/DSECT für den DEVICE-Ausgabesatz (Operand DEVICE).

Hinweis
Bedingt durch die Funktionserweiterung „Dynamische I/O-Konfigurationsänderung“ können im DEVICE-Record jetzt auch Records für jene Dummy-Geräte enthalten sein, die in der BS2000-Gerätetabelle als Platzhalter vorhanden sind und später durch reale Geräte ersetzt werden. Die Records dieser Geräte enthalten im Feld EXTENDED DEVICE MNEMONIC die Mnemonic „DMMY“ und im Feld DEVICE RECONFIGURATION STATE den Wert X'0F' (INVALID).

DISC
erzeugt eine CSECT/DSECT für den DISC-Ausgabesatz (Operand DISC).

DRV
erzeugt eine CSECT/DSECT für den DRV-Ausgabesatz (Operand DRV).

DVQ
erzeugt eine CSECT/DSECT für den DVQ-Ausgabesatz (Operand DVQ).

GLOBAL
erzeugt eine CSECT/DSECT für den GLOBAL-Ausgabesatz (Operand GLOBAL).

HEADER
erzeugt eine CSECT/DSECT für den Output-Control-Record.
Der Output-Control-Record enthält:
– Zeiger zu den verschiedenen Ausgabesätzen
– Zähler
– Längenangaben (Länge des Ausgabebereiches, Länge der einzelnen Ausgabesätze) und
– Returncodes für die verschiedenen Ausgabesätze.

Das Layout ist am Ende der Beschreibung wiedergegeben.

LOC
erzeugt eine CSECT/DSECT für den LOC-Ausgabesatz (Operand LOC).

SUMMARY
erzeugt eine CSECT/DSECT für den SUMMARY-Ausgabesatz (Operand SUMMARY).

TAPE
erzeugt eine CSECT/DSECT für den TAPE-Ausgabesatz (Operand TAPE).

TASK
erzeugt eine CSECT/DSECT für den TASK-Ausgabesatz (Operand TASK).

TYPTASK
erzeugt eine CSECT/DSECT für den TYPTASK-Ausgabesatz (Operand TYPTASK).

UNMONIT
erzeugt eine CSECT/DSECT für den UNMONIT-Ausgabesatz (Operand UNMONIT).

SUMMARY=
fordert für jede(n) angegebene(n) Gerätefamilie (Gerätetyp) einen SUMMARY-Ausgabesatz an. Der Satz enthält Übersichtsinformationen über die Gerätefamilie bzw. den Gerätetyp. Für jede Gerätefamilie wird die Menge der SUMMARY-Sätze ihrer Gerätetypen geliefert.

NO
Diese Funktion wird nicht gewünscht.

ALL
fordert für jeden im System definierten Gerätetyp einen SUMMARY-Ausgabesatz an.

(fc,...)
Der Anwender übergibt in einer Liste (s. unten) die Codes der Gerätefamilien (**F**amily-**C**odes), über deren Gerätetypen er Informationen anfordert.

(tt,...)
Der Anwender übergibt in einer Liste (s. unten) die Codes der Gerätetypen, über die er Informationen anfordert.

adr
symbolische Adresse (Name) eines Feldes; Das Feld enthält eine Liste von Familycodes oder von Gerätetypcodes. Es ist auf Wortgrenze auszurichten.

(r)
r = Register mit dem Adresswert von adr. Die Angabe eines Registers ist nur in Verbindung mit MF=M erlaubt.

Aufbau der **Liste** zur Übergabe der Familycodes oder der Gerätetypcodes:
- Das erste Wort enthält rechtsbündig hexadezimal die Anzahl der Einträge in der Liste.
- Ihm folgen die Einträge: Jeder Eintrag ist 2 Byte lang und enthält linksbündig den Familycode oder den Gerätetypcode.

TAPE=
für jedes angegebene Bandgerät wird ein TAPE-Ausgabesatz (TAPE-Record) geschrieben.

NO
Diese Funktion wird nicht gewünscht.

ALL
Informationen über alle von NDM überwachten Bänder werden angefordert.

(mn,...)
Der Anwender übergibt in einer Liste (s. unten) die mnemotechnischen Namen der Bandgeräte, über die er Informationen anfordert.

(vsn,...)
Der Anwender übergibt in einer Liste (s. unten) die VSNs (Volume Serial Numbers) der Bänder, über die er Informationen anfordert.

adr
symbolische Adresse (Name) eines Feldes; Das Feld enthält entweder eine Liste von VSNs oder von mnemotechnischen Gerätenamen. Es ist auf Wortgrenze auszurichten.

(r)
r = Register mit dem Adresswert von adr. Die Angabe eines Registers ist nur in Verbindung mit MF=M erlaubt.

Aufbau der **Liste** zur Übergabe der mnemotechnischen Gerätenamen oder VSNs:
- Das erste Wort enthält rechtsbündig hexadezimal die Anzahl der Einträge in der Liste.
- Ihm folgen die Einträge: Jeder Eintrag ist 8 Byte lang und enthält linksbündig die VSN oder den mnemotechnischen Gerätenamen. Nicht benötigte Byte sind mit X'00' zu überschreiben.

CAR
Für MBK-Geräte, die sich im „Random“-Modus befinden, werden zusätzlich Records für die im „Cartridge-Loader“ vorhandenen und bereits von der Geräteverwaltung registrierten Kassetten ausgegeben. Zu einem Gerät können mehrere Records bereit gestellt werden.

TASK=
Es wird ein TASK-Ausgabesatz (TASK-Record) geschrieben. Der Satz enthält Informationen über Geräte, Platten und Bänder, die von der angegebenen Task belegt sind.

NO
Diese Funktion wird nicht gewünscht.

OWN
Die Angaben beziehen sich auf die aufrufende Task.

(tsn,...)
Der Anwender übergibt in einem Wort die TSN (Task Sequence Number) der Task, über deren Geräte- und Volumebelegung er Informationen anfordert.

(tid,...)
Der Anwender übergibt in einem Wort die TID (Task Identifier; internes Taskkennzeichen) der Task, über deren Geräte- und Volumebelegung er Informationen anfordert.

adr
symbolische Adresse (Name) des Wortes mit der TSN bzw. TID einer Task

(r)
r = Register mit dem Adresswert von adr. Die Angabe eines Registers ist nur in Verbindung mit MF=M erlaubt.

TYPTASK=
fordert für jede(n) angegebene(n) Gerätefamilie (Gerätetyp) einen TYPTASK-Ausgabesatz an. Der Satz beschreibt, wie viele Geräte des angegebenen Typs eine Task reserviert hat. Für jede Gerätefamilie wird die Menge der TYPTASK-Sätze ihrer Gerätetypen ausgegeben.

NO
Diese Funktion wird nicht gewünscht.

ALL
fordert für jeden im System definierten Gerätetyp einen TYPTASK-Ausgabesatz an.
Dieser Wert kann nur unter der Kennung der Systemverwaltung (TSOS) angegeben werden.

(fc,...)
Der Anwender übergibt in einer Liste (s. unten) die Codes der Gerätefamilien (**F**amily-**C**odes), über deren Gerätetypen er Informationen anfordert.
Dieser Wert kann nur unter der Kennung der Systemverwaltung (TSOS) angegeben werden.

(tt,...)
Der Anwender übergibt in einer Liste (s. unten) die Codes der Gerätetypen, über die er Informationen anfordert.
Dieser Wert kann nur unter der Kennung der Systemverwaltung (TSOS) angegeben werden.

adr
symbolische Adresse (Name) eines Feldes; Das Feld enthält eine Liste von Familycodes oder von Gerätetypcodes. Es ist auf Wortgrenze auszurichten.

(r)
r = Register mit dem Adresswert von adr. Die Angabe eines Registers ist nur in Verbindung mit MF=M erlaubt.

Aufbau der **Liste** zur Übergabe der Familycodes oder der Gerätetypcodes:
- Das erste Wort enthält rechtsbündig hexadezimal die Anzahl der Einträge in der Liste.
- Ihm folgen die Einträge: Jeder Eintrag ist 2 Byte lang und enthält linksbündig den Familycode oder den Gerätetypcode.

UNMONIT=
fordert für jedes angegebene Bandvolume, das „offline“ reserviert wurde, einen UNMONIT-Ausgabesatz an. Der Satz beschreibt, welches Bandvolume (VSN) von welcher Task (TSN) reserviert wurde.
„Offline“-Reservierung bedeutet: Es existiert eine Geräte-/Volumeanforderung für ein Gerät, das von NDM (noch) nicht überwacht wird.

NO
Diese Funktion wird nicht gewünscht.

ALL
fordert für jedes „offline“ reservierte Bandvolume einen UNMONIT-Ausgabesatz an.

(vsn,...)
Der Anwender übergibt in einer Liste (s. unten) die VSNs (Volume Serial Numbers) der Bänder, über die er Informationen anfordert.

adr
symbolische Adresse (Name) eines Feldes; Das Feld enthält eine Liste von VSNs. Es ist auf Wortgrenze auszurichten.

(r)
r = Register mit dem Adresswert von adr. Die Angabe eines Registers ist nur in Verbindung mit MF=M erlaubt.

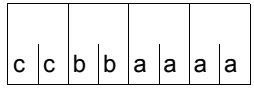
Aufbau der **Liste** zur Übergabe der VSNs:
- Das erste Wort enthält rechtsbündig hexadezimal die Anzahl der Einträge in der Liste.
- Ihm folgen die Einträge: Jeder Eintrag ist 8 Byte lang und enthält linksbündig die VSN. Nicht benötigte Bytes sind mit X'00' zu überschreiben.

Registerverwendung

Beim Aufruf des Makros **NKDINF** mit MF=M und der Angabe von Adressen wird Register R15 intern als Arbeitsregister verwendet.

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros NKDINF wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Funktion erfolgreich ausgeführt. Alle angeforderten Informationen sind im Ausgabebereich bereitgestellt.
X'01' [1]	X'00'	X'0008'	Ausgabebereich wurde erstellt, aber nicht alle angeforderten Ausgabesätze sind vorhanden: Einige oder alle der angegebenen Codes (Gerätetypcodes, VSNs, mnemotechnische Bezeichnungen usw.) sind unbekannt.
X'02' [1]	X'00'	X'0008'	Ausgabebereich wurde erstellt, aber nicht alle angeforderten Ausgabesätze sind vorhanden: Benutzer ist nicht ausreichend privilegiert.
X'04' [1]	X'00'	X'0008'	Ausgabebereich wurde erstellt, aber in einigen Ausgabesätzen ist die Information unvollständig: Disk-Monitor nicht verfügbar.
X'08' [1]	X'00'	X'0008'	Ausgabebereich wurde erstellt, aber in einigen Ausgabesätzen ist die Information unvollständig: Tape-Monitor nicht verfügbar.
X'10' [1]	X'00'	X'0008'	Ausgabebereich wurde erstellt, aber der Operand DRV konnte nicht bearbeitet werden: Subsystem DRV ist nicht verfügbar.
X'20' [1]	X'00'	X'0008'	Ausgabebereich wurde erstellt, aber in einigen Ausgabesätzen ist die Information unvollständig: Es sind vierstellige mnemotechnische Gerätenamen vorhanden, und im Makroaufruf wurde EXTMN=NO angegeben. Das 2-Byte-Feld in den betroffenen Ausgabesätzen wurde gelöscht.
X'xx' [2]	X'01'	X'0010'	Operandenfehler, kein Ausgabebereich wurde erstellt: Typ der angeforderten Information ist unbekannt oder nicht zulässig.
X'xx' [2]	X'01'	X'0011'	Operandenfehler, kein Ausgabebereich wurde erstellt: Eine der Angaben MN, VSN, DEV# usw. ist unbekannt oder nicht zulässig.
X'00'	X'01'	X'0013'	Operandenfehler, kein Ausgabebereich wurde erstellt: Operand EXTMN ist unbekannt.
X'00'	X'20'	X'0004'	Systemfehler, es wurde kein Ausgabebereich erstellt.
X'xx' [2]	X'40'	X'0012'	Typ eines angeforderten Ausgabesatzes (Recordtyp) für den nichtprivilegierten Anwender nicht erlaubt.
X'xx' [2]	X'40'	X'0020'	Liste mit den bereitgestellten Gerätetypcodes, VSNs, mnemotechnischen Namen usw. ist nicht zugewiesen.

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'xx' [2]	X'40'	X'0021'	Liste mit den bereitgestellten Gerätetypcodes, VSNs, mnemotechnischen Namen usw. ist nicht auf Wortgrenze ausgerichtet.
X'xx' [2]	X'40'	X'0022'	In einer Liste wurden mehr Gerätetypcodes, VSNs, mnemotechnischen Namen usw. angegeben als erlaubt sind.
X'xx' [2]	X'40'	X'0023'	Versionsnummer eines Layouts ist unbekannt.
X'00'	X'80'	X'0040'	Zurzeit kein Speicher verfügbar, es wurde kein Ausgabebereich erstellt.

[1] Die Subreturncodes zum Maincode X'0008' sind additiv: Bei einem Makroaufruf können mehrere von ihnen gleichzeitig auftreten.

[2] X'xx' bezeichnet den Typ des Ausgabesatzes (Recordtyp), der zum Returncode führte. Die für xx möglichen Werte haben die folgenden Bedeutungen:

01 TASK-Ausgabesatz
02 GLOBAL-Ausgabesatz
03 DEVICE-Ausgabesatz
04 TAPE-Ausgabesatz
05 DISC-Ausgabesatz
06 CONFIG-Ausgabesatz
07 DVQ-Ausgabesatz
08 SUMMARY-Ausgabesatz
09 TYPTASK-Ausgabesatz
0A UNMONIT-Ausgabesatz
0B DRV-Ausgabesatz
0C DEPOT-Ausgabesatz
0D LOC-Ausgabesatz

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Das aufrufende Programm wird beendet, wenn folgende Fehler auftreten:
– Der Datenbereich ist dem Aufrufer nicht zugewiesen.
– Der Datenbereich ist nicht auf Wortgrenze ausgerichtet.
– Der Datenbereich ist gegen Schreibzugriff geschützt.

Layout der DSECT für den Output-Control-Record (RECORD=HEADER)

```
          NKDINF MF=D,RECORD=HEADER
1               *,NKDINF VERSION 500
1         #INTF INTNAME=NKDINF,REFTYPE=REQUEST,INTCOMP=1
1 *
1 *       GENERATION OF OUTPUT-LAYOUTS
1 *
1 *
1 *    C/DSECT FOR OUTPUT-CONTROL RECORD
1 *
1         MFCHK MF=D,PREFIX=N,SUPPORT=(C,D),MACID=KDI,                  C
1               DMACID=KDO,DNAME=KDOENT
2 NKDOENT  DSECT ,
2               *,##### PREFIX=N, MACID=KDO #####
1 NKDOENTR DS    OF              1 OUTPUT_CONTROL_RECORD
1 NKDOBLEN DS    F                 2 LENGTH_OF_BUFFER_OBTAINED
1 NKDOSBUF DS    OF                2 SUBBUFFER_DESCRIPTION
1 NKDOTRBP DS    A                   3 PTR_TO_FIRST_TASK_RECORD
1 NKDOTRB# DS    H                   3 #_OF_TASK_RECORDS
1 NKDOTRBL DS    H                   3 LENGTH_OF_TASK_RECORD
1 NKDOTRBR DS    X                   3 RESULT_FOR_TASK_RECORDS SET
1 NKDOOKRO EQU   X'00'                 (OK_RECORDS_OUTPUTED
1 NKDOOKPT EQU   X'02'                  OK_RECORDS_PARTIAL_OUTPUT
1 NKDOOKRN EQU   X'04'                  OK_RECORDS_NOT_REQUESTED
1 NKDOOKRE EQU   X'08'                  OK_NO_RECORDS_EXISTS
1 NKDONOTA EQU   X'0C'                  SCOPE_HIDES_REQUESTED_RECORDS
1 NKDOOKMN EQU   X'0E'                  OK_NO_MN_DUE_TO_EXTENDED_MN
1 NKDODRBD DS    OF                2 DEVICE_RECORD_BUFFER_DESCRIPTION
1 NKDODRBP DS    A                   3 PTR_TO_FIRST_DEVICE_RECORD
1 NKDODRB# DS    H                   3 #_OF_DEVICE_RECORDS
1 NKDODRBL DS    H                   3 LENGTH_OF_DEVICE_RECORD
1 NKDODRBR DS    X                   3 RESULT_FOR_DEVICE_RECORDS SET
1 * &P.OKRO  EQU   X'00'                (OK_RECORDS_OUTPUTED
1 * &P.OKPT  EQU   X'02'                 OK_RECORDS_PARTIAL_OUTPUT
1 * &P.OKRN  EQU   X'04'                 OK_RECORDS_NOT_REQUESTED
1 * &P.OKRE  EQU   X'08'                 OK_NO_RECORDS_EXISTS
1 * &P.NOTA  EQU   X'0C'                 SCOPE_HIDES_REQUESTED_RECORDS
1 * &P.OKMN  EQU   X'0E'                 OK_NO_MN_DUE_TO_EXTENDED_MN
1 NKDOGRBD DS    OF                2 GLOBAL_RECORD_BUFFER_DESCRIPTION
1 NKDOGRBP DS    A                   3 PTR_TO_GLOBAL_RECORD
1 NKDOGRB# DS    H                   3 #_OF_GLOBAL_RECORDS "ALWAYS 1"
1 NKDOGRBL DS    H                   3 LENGTH_OF_GLOBAL_RECORD
1 NKDOGRBR DS    X                   3 RESULT_FOR_GLOBAL_RECORD SET
1 * &P.OKRO  EQU   X'00'                (OK_RECORDS_OUTPUTED
1 * &P.OKPT  EQU   X'02'                 OK_RECORDS_PARTIAL_OUTPUT
1 * &P.OKRN  EQU   X'04'                 OK_RECORDS_NOT_REQUESTED
1 * &P.OKRE  EQU   X'08'                 OK_NO_RECORDS_EXISTS
```

```
1 * &P.NOTA  EQU   X'0C'                SCOPE_HIDES_REQUESTED_RECORDS
1 NKDOCRBD DS    0F                   2 CONFIG_RECORD_BUFFER_DESCRIPTION
1 NKDOCRBP DS    A                      3 PTR_TO_FIRST_CONFIG_RECORD
1 NKDOCRB# DS    H                      3 #_OF_CONFIG_RECORDS
1 NKDOCRBL DS    H                      3 LENGTH_OF_CONFIG_RECORD
1 NKDOCRBR DS    X                      3 RESULT_FOR_CONFIG_RECORDS SET
1 * &P.OKRO  EQU   X'00'                   (OK_RECORDS_OUTPUTED
1 * &P.OKPT  EQU   X'02'                    OK_RECORDS_PARTIAL_OUTPUT
1 * &P.OKRN  EQU   X'04'                    OK_RECORDS_NOT_REQUESTED
1 * &P.OKRE  EQU   X'08'                    OK_NO_RECORDS_EXISTS
1 * &P.NOTA  EQU   X'0C'                    SCOPE_HIDES_REQUESTED_RECORDS
1 * &P.OKMN  EQU   X'0E'                    OK_NO_MN_DUE_TO_EXTENDED_MN
1 NKDOBRBD DS    0F                   2 TAPE_RECORD_BUFFER_DESCRIPTION
1 NKDOBRBP DS    A                      3 PTR_TO_FIRST_TAPE_RECORD
1 NKDOBRB# DS    H                      3 #_OF_TAPE_RECORDS
1 NKDOBRBL DS    H                      3 LENGTH_OF_TAPE_RECORD
1 NKDOBRBR DS    X                      3 RESULT_FOR_TAPE_RECORDS SET
1 * &P.OKRO  EQU   X'00'                   (OK_RECORDS_OUTPUTED
1 * &P.OKPT  EQU   X'02'                    OK_RECORDS_PARTIAL_OUTPUT
1 * &P.OKRN  EQU   X'04'                    OK_RECORDS_NOT_REQUESTED
1 * &P.OKRE  EQU   X'08'                    OK_NO_RECORDS_EXISTS
1 * &P.NOTA  EQU   X'0C'                    SCOPE_HIDES_REQUESTED_RECORDS
1 * &P.OKMN  EQU   X'0E'                    OK_NO_MN_DUE_TO_EXTENDED_MN
1 NKDOPRBD DS    0F                   2 DISC_RECORD_BUFFER_DESCRIPTION
1 NKDOPRBP DS    A                      3 PTR_TO_FIRST_DISC_RECORD
1 NKDOPRB# DS    H                      3 #_OF_DISC_RECORDS
1 NKDOPRBL DS    H                      3 LENGTH_OF_DISC_RECORD
1 NKDOPRBR DS    X                      3 RESULT_FOR_DISC_RECORDS SET
1 * &P.OKRO  EQU   X'00'                   (OK_RECORDS_OUTPUTED
1 * &P.OKPT  EQU   X'02'                    OK_RECORDS_PARTIAL_OUTPUT
1 * &P.OKRN  EQU   X'04'                    OK_RECORDS_NOT_REQUESTED
1 * &P.OKRE  EQU   X'08'                    OK_NO_RECORDS_EXISTS
1 * &P.NOTA  EQU   X'0C'                    SCOPE_HIDES_REQUESTED_RECORDS
1 * &P.OKMN  EQU   X'0E'                    OK_NO_MN_DUE_TO_EXTENDED_MN
1 NKDOQRBD DS    0F                   2 DVQ_RECORD_BUFFER_DESCRIPTION
1 NKDOQRBP DS    A                      3 PTR_TO_DVQ_RECORD
1 NKDOQRB# DS    H                      3 #_OF_DVQ_RECORDS "ALWAYS 1"
1 NKDOQRBL DS    H                      3 LENGTH_OF_DVQ_RECORD
1 NKDOQRBR DS    X                      3 RESULT_FOR_DVQ_RECORD SET
1 * &P.OKRO  EQU   X'00'                   (OK_RECORDS_OUTPUTED
1 * &P.OKPT  EQU   X'02'                    OK_RECORDS_PARTIAL_OUTPUT
1 * &P.OKRN  EQU   X'04'                    OK_RECORDS_NOT_REQUESTED
1 * &P.OKRE  EQU   X'08'                    OK_NO_RECORDS_EXISTS
1 * &P.NOTA  EQU   X'0C'                    SCOPE_HIDES_REQUESTED_RECORDS
1 * &P.OKMN  EQU   X'0E'                    OK_NO_MN_DUE_TO_EXTENDED_MN
1 NKDOSRBD DS    0F                   2 SUMMARY_RECORD_BUFFER_DESCRIPTION
1 NKDOSRBP DS    A                      3 PTR_TO_SUMMARY_RECORD
1 NKDOSRB# DS    H                      3 #_OF_SUMMARY_RECORDS
```

```
1 NKDOSRBL DS    H                  3 LENGTH_OF_SUMMARY_RECORD
1 NKDOSRBR DS    X                  3 RESULT_FOR_SUMMARY_RECORD SET
1 * &P.OKRO  EQU   X'00'                (OK_RECORDS_OUTPUTED
1 * &P.OKPT  EQU   X'02'                 OK_RECORDS_PARTIAL_OUTPUT
1 * &P.OKRN  EQU   X'04'                 OK_RECORDS_NOT_REQUESTED
1 * &P.OKRE  EQU   X'08'                 OK_NO_RECORDS_EXISTS
1 * &P.NOTA  EQU   X'0C'                 SCOPE_HIDES_REQUESTED_RECORDS
1 NKDOYRBD DS    0F               2 TYPTASK_RECORD_BUFFER_DESCRIPTION
1 NKDOYRBP DS    A                  3 PTR_TO_TYPTASK_RECORD
1 NKDOYRB# DS    H                  3 #_OF_TYPTASK_RECORDS
1 NKDOYRBL DS    H                  3 LENGTH_OF_TYPTASK_RECORD
1 NKDOYRBR DS    X                  3 RESULT_FOR_TYPTASK_RECORD SET
1 * &P.OKRO  EQU   X'00'                (OK_RECORDS_OUTPUTED
1 * &P.OKPT  EQU   X'02'                 OK_RECORDS_PARTIAL_OUTPUT
1 * &P.OKRN  EQU   X'04'                 OK_RECORDS_NOT_REQUESTED
1 * &P.OKRE  EQU   X'08'                 OK_NO_RECORDS_EXISTS
1 * &P.NOTA  EQU   X'0C'                 SCOPE_HIDES_REQUESTED_RECORDS

1 NKDOURBD DS    0F               2 UNMONIT_RECORD_BUFFER_DESCRIPTION

1 NKDOURBP DS    A                  3 PTR_TO_UNMONIT_RECORD
1 NKDOURB# DS    H                  3 #_OF_UNMONIT_RECORDS
1 NKDOURBL DS    H                  3 LENGTH_OF_UNMONIT_RECORD
1 NKDOURBR DS    X                  3 RESULT_FOR_UNMONIT_RECORD SET
1 * &P.OKRO  EQU   X'00'                (OK_RECORDS_OUTPUTED
1 * &P.OKPT  EQU   X'02'                 OK_RECORDS_PARTIAL_OUTPUT
1 * &P.OKRN  EQU   X'04'                 OK_RECORDS_NOT_REQUESTED
1 * &P.OKRE  EQU   X'08'                 OK_NO_RECORDS_EXISTS
1 * &P.NOTA  EQU   X'0C'                 SCOPE_HIDES_REQUESTED_RECORDS
1 NKDOVRBD DS    0F               2 DRV_RECORD_BUFFER_DESCRIPTION
1 NKDOVRBP DS    A                  3 PTR_TO_DRV_RECORD
1 NKDOVRB# DS    H                  3 #_OF_DRV_RECORDS
1 NKDOVRBL DS    H                  3 LENGTH_OF_DRV_RECORD
1 NKDOVRBR DS    X                  3 RESULT_FOR_DRV_RECORD SET
1 * &P.OKRO  EQU   X'00'                (OK_RECORDS_OUTPUTED
1 * &P.OKPT  EQU   X'02'                 OK_RECORDS_PARTIAL_OUTPUT
1 * &P.OKRN  EQU   X'04'                 OK_RECORDS_NOT_REQUESTED
1 * &P.OKRE  EQU   X'08'                 OK_NO_RECORDS_EXISTS
1 * &P.NOTA  EQU   X'0C'                 SCOPE_HIDES_REQUESTED_RECORDS
1 NKDOLRBD DS    0F               2 DEPOT_RECORD_BUFFER_DESCRIPTION
1 NKDOLRBP DS    A                  3 PTR_TO_DEPOT_RECORD
1 NKDOLRB# DS    H                  3 #_OF_DEPOT_RECORDS
1 NKDOLRBL DS    H                  3 LENGTH_OF_DEPOT_RECORD
1 NKDOLRBR DS    X                  3 RESULT_FOR_DEPOT_RECORD SET
1 * &P.OKRO  EQU   X'00'                (OK_RECORDS_OUTPUTED
1 * &P.OKPT  EQU   X'02'                 OK_RECORDS_PARTIAL_OUTPUT
1 * &P.OKRN  EQU   X'04'                 OK_RECORDS_NOT_REQUESTED
1 * &P.OKRE  EQU   X'08'                 OK_NO_RECORDS_EXISTS
```

```
1 * &P.NOTA  EQU   X'0C'                    SCOPE_HIDES_REQUESTED_RECORDS
1 NKDOARBD DS    OF                    2 LOCATION_REC_BUFFER_DESCRIPTION
1 NKDOARBP DS    A                       3 PTR_TO_LOCATION_RECORD
1 NKDOARB# DS    H                       3 #_OF_LOCATION_RECORDS
1 NKDOARBL DS    H                       3 LENGTH_OF_LOCATION_RECORD
1 NKDOARBR DS    X                       3 RESULT_FOR_LOCATION_RECORD SET
1 * &P.OKRO  EQU   X'00'                     (OK_RECORDS_OUTPUTED
1 * &P.OKPT  EQU   X'02'                      OK_RECORDS_PARTIAL_OUTPUT
1 * &P.OKRN  EQU   X'04'                      OK_RECORDS_NOT_REQUESTED
1 * &P.OKRE  EQU   X'08'                      OK_NO_RECORDS_EXISTS
1 * &P.NOTA  EQU   X'0C'                      SCOPE_HIDES_REQUESTED_RECORDS
1          DS    XL3                 RESERVED
1 NKDO#    EQU   *-NKDOENTR          LENGTH_OF_CONTROL_RECORD
1 NKDOLGTH EQU   NKDO#               OLD NAME OF LENGTH
```

NKGTYPE – Geräteinformationen ausgeben

Allgemeines

Anwendungsgebiet: Abfragen und Zugriff zu Listen und Tabellen; siehe Seite 158
Makrotyp: S-Typ, MF-Format **1**:
31-Bit-Schnittstelle: Standardform/E-/L-/D-Form; siehe Seite 29

Gerätetyp: Jedes Gerät ist einem Gerätetyp (DEVICE-Typ) zugeordnet. Die Geräte eines Gerätetyps besitzen gleiche Geräteeigenschaften. Ein Gerätetyp kann über seinen (abdruckbaren) Namen oder über eine Kurzbezeichnung, dem Gerätetypcode, angesprochen werden. Namen für Gerätetypen sind z.B HNC, D3435; Gerätetypcodes dafür X'6D', X'A5'.

Volumetyp: Der Volumetyp bezeichnet eine Kombination von (Geräte-) Eigenschaften, die für das Benutzen eines Volumes (Datenträgers) von Bedeutung sind, z.B. Schreibdichte bei Bändern. Gerätetypen, die den geforderten Eigenschaften genügen, sind einem Volumetyp zugeordnet; z.B. gehört der Gerätetyp LTO-U4 zum Volumetyp TAPE-U4 (Magnetbandkassette 896 Spur). Für Platten ist die Zuordnung von Gerätetypen zu Volumetypen eineindeutig (identische Namen). Ein Volumetyp kann über seinen (abdruckbaren) Namen oder über den Volumetypcode angesprochen werden. Namen sind z.B. TAPE-U4; Volumetypcode X'CE'.

Gerätefamilie: Eine Menge von Gerätetypen mit bestimmten Geräteeigenschaften ist einer Gerätefamilie (FAMILY-Typ) zugeordnet. Ein Gerätetyp gehört immer nur zu einer Gerätefamilie. Eine Gerätefamilie kann über den (abdruckbaren) Namen oder über eine Kurzbezeichnung, dem Familycode, angesprochen werden. Namen für Gerätefamilien sind z.B. DISK, MBK; Familycodes dafür X'A0', X'C0'.

Geräteklasse: Eine Menge von Gerätetypen ist einer Geräteklasse (CLASS-Typ) zugeordnet. Es gibt die drei Geräteklassen UR, TAPE und DISK (UR für UNIT-RECORD-Geräte). Geräteklassen können nur über ihren (abdruckbaren) Namen angesprochen werden.

In der „Gerätetyp-Tabelle“ (Handbuch „Systeminstallation“ [11]) sind die Namen von Gerätetypen und Gerätefamilien mit den zugehörigen Gerätetyp- und Familycodes enthalten.

Makrobeschreibung

Der Makro **NKGTYPE** informiert über Namen, Gerätetypcode, Geräteeigenschaften, Pfadadressen und Pfadeigenschaften eines Geräte- oder Volumetyps oder über Namen und Gerätetypcodes der Gerätetypen, die zu einer Gerätefamilie oder Geräteklasse gehören.
Der Umfang der auszugebenden Informationen kann mit dem Operanden INF=... gesteuert werden. Der Datenbereich beginnt mit dem Standardheader. Der Anwender kann sich eine DSECT zu dem In-/Output-Bereich generieren lassen. Die Größe der DSECT wird ebenfalls mit dem Operanden INF gesteuert.

Makroaufrufformat und Operandenbeschreibung

NKGTYPE
[p] [,MF=(E,..) / L / D]
,FORMAT=INTERNAL / PRINTABLE
,TYPE=UNKNOWN / DEVICE / VOLUME / FAMILY / CLASS
[,INTYP=adr]
,INF=STD / LIST / LOCLIST / VLIST / GEN / STD-LIST

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei MF=D kann ein Präfix (p = 1 Buchstabe), wie im Aufrufformat dargestellt, angegeben werden.

Fehlt der MF-Operand, so werden Parameterliste und Sprungfolge erzeugt. R1 zeigt dann nach Ausführung auf die Parameterliste.

FORMAT=
gibt an, ob der Name oder die Kurzbezeichnung angegeben ist.

INTERNAL
Die Kurzbezeichnung (Gerätetypcode/ Volumetypcode/Familycode) ist angegeben.

PRINTABLE
Der (abdruckbare) Name ist angegeben.

TYPE=
bezeichnet die Zugehörigkeit zu einem Gerätetyp, Volumetyp, einer Gerätefamilie oder Geräteklasse.

UNKNOWN
Die Zugehörigkeit ist nicht bekannt. Die Typlisten werden in der Reihenfolge Geräteklasse, Gerätefamilie, Volumetyp, Gerätetyp durchsucht (ebenfalls die Listen der entsprechenden Aliasnamen). Die Liste der Geräteklasse wird nur durchsucht, wenn der (abdruckbare) Name angegeben ist.

DEVICE
Es wird nur die Liste der Gerätetypen und die Liste der entsprechenden Aliasnamen durchsucht.

VOLUME
Es wird nur die Liste der Volumetypen und die Liste der entsprechenden Aliasnamen durchsucht.

FAMILY
Es wird nur die Liste der Gerätefamilien durchsucht.

CLASS
Es wird nur die Liste der Geräteklassen durchsucht.

INTYP=
bezeichnet das Feld, das den Namen oder die Kurzbezeichnung enthält. Wenn der Operand INTYP nicht spezifiziert wird, muss der Aufrufer das Inputfeld NKGDIPTI/NKGDIPPR im Datenbereich versorgen. Feldlänge:
– 8 Byte bei Angabe des (abdruckbaren) Namens. Eintrag linksbündig mit nachfolgenden Leerzeichen.
– 2 Byte bei Angabe des Gerätetyp- oder Familycodes. Eintrag linksbündig mit nachfolgenden Nullen.

adr
symbolische Adresse des Feldes

INF=
bestimmt den Umfang der auszugebenen Information.

STD
Die Standardausgabe wird generiert:
– Typliste, in der der Name oder die Kurzbezeichnung gefunden wurde;
– Geräteklasse (CLASS-Typ).
– Name und Kurzbezeichnung.
– Bei Angabe eines Gerätetyps wird zusätzlich eine Liste mit Gerätetypmerkmalen ausgegeben (Familycode, Aufzeichnungsdichte eines Magnetbandgeräts, ...).

LIST
Zusätzlich zur Standardausgabe wird eine Liste der korrespondierenden Gerätetypen ausgegeben:
– Bei Angabe eines Gerätetyps, Ausgabe einer Liste der korrespondierenden Volumetypen und umgekehrt.
– Bei Angabe eines Gerätefamilie oder Geräteklasse, Ausgabe einer Liste mit Namen und Gerätetypcodes, der zu der Gerätefamilie oder Geräteklasse gehörenden Gerätetypen.

LOCLIST
Zusätzlich zur Standardausgabe wird eine Liste der korrespondierenden Typen im angegebenen Lagerort ausgegeben. Der Name des Lagerorts ist in der Parameterliste explizit zu versorgen. Ist das Feld für den Lagerort nicht versorgt, werden die korrespondierenden Typen aus der Menge der Geräte ermittelt, die keinem Lagerort zugeordnet sind (Restpool).
Es ist nur möglich, einen Gerätetyp oder einen Volumetyp anzugeben.
Bei Angabe eines Gerätetyps (TYPE=DEVICE) erhält man die Liste der korrespondierenden Volumetypen, bei Angabe eines Volumetyps (TYPE=VOLUME) die der korrespondierenden Gerätetypen.

VLIST
Angabe nur für eine Geräteklasse sinnvoll.
Zusätzlich zur Standardausgabe wird eine Liste der zu der Geräteklasse gehörenden Volumetypen ausgegeben.

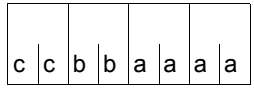
GEN
Angabe nur für einen Gerätetyp sinnvoll.
Zusätzlich zur Standardausgabe wird eine Liste der Generierungsmöglichkeiten (Pfadadressen und Pfadeigenschaften, wie Kanaltyp, Steuerungstyp, ...) ausgegeben.

STD-LIST
Angabe nur bei TYPE=CLASS sinnvoll.
Im Gegensatz zu INF=LIST wird nur die Liste der bis BS2000/OSD-BC V2.0 unterstützten Plattengeräte und `STDDISK` ausgegeben.
Dieser Wert wurde zur Unterstützung des SDF-Datentyps <device> eingeführt. Neue Platten sollen in Benutzerkommandos nur noch mit dem Standard-Plattentyp `STDDISK` angefordert werden.

Rückinformation und Fehleranzeigen

Register R1 enthält die Adresse des Datenbereichs.

R15:



Über die Ausführung des Makros NKGTYPE wird ein gegliederter Returncode (cc=Subcode2, bb=Subcode1, aaaa=Maincode) im Register R15 übergeben.

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Normale Ausführung, Eintrag gefunden.
X'00'	X'04'	X'0004'	Eintrag nicht gefunden.
X'00'	X'08'	X'0004'	Angabe für INF=... ist widersprüchlich zu der Angabe bei TYPE=... .
X'00'	X'10'	X'0004'	Der (abdruckbare) Name ist unvollständig oder nicht eindeutig angegeben.
X'00'	X'20'	X'0004'	Systemfehler.
X'00'	X'0C'	X'0004'	Ungültige Adresse des Datenbereichs.
X'00'	X'FF'	X'0004'	Operandenfehler.

Layout der DSECT für den Ein-/Ausgabebereich bei INF=STD

```
           NKGTYPE MF=D
1               *,VERSION 701
1          #INTF MACNAME=NKGTYPE,REFTYPE=REQUEST,MACVERS=701,            C
1                INTNAME=011.001,INTVERS=1
1          MFPRE DNAME=KGDTYP,MF=D,MACID=KGD,ALIGN=F,PREFIX=N,          C
1                DMACID=KGD
2 NKGDTYP  DSECT ,
2                *,##### PREFIX=N, MACID=KGD #####
1 NKGDPLS  DS    OF       DCL 1 NKGTYPE_PARAMETERLIST_START
1 NKGDFHDR FHDR  MF=(C,NKGD),EQUATES=NO
2 NKGDFHDR DS    0A
2 NKGDFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
2 NKGDIFID DS    0A              0   INTERFACE IDENTIFIER
2 NKGDFCTU DS    AL2             0   FUNCTION UNIT NUMBER
2 *                                  BIT 15    HEADER FLAG BIT,
2 *                                  MUST BE RESET UNTIL FURTHER NOTICE
2 *                                  BIT 14-12 UNUSED, MUST BE RESET
2 *                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
2 NKGDFCT  DS    AL1             2   FUNCTION NUMBER
2 NKGDFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
```

```
2 NKGDRET  DS    0A              4   GENERAL RETURN CODE
2 NKGDSRET DS    0AL2            4   SUB RETURN CODE
2 NKGDSR2  DS    AL1             4   SUB RETURN CODE 2
2 NKGDSR1  DS    AL1             5   SUB RETURN CODE 1
2 NKGDMRET DS    0AL2            6   MAIN RETURN CODE
2 NKGDMR2  DS    AL1             6   MAIN RETURN CODE 2
2 NKGDMR1  DS    AL1             7   MAIN RETURN CODE 1
2 NKGDFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1          SPACE 1
1 *        EQUATES FOR THE MAIN RETURN CODE
1          SPACE 1
1 NKGDSUCC EQU   X'0000'          SUCCESS : ENTRY FOUND
1 *                               OUTPUT AREA CONTAINS VALID DATA
1 NKGDPERR EQU   X'0004'          UNSUCCESS
1 *                               OUTPUTAREA IS SET TO ZERO (X'00')
1 *                               SEE SUB RETURN CODE 1 FOR DETAIL
1          SPACE 1
1 *        EQUATES FOR THE SUB RETURN CODE 1 (RR IN X'00RR0004)
1          SPACE 1
1 NKGDSENF EQU   X'04'            ENTRY NOT FOUND
1 NKGDICIT EQU   X'08'            UNCONSISTENCY BETWEEN INF AND TYPE P.
1 NKGDS1SY EQU   X'20'            SYSTEM ERROR   (RC-CLASS C)         215
1 NKGDSPIA EQU   X'10'            PRINTABLE INPUT IS AMBIGUOUS
1 NKGDSPER EQU   X'FF'            PARAMETER ERROR
1 NKGDNAP1 EQU   X'0C'            PARAMETER LIST OR INPUT-TYPE NOT
1 *                               ACCESSIBLE (THIS RC IS ONLY
1 *                               POSSIBLE FOR P1-USERS, FOR WHICH A
1 *                               ADDRESS VALIDATION IS DONE, THIS RC
1 *                               IS ONLY SET TO CALLERS R15)
1          SPACE 1
1 *        EQUATES FOR THE SUB RETURN CODE 2  (SS IN X'SS000000)      101
1          SPACE 1
1 *********************************
1 *        INPUT PARAMETER        *
1 *********************************
1 NKGDINP  DS    0F        2 INPUT_PARAMETER
1 NKGDTPES DS    X           3 TYPE_SET
1 NKGDDEVC EQU   C'D'           (DEVICE  = D,
1 NKGDVOLM EQU   C'V'            VOLUME  = V,
1 NKGDFAML EQU   C'F'            FAMILY  = F,
1 NKGDCLSS EQU   C'C'            CLASS   = C,
1 NKGDUNKN EQU   C'U'            UNKNOWN = U)
1 NKGDFRMT DS    X           3 FORMAT_SET
1 NKGDINTR EQU   X'01'           (INTERNAL  = 1,
1 NKGDPRNT EQU   X'02'            PRINTABLE = 2)
```

```
1 NKGDTYPE DS    X           3 INPUT_TYPE_SET
1 NKGDTSTD EQU   X'00'         (TYPE_EXPLIZIT_IN_PARAMLIST   = 0,
1 NKGDADDR EQU   X'01'          TYPE_ADDR_SPECIFIED          = 1,
1 NKGDREG  EQU   X'02'          NO MORE USED SINCE V11.2A )
1 NKGDINFS DS    X           3 INFORMATION_SET
1 NKGDISTD EQU   X'01'         (STANDARD                     = 1,
1 NKGDIGEN EQU   X'02'          GENERATION                   = 2,
1 NKGDILST EQU   X'04'          LIST_OF_CORRESPONDING_TYPES  = 4,
1 *I.ICHR  EQU   X'08'          DISK_CHARACTERISTICS NO LONGER SUPP.
1 NKGDVLST EQU   X'10'          LIST_OF_CLASS_SPEC_VOL_TYPES = 16,
1 NKGDSLST EQU   X'20'          Standard_List_of_CLASS       = 32,
1 NKGDLLST EQU   X'40'          LIST-OF-CORR-TYPES-IN-LOCATION=64 )
1          SPACE 1
1 NKGDIADR DS    A           3 ADDR(INPUT_TYPE)
1          ORG   NKGDIADR
1 NKGDIREG DS    A           3 NO MORE USED SINCE V11.2A
1 NKGDIPTI DS    XL2         3 INPUT_TYPE_INTERNAL_FORMAT
1          DS    0A
1 NKGDIPPR DS    CL8         3 INPUT_TYPE_PRINTABLE_FORMAT
1 NKGDILOC DS    CL8         3 INPUT_LOCATION
1          SPACE 2
1 *********************************
1 *        OUTPUT PARAMETER       *
1 *********************************
1 NKGDOUTP DS    0F        2 OUTPUT_PARAMETER
1 NKGDOPLI DS    CL1         3 OUTPUT_LIST_INDICATOR SET
1 NKGDOTTL EQU   C'D'           (ENTRY_FOUND_IN_DEVICE_TYPE_LIST=D,
1 NKGDOVTL EQU   C'V'            ENTRY_FOUND_IN_VOLUME_TYPE_LIST=V,
1 NKGDOFTL EQU   C'F'            ENTRY_FOUND_IN_FAMILY_TYPE_LIST=F,
1 NKGDOCTL EQU   C'C'            ENTRY_FOUND_IN_CLASS_TYPE_LIST =C)
1 NKGDSECL DS    XL1         3 SECOND_LIST_INDICATOR
1 NKGDSVTL EQU   X'01'          4 E._FOUND_ALSO_IN_VOL_TYPE_L = 1
1 NKGDSDTL EQU   X'02'          4 E._FOUND_ALSO_IN_DEV_TYPE_L = 2
1 NKGDSFML EQU   X'04'          4 E._FOUND_ALSO_IN_FAM_TYPE_L = 4
1 NKGDOPNI DS    XL1         3 NAME_INDICATOR
1 *                             /* TO TEST ON BIT LEVEL */
1 NKGDOINS EQU   X'01'          4 INPUT_NAME_IN_SHORT_FORMAT = 1
1 NKGDOANT EQU   X'02'          4 ALIAS_NAME_TRANSLATED      = 2
1 NKGDDCLS DS    X           3 DEVICE_CLASS SET
1 NKGDURD  EQU   C'U'              (UR_DEVICE  =U,
1 NKGDDISC EQU   C'D'               DISC_DEVICE=D,
1 NKGDTAPE EQU   C'T'               TAPE_DEVICE=T)
1          DS    X           3 RESERVED
1 NKGDDVTI DS    XL2         3 TYPE_INTERNAL_FORMAT
1 NKGDDVTP DS    CL8         3 TYPE_PRINTABLE_FORMAT
```

```
1 NKGDRESP DS    X           3 RESERVATION_POSSIBILITY_INDICATOR
1 NKGDRESN EQU   X'01'         4 RESERVATION_BY_TYPE_NOT_POSSIBLE = 1
1 NKGDALNP EQU   X'02'         4 ALLOCATION_BY_TYPE_NOT_POSSIBLE  = 2
1 NKGDRMNN EQU   X'04'         4 RESERVATION_BY_MNEM_NOT_POSSIBLE = 4
1 NKGDAMNN EQU   X'08'         4 ALLOCATION_BY_MNEM_NOT_POSSIBLE  = 8
1 NKGDATNA EQU   X'10'         4 ATTACH_NOT_ALLOWED               =16
1          DS    F           3 RESERVED
1          SPACE 2
1 ***********************************************************
1 *
1 *   VOLUME-TYPE-ATTRIBUTES
1 *   ONLY VALID IF NKGDOPLI=NKGDOVTL
1 *   (ENTRY_FOUND_IN_VOLUME_TYPE_LIST_
1 *
1 ***********************************************************
1 NKGDDNSF DS    X           3 DENSITY_FLAG
1 NKGDCORD EQU   X'01'         4 VOL_TYPE_CORRESPONDS_TO_ONE_DENSITY
1 NKGDVLEV DS    X           3 TAPE_LEVEL_INDICATOR
1 NKGDLV9  EQU   X'01'         4 9_LEVEL_TAPE
1 NKGDLV18 EQU   X'02'         4 MBK_Mode
1 NKGDLVV  EQU   X'04'         4 VIDEO_TAPE
1 NKGDLVSC EQU   X'08'         4 STREAMING_CARTRIGE_TAPE
1 NKGDITYP DS    XL2         3 VOLUME_TYPE_FOR_LABEL_INITIALISATION
1 NKGDVL#2 EQU   *             VOLUME_TYPE_DESCRIPTION_EXTENSION
1 NKGDVRCM EQU   *           3 VOLUME_TYPE_RECORDING_MODE
1 NKGDVTPM DS    X             4  VOLUME_TAPE_MODE ( /* 9_LEVEL_TAPE*/
1 *I.800   EQU   X'01'               DENSITY_800_BPI  = 1,
1 *I1600   EQU   X'02'               DENSITY_1600_BPI = 2,
1 *I6250   EQU   X'04'               DENSITY_6250_BPI = 4 )
1          ORG   NKGDVTPM
1 NKGDV18M DS    X            4   VOLUME_MBK_MODE  (/* 18_LEVEL_TAPE*/
1 *I.COMP  EQU   X'01'               DATA_COMPACTION        = 1
1 *I.256P  EQU   X'02'               256kB_blocks_possible  = 2
1 *I.MLTE  EQU   X'20'               LTO_encrypted          = 32
1 *I.MLTO  EQU   X'40'               MODE LTO               = 64 )
1 NKGDWORN EQU   X'80'               WORM-Medium            = 128 )
1 NKGDTRCT DS    CL2        3 DMS_TAPE_RECORDING_TECHNIQUE(
1 NKGDDLNR EQU   C'  '               NORMAL_DENSITY       ,
1 NKGDDLCM EQU   C'P '               COMPACTION           )
1 NKGDDHDI DS    C          3 DMS_HDR2_DENSITY_INDICATOR(
1 NKGDDH8  EQU   C'2'                HDR2_800_BPI         ,
1 NKGDDH16 EQU   C'3'                HDR2_1600_BPI        ,
1 NKGDDH62 EQU   C'4'                HDR2_6250_BPI        ,
1 NKGDDHDM EQU   C' '                HDR2_MBK             )
1 NKGDTCRS DS    X           3 TYPE_CONCERNED_BY_RESTRICTION
1 NKGDRCNW EQU   X'01'          CONCERNED_BY_RESTRICTION_NO_WRITE
1 NKGDLTOT DS    X           3 LTO_MEDIUM_TYPE
1          ORG   NKGDVL#2
```

```
1          SPACE 2
1 ********************************************************
1 *
1 *   GENERATION-UNDEPENDENT DEVICE-TYPE-ATTRIBUTES
1 *       ONLY VALID IF NKGDOPLI=NKGDOTTL
1 *       (ENTRY_FOUND_IN_DEVICE_TYPE_LIST)
1 *
1 ********************************************************
1 NKGDFAMD DS    0XL10       3 FAMILY_DESCRIPTION
1 NKGDFAMI DS    XL2           4 FAMILY_INTERNAL_FORMAT
1 NKGDFAMP DS    CL8           4 FAMILY_PRINTABLE_FORMAT
1 NKGDDEFI DS    X           3 DEFAULT_INTERRUPTION_SUBCLASS_CODE_SET
1 NKGDDEF0 EQU   X'00'
1 NKGDDEF1 EQU   X'01'
1 NKGDDEF2 EQU   X'02'
1 NKGDDEF3 EQU   X'03'
1 NKGDDEF4 EQU   X'04'
1 NKGDDEF5 EQU   X'05'
1 NKGDDEF6 EQU   X'06'
1 NKGDDEF7 EQU   X'07'
1 NKGDHWA2 DS    X           3 HARDWARE_ATTRIBUTES BYTE 2
1 *        EQUATES  CF BELOW
1 NKGDDSSP DS    0XL2        3 DISC_SUB_ATTRIBUTES
1 NKGDDMDE DS    X             4 DISC_MODE
1 NKGDMOVJ EQU   X'01'           5 MOVABLE  =1
1 NKGDFIXH EQU   X'02'           5 FIX_HEAD =2
1 NKGDSTRU DS    X             4 DISK_STRUCTUR_SET
1 NKGDSFBA EQU   X'01'           (FBA-DISK          =1,
1 NKGDSCKD EQU   X'02'            CKD-DISK          =2,
1 NKGDSPAD EQU   X'04'            PAD_DISK          =4,
1 NKGDSCHD EQU   X'08'            CHANNEL_DEPENDENT =8, &CTL-DEPENDENT
1 NKGDSECK EQU   X'10'            ECKD_DISK         =16,
1 NKGDSEMU EQU   X'20'            EMULATED_DISK     =32)
1          ORG   NKGDDSSP
1 NKGDTSSP EQU   *           3 TAPE_SUB_ATTRIBUTES
1 NKGDTMDE DS    X             4 TAPE_MODE  /* 9_LEVEL_TAPE */
1 NKGDL9M  EQU   NKGDTMDE
1 NKGD800  EQU   X'01'           5 No longer used
1 NKGD1600 EQU   X'02'           5 1600BPI_SUPPORTED        =2
1 NKGD6250 EQU   X'04'           5 6250BPI_SUPPORTED        =4
1          ORG   NKGDTMDE
1 NKGDL18M DS    X             4 MBK_MODE
1 NKGDCOMP EQU   X'01'           5 DATA_COMPATCION_SUPPORTED = 1
1 NKGDTPLV DS    X           3 TAPE_LEVEL_INDICATOR
1 *&I.LV9  EQU   X'01'         4 9_LEVEL_TAPE             = 1
1 *&I.LV18 EQU   X'02'         4 MBK_Mode                 = 2
1 *&I.LVV  EQU   X'04'         4 VIDEO_TAPE
1 *&I.LVSC EQU   X'08'         4 STREAMING_CARTRIGE_TAPE
```

```
1          ORG   NKGDDSSP
1 NKGDCNSP EQU   *           3 CONSOLE_SUB_ATTRIBUTES
1 NKGDCMDE DS    X             4 CONSOLE_MODE
1 NKGDCDIS EQU   X'01'           5 DISPLAY =1
1 NKGDCHDC EQU   X'02'           5 HARDCOPY=2
1          ORG   NKGDDSSP+L'NKGDDSSP
1 NKGDHWAT DS    X           3 HARDWARE_ATTRIBUTES
1 NKGDRERE EQU   X'01'         4 RESERVE_RELEASE_SUPPORTED = 1  (D)
1 NKGDSDNC EQU   X'02'         4 SET_DENSITY_NECCESSARY    = 2  (T)
1 NKGDSTRM EQU   X'04'         4 STREAMING_MODUS_POSSIBLE  = 4  (T)
1 NKGDDISP EQU   X'08'         4 DISPLAY                   = 8  (T,D)
1 NKGDBUFF EQU   X'10'         4 CONTROLLER_WITH_BUFFER    =16  (T,D)
1 NKGDPOSI EQU   X'20'         4 CTL_SUPPORTS_POSITIONING  =32  (T)
1 NKGDNRRV EQU   X'40'         4 NO_READ_REVERSE_SUPPORTED =64  (T)
1 NKGDSTKP EQU   X'80'         4 STACKER_POSSIBLE          =128 (T)
1 *  HWA2                    3 HARDWARE_ATTRIBUTES BYTE 2
1 NKGDSSDD EQU   X'01'         4 SOLID_STATE_DISC_DEVICE   = 1  (D)
1 NKGDRSTP EQU   X'02'         4 RANDOM_STACKER_POSSIBLE   = 2  (T)
1 NKGDFORI EQU   X'04'         4 FORMATTING_DURING_INIT_NECESSARY= 4(T)
1 NKGDGENF DS    X           3 GENERATION_FACILITIES
1 NKGDSICH EQU   X'01'         4 CFCS12_CHANNEL_TYPE =1
1 NKGDFJCH EQU   X'02'         4 CFCS3_CHANNEL_TYPE  =2
1 NKGDPADT DS    A           3 ADRESS_OF_ADAM_TRANSLATION
1 NKGDCBUF DS    F           3 CONTROLLER_BUFFER_SIZE (BYTES)
1 NKGDDCAP DS    F           3 MAXIMAL DISC CAPACITY (#HP'S PER VOL)
1 NKGD#CYL DS    H           3 MAXILAL NUMBER OF CYCLINDERS PER VOL
1 NKGDDFVT DS    XL2         3 DEFAULT_VOLUME_TYPE (DEFAULT_DENSITY)
1 NKGDURST DS    X           3 USE_RESTRICTION
1 NKGDRNWR EQU   X'01'         RESTRICTION_NO_WRITE_POSSIBLE
1 NKGDLTOL DS    X           3 LTO_LEVEL
1          DS    XL2         3 RESERVED
1 NKGDOPL1 EQU   *-NKGDOUTP
1 NKGDOPLN EQU   *-NKGDOUTP  LENGTH_OF_THE_OUTPUT_PARAMETERLIST
1          DS    OXL(1-(NKGDOPLN-60)*(NKGDOPLN-60))
1 NKGDPLLN EQU   *-NKGDPLS   TOTAL_LENGTH_OF_THE_PARAMETERLIST
```

Beispiel

Die zu der Gerätefamilie X'A0' (=Plattenspeichergeräte) gehörenden Gerätetypen sollen ausgegeben werden. Der Makroaufruf erfolgt in der E-/L-Form.

```
NKGTYPE  START
         PRINT NOGEN
         BALR  3,0
         USING *,3
         NKGTYPE MF=(E,NKGL)
END      TERM
NKGL     NKGTYPE MF=L,TYPE=FAMILY,INTYP=ADDR,INF=LIST
ADDR      DC    X'A000' ------------------------------------------------  (1)
         END
```

Ablaufprotokoll:

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,nkgtype), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,nkgtype)), -
//        test-support=*aid
%  ASS6011 ASSEMBLY TIME: 306 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 94 MSEC
//end
%  ASS6012 END OF ASSEMBH
/load-executable-program library=macexmp.lib,element-or-symbol=nkgtype, -
/     test-options=*aid
%  BLS0523 ELEMENT 'NKGTYPE', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'NKGTYPE', VERSION ' ' OF '<date> <time>' LOADED
/%on %any <%d %§(nkgl)-> %xl568>;%r ------------------------------------  (2)
*** TID: 00710070 *** TSN: 6LHZ ****************************************
CURRENT PC: 0000002C    CSECT: NKGTYPE  ********************************
V'00000030' = NKGTYPE  + #'00000030'
00000030 (00000030) 000B0101 00000000 C6010104 000002BA    ........F.......
00000040 (00000040) 00000000 00000000 00000000 00000000    ................
00000050 (00000050) 00000000 C60000C4 00A000C4 C9E2D240    ....F..D...DISK
00000060 (00000060) 40404003 00000000 00000000 00000000       .............
00000070 (00000070) 00000000 00000000 00000000 00000000    ................
00000080 (00000080) 00000000 00000000 00000000 00000000    ................
00000090 (00000090) 001A0000 8100C4F3 F4F3F460 F1F08200    ....a.D3434-10b.
000000A0 (000000A0) C4F3F4F3 F460F2F0 8300C4F3 F4F3F860    D3434-20c.D3438-
```

```
000000B0 (000000B0) F3F08400 C4F3F4F2 F1F160F4 8500C4F3    30d.D34211-4e.D3
000000C0 (000000C0) F4F0F940 40408600 C4F3F4F2 F1F160F2    409   f.D34211-2
000000D0 (000000D0) 8700C4F3 F4F3F460 F3F08800 C4F3F4F2    g.D3434-30h.D342
000000E0 (000000E0) F1F160F3 8900C4F3 F4F9F060 F3F08A00    11-3i.D3490-30..
000000F0 (000000F0) C4F3F4F9 F060F4F0 8B00C4F3 F4F2F1F1    D3490-40..D34211
00000100 (00000100) 60F58E00 C4F3F4F9 F2404040 8F00C4F3    -5..D3492   ..D3
00000110 (00000110) F4F7F560 F8C6A100 C4F3F4F3 F960F1F0    475-8F..D3439-10
00000120 (00000120) A200C4F3 F4F3F640 4040A300 C4F3F4F3    s.D3436   t.D343
00000130 (00000130) F7404040 A400C4F3 F4F3F860 F2F0A500    7   u.D3438-20v.
00000140 (00000140) C4F3F4F3 F5404040 A700C4F3 F4F9F060    D3435   x.D3490-
00000150 (00000150) F1F0AAF0 C4F3F4F0 F960C7E2 AB00C4F3    10.0D3409-GS..D3
00000160 (00000160) F4F7F540 4040AC00 C4F3F4F8 F0404040    475   ..D3480
00000170 (00000170) AD00C4F3 F4F8C540 4040AE00 C4F3F4F8    ..D348E   ..D348
00000180 (00000180) C6404040 AF00C4F3 F4F9F060 F2F0AA00    F   ..D3490-20..
00000190 (00000190) E2E3C4C4 C9E2D240 00000000 00000000    STDDISK ........
000001A0 (000001A0) 00000000 00000000 00000000 00000000    ................
          REPEATED LINES:    10
00000250 (00000250) 00000000 00000000 00000000 00000000    ................
00000260 (00000260) 00000000 00000000                      ........
```

(1) Eingabe des Familycodes für Plattenspeichergeräte.

(2) Der Ein-/Ausgabebereich wird mit der Testhilfe AID angezeigt.

Der Ein-/Ausgabebereich beginnt bei Adresse X'000034'.
Die Bytes 3 - 7 enthalten den Returncode X'00000000'.
Es folgen die Inputdaten (DSECT-Bereich „INPUT PARAMETER“):
X'C6': TYPE=FAMILY,
X'01': FORMAT=INTERNAL,
X'01': INTYP=ADR,
X'04': INF=LIST,
X'000002BA' ist die virtuelle Adresse des Feldes ADR.

Der Ausgabebereich (DSECT-Bereich „OUTPUT PARAMETER“) beginnt bei Adresse X'000054'. Es bedeuten:
X'C6': ‚Der Eintrag wurde in der Liste der Gerätefamilien gefunden.
X'C4': ‚Plattengerät.
X'A0': ‚Familycode.
X'C4C9E2D240404040': DISK = Name der Gerätefamilie.

Ab Adresse X'000090' steht die erweiterte Ausgabe für INF=LIST.
X'001A': Die Gerätetypliste enthält 26 Einträge.
Nach 2 reservierten Byte folgen hintereinander Gerätetypcode und (abdruckbarer) Name der Gerätetypen.

NSIINF – Systeminformationen ausgeben

Allgemeines

Anwendungsgebiet: Abfragen und Zugriff zu Listen und Tabellen; siehe Seite 158
Makrotyp: S-Typ, MF-Format **3**: D-/C-/E-/L-Form; siehe Seite 29

Makrobeschreibung

Der Makro **NSIINF** informiert über

CPU: Seriennummern und Identifikationen der verfügbaren CPUs
BS2000: Bezeichnung und Version des Betriebssystems, sowie Adressierungsmodus des Betriebssystems und mit der Generierung eingestellte Betriebssystemoptionen
Speicher: Größe des (physikalischen) Hauptspeichers
HSI: HSI-Basistyp
Server: Server-Typ (Modellreihe)
Live Migration: Anzahl der Live Migrations, die stattgefunden haben

Pro Makroaufruf kann nur eine Information abgefragt werden. Die Ausgabe der gewünschten Information erfolgt im Feld `<PREFIX><MACID>OUTP`. Datentyp und Länge des Ausgabefeldes ist von der jeweiligen Information abhängig und in der Tabelle auf der nächsten Seite dargestellt.

Makroaufrufformat und Operandenbeschreibung

```
NSIINF

INFO=BIGPGSIZ / BS2ID / CONFNAMX / CPUIDIPL / (CPUID,8) / (CPUID,16) / EPOCH / HSIASF / HSIBASE /
     HSILINE / HSIVM / IPLDTTM / IPLMODE / NEWMSIZE / MEMSIZE / OSDVERS / OSIOID /
     PAGESIZE / SPSUFFIX / SYSNAME / MIGCOUNT
,SRVUNIT=STD / INITIAL / CURRENT
,MF=D / C / E / L
[,PARAM=adr / (r)]
,PREFIX=N / p
,MACID=SII / macid
```

INFO=
bezeichnet die Information, die ausgegeben werden soll.
Zu Inhalt und Länge der gewünschten Information siehe die Tabelle auf Seite 722.

SRVUNIT =
Spezifiziert die Server Unit, deren Daten angezeigt werden sollen.
Die explizite Angabe dieses Operanden ist nur dann sinnvoll, wenn eine Live Migration stattgefunden hat und wenn die gewünschte Information von der Server Unit abhängt.

SRVUNIT = STD
Die derzeit gültige Einstellung der BS2000-Session soll verwendet werden.
Systemglobale Voreinstellung: INITIAL.

SRVUNIT = INITIAL
Server Unit, auf der der IPL ausgeführt wurde (IPL-Server).

SRVUNIT = CURRENT
Server Unit, auf der die BS2000-Session (evtl. nach Live Migration) derzeit abläuft.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Bei der C-Form oder D-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

Tabelle mit Erläuterungen zu den angeforderten Informationen

Bezeichnung	Länge in Byte	Erläuterung der ausgegebenen Werte
BIGPGSIZ	4	Größe einer „Big Page“ im Speicher (in Byte). Diese Information ist nur sinnvoll auf x86-Servern. Sonst wird sie mit dem entsprechender Returncode abgewiesen.
BS2ID		informiert über das laufende Betriebssystem. Die Information beinhaltet folgende Teile:
	8	<PREFIX> <MACID> BSNP: Programmname (8 Byte, linksbündig), z.B. 'BS2V190'
	10	<PREFIX> <MACID> BSVR: Versionsangabe im DOD-Format, z.B. 'V19.0A00pp', pp ist die PVLU-Information
	10	<PREFIX> <MACID> BSDT: Datum der Generierung des Betriebssystems im ISO4-Format (YYYY-MM-DD)
	8	<PREFIX> <MACID> BSTM: Zeitpunkt der Generierung des Betriebssystems im ISO4-Format (HH:MM:SS)
CONFNAMX	21	Server-Typ (Modellreihe) im neuen, erweiterten Format, z.B.: '7.500-␣S210-F␣␣␣␣␣␣␣␣' Ist der Server-Typ im System nicht eingetragen, so wird 7.500-7.000␣␣␣␣␣␣␣␣␣␣ ausgegeben. Die Serverbezeichnung gliedert sich in vier Abschnitte: Abschnitt 1: Byte 0- 4: Basistyp Abschnitt 2: Byte 6-10: Modellreihe Abschnitt 3: Byte 12-15: Modellkennzeichen Abschnitt 4: Byte 17-20: besondere Modelleigenschaften Das 5., 11. und 16. Byte trennen diese Abschnitte voneinander und enthalten stets das Zeichen '-' (Bindestrich). Wenn zu Abschnitt 4 keine Informationen vorliegen, enthalten Byte 16-20 Leerzeichen. *Hinweis* Das alte Format der Bezeichnung des Server-Typs kann den Bytes 6 - 13 entnommen werden (`SINF INFO=CONFNAME`).
(CPUID,n)	8*n (n=8 oder n=16)	Identifikationen der CPUs. Das i-te Element enthält die Identifikation der i-ten CPU in sedezimaler Darstellung, wie sie von der Hardware übergeben wird. Steht die i-te CPU nicht zur Verfügung, enthält das i-te Element binär Null (8-mal X'00'). I.A. ist die Elementnummer nicht mit der CPU-Adresse identisch.
CPUIDIPL	8	Identifikation der IPL CPU in sedezimaler Darstellung, wie sie von der Hardware übergeben wird. Siehe auch Beschreibung (CPUID,n)
EPOCH	1	Epoche für das TOD-Register (siehe Handbuch „Systembetreuung“ [10])

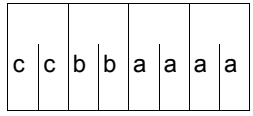
Bezeichnung	Länge in Byte	Erläuterung der ausgegebenen Werte
HSIASF	2	Zusätzliche Informationen über das HSI werden angezeigt. Folgende Werte sind möglich: AF: Das Betriebssystem läuft auf einem Server mit der Möglichkeit zur Erweiterung des virtuellen Adressraums. Damit ist der wahlweise Zugriff auf einen Programmraum und mehrere Datenräume möglich. Dieser Zugriff wird durch den Einsatz eines zusätzlichen Registersatzes (Zugriffsregister) ermöglicht (siehe Abschnitt „Erweiterung durch Datenräume“ auf Seite 61). NA: Es steht kein erweiterter Adressraum zur Verfügung.
HSIBASE	6	CFCS3: HSI-Basistyp = CFCS3 (/390-Server). X86: HSI-Basistyp = X86 (x86-Server).
HSILINE	2	Zusätzliche Informationen über das HSI werden angezeigt. I: Der HSI-Basistyp ist CFCS3; CPU-Typ = IX K: Der HSI-Basistyp ist X86; CPU-Typ = KM UD: Der HSI-Basistyp ist undefiniert
HSIVM	2	Informiert darüber, ob eine reale oder eine virtuelle Maschine vorliegt. Mögliche Werte: V2 Das Betriebssystem läuft auf einer virtuellen Maschine unter VM2000. NV Das Betriebssystem läuft auf einer realen Maschine.
IPLDTTM	10 8	informiert über: <PREFIX> <MACID> IPDA: Datum des Startup der laufenden Session im ISO4-Format (YYYY-MM-DD) <PREFIX> <MACID> IPTM: Zeitpunkt des Startup der laufenden Session im ISO4-Format (HH:MM:SS)
IPLMODE	1	informiert über den Modus der Systemeinleitung. Folgende Werte sind möglich: <PREFIX> <MACID> IPMD: Dialogmodus <PREFIX> <MACID> IPMA: Automatic- oder Fast-Modus
NEWMSIZE	4	Größe des für die Software nutzbaren (physikalischen) Hauptspeichers (Angabe in Megabyte)
MEMSIZE	4	Größe des für die Software nutzbaren (physikalischen) Hauptspeichers (Angabe in Byte). *Hinweis* Verwenden Sie vorzugsweise NEWMSIZE, da die Größe von MEMSIZE bei aktuellen Speichergrößen möglicherweise nicht ausreicht.
MIGCOUNT	4	informiert über die Anzahl Live Migrations, die stattgefunden haben. 0 bedeutet, dass in der Session noch keine Live Migration stattgefunden hat.

Bezeichnung	Länge in Byte	Erläuterung der ausgegebenen Werte
OSDVERS	10	Version von BS2000 im DOD-Format, z.B. 'V10.0A00pp' (pp ist die PVLU-Information)
OSIOID		informiert über die IO-Konfiguration des laufenden Betriebssytems. Die Information beinhaltet folgende Teile:
	8	<PREFIX> <MACID> IONP: Programmname (8 Byte, linksbündig), z.B. 'OSIOV190'
	10	<PREFIX> <MACID> IOVR: Versionsangabe im DOD-Format, z.B. 'V19.0A00pp' pp ist die PVLU-Information
	10	<PREFIX> <MACID> IODT: Datum der Generierung der IO-Konfiguration im ISO4-Format (YYYY-MM-DD)
	8	<PREFIX> <MACID> IOTM: Zeitpunkt der Generierung der IO-Konfiguration im ISO4-Format (HH:MM:SS)
PAGESIZE	4	Hardware-Seitengröße (Angabe in Kilobyte)
SPSUFFIX	15	informiert über den Ursprung der Systemparameter beim Systemstart *NONE aus <std-name> ohne Suffix *SYSTEM-NAME aus <std-name>.<sys-name>, wobei <sys-name> dem Systemnamen entspricht (siehe Parameter SYSNAME) *VM-NAME aus <std-name>.<vm-name>, wobei <vm-name> dem VM-Namen entspricht *IOCONF-ID-NAME aus <std-name>.<ioconf-name>, wobei <ioconf-name> dem IOCONF-Programmnamen entspricht *DIALOG die Systemparameter wurden vollständig bzw. teilweise über im Dialog angegebene Parameterdateien oder direkt über die Konsole eingegeben. *UNKNOWN Information über die Namensbildung liegt nicht vor (nur im Fehlerfall) <std-name> entspricht jeweils SYSPAR.BS2.vvv, wobei vvv = BS2000-Version ohne Punkt; also z.B. SYSPAR.BS2.190
SYSNAME	8	Systemname im Format <name 1..8>

Rückinformation und Fehleranzeigen

Die Ausgabe der gewünschten Information erfolgt im Feld <PREFIX><MACID>OUTP.

Standard-header:



Über die Ausführung des Makros NSIINF wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	**X'bb'**	**X'aaaa'**	**Erläuterung**
X'00'	X'00'	X'0000'	Funktion erfolgreich ausgeführt
X'01'	X'01'	X'0001'	Keine Aktion: Ungültiger Wert des Operanden INFO
X'02'	X'01'	X'0001'	Keine Aktion: Ungültiger Wert bei INFO=(CPUID,n)
X'06'	X'01'	X'0001'	Keine Aktion: Parameter für aktuelles HSI nicht anwendbar (BIGPGSIZ)

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Beispiel

```
NSIINF   START
         PRINT NOGEN
NSIINF   AMODE ANY
         GPARMOD 31
         BALR   3,0
         USING  *,3
*
NSIOPLST NSIINF MF=D,INFO=BS2ID ------------------------------------------  (1)
NSIINF   CSECT
         LA    4,NSILIST
         USING NSIOPLST,4
         NSIINF MF=E,PARAM=NSILIST
         MVC   MESSNAME,=C'NP'
         MVC   MESSTXT(L'NSIIBSNP),NSIIBSNP
         BAL   7,OUTPUT               Call output routine ---------->
         MVC   MESSNAME,=C'VR'
         MVC   MESSTXT,NSIIBSVR
         BAL   7,OUTPUT               Call output routine ---------->
         MVC   MESSNAME,=C'DT'
         MVC   MESSTXT,NSIIBSDT
         BAL   7,OUTPUT               Call output routine ---------->
         MVC   MESSNAME,=C'TM'
         MVC   MESSTXT(L'NSIIBSTM),NSIIBSTM
         BAL   7,OUTPUT               Call output routine ---------->
END      TERM
*
*        Output routine
*
OUTPUT   WROUT MESSAGE,END,PARMOD=31
         MVI   MESSTXT,C' '           Clear MESSTXT for next output
         MVC   MESSTXT+1(L'MESSTXT-1),MESSTXT
         BR    7                      Return ->
*
****  Definitions ****
         DS    0H
NSILIST  NSIINF MF=L,INFO=BS2ID ------------------------------------------  (2)
MESSAGE  DC    Y(ENDMESS-MESSAGE)     Record length
         DS    CL2                    Reserved
         DC    X'01'                  Print feed control character
         DC    C'NSIIBS'
MESSNAME DC    CL2' '                 Field indicator
         DC    C' = '
MESSTXT  DC    CL10' '                Contents
ENDMESS  EQU   *
         END
```

Ablaufprotokoll

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,nsiinf), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,nsiinf))
%  ASS6011 ASSEMBLY TIME: 368 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 85 MSEC
//end
%  ASS6012 END OF ASSEMBH
/start-executable-program library=macexmp.lib,element-or-symbol=nsiinf
%  BLS0523 ELEMENT 'NSIINF', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'NSIINF', VERSION ' ' OF '<date> <time>' LOADED
NSIIBSNP = I10BXS ------------------------------------------------------  (3)
NSIIBSVR = V19.0A00I1
NSIIBSDT = <date>
NSIIBSTM = <time>
```

(1) Es wird die DSECT für die Information BS2ID (Informationen über das laufende Betriebssystem) generiert.

(2) Der Datenbereich wird mit den Informationen versorgt.

(3) Folgende Informationen über das laufende Betriebssytem werden übergeben:
I10BXS: Programmname
V19.0A00I1: Versionsangabe (V19.0A00) und PVLU-Information (I1)
<date>: Generierungsdatum des Betriebssystems
<time>: Generierungszeitpunkt des Betriebssystems

NSIOPT – Systemparameter ausgeben

Allgemeines

Anwendungsgebiet: Abfragen und Zugriff zu Listen und Tabellen; siehe Seite 158
Makrotyp: S-Typ, MF-Format **3**: D-/C-/E-/L-Form; siehe Seite 29

Systemparameter legen bestimmte Optionen des individuellen Betriebssytems fest.

Systemparameter werden in die Startup-Parameterdatei eingetragen bzw. es werden die dafür vorgesehenen Standardwerte verwendet. Während des Systemlaufs können Systemparameter individuell verändert werden. Diese Änderungen müssen zusätzlich in die Startup-Parameterdatei eingetragen werden, wenn sie bei einem neuen Systemstart weiter gelten sollen.

Makrobeschreibung

Der Makro **NSIOPT** informiert über nicht-privilegierte Systemparameter. Wenn der Aufrufer das Privileg TSOS besitzt, dann werden auch die privilegierten Systemparameter ausgegeben.

Alle Systemparameter sind beim Kommando SHOW-SYSTEM-PARAMETERS im Handbuch „Kommandos“ [19] beschrieben.

Pro Makroaufruf kann nur ein Systemparameter abgefragt werden. Die Auswahl erfolgt durch den Operanden INFO. Die gewünschte Information wird in ein Ausgabefeld übertragen, das durch die Operanden FIELD und LENG bestimmt wird. Dabei ist der Datentyp abhängig vom jeweiligen Systemparameter.

Makroaufrufformat und Operandenbeschreibung

NSIOPT
[INFO='info',FIELD=adr,LENG=länge]
,MF=D / C / E / L
[,PARAM=adr / (r)]
,PREFIX=N / p
,MACID=SIO / macid

INFO=
bezeichnet die Information, die ausgegeben werden soll.

'info'
info = Kurzbezeichnung des Systemparameters. Die Zeichenkette info muss genau 8 Byte enthalten. Ist der Name des gewünschten Systemparameters kleiner als 8 Byte, müssen die übrigen Stellen mit Leerzeichen aufgefüllt werden.

FIELD=
bezeichnet die Adresse des Feldes, in das die Information ausgegeben wird. Die Feldlänge wird durch den Operanden LENG bestimmt. Die Information wird linksbündig eingetragen.

adr
symbolische Adresse des Feldes

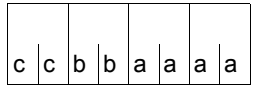
LENG=
beschreibt die Länge des Eintrags in das bei FIELD angegebene Feld. Die Länge ist dem Handbuch „Systembetreuung“ [10] zu entnehmen. Bei falscher Längenangabe erfolgt kein Eintrag.

länge
Länge in Byte, entsprechend der Liste; Angabe als Dezimalzahl
Für Systemparameter des Typs C erfolgt die Ausgabe auch in ein zu kleines Ausgabefeld, wenn vom Inhalt des Systemparameters nur Leerzeichen abgeschnitten werden.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Bei der C-Form oder D-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form zusätzlich eine Macid MACID angegeben werden.

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros NSIOPT wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Funktion erfolgreich ausgeführt.
X'01'	X'01'	X'0001'	Keine Aktion: Ungültiger Wert für den Operanden INFO.
X'02'	X'01'	X'0001'	Keine Aktion: Ungültiger Wert für den Operanden FIELD oder für die Operanden FIELD und LENG
X'03'	X'01'	X'0001'	Keine Aktion: Ungültiger Wert für den Operanden LENG, z.B. LENG nicht versorgt.
X'04'	X'01'	X'0001'	Keine Aktion: Der Operand LENG stimmt nicht mit der Länge des geforderten Systemparameters überein.
X'05'	X'01'	X'0001'	Keine Aktion: Informationen über den angeforderten Systemparameter stehen dem nichtprivilegierten Anwender nicht zur Verfügung.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Beispiel

```
  NSIOPT   START
  NSIOPT   AMODE ANY
           GPARMOD 31
           BALR   3,0
           USING  *,3
  *
  NSIOPLST NSIOPT MF=D ──────────────────────────────────────────────  (1)
1 *              NSIOPT, VERSION=101, DATE=900911                     101
1 NSIOPLST MFCHK MF=D,                                                101C
1                PREFIX=N,                                            101C
1                MACID=SIO,                                           101C
1                DMACID=SIO,                                          102C
1                DNAME=SIOPL,                                         101C
1                SUPPORT=(D,C,L,E),                                   101C
1                PARAM=,                                              101C
1                SVC=135                                              101
2 NSIOPLST DSECT ,
2                *,##### PREFIX=N, MACID=SIO #####
1 *                                                                   101
1           #INTF INTNAME=NSIOPT,REFTYPE=REQUEST,                     101C
1                 INTCOMP=1                                           101
1 *                                                                   101
1 NSIORCNE EQU   0                       NO ERROR                     101
1 NSIORCII EQU   1                       INFO INVALID                 101
1 NSIORCFI EQU   2                       FIELD INVALID                101
1 NSIORCLI EQU   3                       LENG INVALID                 101
1 NSIORCLS EQU   4                       LENG TOO SHORT               101
1 *                                                                   101
1 NSISOOO2     EQU  *                                                 101
1 NSIOFHDR FHDR  MF=(C,NSIO),EQUATES=NO                               101
2 NSIOFHDR DS    0A
2 NSIOFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
2 NSIOIFID DS    0A              0   INTERFACE IDENTIFIER
2 NSIOFCTU DS    AL2             0   FUNCTION UNIT NUMBER
2 *                                  BIT 15    HEADER FLAG BIT,
2 *                                  MUST BE RESET UNTIL FURTHER NOTICE
2 *                                  BIT 14-12 UNUSED, MUST BE RESET
2 *                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
2 NSIOFCT  DS    AL1             2   FUNCTION NUMBER
2 NSIOFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 NSIORET  DS    0A              4   GENERAL RETURN CODE
2 NSIOSRET DS    0AL2            4   SUB RETURN CODE
2 NSIOSR2  DS    AL1             4   SUB RETURN CODE 2
2 NSIOSR1  DS    AL1             5   SUB RETURN CODE 1
```

```
2 NSIOMRET DS    OAL2            6   MAIN RETURN CODE
2 NSIOMR2  DS    AL1             6   MAIN RETURN CODE 2
2 NSIOMR1  DS    AL1             7   MAIN RETURN CODE 1
2 NSIOFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 NSIOINFO DS    CL8                 INFO                          101
1 NSIOFILD DS    A                   POINTER                       101
1 NSIOLENG DS    H                   LENGTH                        101
1 NSIE0002     EQU  *                                              101
1 NSIO#        EQU  (NSIE0002-NSIS0002)                            101
           PRINT NOGEN
  NSIOPT   CSECT
           LA    4,NSILST
           USING NSIOPLST,4
           BAL   7,OUTPUT                Call output routine ----------->  (2)
           MVC   NSIOINFO,=C'SSMLGOF1' ------------------------------------  (3)
           MVC   NSIOLENG,=H'9'
           BAL   7,OUTPUT                Call output routine ----------->
           MVC   NSIOINFO,=C'ENCRYPT ' ------------------------------------  (4)
           MVC   NSIOLENG,=H'1'
           BAL   7,OUTPUT                Call output routine ----------->
  END      TERM
  *
  *        Output routine
  *
  OUTPUT   NSIOPT MF=E,PARAM=NSILST ------------------------------------  (5)
           MVC   PARNAME,NSIOINFO
           WROUT MESSAGE,END,PARMOD=31
2                *,@DCEO      999    921011   53531004
           MVI   OPTTXT,C' '             Clear OPTSTXT for next output
           MVC   OPTTXT+1(L'OPTTXT-1),OPTTXT
           BR    7                       Return ->
  *
  **** Definitions ****
           DS    0H
  NSILST   NSIOPT MF=L,INFO='BLKCTRL ',FIELD=OPTTXT,LENG=6 -------------  (6)
  MESSAGE  DC    Y(ENDMESS-MESSAGE)     Record length
           DS    CL2                    Reserved
           DC    X'01'                  Print feed control character
  PARNAME  DC    CL8' '                 Field indicator
           DC    C' = '
  OPTTXT   DC    CL10' '                Contents
  ENDMESS  EQU   *
           END
                 =C'SSMLGOF1'
                 =C'ENCRYPT '
```

Ablaufprotokoll

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,nsiopt), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,nsiopt))
%  ASS6011 ASSEMBLY TIME: 366 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 92 MSEC
//end
%  ASS6012 END OF ASSEMBH
/start-executable-program library=macexmp.lib,element-or-symbol=nsiopt
%  BLS0523 ELEMENT 'NSIOPT', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'NSIOPT', VERSION ' ' OF '<date> <time>' LOADED
BLKCTRL  = PAMKEY ------------------------------------------------------  (7)
SSMLGOF1 = REQ-SPOOL
ENCRYPT  = Y
```

(1) Generierung der DSECT zur Adressierung der auszugebenden Informationen.

(2) Erster Aufruf der Ausgaberoutine (5).

(3) Modifikation des Datenbereichs von **NSIOPT** für den zweiten Aufruf: Der Systemparameter SSMLGOF1 soll in der Länge 9 ausgegeben werden. Anschließend wird die Ausgaberoute (5) zum zweiten Mal aufgerufen.

(4) Modifikation des Datenbereichs von **NSIOPT** für den dritten Aufruf: Der Systemparameter ENCRYPT soll in der Länge 1 ausgegeben werden. Anschließend wird die Ausgaberoute (5) zum dritten Mal aufgerufen.

(5) Ausgaberoutine.
In ihr wird zunächst mit dem Makro **NSIOPT** der Wert eines Systemparameters ermittelt. Danach wird der ermittelte Wert mit **WROUT** auf SYSOUT ausgegeben. Die Ausgaberoutine wird in diesem Beispiel drei Mal aufgerufen.

(6) Aufruf des Makros **NSIOPT** mit MF=L zur Initialisierung des Datenbereichs. Das Dateiattribut BLKCTRL soll ausgegeben werden. Die Länge der auszugebenden Information beträgt 6 Byte.

(7) Ausgabe der ermittelten Systemparameter:

- Der Systemparameter BLKCTRL zeigt den Wert „PAMKEY“ als Standardwert für das Dateiattribut BLKCTRL an.
- Der Systemparameter SSMLGOF1 zeigt den Wert „REQ-SPOOL“ an, d.h. Spoolout-Aufträge werden immer angenommen.
- Der Systemparameter ENCRYPT zeigt den Wert „Y“ an. Kennwörter werden verschlüsselt in den Dateikatalog eingetragen.

OPCOM – Teilnahme an Intertaskkommunikation erklären

Allgemeines

Anwendungsgebiet: Intertaskkommunikation; siehe Seite 76
Kommunikation; siehe Seite 167
Makrotyp: O-Typ; siehe Seite 28

Makrobeschreibung

Mit dem Makro **OPCOM** erklärt der Anwender seine Teilnahme an der Intertaskkommunikation (ITC). Der Aufrufer gibt einen Namen an, der als ITC-Name zur Identifikation des Teilnehmers beim Senden oder Empfangen einer Nachricht dient. Der Name wird in die ITC-Teilnehmerliste eingetragen, wenn er nicht schon von einem anderen ITC-Teilnehmer benutzt wird (siehe Rückinformation).
Der Teilnehmer, der zuerst den Makro **OPCOM** aufruft, eröffnet damit auch implizit die Intertaskkommunikation (Einrichten einer Kommunikationstabelle).

Makroaufrufformat und Operandenbeschreibung

OPCOM
name / (1)

name
Name, den der Aufrufer als ITC-Name benutzen möchte. Der Name darf maximal 8 Zeichen lang sein. Zulässig sind alphanumerische Zeichen und Sonderzeichen mit Ausnahme von:

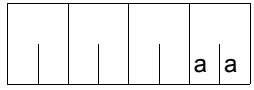
X'00' bis X'3F'
X'41' bis X'49'
X'51' bis X'59'
X'62' bis X'69'
X'70' bis X'79'
X'80' bis X'C0'
X'CA' bis X'D0'
X'DA' bis X'E1'
X'EA' bis X'EF'
X'FA' bis X'FE'

Das Zeichen X'40' darf man nur als letztes Zeichen von „name“ angeben.

(1)
Register R1 enthält den Adresswert des Feldes mit dem ITC-Namen. Feldlänge = 8 Byte.

Rückinformation und Fehleranzeigen

R15:



Über die Ausführung des Makros OPCOM wird im rechtsbündigen Byte des Registers R15 ein Returncode übergeben.

X'aa'	**Erläuterung**
X'00'	Die ITC-Teilnahme ist eröffnet.
X'04'	Fehler in der Operandenangabe. Die ITC-Teilnahme ist nicht eröffnet.
X'08'	ITC-Name ist schon vergeben. Die ITC-Teilnahme ist nicht eröffnet.
X'0C'	Kein Systemspeicher zur Eröffnung der ITC verfügbar oder die systeminterne Größe für Empfangswarteschlangen ist überschritten.
X'10'	Die ITC-Teilnahme wurde bereits erklärt.

OPSGEN – Steuern der S-Variablen-Generierung durch MIP

Allgemeines

Anwendungsgebiet: Meldungswesen; siehe Seite 165
Makrotyp: S-Typ; siehe Seite 29

Makrobeschreibung

Der Makro **OPSGEN** steuert Beginn oder Ende der S-Variablen-Generierung durch das Meldungssystem und übergibt den Variablennamen, unter dem die erstellte Variable abgelegt werden soll.

Makroaufrufformat und Operandenbeschreibung

OPSGEN
[SIGNAL=*START / *STOP] ,OUTPUT=*SYSOUT / *NONE ,MODE=*REPLACE / *EXTEND [,MSGVAR=adr / (r)] [,MSGVARL=länge / adr / (r)] [,MF=D / C / E / L / M] [,PARAM=adr / (r)] ,PREFIX=N / p ,MACID=MHG / macid

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

SIGNAL=
gibt an die laufende Task das Signal zum Beginnen oder Beenden der Generierung von strukturierten Meldungsausgaben in S-Variable. Die Ausgabe enthält den Meldungstext, den Meldungsschlüssel, die Einfügungen und den Antwortbereich.

***START**
Wird dieses Signal gegeben, beginnt MIP, strukturierte Meldungsausgaben zu generieren und in S-Variablen anzulegen.

***STOP**
Die Generierung von strukturierten Meldungsausgaben wird beendet.

OUTPUT=
bestimmt, ob Meldungen vom Programm nach SYSOUT ausgegeben werden sollen oder nicht.

***SYSOUT**
Die Meldungen werden nach SYSOUT ausgegeben.

***NONE**
Die Meldungen werden verworfen und nicht ausgegeben.

MSGVAR=
verweist auf die Adresse eines Bereiches, in dem der Name der benötigten Variable steht. Die Angabe dieses Operanden ist nur mit MF=M möglich.

adr
symbolische Adresse eines Bereiches, der wiederum die Adresse des Variablennamens enthält.

(r)
r = Register, das die Adresse des Bereichs enthält.

MSGVARL=
gibt die Länge (max. 255) des bei MSGVAR adressierten Bereiches an.

länge
anzugebende Länge des Feldes in Byte.

adr
symbolische Adresse eines 2 Byte langen Feldes, das die Länge enthält. Dieser Wert kann nur zusammen mit MF=M angegeben werden.

(r)
r = Register, das die Adresse des 2 Byte langen Feldes enthält.

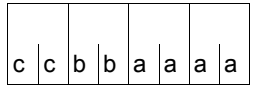
MODE=
gibt an, ob der bereits bestehende Inhalt der S-Variable überschrieben werden oder die neu generierten Meldungen an den alten Inhalt angehängt werden sollen.

***REPLACE**
Der Inhalt der S-Variablen wird vor dem Beschreiben mit den neu generierten Meldungen gelöscht.

***EXTEND**
Die neu generierten Meldungen werden an den bereits bestehenden Inhalt der S-Variable angehängt.

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros OPSGEN wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Funktion erfolgreich ausgeführt.
X'00'	X'01'	X'0002'	Fehler in der Parameterliste.
X'00'	X'01'	X'FFFF'	Funktion nicht bekannt.
X'00'	X'02'	X'FFFF'	Falsche Angabe für UNIT/FUNCTION im Standardheader.
X'00'	X'03'	X'FFFF'	Falsche Angabe für VERSION im Standardheader.
X'00'	X'20'	X'0003'	Probleme bei der Variablen-Generierung.
X'00'	X'20'	X'0004'	Fehler beim Ausführen der Funktion.
X'00'	X'40'	X'0001'	Operand SIGNAL=*STOP angegeben, ohne vorher die Variablen-Generierung mit SIGNAL=*START eingeschaltet zu haben.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können derTabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

PASS – Eine Sekunde warten

Allgemeines

Anwendungsgebiet: Starten, Unterbrechen und Beenden; siehe Seite 72
Makrotyp: O-Typ; siehe Seite 28

Makrobeschreibung

Der Makro **PASS** versetzt die Task 1 Sekunde lang in den Wartezustand.

Makroaufrufformat

PASS

Beispiel

```
PASS     START
         PRINT NOGEN
         BALR  3,0
         USING *,3
         GEPRT ,CPU                REMAINING PROGRAM TIME
         GDATE TOD=CLOCK           TIME OF DAY
         PASS                      WAIT ONE SECOND
         GDATE TOD=CLOCK1          TIME OF DAY
         GEPRT ,CPU1               REMAINING PROGRAM TIME
DTH1     TERM
CLOCK    DS    CL8
         DC    C' '
CLOCK1   DS    CL8
         DC    C' '
CPU      DS    CL6
         DC    C' '
CPU1     DS    CL6
         DC    C' '
         END
```

Ablaufprotokoll

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,pass), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,pass)), -
//        test-support=*aid
%  ASS6011 ASSEMBLY TIME: 322 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 78 MSEC
//end
%  ASS6012 END OF ASSEMBH
/load-executable-program library=macexmp.lib,element-or-symbol=pass, -
/     test-options=*aid
%  BLS0523 ELEMENT 'PASS', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'PASS', VERSION ' ' OF '<date <time>' LOADED
/%on %term<%d clock,clock1,cpu,cpu1>
/%r
*** TID: 005000D8 *** TSN: 2QSE ******************************************
**
SRC_REF:    54 SOURCE: PASS  PROC: PASS  ************************************
**
CLOCK           = |13:12:36| ------------------------------------------  (1)
CLOCK1          = |13:12:37|
CPU             = |022900| ------------------------------------------  (2)
CPU1            = |022900|
```

(1) Ausgabe der Uhrzeit vor und nach der Ausführung des Makros **PASS**. Es ist eine Sekunde vergangen.

(2) Ausgabe der CPU-Zeit vor und nach der Ausführung des Makros **PASS**. Es wurde keine CPU-Zeit verbraucht.

PINF – Globale Programminformationen ausgeben

Allgemeines

Anwendungsgebiet: Binden und Laden; siehe Seite 47
Makrotyp: S-Typ, MF-Format **2**: Standardform/C-/D-/L-/E-/M-Form; siehe Seite 29

Zum dynamischen Bindelader DBL siehe auch Handbuch „BLSSERV“ [4].

Makrobeschreibung

Der Makroaufruf **PINF** informiert den Benutzer über Programme, die mit den Kommandos LOAD-EXECUTABLE-PROGRAM oder START-EXECUTABLE-PROGRAM (bzw. LOAD-PROGRAM oder START-PROGRAM) geladen wurden. Folgende Informationen können abgefragt werden:

– der internen Programmname (SELECT=INTNAME),
– die interne Programmversion (SELECT=INTVERS),
– das Erzeugungsdatum des Programmes (SELECT=INTDATE),
– der Copyright-Name des Programmes (SELECT=COPRIGHT),
– der Name der Bibliothek, in der sich das Programm befand (SELECT=FILENAME),
– der Elementname des Programmes in der Bibliothek (SELECT=ELEMNAME),
– die Elementversion (SELECT=ELEMVERS),
– den Elementtyp (SELECT=ELEMTYPE),
– der Name, der im Ladeaufruf (bei START-PROGRAM bzw. LOAD-PROGRAM) angegeben wurde (SELECT=SPECNAME),
– einen Indikator, der anzeigt, ob das geladene Programm vom statischen Lader ELDE geladen wurde oder das erste Modul einer Ladeeinheit war und vom DBL geladen wurde (SELECT=LOADTYPE).

Das Ausgabefeld enthält alle Informationen in sedezimaler Form.

Makroaufrufformat und Operandenbeschreibung

PINF
SELECT=INTNAME / INTVERS / INTDATE / COPRIGHT / FILENAME / ELEMNAME / ELEMVERS / ELEMTYPE / SPECNAME / LOADTYPE ,VERSION=001 / 002 ,ADDR=adr / (r) [,LEN=integer] ,MF=S / C / D / E / L / M [,PARAM=adr / (r)] ,PREFIX=P / p ,MACID=BPI / macid

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

ADDR=adr
gibt die symbolische Adresse eines Feldes an, in dem der DBL die Informationen ablegen soll.

(r)
r = Register mit dem Adresswert adr. Angabe nur mit MF=M.

LEN=
gibt die Länge des bei ADDR angegebenen Ausgabefeldes in Byte an.
Wenn der Operand nicht angegeben wird, dann wird die Mindestlänge (4 Byte) verwendet.

integer
Die Mindestlänge ist 4 Byte. Werden bei SELECT mehrere Informationen angegeben, so muss die Gesamtlänge des Ausgabefeldes gleich der Summe aller Informationslängen (siehe Operand SELECT) sein.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der angegebenen Operanden PARAM, PREFIX und MACID siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

SELECT=
Legt die Art der Information fest, die ausgegeben wird. Anzugeben sind die symbolischen Namen der gewünschten Informationen. Gleichzeitig dürfen maximal 8 Informationen gefordert werden. Die folgenden Tabellen enthalten die Beschreibung der Teilinformationen mit symbolischen Namen und ihrer Länge.

Bedeutung der Ausgabe bei VERSION=001

In der Spalte Beschreibung bedeutet:

(D): Das Programm wurde durch den dynamischen Bindelader DBL geladen, die Information bezieht sich auf das erste geladene Modul (LLM oder Bindemodul) der Ladeeinheit.

(S): Das Programm wurde durch den statischen Lader ELDE geladen (Lademodul, Typ-C-Element einer Bibliothek), oder es handelt sich um ein PAM-LLM, das durch den dynamischen Bindelader DBL geladen wurde.

symb. Name	Länge (Byte)	Beschreibung
INTNAME	41	interner Programmname: (D): – interner Name des LLM oder der erste CSECT-Name des Bindemoduls, wenn es aus der EAM-Bindemoduldatei geladen wurde (S): – interner Name des Lademoduls oder des PAM-LLM
INTVERS	24	interne Programmversion: (D): – interne Version des LLM (S): – interne Version des Programms oder des PAM-LLM
INTDATE	10	Erzeugungsdatum des Programmes in der Form yyyy-mm-dd yyyy=Jahr, mm=Monat, dd=Tag (D): bei LLMs deren Erzeugungsdatum bei Bindemodulen das Ausführungsdatum des Type-R-Elementes bzw. des Elementes aus der EAM-Bindemoduldatei (S): Erzeugungsdatum des Lademoduls oder des PAM-LLM
COPRIGHT	64	Copyright-Name des Programmes (D): Copyright-Name des LLM (S): Copyright-Name des Lademoduls oder des PAM-LLM
FILENAME	54	(D): Name der Bibliothek (Programmbibliothek oder OML), beim Laden aus der EAM-Bindemoduldatei enthält das Feld Leerzeichen (S): Name der katalogisierten Datei (des Lademoduls oder des PAM-LLMs) oder der Name der Bibliothek, die das Element vom Typ C (das Lademodul) enthält

symb. Name	Länge (Byte)	Beschreibung
ELEMNAME	64	(D): bei LLMs der Typ-L-Elementname, bei Bindemodulen entweder: – der Typ-R-Elementname in der Programmbibliothek – der Elementname in einer OML oder – der erste CSECT-Name des Bindemoduls ,wenn er aus der EAM-Bindemoduldatei geladen wurde. (S): Name des Typ-C-Elementes in einer Programmbibliothek.
ELEMVERS	24	(D): bei LLMs die Version des Typ-L-Elements, bei Bindemodulen die Version des Typ-R-Elementes in der Programmbibliothek (S): die Version des Typ-C-Elementes in der Programmbibliothek
ELEMTYPE	8	Enthält einen Buchstaben, der den Elementtyp kennzeichnet und 7 Leerzeichen. (D): L = LLM, R = Bindemodul (S): C = Lademodul
SPECNAME	64	Enthält den Namen, der im Ladeaufruf (bei START-PROGRAM oder LOAD-PROGRAM) angegeben wurde: (D): das angegebene Symbol oder die Zeichenfolge „EAM OMF“ für die EAM-Bindemodulbibliothek (S): der Name der katalogisierten Datei (des Lademoduls oder des PAM-LLMs) oder der Name des Typ-C-Elementes in einer Programmbibliothek
LOADTYPE	1	Indikator für den Typ des geladenen Programmes: = 0 : (S) – das Programm wurde vom statischen Lader ELDE geladen oder es handelt sich um ein PAM-LLM. ≠ 0 : (D) – das Programm wurde durch den DBL geladen und die Informationen beziehen sich auf das erste Modul der Ladeeinheit

Bedeutung der Ausgabe bei VERSION=002

In der Spalte Beschreibung bedeutet:

(D): Das Programm wurde durch den dynamischen Bindelader DBL geladen, die Information bezieht sich auf das erste geladene Modul (LLM oder Bindemodul) der Ladeeinheit oder auf ein PAM-LLM.

(S): Das Programm wurde durch den statischen Lader ELDE geladen (Lademodul, Typ-C-Element einer Bibliothek).

symb. Name	Länge (Byte)	Beschreibung
INTNAME	41	interner Programmname: (D): – interner Name des (PAM-)LLM oder der erste CSECT-Name des Bindemoduls, wenn es aus der EAM-Bindemoduldatei geladen wurde (S): – interner Name des Lademoduls
INTVERS	24	interne Programmversion: (D): – interne Version des (PAM-)LLM (S): – interne Version des Programms
INTDATE	10	Erzeugungsdatum des Programmes in der Form yyyy-mm-dd yyyy=Jahr, mm=Monat, dd=Tag (D): bei (PAM-)LLMs deren Erzeugungsdatum bei Bindemodulen das Ausführungsdatum des Type-R-Elementes bzw. des Elementes aus der EAM-Bindemoduldatei (S): Erzeugungsdatum des Lademoduls
COPRIGHT	64	Copyright-Name des Programmes (D): Copyright-Name des (PAM-)LLM (S): Copyright-Name des Lademoduls
FILENAME	54	(D): Name der Bibliothek (Programmbibliothek oder OML) oder PAM-LLM-Dateiname, beim Laden aus der EAM-Bindemoduldatei enthält das Feld Leerzeichen (S): Name der katalogisierten Datei (des Lademoduls) oder der Name der Bibliothek, die das Element vom Typ C (das Lademodul) enthält
ELEMNAME	64	(D): bei LLMs der Typ-L-Elementname, bei Bindemodulen entweder: – der Typ-R-Elementname in der Programmbibliothek – der Elementname in einer OML oder – der erste CSECT-Name des Bindemoduls ,wenn er aus der EAM-Bindemoduldatei geladen wurde. (S): Name des Typ-C-Elementes in einer Programmbibliothek.
ELEMVERS	24	(D): bei LLMs die Version des Typ-L-Elements, bei Bindemodulen die Version des Typ-R-Elementes in der Programmbibliothek (S): die Version des Typ-C-Elementes in der Programmbibliothek

symb. Name	Länge (Byte)	Beschreibung
ELEMTYPE	8	Enthält einen Buchstaben, der den Elementtyp kennzeichnet und 7 Leerzeichen. (D): L = LLM, R = Bindemodul (S): C = Lademodul
SPECNAME	64	Enthält den Namen, der im Ladeaufruf (bei START-PROGRAM oder LOAD-PROGRAM) angegeben wurde: (D): das angegebene Symbol oder die Zeichenfolge „EAM OMF“ für die EAM-Bindemodulbibliothek oder der PAM-LLM-Dateiname (S): der Name der katalogisierten Datei (des Lademoduls) oder der Name des Typ-C-Elementes in einer Programmbibliothek
LOADTYPE	1	Indikator für den Typ des geladenen Programmes: = 0 : (S) – das Programm wurde vom statischen Lader ELDE geladen. ≠ 0 : (D) – das Programm wurde durch den DBL geladen und die Informationen beziehen sich auf das erste Modul der Ladeeinheit

VERSION=

Legt die gewünschte Makro-Version und damit das Layout der Ausgabe fest.

001

Version 1 des PINF-Makros wird verwendet. In diesem Fall wird bei der Ausgabe nicht zwischen Lademodulen und PAM-LLMs unterschieden.

002

Version 2 des PINF-Makros wird verwendet. Diese Angabe wird ab BLSSERV V2.5 unterstützt.

Die ausgegebene Information für PAM-LLMs unterscheidet sich von der Information für Lademodule.

Hinweise zum Makroaufruf

Eine dynamische Änderung des Datenbereichs wird normalerweise mit einem Makroaufruf der Form MF=M durchgeführt.

Wird die Änderung jedoch direkt mit symbolischen Namen von MF=D vorgenommen, sollte der Benutzer beachten, dass für den Makro-Prozessor

– die Einträge sequenziell ab der ersten Position erfolgen müssen

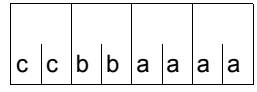
– Einträge, die den aktuell vorhandenen folgen, ignoriert werden

– ein Null-Eintrag in den aktuell vorhandenen Einträgen ignoriert, aber trotzdem als gültiger Eintrag gewertet wird.

Rückinformation und Fehleranzeigen

Die geforderten Informationen werden in dem Feld übergeben, das mit dem Operanden ADDR festgelegt wurde.

Standard-header:



Über die Ausführung des Makros PINF wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Der Makro wurde normal ausgeführt.
X'00'	X'40'	X'0001'	Die gewünschte Information ist noch nicht definiert (Warnung) oder wird nicht mehr unterstützt.
X'00'	X'01'	X'0010'	Das Ausgabefeld ist zu klein.
X'00'	X'01'	X'0020'	Bei SELECT wurde ein unbekannter symbolischer Name angegeben.
X'00'	X'01'	X'0070'	Der Speicherbereich für das Ausgabefeld ist nicht zugewiesen.
X'00'	X'20'	X'0090'	Interner Fehler.
X'00'	X'01'	X'0100'	Der SELECT-Operand fehlt.
X'00'	X'01'	X'0110'	Bei SELECT wurden mehr als 8 Informationen angegeben.
X'00'	X'01'	X'FFFF'	Die Funktion wird nicht mehr oder noch nicht unterstützt.
X'00'	X'02'	X'FFFF'	Die Funktion ist nicht verfügbar.
X'00'	X'03'	X'FFFF'	Die Version der Schnittstelle wird nicht unterstützt.

Beispiel

Das Programm PINFBSP ruft den Makro **PINF** auf und gibt folgende Informationen aus:

- den internen Programmnamen
- den Bibliotheksnamen
- den Namen des Elements in der Bibliothek
- den Elementtyp
- die Elementversion
- den im Ladeaufruf angegebenen Namen.

```
PINFBSP  CSECT
PINFBSP  AMODE ANY
PINFBSP  RMODE ANY
         BASR   10,0
         USING  *,10
         GPARMOD 31
         PRINT  GEN
         EJECT
*
***** PINF SELECT = INTNAME *****
*
INTNAM   MVC    MTEXT(24),=CL24'PINF SEL=INTNAME'
         MVC    MEXPECT(4),=XL4'00000000'
         PINF   SELECT=INTNAME,ADDR=ZONE,LEN=41 ------------------------  (1)
         L      15,4(1) ---------------------------------------------------  (2)
         CL     15,MEXPECT
         BE     WINTNAM ---------------------------------------------------  (3)
         BAS    12,MEREXP -------------------------------------------------  (4)
         B      FILNAM ----------------------------------------------------  (5)
WINTNAM  MVC    MAREA,MTEXT -----------------------------------------------  (6)
         MVC    ZAREA(41),ZONE
         BAS    12,WRINFO
*
***** PINF SELECT = FILENAME *****
*
FILNAM   MVC    MTEXT(24),=CL24'PINF SEL=FILENAME'
         MVC    MEXPECT(4),=XL4'00000000'
         PINF   SELECT=FILENAME,ADDR=ZONE,LEN=54 ------------------------  (7)
         L      15,4(1)
         CL     15,MEXPECT
         BE     WFILNAM
         BAS    12,MEREXP
         B      ELMT
WFILNAM  MVC    MAREA,MTEXT -----------------------------------------------  (8)
         MVC    ZAREA(54),ZONE
         BAS    12,WRINFO
*
```

```
***** PINF SELECT = ELEMNAME, ELEMVERS, ELEMTYPE *****
*
ELMT     MVC    MTEXT(24),=CL24'PINF SEL=ELEMxxxx'
         MVC    MEXPECT(4),=XL4'00000000'
         PINF   SELECT=(ELEMNAME,ELEMVERS,ELEMTYPE),ADDR=ZONE,LEN=96 ---  (9)
         L      15,4(1)
         CL     15,MEXPECT
         BE     WELMT
         BAS    12,MEREXP
         B      SPNAM
WELMT    MVC    MAREA(24),=CL24'PINF SEL=ELEMNAME' ------------------------ (10)
         MVC    ZAREA(64),ZONE
         BAS    12,WRINFO
*
         MVC    MAREA(24),=CL24'PINF SEL=ELEMVERS' ------------------------ (11)
         MVC    ZAREA(24),ZONE+64
         BAS    12,WRINFO
*
         MVC    MAREA(24),=CL24'PINF SEL=ELEMTYPE' ------------------------ (12)
         MVC    ZAREA(8),ZONE+88
         BAS    12,WRINFO
*
***** PINF SELECT = SPECNAME *****
*
SPNAM    MVC    MTEXT(24),=CL24'PINF SEL=SPECNAME'
         MVC    MEXPECT(4),=XL4'00000000'
         PINF   SELECT=SPECNAME,ADDR=ZONE,LEN=64 ------------------------- (13)
         L      15,4(1)
         CL     15,MEXPECT
         BE     WSPNAM
         BAS    12,MEREXP
         B      MTERM
WSPNAM   MVC    MAREA,MTEXT ----------------------------------------------- (14)
         MVC    ZAREA(64),ZONE
         BAS    12,WRINFO
*
MTERM    TERM ------------------------------------------------------------ (15)
MTERMD   TERM DUMP=Y ---------------------------------------------------- (16)
*
****  Definitions  ****
MEREXP   DS     0H
         MVC    MPACK(4),MEXPECT ------------------------------------------ (17)
         UNPK   MEXP+1(9),MPACK(5)
         NC     MEXP+1(8),=X'0F0F0F0F0F0F0F0F'
         TR     MEXP+1(8),MTAB
         MVI    MEXP,C'('
         MVC    MEXP+9(10),MCEXP
*
```

```
MEREAL   ST     15,MPACK ------------------------------------------------ (18)
         UNPK   MRS(9),MPACK(5)
         NC     MRS(8),=X'OFOFOFOFOFOFOFOF'
         TR     MRS(8),MTAB
         MVI    MRS+8,MBLANK
*
MEWROUT  DS     OH
         WROUT  MMSG,MTERMD ------------------------------------------------ (19)
         MVI    MTEXT,MBLANK ------------------------------------------------ (20)
         MVC    MTEXT+1(51),MTEXT
         BR     12
*
WRINFO   DS     OH
         WROUT  AREA,MTERMD ------------------------------------------------ (21)
         MVI    ZAREA,MBLANK
         MVC    ZAREA+1(95),ZAREA
         BR     12
*
**** DATA DEFINITION ****
*
AREA     DC     Y(LAREA)
         DC     CL3' '
MAREA    DC     CL24' '
ZAREA    DC     CL96' '
LAREA    EQU    *-AREA
MMSG     DC     Y(MMSGE)
         DC     C'    ==ERROR==    '
MTEXT    DC     CL24' '
MRS      DC     CL9' '
MEXP     DC     CL19' '
MMSGE    EQU    *-MMSG
MBLANK   EQU    X'40'
MCEXP    DC     C' EXPECTED)'
MPACK    DS     F
         DC     X'00'
MTAB     DC     C'0123456789ABCDEF'
MEXPECT  DC     F'0'
ZONE     DS     CL96
         END
```

(1) Der Makro **PINF** wird aufgerufen. Angefordert wird der interne Programmname, der mit einer maximalen Länge von 41 Byte im Feld ZONE abgelegt werden soll.

(2) Der Makro-Returncode, der im 2. Wort des Standardheader abgelegt ist, wird in Register R15 geladen. Danach wird der tatsächliche Returncode mit dem zu erwartenden Returncode X'00000000' verglichen.

(3) Bei fehlerfreier Makroausführung wird zur Ausgabe der von **PINF** gelieferten Information verzweigt.

(4) Bei fehlerhafter Makroausführung wird zur Fehlerbehandlungsroutine verzweigt.

(5) Nach der Fehlerbehandlung wird das Programm mit dem nächsten **PINF**-Aufruf fortgesetzt.

(6) Ausgabe des internen Programmnamens, der vom **PINF**-Makro im Feld `ZONE` abgelegt wurde.

(7) Der Makro **PINF** wird erneut aufgerufen. Angefordert wird der Name der Programmbibliothek, der mit einer maximalen Länge von 54 Byte im Feld `ZONE` abgelegt werden soll.

(8) Ausgabe des Bibliotheksnamens.

(9) Der Makro **PINF** wird wieder aufgerufen. Angefordert werden:
– der Name des Elementes in der Programmbibliothek,
– die Elementversion,
– der Elementtyp.
Für diese 3 Informationen sind insgesamt 96 Byte zu reservieren.

(10) Ausgabe des Elementnamens. Er beginnt am Anfang des Feldes `ZONE` und hat eine Länge von 64 Byte.

(11) Ausgabe der Elementversion. Sie beginnt an der symbolischen Adresse ZONE+64 und hat eine Länge von 24 Byte.

(12) Ausgabe des Elementtyps. Er beginnt an der symbolischen Adresse ZONE+88 und hat eine Länge von 8 Byte.

(13) Der Makro **PINF** wird erneut aufgerufen: Angefordert wird der Symbolname, der im Ladeaufruf (z.B. bei START-EXECUTABLE-PROGRAM) angegeben ist. Der Symbolname soll mit einer maximalen Länge von 64 Byte im Feld `ZONE` abgelegt werden.

(14) Ausgabe des Symbolnamens aus dem Ladeaufruf.

(15) Normales Programmende.

(16) Programmende mit DUMP, falls Fehler im **WROUT**-Makro auftreten.

(17) Aufbereitung des zu erwartenden Makro-Returncodes als Zeichenkette, die später ausgegeben wird.

(18) Aufbereitung des tatsächlichen Makro-Returncodes als Zeichenkette, die später ausgegeben wird.

(19) Ausgabe einer Fehlermeldung, falls der **PINF**-Makro einen Returncode ≠ X'00000000' geliefert hat. Folgende Meldung wird in diesem Fall ausgegeben:

```
==ERROR==     PINF SEL=zzzzzzzz      xxxxxxxx (00000000 EXPECTED)
```

Dabei ist:

zzzzzzzz	die ausgewählte Information
xxxxxxxx	der Wert des tatsächlichen Makro-Returncodes

(20) Die Felder MTEXT, MRS und MEXP werden mit Leerzeichen (X'40') belegt und das Programm wird fortgesetzt.

(21) Ausgabe der geforderten Programminformation. Danach wird das Feld ZAREA mit Leerzeichen (X'40') belegt und das Programm wird fortgesetzt.

Ablaufprotokoll

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,pinfbsp), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,pinfbsp))
%  ASS6011 ASSEMBLY TIME: 381 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 143 MSEC
//end
%  ASS6012 END OF ASSEMBH
/start-executable-program library=macexmp.lib,element-or-symbol=pinfbsp
%  BLS0523 ELEMENT 'PINFBSP', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'PINFBSP', VERSION ' ' OF '<date> <time>' LOADED
PINF SEL=INTNAME        PINFBSP ————————————————————————————————— (22)
PINF SEL=FILENAME       :2OSG:$QM212.MACEXMP.LIB
PINF SEL=ELEMNAME       PINFBSP
PINF SEL=ELEMVERS       ~
PINF SEL=ELEMTYPE       L
PINF SEL=SPECNAME       PINFBSP
```

(22) Ausgabe der angeforderten Informationen auf SYSOUT.
Der interne Programmname ist PINFBSP. Das Modul wurde aus der Programmbibliothek MACEXMP.LIB geholt. Es ist dort als LLM mit dem Namen PINFBSP gespeichert. Im Ladeaufruf (bei START-EXECUTABLE-PROGRAM) wurde der Name PINFBSP angegeben.

POSSIG – Ereignis signalisieren

Allgemeines

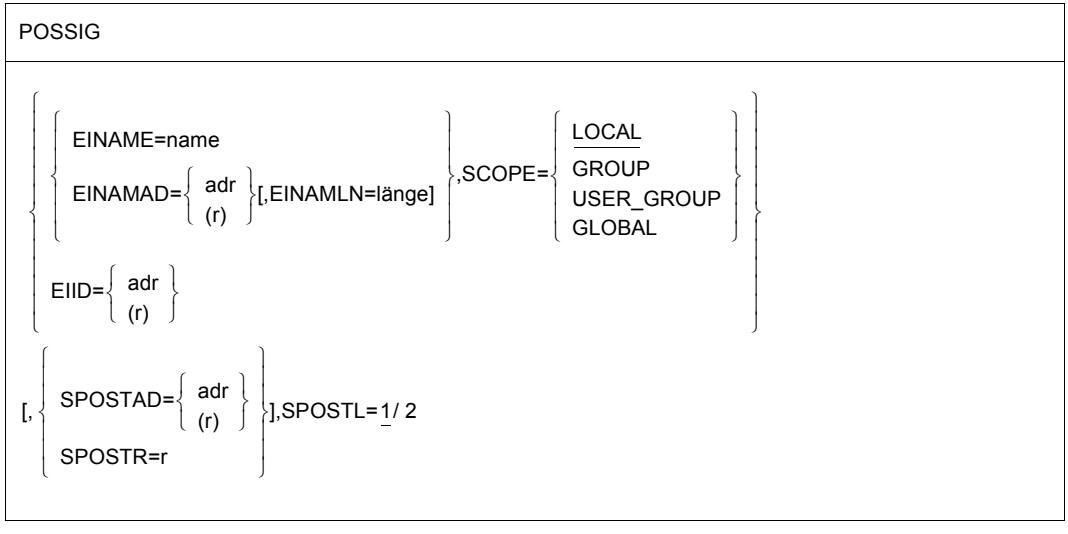
Anwendungsgebiet: Ereignissteuerung; siehe Seite 95
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

Bei Makrokettung müssen alle Makros der Kette dieselbe Schnittstelle benutzen. Bei Verwendung der 24-Bit-Schnittstelle wird für den Post Code nur ein 4 Byte langes Feld generiert. Bei Verwendung der 31-Bit-Schnittstelle kann der Post Code 4 oder 8 Byte lang sein.

Makrobeschreibung

Der Makroaufruf dient dazu, das Eintreten eines Ereignisses der Ereigniskennung anzuzeigen. Die Ereigniskennung muss zuvor der Task des Aufrufers zugeordnet worden sein (**ENAEI**-Makroaufruf).
Das den **POSSIG**-Aufruf gebende Programm wird immer fortgesetzt. Es kann auch einen Contingency-Prozess spezifizieren, der aktiviert wird, sobald der POSSIG-Aufruf („Sende Signal“) einen SOLSIG-Aufruf („Anfordern eines Signals“) befriedigt hat oder sobald eine vorgegebene Zeitperiode verstrichen ist. Dieser Contingency-Prozess übergibt eine Information, ob das Signal innerhalb einer Zeitperiode benutzt oder nicht benutzt wurde.
Der Contingency-Prozess muss zuvor definiert worden sein (**ENACO**-Makroaufruf).

Makroaufrufformat und Operandenbeschreibung



POSSIG (Fortsetzung)
[,LIFETIM=sec / (r)]
,CONTINU=NO / YES / SOLSIG
[,COID=adr / (r)]
[,COMAD=adr / (r)]
[,PARMOD=24 / 31]
[,MF=L / (E,..)]

EINAME=name
gibt den Namen der Ereigniskennung an, an die ein Ereignis gemeldet werden soll. Die Ereigniskennung muss zuvor mit dem Aufruf **ENAEI** definiert worden sein. Der Name der Ereigniskennung ist nur zusammen mit dem Geltungsbereich (SCOPE) eindeutig.

EINAMAD=
bezeichnet den Namen der Ereigniskennung. Diese Angabe ist nur zusammen mit dem Geltungsbereich (Operand SCOPE) eindeutig.

adr
symbolische Adresse des Feldes, das den Namen enthält.

(r)
r = Register, das die Adresse enthält.

EINAMLN=
gibt die Länge des Namens der Ereigniskennung in Byte an. Fehlt dieser Operand, so wird das Längenattribut des EINAMAD-Operandenwertes angenommen, wenn EINAMAD=adr angegeben ist; ist EINAMAD=(r) angegeben, so wird die maximale Länge (54 Byte) angenommen.

länge
Länge des Namens der Ereigniskennung.

SCOPE=
beschreibt den Geltungsbereich (Teilnehmerkreis) für die Ereigniskennung.

LOCAL
Die Ereigniskennung wird nur von der Task des Aufrufers benutzt.

GROUP
Teilnehmer sind alle Tasks unter der Benutzerkennung des Aufrufers.

USER_GROUP
Teilnehmer können alle Tasks sein, deren Benutzerkennungen der gleichen Benutzergruppe angehören wie die Benutzerkennung des einrichtenden Teilnehmers.
Der Operandenwert setzt die Existenz von Benutzergruppen voraus und kann daher nur angegeben werden, wenn die Funktionseinheit SRPM des Software-Produkts SECOS im System vorhanden ist. Vor einem Makroaufruf mit SCOPE=USER_GROUP muss deshalb mit dem Makro GETUGR (siehe Handbuch „SECOS“ [14]) geprüft werden, ob SRPM zur Verfügung steht; abhängig vom Ergebnis (Returncode) ist im Programm zu reagieren.

GLOBAL
Teilnehmer sind alle Tasks im System.

EIID=
bezeichnet die Kurzkennung der Ereigniskennung. Die Kurzkennung wird dem Benutzer durch den **ENAEI**-Makroaufruf zur Verfügung gestellt. Die Verwendung der Kurzkennung an Stelle des Namens zur Identifizierung der Ereigniskennung beschleunigt die Verarbeitung und ist eindeutig.

adr
symbolische Adresse eines 4 Byte langen Feldes, das die Kurzkennung enthält.

(r)
r = Register, das die Adresse des Feldes enthält.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wird PARMOD nicht spezifiziert, so erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB).

SPOSTAD=
bezeichnet das Feld mit der Post-Code-Information, die dem korrespondierenden Programm (das den **SOLSIG**-Makro aufruft) übergeben werden soll (siehe Abschnitt „Ereignisgesteuerte Verarbeitung (Eventing)“ auf Seite 95). Der Post Code ist 4 bzw. 8 Byte lang. Der Post Code wird auch dem Contingency-Prozess, falls vorhanden, übergeben (Operand COID). Der Post Code X'00000000' wird nicht übertragen.
Der Operand SPOSTR führt dieselbe Funktion bei schnellerer Verarbeitung aus.

adr
symbolische Adresse des Feldes, das die Post Code-Information enthält.
Länge = 4 oder 8 Byte.

(r)
r = Register mit dem Adresswert adr.

SPOSTR=
bezeichnet ein Register mit der Post-Code-Information, die dem korrespondierenden Programm (das den **SOLSIG**-Makro aufruft) übergeben werden soll. Bei 2 Worten (8 Byte) Post Code muss das dem angegebenen Register folgende Register (der Zahl nach) das 2. Wort des Post Codes enthalten. Post Code X'00000000' wird nicht übertragen.

(r)
r = Register mit dem Post Code. Register R0 und R1 dürfen nicht benutzt werden.

SPOSTL=
bezeichnet die Länge des Post Codes in Worten. Bei Verwendung der 24-Bit-Schnittstelle (PARMOD=24) kann nur SPOSTL=1 angegeben werden.

1
Der Post Code ist 1 Wort (4 Byte) lang.

2
Der Post Code ist 2 Worte lang.

LIFETIM=
Zeit, während der das gesendete Signal durch eine korrespondierende SOLSIG-Anforderung benutzt werden soll.
Falls ein Contingency-Prozess existert, teilt ihm der Ereignis-Informationscode mit, ob das Signal innerhalb der gesetzten Zeitperiode benutzt oder nicht benutzt wurde.

sec
Zeitangabe in Sekunden. $1 \leq sec \leq 43200$
Die Genauigkeit für die Bearbeitung liegt bei +10 Sekunden.
Voreinstellung: 600 sec.

(r)
r = Register, das die Angabe in Sekunden enthält.

CONTINU=
ermöglicht eine Verkettung des **POSSIG**-Makros mit weiteren **POSSIG**-Makros oder einem **SOLSIG**-Makro. Alle Makros der Kette müssen dasselbe Schnittstellenformat benutzen.

NO
Kein weiterer **POSSIG**- oder **SOLSIG**-Makro folgt unmittelbar auf den **POSSIG**-Makro.

YES
Ein weiterer **POSSIG**-Makro folgt unmittelbar nach dem **POSSIG**-Makro.

SOLSIG
Ein **SOLSIG**-Makro folgt unmittelbar auf den **POSSIG**-Makro.

Operanden für die Angabe eines Contingency-Prozesses

COID=
bezeichnet die Kurzkennung des Contingency-Prozesses. Die Kurzkennung wird dem Anwender bei Aufruf des Makros **ENACO** übergeben.

adr
symbolische Adresse des Feldes mit der Kurzkennung.

(r)
r = Register mit dem Adresswert adr.

COMAD=
bezeichnet eine Contingency-Mitteilung. Diese Contingency-Mitteilung wird dem Contingency-Prozess übergeben (Register R1). Eine hier gegebene Contingency-Mitteilung ersetzt eine eventuell bei der Definition der Contingency gegebene Mitteilung (siehe Abschnitt „Contingency-Prozesse“ auf Seite 111).

adr
symbolische Adresse eines Wortes, das eine Contingency-Mitteilung enthält.

(r)
r = Register, das die Adresse enthält.

Hinweise zum Makroaufruf

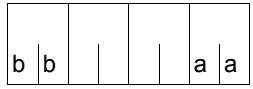
– Die POSSIG-Warteschlange einer Ereigniskennung kann nicht beliebig viele Anforderungen aufnehmen. Um das System zu schützen, ist die Anzahl der POSSIG-Anforderungen in der Warteschlange einer Ereigniskennung auf einen maschinenabhängigen Maximalwert begrenzt.

– Wenn ein Programm(paket) einen Contingency-Prozess definiert hat, der in SPL geschrieben ist, muss bei allen **ENACO**-, **SOLSIG**- und **POSSIG**-Aufrufen das Register R12 die Adresse des SPL-Program Managers enthalten.

Rückinformation und Fehleranzeigen

Während der Makrobearbeitung enthält Register R1 die Adresse der Operandenliste.

R15:



Über die Ausführung des Makros POSSIG wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'bb'	X'aa'	Erläuterung
X'00'	X'00'	Funktion ausgeführt: Das Signal wurde erfolgreich abgeschickt.
X'0C'	X'04'	Keine Aktion: Die vom System erstellte Ereigniskennung ist der Task des Aufrufers nicht zugeordnet.
X'10'	X'04'	Keine Aktion: Es wurden ungültige Operanden angegeben.
X'14'	X'04'	Keine Aktion: Ungültiger Name bzw. ungültige Kurzkennung; die spezifizierte Ereigniskennung existiert nicht.
X'18'	X'04'	Keine Aktion: Maximal erlaubte Anzahl (400) an Contingency-Prozessen pro Basisprozess überschritten.
X'24'	X'04'	Keine Aktion: Ungültige Kurzkennung des Contingency-Prozesses. Es existiert kein Contingency-Prozess mit dieser Identifikation.
X'28'	X'04'	Keine Aktion: Die maximale Anzahl der Anforderungen in der POSSIG-Queue wurde erreicht.

Bei Makroverkettung (Operand CONTINU) gilt:

aa = X'00' Alle Makroaufrufe der Kette waren erfolgreich.

aa = X'04' Ein Makro der Kette war nicht erfolgreich (alle Makros der Kette, die vor diesem Makro liegen, waren erfolgreich.) Die Kette wird an dieser Stelle abgebrochen. Die Fehlerursache zeigt der entsprechende Sekundärindikator an.

Die vorgeschriebene Belegung des Post Codes ist auf Seite 98 angegeben.

Die Bedeutung der Werte des Ereignis-Informationscodes (für Contingency-Prozesse) ist in Tabelle 8 auf Seite 118 beschrieben.

Beispiel

```
  POSSIG   START
           PRINT NOGEN
  POSSIG   AMODE ANY
           GPARMOD 31
1                *,MACRO: GPARMOD, VERSION: VER121
           BALR  5,0
           USING *,5
           ENAEI EINAME=EVENT,SCOPE=GROUP,EIIDRET=KKEV ————————————————  (1)
           POSSIG EIID=KKEV ———————————————————————————————————————————  (2)
           ST    15,RCFIELD1
           ENACO CONAME=CONT,COADAD=COANFAD,COIDRET=KKCO ———————————————  (3)
           POSSIG EIID=KKEV,COID=KKCO,LIFETIM=60 ——————————————————————  (4)
           ST    15,RCFIELD2
  LOOP     CLI   SWITCH,'0'
           BE    LOOP
           ST    2,ACKNO ———————————————————————————————————————————  (5)
           DISCO COID=KKCO
           ST    15,RCFIELD3
           DISEI EIID=KKEV
           ST    15,RCFIELD4
  DTH1     TERM
  COANF    BALR  6,0
           USING *,6
           CONTXT STACKR=(2),OWNR=(2),FUNCT=WRITE ——————————————————————  (6)
           MVI   SWITCH,'1'
           RETCO
  KKEV     DS    F
  KKCO     DS    F
  COANFAD  DC    A(COANF)
  SWITCH   DC    C'0'
  *
  ACKNO    DS    F
  RCFIELD1 DS    F
  RCFIELD2 DS    F
  RCFIELD3 DS    F
  RCFIELD4 DS    F
           END
```

Ablaufprotokoll

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,possig), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,possig)), -
//        test-support=*aid
%  ASS6011 ASSEMBLY TIME: 509 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 84 MSEC
//end
%  ASS6012 END OF ASSEMBH
/load-executable-program library=macexmp.lib,element-or-symbol=possig, -
/     test-options=*aid
%  BLS0523 ELEMENT 'POSSIG', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'POSSIG', VERSION ' ' OF '<date> <time>' LOADED
/%in dth1;%r
STOPPED AT LABEL: DTH1 , SRC_REF: 158, SOURCE: POSSIG , PROC: POSSIG
/%d %@(rcfield1) -> %x
*** TID: 005000D8 *** TSN: 2QSE ******************************************
CURRENT PC: 000000A6    CSECT: POSSIG  ************************************
V'000000FC' = POSSIG   + #'000000FC' ------------------------------------  (7)
000000FC (000000FC) 00000000                                 ....
/%d %@(rcfield2) -> %x
V'00000100' = POSSIG   + #'00000100' ------------------------------------  (8)
00000100 (00000100) 00000000                                 ....
/%d %@(quitt) -> %x
V'000000F8' = POSSIG   + #'000000F8' ------------------------------------  (9)
000000F8 (000000F8) 00000004                                 ....
/%d %@(rcfield3) -> %x, %@(rcfield4) -> %x ------------------------------ (10)
V'00000104' = POSSIG   + #'00000104'
00000104 (00000104) 04000000                                 ....
V'00000108' = POSSIG   + #'00000108'
00000108 (00000108) 04000000                                 ....
/%r
```

(1) Die Ereigniskennung EVENT wird definiert. Adresse der Kurzkennung: KKEV

(2) Mit **POSSIG** wird an die Ereignissteuerung ein Signal gegeben. Ein Contingency-Prozess als Quittung ist nicht angegeben. Da kein weiterer Teilnehmer das Signal mit **SOLSIG** anfordert, bleibt das Signal in der POSSIG-Warteschlange der Ereignissteuerung, bis die Wartezeit abgelaufen ist (Standardwert: 600 Sekunden).

(3) Der Contingency-Prozess COANF wird mit der Kurzkennungsadresse KKCO definiert (siehe **ENACO**-Makro).

(4) Ein zweites Signal wird an die Ereignissteuerung gegeben. Der Contingency-Prozess (Kurzkennungsadresse KKCO) soll nach 60 Sekunden gestartet werden, falls nicht schon früher ein **SOLSIG**-Aufruf seinen Start ausgelöst hat. Anschließend läuft das Programm in einer Warteschleife.

(5) Der Basisprozess speichert Register R2 (Ereignis-Informationscode) unter ACKNO ab und schließt die Contingency-Definition (**DISCO**) und die Ereignissteuerung (**DISEI**).

(6) Der Contingency-Prozess COANF ist gestartet. Der Ereignis-Informationscode in Register R2 gibt an, dass kein **SOLSIG**-Aufruf eingetroffen war. Mit dem Makro **CONTXT** wird er vom Register R2 des Contingency-Prozesses ins Register R2 des Basisprozesses übertragen (siehe **CONTXT**-Makro). Anschließend wird die Schleifenvariable geändert, um die Warteschleife zu öffnen, und der Contingency-Prozess wird mit dem Makro **RETCO** beendet.

(7) Rücksprungschalter nach dem ersten **POSSIG**-Aufruf (vgl.(2)): Das Signal wurde erfolgreich abgeschickt.

(8) Rücksprungschalter nach dem zweiten **POSSIG**-Aufruf (vgl.(4)): Das Signal wurde erfolgreich abgeschickt.

(9) Ereignis-Informationscode des Contingency-Prozesses: Das erwartete Ereignis trat innerhalb der Zeitperiode nicht ein. Weder Contingency-Mitteilung noch Post Code ist vorhanden.

(10) Rücksprungschalter nach dem **DISCO**- und nach dem **DISEI**-Aufruf:
Die Contingency-Definition bzw. die Ereigniskennung wurde gelöscht.

Weitere **Beispiele** enthält der Abschnitt „Ereignisgesteuerte Verarbeitung (Eventing)“ (Seite 107) und der Abschnitt „Contingency-Prozesse“ (Seite 119).

RDATA – Satz von SYSDTA (Datenstation) lesen

Allgemeines

Anwendungsgebiet:	Ein-/Ausgabe von Dateien und Datensätzen; siehe Seite 159 Verkehr mit Datenstationen; siehe Seite 164
Makrotyp:	S-Typ, MF-Format **1**: 24-Bit-Schnittstelle: Standardform/E-/L-Form 31-Bit-Schnittstelle: Standardform/E-/L-/C-/D-Form, siehe Seite 29

Stand der Beschreibung: TIAM V13.2A

Für die 31-Bit-Schnittstelle zu beachten:

– Bei Verwendung von MF=C/D werden für den Standardheader keine symbolischen Namen und Equates erzeugt. Bei dynamischer Versorgung des Datenbereichs sollten die Initialisierungswerte für den Standardheader aus einem mit MF=L erzeugten Datenbereich übernommen werden. Bei fehlerhafter Versorgung des Feldes für UNIT wird ein Userdump erzeugt.
– Im Standardheader wird kein Returncode übergeben.

Der Makro **CUPAB** generiert eine DSECT des Datenbereichs des **RDATA** für das 24-Bit-Schnittstellenformat.

Makrobeschreibung

Mit **RDATA** kann der nächste Datensatz von SYSDTA gelesen werden.
SYSDTA kann einem Element einer PLAM-Bibliothek, einer S-Variablen, einer katalogisierten SAM- oder ISAM-Datei, insbesondere der prozessführenden Datenstation zugeordnet sein.
Der Satz (im Fall der Datenstation: die Nachricht) wird in einen Bereich des Benutzerprogramms als Satz variabler Länge übertragen.

Nach einer Eingabe mit **RDATA** wird für die Datensichtstationen 8160, 9749, 975x und 9763 die Tastatur gesperrt, sodass bis zur nächsten Ausgabe keine weitere Eingabe möglich ist. Lediglich Kurztelegramme sind zugelassen.

Wird während der Lese-Operation ein „BREAK“ erkannt, wird der Befehlszähler auf den Anfang der Makroauflösung zurückgesetzt, sodass nach der Behandlung der Unterbrechung der Makroaufruf wiederholt wird.

Beim Ablauf des Makros werden im Fall von Format 1 die spezifizierten Operanden in einer Operandentabelle abgespeichert und die Anfangsadresse dieser Tabelle in Register R1 geladen. Im Fall von Format 2 wird die im Anwenderprogramm spezifizierte Tabelle verwendet.

Makroaufrufformat 1 und Operandenbeschreibung

```
RDATA

satz,fehler[,länge][,edit][,A]
,KEYOUT=N / Y
,KEYPOS=N / Y
,KEYLEN=N / Y
[{ ,MODE=COMP ,ITRSUP={NO|YES},ILINEND={NO|YES}
                ,ILCASE={NO|YES},IHDR={NO|YES}
                ,IGETBS={NO|YES}
 ,MODE=LINE ,ILCASE={NO|YES},IGETBS={NO|YES}
                ,IGETFC={NO|YES},IGETIC={NO|YES}
                ,ICFD={NO|YES}                  }]
,RC=OLD / NEW
[,VTSUCBA=adr]
[,TIMER=wert]
,PARMOD=24 / 31
[,MF=C / (C,pre) / (E,...) / (D,pre) / D / L]
```

satz
symbolische Adresse des Feldes, in das der einzulesende Datensatz übertragen wird. Das Feld beginnt mit dem Satzlängenfeld. Aufbau:
Byte 0-1: Länge des Satzes + 4 Byte Satzlängenfeld
Byte 2-3: reserviert
Byte 4-n: Datensatz.
Beispiel

```
SATZ     DS      OCL74
LÄNGE    DS      CL2
RESERV   DS      CL2
DATEN    DS      CL70
```

fehler
symbolische Adresse (Name) im Benutzerprogramm, zu der verzweigt wird:
– im Fehlerfall (Einlesefehler, Dateiende, ...)
– wenn der Operand A gesetzt ist.

Im Fehlerfall enthält Register R14 die Adresse des dem **RDATA**-Aufruf folgenden Befehls. Der Fehlercode wird im Register R15 übergeben.
31-Bit-Schnittstelle: Bei Angabe fehler = 0 (Adresse X'00..0) wird das Programm mit dem Befehl fortgesetzt, der dem **RDATA**-Aufruf folgt.

länge
Größe des Einlesebereichs einschließlich 4 Byte für das Satzlängenfeld. Die maximal zulässige Länge beträgt 32767 Byte. Fehlt der Operand, wird das Längenattribut des Einlesebereichs angenommen.

edit
Aufbereitungsfunktion (Edit-Options) für eine Eingabe-Nachricht von der Datenstation. Dieser Operand ist nicht erforderlich, wenn Standardfunktionen (alle Edit-Bits=0) verwendet werden, bei einer MODE-Angabe oder bei Nutzung des VTSU-Control-Blocks. Durch Direktangabe (X'xx') kann nur das 1. Edit-Byte für Eingabe auf die beim **CUPAB**-Makro beschriebene Bedeutung gesetzt werden.

Hinweis
Dieser Operand wird nur noch aus Kompatibilitätsgründen unterstützt.
Die Edit-Optionen sollten über MODE-Angaben (siehe Operand MODE) oder über den VTSU-Control-Block (siehe Operand VTSUCBA) gesteuert werden.

A
Das Benutzerprogramm wird von der anfänglichen Standardzuweisung und von jeder folgenden Zuweisung für SYSDTA verständigt. Diese Verständigung erfolgt über die Fehleradresse, sobald der Lesevorgang beendet ist.
Der Zuweisungsschlüssel wird wie folgt abgespeichert:
24-Bit-Schnittstelle: im linksbündigen Byte des Registers R15.
31-Bit-Schnittstelle: im Feld CURAIND des Datenbereichs.

KEYLEN=
bestimmt, ob die Länge des ISAM-Schlüssels abgespeichert werden soll oder nicht. Dieser Operand wird nur ausgewertet, wenn SYSDTA einer ISAM-Datei zugeordnet ist.

N
Die Länge des ISAM-Schlüssels wird nicht abgespeichert.

Y
Die um 1 verringerte Länge des ISAM-Schlüssels wird wie folgt abgespeichert:
– 24-Bit-Schnittstelle: Im rechtsbündigen Byte des Registers R0.
– 31-Bit-Schnittstelle: Im Feld CURKEYL der Operandenliste.

KEYOUT=
bestimmt, ob der Datensatz mit oder ohne ISAM-Schlüssel übergeben wird. Dieser Operand wird nur ausgewertet, wenn SYSDTA einer ISAM-Datei zugeordnet ist.

N
Der ISAM-Schlüssel wird nicht entfernt.

Y
Der ISAM-Schlüssel wird entfernt.

KEYPOS=
bestimmt, ob die Position des ISAM-Schlüssels abgespeichert werden soll oder nicht. Dieser Operand wird nur ausgewertet, wenn SYSDTA einer ISAM-Datei zugewiesen ist.

N
Die Position des ISAM-Schlüssels wird nicht abgespeichert.

Y
Die um 1 verringerte Position des ISAM-Schlüssels wird wie folgt abgespeichert:
24-Bit-Schnittstelle:
– In den beiden mittleren Byte des Registers R0, wenn KEYLEN=Y angegeben.
– In den beiden rechtsbündigen Byte des Registers R0, wenn KEYLEN=N angegeben.
31-Bit-Schnittstelle: Im Feld CURKEYP des Datenbereichs.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Bei der C-Form und D-Form des Makroaufrufs kann ein Präfix (pre = 1..3 Buchstaben), wie im Aufrufformat dargestellt, angegeben werden. Voreinstellung: pre = CUR

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

Hinweis

Dieselbe Bedeutung wie „Y“ hat auch die Angabe einer Zeichenfolge, die mit „Y“ beginnt. Jedes von „Y“ verschiedene Zeichen wird wie die Angabe von „N“ interpretiert. Jede Zeichenfolge, die nicht mit „Y“ beginnt, wirkt wie die Angabe von „N“ und verursacht eine MNOTE-Meldung.

Folgende Operanden werden nur ausgewertet, wenn SYSDTA der Datensichtstation zugeordnet ist. Die MODE-Angaben zusammen mit den Edit-Optionen werden nur noch aus Kompatibilitätsgründen unterstützt. Sie werden jetzt im VTSU-Control-Block (VTSUCB, siehe Makro **VTSUCB**) zusammengefasst.

MODE=COMP
Kompatibler Betriebsmodus. Vom Benutzerprogramm können über symbolische Operanden sämtliche Edit-Options entsprechend der Edit Option Tabelle (siehe Makro **CUPAB**) verwendet werden. Eventuell im Operanden edit gemachte Angaben werden ignoriert. Steuerzeichen im Text werden unbesehen dem Benutzerprogramm übergeben. Diese Betriebsweise ist zu früheren Versionen kompatibel (kompatibel unterstützte Datenstationen: 8103, 8110, 8150, 8152 und 8161).
Für die Geräte 3270, 8160, 8162, 9749, 975x wird dieser Modus wie MODE=LINE behandelt. Alle Edit Options außer ILCASE, IGETBS werden abgewiesen (RC: X'08').

MODE=LINE
Ist SYSDTA eine Datenstation, so wird diese als logische Zeilendatenstation behandelt. Die Nachricht kann mit logischen Steuerzeichen strukturiert sein (siehe Makro **VTCSET**). Der gerätespezifische Nachrichtenkopf wird nicht mitgeliefert. Die Operanden für die Nachrichtenaufbereitung werden ausgewertet.
Ist SYSDTA eine katalogisierte Datei (also keine Datenstation), so werden die Operanden zur Nachrichtenaufbereitung nicht ausgewertet. Zum Beispiel werden Kleinbuchstaben nicht in Großbuchstaben übersetzt, wenn SYSDTA eine katalogisierte Datei ist und ILCASE den Wert NO (Standardwert) hat. Logische Steuerzeichen in der Nachricht werden ebenfalls nicht ausgewertet.

ICFD=
gibt an, ob vertrauliche Daten geschützt werden sollen.

NO
Es sollen keine Vorkehrungen zum Schutz vertraulicher Daten getroffen werden.

YES
Die Eingabedaten sind vertraulich und sollen an der Datenstation unsichtbar bleiben. Dies erfolgt je nach Datenstation durch Dunkelsteuerung bzw. Löschen des Bildschirms oder durch Überschreiben der Eingabezeile bei Schreibstationen.

IGETBS=
bestimmt, ob „Underline“ (X'6D') in das Benutzerprogramm übertragen wird. Die Angabe des Operanden ist nur für 8103-Datensichtstationen sinnvoll.

NO
„Underline“ wird nicht in das Benutzerprogramm übertragen. Stattdessen wird vom System die Korrekturfunktion durchgeführt.

YES
Die Zeichen „Underline“ werden dem Benutzerprogramm übergeben. Es findet keine Auswertung durch das System statt.

IGETFC
bestimmt, ob ein Funktionstastencode übergeben wird.

NO
Es soll kein Funktionstastencode übergeben werden.

YES
Das 5. Byte des Einlesebereichs soll den normierten Funktionstastencode enthalten. Dieser identifiziert die zur Auslösung der Datenübertragung an der Datenstation betätigte Taste. Zur Tabelle der normierten Funktionstastencodes siehe Anhang, Seite 1193.

IGETIC=
legt fest, ob die Eingabequelle verändert werden soll.

NO
Die Eingabequelle soll nicht verändert werden.

YES
Die Eingabe soll vom angeschlossenen Ausweisleser erfolgen. Die Eingabedaten können nur aus der Ausweisinformation oder aus dem Kurztelegramm K14 bestehen. Diese Angabe ist nur bei den Datenstationen 8160, 9749, 975x und 3270 mit einem definierten Ausweisleser möglich (siehe auch Makro **TSTAT** TYPE=TCHAR). Der Operand IGETIC wird ignoriert, wenn gleichzeitig ICFD=YES angegeben wird oder wenn kein Ausweisleser angeschlossen ist.

IHDR=
gibt an, wie mit dem Nachrichtenkopf verfahren wird.

NO
Der Nachrichtenkopf wird nicht an das Benutzerprogramm übergeben.

YES
Der gesamte Nachrichtenkopf wird an das Benutzerprogramm übergeben. Bei 3270-Datenstationen besteht der Nachrichtenkopf aus dem Anwendungskennzeichen (AID-Byte) und der zwei Byte langen Schreibmarkenposition.

ILCASE=
legt fest, ob zwischen Klein- und Großschreibung unterschieden werden soll.

NO
Alle Kleinbuchstaben werden dem Benutzerprogramm als Großbuchstaben übergeben.

YES
Dem Benutzerprogramm werden auch Kleinbuchstaben übergeben.

ILINEND=
gibt an, wie mit den Wagenrücklauf-/Zeilenvorschubzeichen verfahren wird.

NO
Die Wagenrücklauf-/Zeilenvorschubzeichen werden dem Benutzerprogramm nicht übergeben.

YES
Die Zeichen Wagenrücklauf/Zeilenvorschub werden in das Benutzerprogramm übertragen.

ITRSUP=
gibt an, ob die Übersetzung von Gerätecode in EBCDIC unterdrückt werden soll.

NO
Die Übersetzung von Gerätecode in EBCDIC wird nicht unterdrückt. Das Benutzerprogramm erhält die Nachricht im EBCDI-Code.

Ausnahme
Der Nachrichtenkopf bei der Datensichtstation 8161 wird immer im Gerätecode geliefert.

YES
Die Übersetzung von Gerätecode in EBCDIC wird unterdrückt. Das Benutzerprogramm erhält also die Nachricht im Gerätecode. Die Angabe ist für die Datensichtstation 8161 unzulässig.

RC=
legt fest, wo der Returncode abgelegt wird.
Dieser Operand ist nur für eine 31-Bit-Schnittstelle zulässig.

OLD
Der Returncode wird im rechtsbündigen Byte des Register R15 abgelegt.

NEW
Der Returncode wird sowohl im Register R15 als auch im Standardheader abgelegt. Alle 4 Byte des Registers R15 sind für die Auswertung belegt. Ein 4-Byte-Returncode wird nur zurückgeliefert, wenn SYSDTA von der Datensichtstation liest. In allen anderen Fällen wird nur ein 1-Byte-Returncode zurückgeliefert, unabhängig vom Wert des Returncodes.

TIMER=wert
legt eine max. Wartezeit für die Eingabe fest. Falls innerhalb der festgelegten Wartezeit keine Eingabe erfolgt, wird ein Returncode zurückgegeben. Die Angabe dieses Operanden ist nur für die 31-Bit-Schnittstelle zulässig.

wert
Wartezeit von 10 bis 3600 Sekunden.
Standardwert ist UNLIMITED, d.h. es wird kein Timer benutzt.

VTSUCBA=adr
bestimmt die Adresse eines mit MF=L erzeugten VTSUCB.
Bei Benutzung des Operanden VTSUCBA wird der Operand MODE und die folgenden Edit-Optionen ignoriert (im Datenbereich wird ihr Wert auf X'FF' gesetzt). Das bedeutet, dass alle gewünschten Edit-Optionen im VTSUCB angegeben werden müssen.
Die Angabe dieses Operanden ist nur für die 31-Bit-Schnittstelle zulässig. Standardmäßig erfolgt keine Verwendung des VTSUCB.

adr
symbolische Adresse (Name) des VTSUCB.

Makroaufrufformat 2 und Operandenbeschreibung

RDATA
(1) [,PARMOD=24 / 31]

(1)
Register R1 enthält die Adresse des Datenbereichs (Liste ist auf Wortgrenze auszurichten).

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert. Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

Aufbau des Datenbereichs

Schnittstelle	Byte	Inhalt
31-Bit-Schnittstelle	0 - 7	Standardheader. Aufbau siehe Seite 43 Die Initialisierungswerte sind einem mit MF=L erzeugten Datenbereich zu entnehmen. Bei RC=OLD wird kein Returncode im Standardheader übergeben.
	8 - 11	Adresse, zu der im Fehlerfall verzweigt wird (Operand fehler).
	12 - 15	Adresse des Feldes, in das der eingelesene Datensatz übertragen wird (Operand satz).
	16	Input Edit Byte 1
	17	Input Edit Byte 2
	18	Zuweisungsschlüssel für SYSDTA.
	19	Flag, das die Verwendung des VTSUCB und das Returncode-Verhalten anzeigt.
	20 - 21	max. Länge des einzulesenden Datensatzes (Operand länge).
	22	Flag für Aufbereitungsbyte des ISAM-Schlüssels.
	23	SYSDTA-Zuweisungsindikator (Bit 2^0=1 ≙ Operand A).
	24 - 25	Position des ISAM-Schlüssels.
	26 - 27	Länge des ISAM-Schlüssels.
	28 - 31	Adresse des VTSUCB
	32 - 33	Werte des Timers
	34 - 35	reserviert (X'00000000')
24-Bit-Schnittstelle	0	Input Edit Byte 1
	1 - 3	Adresse des Feldes, in das der eingelesene Datensatz übertragen wird (Operand satz).
	4	Flag
	5	Input Edit Byte 2
	6 - 7	max. Länge des einzulesenden Datensatzes (Operand länge).
	8	SYSDTA-Zuweisungsindikator (Bit 2^0=1 ≙ Operand A).
	9 - 11	Adresse, zu der im Fehlerfall verzweigt werden soll (Operand fehler).

– Bei Verwendung der 24-Bit-Schnittstelle sind die Werte für Input Edit Byte 1/2 der beim Makro **CUPAB** angegebenen Tabelle zu entnehmen.

– Bei Verwendung der 31-Bit-Schnittstelle sind diese Werte einer mit MF=C/D erzeugten Datenliste zu entnehmen.

– Zuweisungsschlüssel für SYSDTA:

  24-Bit-Schnittstelle: Der Zuweisungsschlüssel wird im linksbündigen Byte des Registers R15 übergeben.

  31-Bit-Schnittstelle: Der Zuweisungsschlüssel wird im Feld CURAIND des **RDATA**-Datenbereichs übergeben.

Schlüsselwerte und ihre Bedeutung

Wert	Bedeutung
X'00'	Zuordnung für SYSDTA nicht geändert.
X'04'	SYSDTA ist einer SAM-Datei zugeordnet.
X'08'	SYSDTA ist einer ISAM-Datei zugeordnet.
X'14'	SYSDTA ist einer Datenstation zugeordnet.
X'18'	SYSDTA ist einer S-Variablen zugeordnet.
X'20'	SYSDTA ist einem Element einer PLAM-Bibliothek zugeordnet.

– Flag-Byte und Bedeutung:
Bit 2^7 = 1/0 entspricht der Angabe KEYOUT=Y/N.
Bit 2^6 = 1/0 entspricht der Angabe KEYPOS=Y/N.
Bit 2^5 = 1/0 entspricht der Angabe KEYLEN=Y/N.

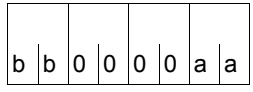
Rückinformation und Fehleranzeigen

– Für ein Gerät oder für den Betriebsmodus MODE nicht zugelassene Edit-Options werden vom System soweit wie möglich korrigiert.

– Während der Makrobearbeitung enthält Register R1 die Adresse des Datenbereichs.

bei PARMOD=24

R15:



Über die Ausführung des Makros RDATA wird im Register R15 ein gegliederter Returncode übergeben (aa=Maincode, bb=Zuweisungsschlüssel, wenn der Operand A in Verbindung mit der 24-Bit-Schnittstelle angegeben wurde; sonst bb = X'00').

X'aa'	Erläuterung
X'00'	Normale Beendigung.
X'04'	Nicht behebbarer Fehler.
X'08'	Operandenfehler.
X'0C'	Abschneiden des Satzes. Satzlänge > spezifizierte Länge.
X'10'	Dateiende (EOF).
X'14'	SYSDTA ist nicht zugewiesen
X'18'	Fehler während des Datenträger-Zugriffs
X'20'	Ungültiges Edit Option Byte; Fehler wurde vom System korrigiert
X'38'	Problem im Zusammenhang mit POSIX

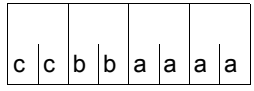
bei PARMOD=31, RC=OLD

Zusätzlich zu den unter PARMOD=24 beschriebenen Returncodes kann der Returncode X'24' (Fehler im VTSUCB) auftreten, sowie die Returncodes, die durch Konvention makroübergreifend festgelegt sind (siehe Tabelle „Standard-Returncodes“ auf Seite 43).

bei PARMOD=31, RC=NEW:

Die Returncodes werden sowohl im Standardheader als auch im Register R15 eingetragen.

Standard-header:



Über die Ausführung des Makros RDATA wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Erfolgreiche Bearbeitung der Funktion.
X'00'	X'00'	X'0014'	Erfolgreiche Bearbeitung der Funktion, aber SYSDTA nicht zugewiesen.
X'00'	X'00'	X'0018'	Erfolgreiche Bearbeitung der Funktion, aber Fehler beim Zugriff auf den Datenträger.
X'00'	X'00'	X'0020'	Erfolgreiche Bearbeitung der Funktion, ein aufgetretener Operanden-fehler ist durch TIAM/ VTSU korrigiert worden.
X'00'	X'01'	X'0008'	Nicht korrigierter Operandenfehler.
X'07'	X'01'	X'0008'	Nicht korrigierter Operandenfehler: die RESERVED-Felder sind nicht 0
X'08'	X'01'	X'0008'	Nicht korrigierter Operandenfehler: der Wert des Operanden TIMER ist nicht im erlaubten Intervall von 10 bis 3600 Sekunden.
X'00'	X'20'	X'0004'	Interner Fehler.
X'02'	X'20'	X'0004'	Interner Fehler: BCAM-Nachricht verloren.
X'05'	X'20'	X'0004'	Interner Fehler: Eingabenachricht zu lang.
X'06'	X'20'	X'0004'	Interner Fehler: negative Transportquittung.
X'00'	X'40'	X'0004'	Ein-/Ausgabe abgebrochen.
X'00'	X'40'	X'000C'	Eingabesatzlänge > spezifizierte Länge: Eingabesatz wurde abgeschnitten.
X'00'	X'40'	X'0010'	Dateiende (EOF)
X'00'	X'40'	X'0034'	Timer-Ablauf (innerhalb der festgelegten Wartezeit erfolgte keine Eingabe).
X'01'	X'80'	X'0004'	Interner BCAM-Engpass.
X'09'	X'80'	X'0038'	Fehler im Zusammenhang mit POSIX: Ein-/Ausgabe-Serialisierungsfehler.
X'0A'	X'40'	X'0038'	Fehler im Zusammenhang mit POSIX: Wenn die LOGON-Task im Systemmodus ist, sind keine Ein-/ Ausgaben von mit fork() erzeugten erzeugten Prozessen möglich.
		X'24'	VTSU-Fehler. Außer Maincode (rechtes Byte) siehe Fehlerinformation im VTSUCB-Header.

Hinweise zum Makroaufruf

- Ein Satz wird abgeschnitten, wenn der zu übertragende Satz größer ist als im Längenoperanden angegeben. Der Satz wird nur entsprechend dem Längenoperanden in den angegebenen Einlesebereich übertragen. Der Satzrest geht verloren. Ist der Satz kürzer als der Einlesebereich, wird er in den Einlesebereich linksbündig eingetragen. Der verbleibende Rest des Einlesebereichs wird nicht mit Leerzeichen aufgefüllt. Das Programm wird ohne Fehleranzeige weiter ausgeführt.
- Dateiende (EOF) kann auf folgende Weise initiiert werden:
  - Dialogbetrieb:
    Auslösen der Funktion ESCAPE (Taste K2). Die Jobverarbeitung wechselt in den Kommandomodus. Eingabe der Kommandos EOF und (nachfolgend) RESUME-PROGRAM.
  - Prozedur- oder Batch-Betrieb:
    Die (System-) Dateien SYSDTA und SYSCMD sind einander zugeordnet, und ein Datensatz wird eingelesen, der mit einem Schrägstrich beginnt.
    Ausnahmen:
    - Das Kommando HOLD-PROGRAM wird eingegeben;
    - Der Schrägstrich wird durch symbolische Operanden ersetzt.
    - Zwei aufeinander folgende Schrägstriche am Anfang des Satzes (z.B. SDF-Anweisung).
  - Die (System-)Dateien SYSDTA und SYSCMD sind nicht einander zugeordnet: Am Anfang des Datensatzes (Spalte 1-4) steht das Kommando EOF.

Beispiele siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29.

RDUID – Benutzerkennung abfragen

Allgemeines

Anwendungsgebiet: Abfragen und Zugriff zu Listen und Tabellen; siehe Seite 158
Makrotyp: S-Typ, MF-Format **2**: Standardform/C-/D-/L-/E-Form; siehe Seite 29

Makrobeschreibung

Der Makro **RDUID** übergibt in seinem Datenbereich einem Benutzerprogramm die Benutzerkennung und die Abrechnungsnummer des Auftrags, unter dem es läuft (siehe Layout des Datenbereichs im Anschluss an die Operandenbeschreibung).

Makroaufrufformat und Operandenbeschreibung

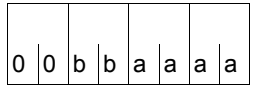
RDUID
MF=S / E / L / C / D [,PARAM=adr / (r)] ,PREFIX=S / p ,MACID=RMR / macid

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form oder D-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros RDUID wird im Standardheader folgender Returncode übergeben (bb=Subcode1, aaaa=Maincode):

X'bb'	X'aaaa'	Erläuterung
X'00'	X'0000'	Funktion erfolgreich ausgeführt.
X'20'	X'00FF'	Systemfehler

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Das aufrufende Programm wird beendet, wenn folgende Fehler auftreten:
- Der Datenbereich ist dem Aufrufer nicht zugewiesen.
- Der Datenbereich ist nicht auf Wortgrenze ausgerichtet.
- Der Datenbereich ist gegen Schreibzugriff geschützt.

Layout des Datenbereichs für RDUID MF=C

```
1           FHDR  MF=(C,SRMR),EQUATES=NO
2           DS    0A
2 SRMRFHE   DS    0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
2 SRMRIFID  DS    0A              0   INTERFACE IDENTIFIER
2 SRMRFCTU  DS    AL2             0   FUNCTION UNIT NUMBER
2 *                                   BIT 15    HEADER FLAG BIT,
2 *                                   MUST BE RESET UNTIL FURTHER NOTICE
2 *                                   BIT 14-12 UNUSED, MUST BE RESET
2 *                                   BIT 11-0  REAL FUNCTION UNIT NUMBER
2 SRMRFCT   DS    AL1             2   FUNCTION NUMBER
2 SRMRFCTV  DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 SRMRRET   DS    0A              4   GENERAL RETURN CODE
2 SRMRSRET  DS    0AL2            4   SUB RETURN CODE
2 SRMRSR2   DS    AL1             4   SUB RETURN CODE 2
2 SRMRSR1   DS    AL1             5   SUB RETURN CODE 1
2 SRMRMRET  DS    0AL2            6   MAIN RETURN CODE
2 SRMRMR2   DS    AL1             6   MAIN RETURN CODE 2
2 SRMRMR1   DS    AL1             7   MAIN RETURN CODE 1
2 SRMRFHL   EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 SRMRUID   DC    CL8' '              USERID
1 SRMRACC   DC    CL8' '              ACCOUNT NUMBER
1 SRMR#     EQU   *-SRMRFHE           LENGTH OF RDUID PARAMETER BLOCK
```

RELBF – Empfangswarteschlange freigeben

Allgemeines

Anwendungsgebiet: Intertaskkommunikation; siehe Seite 76
Kommunikation; siehe Seite 167
Makrotyp: O-Typ; siehe Seite 28

Makrobeschreibung

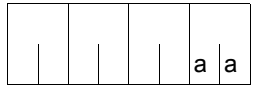
Mit dem Makro **RELBF** kann ein Anwender, der an der Intertaskkommunikation (ITC) teilnimmt, die erste Nachricht in seiner Empfangswarteschlange löschen.
Ein Anwender, der seine Empfangswarteschlange auswerten will, kann die erste Nachricht eines bestimmten Absenders anfordern oder die erste Nachricht in der Warteschlange (FIFO-Prinzip). Geht er nach dem FIFO-Prinzip vor, so muss er die erste Nachricht in der Warteschlange löschen, bevor er auf die nächste Nachricht zugreifen kann. Hat er die erste Nachricht nicht implizit beim Anfordern löschen lassen (REVNT...REL=YES), dann kann er sie explizit mit dem Makro **RELBF** löschen.
Will der Anwender die gesamte Empfangswarteschlange löschen, aber noch nicht seine ITC-Teilnahme beenden (CLCOM), dann kann er den Makro **RELBF** so lange in einer Schleife aufrufen, bis der Returncode angibt, dass die Warteschlange leer ist.

Makroaufrufformat und Operandenbeschreibung

RELBF

Rückinformation und Fehleranzeigen

R15:



Über die Ausführung des Makros RELBF wird im rechtsbündigen Byte des Registers R15 ein Returncode übergeben.

X'aa'	Erläuterung
X' 00'	Die erste Nachricht in der Empfangswarteschlange wurde gelöscht.
X' 04'	Die Empfangswarteschlange war leer.
X' 08'	Die Task des Aufrufers ist kein ITC-Teilnehmer.

RELM – Speicherbereich freigeben

Allgemeines

Anwendungsgebiet: Arbeiten mit virtuellem Speicher; siehe Seite 55
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

Makrobeschreibung

Der Makro **RELM** gibt einen zusammenhängenden Speicherbereich des Klasse-6-Speichers des Aufrufers frei. Der Speicherbereich wird als Vielfaches von einer Seite (4 KByte) freigegeben.
Es ist erlaubt, Speicher freizugeben, der nicht vorher mit **REQM** angefordert wurde.
Mittels mehrerer **REQM**-Makroaufrufe angeforderter Speicherplatz kann mit einem **RELM** freigegeben werden, sofern diese Bereiche zusammenhängend sind.

Makroaufrufformat und Operandenbeschreibung

```
RELM

[{ zahl }],{ seite }
[{ (r)  }]  { (r)  }

[,PARMOD=24 / 31]
,MF=S / (E,...) / L
```

zahl
Anzahl der Seiten (4 KByte), die freigegeben werden sollen.
Voreinstellung: zahl = 1.

(r)
r = Register, das die Angabe zahl enthält.

seite
Seitennummer der ersten Seite (4 KByte) des Bereichs, der freigegeben werden soll.

(r)
r = Register, das die Seitennummer enthält.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wird PARMOD nicht spezifiziert, so erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

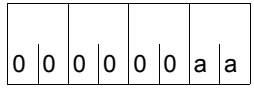
24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

Rückinformation und Fehleranzeigen

Während der Makrobearbeitung enthält Register R1 die Adresse des Datenbereichs.

R15:



Über die Ausführung des Makros RELM wird im Register R15 ein Returncode übergeben.

X'aa'	Erläuterung
X'00'	Der Speicherplatz wurde freigegeben.
X'04'	Funktion nur zum Teil ausgeführt. Im freizugebenden Bereich sind auch nicht angeforderte Seiten enthalten. Die Adresse der ersten nicht angeforderten Seite wird in Register R1 übergeben. Alle Seiten des Bereichs bis zu dieser Adresse wurden freigegeben, alle Seiten danach sind im selben Zustand wie vor dem RELM-Aufruf.
X'0C'	Funktion nicht ausgeführt. – Unzulässige Adresse des Datenbereichs. – Unzulässiger Aufbau des Datenbereichs. – Der freizugebende Bereich liegt (zum Teil) außerhalb des Klasse-6-Speichers. – Der freizugebende Bereich überschneidet sich (zum Teil) mit einem Memory Pool. – Der freizugebende Bereich überschneidet sich (zum Teil) mit einem FASTPAM ENVIRONMENT / IOAREA POOL. Die Freigabe des Bereichs ist erst nach der entsprechenden FASTPAM-Disable-Funktion möglich. – Der freizugebende Bereich überschneidet sich (zum Teil) mit einem DIV-Fenster. Die Freigabe des Bereichs ist erst nach der entsprechenden DIV-Unmap-Funktion möglich.

bei 31-Bit-Schnittstelle:

– Bei fehlerhafter Initialisierung des Standardheaders werden zusätzlich die Returncodes X'0001FFFF' / X'0003FFFF' / X'0004FFFF' übergeben; siehe Tabelle „Standard-Returncodes“ auf Seite 43.
– Im Standardheader wird kein Returncode übergeben.

Beispiele

1. Das Benutzerprogramm belegt die Seiten 1 bis 12.
   Durch **RELM 3,10** werden ab Seite 10 drei Seiten freigegeben, also die Seiten 10, 11 und 12.
2. Das Benutzerprogramm belegt die Seiten 1 bis 6.
   Durch **RELM ,5** wird ab Seite 5 eine Seite freigegeben, also Seite 5.

RELMP – Seiten im Memory Pool freigeben

Allgemeines

Anwendungsgebiet: Memory Pool Technik; siehe Seite 55
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

Ein Anwender kann explizit (zusammenhängenden) Speicherplatz in einem Memory Pool anfordern (**REQMP**) und auch wieder freigeben (**RELMP**). Für die Freigabe von Speicherplatz in einem Memory Pool gilt:

– der Aufrufer muss Pool-Teilnehmer sein (**ENAMP**),

– es ist ohne Bedeutung, wer von den Pool-Teilnehmern den Speicherplatz angefordert hatte und in welchen Portionen dies erfolgte.

Makrobeschreibung

Mit dem Makro **RELMP** kann der Anwender zusammenhängenden Speicherplatz in einem Memory Pool freigeben. Die Freigabe erfolgt in Speicherseiten (4KB).

Hinweise

– Ein Memory Pool wird über den Pool-Namen oder über seine Kurzkennung (siehe **ENAMP**) angesprochen.

– Die Freigabe von Poolseiten verändert nicht die im **ENAMP**-Aufruf festgelegte Größe eines Memory Pools.

– Der freizugebende Speicherplatz muss nicht zusammenhängend zugewiesen worden sein.

– **RELMP** wird abgewiesen, wenn für den Memory Pool mit **CSTMP** Schreibschutz vereinbart wurde.

Makroaufrufformat und Operandenbeschreibung

```
RELMP
{ { MPNAME=name                                                                  }
  { MPNAMAD={adr}[,MPNAMLN={länge}]                                              } [,SCOPE={LOCAL    }]
  {         {(r)}          {(r)}                                                 }         {GROUP    }
  { MPID=adr / (r)                                                               }         {USER_GROUP}
  {                                                                              }         {GLOBAL   }
[,ADDR=adr / (r)]
[,BSIZE=anzahl / (r) / ALL]
[,PARMOD=24 / 31]
[,MF=L / (E,..)]
```

MPNAME=
beschreibt den Namen des Memory Pools (Verbindung mit Operand SCOPE beachten).

name
Name des Memory Pools.

MPNAMAD=
beschreibt die Adresse des Feldes mit name (Verbindung mit Operand SCOPE beachten).

adr
symbolische Adresse (Name) des Feldes.

(r)
r = Register mit dem Adresswert des Feldes.

MPNAMLN=
bezeichnet die Länge des unter MPNAMAD angegebenen Namens.
Wenn nicht spezifiziert:
Längenattribut des Feldes adr oder 54 Byte, wenn MPNAMAD=(r) angegeben wurde.

länge
Länge in Byte.

(r)
r = Register, das länge enthält.

SCOPE=
beschreibt den Geltungsbereich (Teilnehmerkreis) des Memory Pools. Die Angabe dient der eindeutigen Identifizierung des Memory Pools und muss immer in Verbindung mit dem Operanden MPNAME bzw. MPNAMAD spezifiziert werden.

LOCAL
Der Memory Pool wird nur von dem einrichtenden Teilnehmer benutzt.

GROUP
Teilnehmer können alle Tasks unter der Benutzerkennung des einrichtenden Teilnehmers sein.

USER_GROUP
Teilnehmer können alle Tasks sein, deren Benutzerkennungen der gleichen Benutzergruppe angehören wie die Benutzerkennung des einrichtenden Teilnehmers.
Der Operandenwert setzt die Existenz von Benutzergruppen voraus und kann daher nur angegeben werden, wenn die Funktionseinheit SRPM des Software-Produkts SECOS im System vorhanden ist. Vor einem Makroaufruf mit SCOPE=USER_GROUP muss deshalb mit dem Makro GETUGR (siehe Handbuch „SECOS“ [14]) geprüft werden, ob SRPM zur Verfügung steht; abhängig vom Ergebnis (Returncode) ist im Programm zu reagieren.

GLOBAL
Teilnehmer können alle im System laufenden Tasks sein.

MPID=
beschreibt die Adresse eines Feldes (Länge = 4 Byte) mit der Kurzkennung für den Memory Pool (siehe auch **ENAMP**). Die Kurzkennung identifiziert den Memory Pool eindeutig; die Angabe beschleunigt die Verarbeitung.

adr
symbolische Adresse (Name) des Feldes mit der Kurzkennung.

(r)
r = Register mit dem Adresswert des Feldes.

ADDR=
bezeichnet die Anfangsadresse des Speicherbereichs, der freigegeben werden soll. Die Adresse muss auf eine 4KB-Grenze ausgerichtet sein. Der gesamte Bereich muss im angegebenen Memory Pool liegen.

Hinweis
Der Operand ADDR muss spezifiziert werden, wenn nicht BSIZE=ALL angegeben wird.

adr
Anfangsadresse.

(r)
r = Register mit der Anfangsadresse.

BSIZE=
beschreibt die Größe des freizugebenden Speicherbereichs in Speicherseiten (4 KB).
Voreinstellung: BSIZE=1.

Hinweis
Die Angabe BSIZE=0 ist erlaubt: Es wird kein Speicherplatz freigegeben.

anzahl
Anzahl der Speicherseiten.

(r)
r = Register, das anzahl enthält.

ALL
Alle für den Memory Pool angeforderten Speicherseiten werden freigegeben. Diese Angabe kann auch im Register r, in der Form C' ALL', erfolgen.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

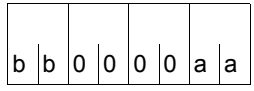
24
Die 24-Bit-Schnittstelle wird generiert; Datenlisten benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB) und beginnen mit dem Standardheader.

Rückinformation und Fehleranzeigen

Nach der Makrobearbeitung enthält Register R1 die Adresse des Datenbereichs.

R15:



Über die Ausführung des Makros RELMP wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.
aa = X'00': normale Ausführung,
aa = X'04': Funktion nicht ausgeführt.

X'bb'	X'aa'	Erläuterung
X'00'	X'00'	Normale Ausführung. Speicherseiten wurden freigegeben.
X'18'	X'00'	Normale Ausführung. Speicherseiten wurden freigegeben; innerhalb des angegebenen Bereichs waren nicht alle Seiten angefordert worden.
X'04'	X'04'	Funktion nicht ausgeführt. Aufrufer ist kein Memory-Pool-Teilnehmer (kein ENAMP-Aufruf).
X'18'	X'04'	Funktion nicht ausgeführt. Ungültiger Speicherbereich: – Anfangsadresse oder eine Adresse des angegebenen Speicherbereichs liegt außerhalb des Memory Pools. – Anfangsadresse nicht auf 4-KB-Grenze ausgerichtet. – Der angegebene Speicherbereich überlappt ganz oder teilweise mit einem Bereich, der von DIV oder FASTPAM genutzt wird.
X'1C'	X'04'	Funktion nicht ausgeführt. Operandenfehler: – unzulässige Adresse des Datenbereichs. – fehlerhafter Aufbau des Datenbereichs. – unzulässige Adressen für MPNAMAD oder MPID im Datenbereich. – Bezeichnung des Memory Pools: – Name enthält unzulässige Zeichen – ungültige Längenangabe (MPNAMLN) – MPNAMLN angegeben, aber MPNAMAD nicht spezifiziert – weder MPNAME noch MPNAMAD noch MPID angegeben – SCOPE angegeben, aber MPNAME/MPNAMAD nicht spezifiziert. – Benennung nicht eindeutig. Es wurde mehr als nur ein Operand spezifiziert (MPNAME/MPNAMAD/MPID). – ungültige SCOPE-Angabe. – ungültige BSIZE-Angabe. – es wurde weder der Operand ADDR noch BSIZE=ALL angegeben – ungültiges Register (R1) angegeben – PARMOD=24 in Verbindung mit 31-Bit-Adressierungsmodus (AMODE 31) angegeben. – SCOPE=USER_GROUP wurde angegeben, obwohl SRPM nicht im System vorhanden ist. – Die Freigabe von Nicht-Memory-Pool-Seiten wird nicht mehr unterstützt.

X'bb'	X'aa'	Erläuterung
X'24'	X'04'	Funktion nicht ausgeführt. Berechtigungsfehler: – Memory Pool ist mit Schreibschutz versehen. – Memory Pool wurde von einem privilegierten Teilnehmer gegen Seitenfreigabe geschützt. Aufruf evtl. wiederholen. – Aufrufer ist nicht berechtigt, Speicherseiten in einem privilegierten oder Klasse-5-MP freizugeben.

31-Bit-Schnittstelle:

– Bei fehlerhafter Ausrichtung oder Initialisierung des Standardheaders werden im Register R15 zusätzlich die Returncodes X'0001FFFF' / X'0003FFFF' / X'0004FFFF' übergeben; siehe Tabelle „Standard-Returncodes“ auf Seite 43.

– Im Standardheader wird kein Returncode übergeben.

REQM – Speicherbereich anfordern

Allgemeines

Anwendungsgebiet: Arbeiten mit virtuellem Speicher; siehe Seite 55
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

Makrobeschreibung

Mit **REQM** kann zusammenhängender Speicherplatz für das Benutzerprogramm angefordert werden. Mit **RELM** kann dieser Bereich wieder freigegeben werden. Der Speicherbereich wird als Vielfaches von einer Seite (4 KByte) angefordert. Es wird immer virtueller Speicher zugeteilt. Die Byte der Seite(n) sind mit X'00' überschrieben (Ausnahme: die Seite wurde bereits angefordert).

Makroaufrufformat und Operandenbeschreibung

```
REQM

[{ zahl }][,{ seite                          }
  { (r)   }  { (r)                           }
             { LOC={ RES   }                 } ]
             {      { ANY   }                }
             {      { BELOW }                }
             {      { ABOVE }                }
             {                               }

,ALIGN=4KB / HW_PAGE
[,PARMOD=24 / 31]
,MF=S / (E,..) / L
```

zahl
Anzahl der anzufordernden Seiten (4 KByte).
Voreinstellung: zahl = 1.

(r)
r = Register mit der Angabe zahl.

seite
seite = Seitennummer (4 KByte), ab der die Zuweisung beginnen soll.
Voreinstellung:
Niedrigste Seitennummer des ersten freien und ausreichend großen Bereichs im Klasse-6-Speicher des Aufrufers. Der Bereich kann bei Funktionsausführung ausgerichtet werden

(z.B. auf ein Vielfaches von 16 Seiten), wenn die Anzahl der angeforderten Seiten einen bestimmten Wert überschreitet.

Hinweis

Der Operand sollte nur spezifiziert werden, wenn vorher Lage und Größe des Klasse-6-Speichers des Aufrufers mit dem Makro **MINF** abgefragt wurde.

(r)
r = Register mit der Angabe seite.

LOC=
bezeichnet den Teil des Klasse-6-Speichers, in dem die Speicherseiten reserviert werden sollen (unterhalb oder oberhalb der 16MB-Grenze).
Der Operand wird ignoriert, wenn die 24-Bit-Schnittstelle benutzt wird.

RES
Die Speicherseiten werden in dem Teil des Klasse-6-Speichers reserviert, in dem der Makroaufruf erfolgt.

ANY
Die Speicherseiten werden abhängig vom Adressierungsmodus wie folgt reserviert:
– Unterhalb der 16MB-Grenze, wenn der 24-Bit-Adressierungsmodus eingeschaltet ist, bzw.
– Oberhalb oder unterhalb der 16MB-Grenze, wenn der 31-Bit-Adressierungsmodus eingeschaltet ist.

BELOW
Die Speicherseiten werden unterhalb der 16MB-Grenze reserviert.

ABOVE
Die Speicherseiten werden oberhalb der 16MB-Grenze reserviert.

ALIGN=
gibt die Ausrichtung des gewünschten Speicherbereichs an.

4KB
Der gewünschte Speicherbereich soll auf einer 4 KByte-Grenze beginnen.

HW_PAGE
Der gewünschte Speicherbereich soll auf einer Seitengrenze beginnen, die durch die Hardware vorgegeben ist. In diesem Fall müssen die Operanden „zahl“ und „seite“, sofern verwendet, ein Vielfaches (in 4 KByte-Einheiten) der Hardware-Seitengröße sein.

Hinweis

Die Hardware-Seitengröße kann mit dem Makro **NSIINF INFO=PAGESIZE** ermittelt werden.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

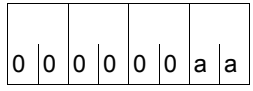
24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

Rückinformation und Fehleranzeigen

Während der Bearbeitung des Makros enthält Register R1 die Adresse der Operandenliste. Nach erfolgreicher Bearbeitung des Aufrufs ist die Anfangsadresse des zugewiesenen Speichers in Register R1 abgespeichert.

R15:



Über die Ausführung des Makros REQM wird im Register R15 ein Returncode übergeben.

X'aa'	Erläuterung
X'00'	Die Anforderung wurde erfüllt. Register R1 enthält die Adresse der ersten Seite, auch wenn die Seite bereits vorher angefordert wurde.
X'04'	Funktion nicht ausgeführt. – Nicht genügend zusammenhängender, freier Speicherplatz im Adressraum verfügbar. – Nicht genügend freier Platz auf dem Hintergrundspeicher (Paging Area) verfügbar.

X'0C'	Funktion nicht ausgeführt.
	– Unzulässige Adresse des Datenbereichs.
	– Unzulässiger Aufbau des Datenbereichs.
	– Die angeforderte Anzahl an Seiten ist größer als
	  – der Wert für den ADDRSPACE-Operanden im Benutzerkatalog (siehe Ausgabe des Kommandos SHOW-USER-ATTRIBUTES).
	  – der Bereich, der unterhalb bzw. oberhalb 16 MB zur Verfügung steht (siehe Ausgabe des Makros MINF).
	– Der angeforderte Bereich liegt (zum Teil) außerhalb des Klasse-6-Speichers.
	– Der angeforderte Bereich überschneidet sich (zum Teil) mit einem Memory Pool.
	– Es ist ALIGN=HW_PAGE angegeben, aber „zahl“ oder „seite“ ist kein Vielfaches der Hardware-Seitengröße (in 4 KByte-Einheiten).

für 31-Bit-Schnittstelle:

– Bei fehlerhafter Initialisierung des Standardheaders werden zusätzlich die Returncodes X'0001FFFF' / X'0003FFFF' / X'0004FFFF' übergeben; siehe Tabelle „Standard-Returncodes“ auf Seite 43.
– Im Standardheader wird kein Returncode übergeben.

REQMP – Seiten im Memory Pool anfordern

Allgemeines

Anwendungsgebiet: Memory Pool Technik; siehe Seite 55
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

Ein Anwender kann explizit (zusammenhängenden) Speicherplatz in einem Memory Pool anfordern, wobei gilt:

– der Aufrufer muss Pool-Teilnehmer sein (**ENAMP**),
– der zugeteilte Speicherbereich kann von jedem Pool-Teilnehmer wieder freigegeben werden,
– auf den zugeteilten Speicherbereich kann jeder Pool-Teilnehmer zugreifen.

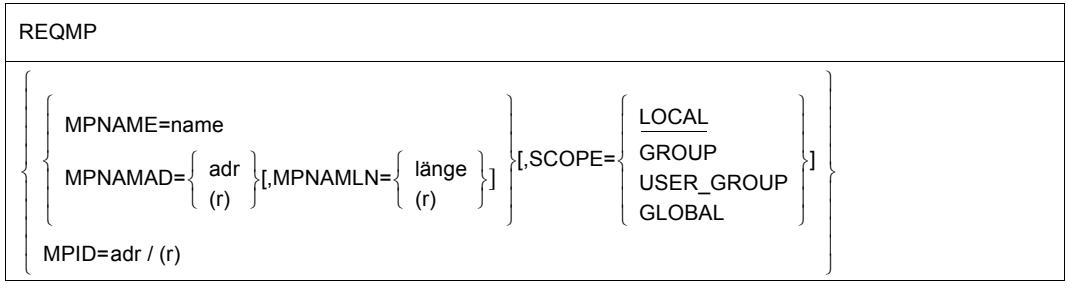
Makrobeschreibung

Mit dem Makro **REQMP** wird zusammenhängender Speicherbereich in einem Memory-Pool angefordert. Die Zuteilung erfolgt in Speicherseiten (4 KB). Die Bytes der Seite(n) sind mit X'00' überschrieben (Ausnahme: die Seite wurde bereits angefordert).

Hinweise

– Ein Memory Pool wird über den Pool-Namen oder über seine Kurzkennung (siehe **ENAMP**) angesprochen.
– **REQMP** wird abgewiesen, wenn für den Memory Pool mit **CSTMP** Schreibschutz vereinbart wurde.
– Residenter Memory Pool: **REQMP** wird abgewiesen, wenn die Anzahl der angeforderten Seiten den Wert des Operanden RESIDENT-PAGES im Kommando START- oder LOAD-EXECUTABLE-PROGRAM überschreitet.
– Größe des Memory Pools und Belegung der Speicherseiten (frei oder nicht frei) können mit **MINF** abgefragt werden.

Makroaufrufformat und Operandenbeschreibung



REQMP (Fortsetzung)
[,ADDR=adr / (r)]
[,BSIZE=anzahl / (r)]
,ALIGN=4KB / HW_PAGE
[,PARMOD=24 / 31]
[,MF=L / (E,..)]

MPNAME=
beschreibt den Namen des Memory Pools (Verbindung mit Operand SCOPE beachten).

name
Name des Memory Pools.

MPNAMAD=
beschreibt die Adresse des Feldes mit name (Verbindung mit Operand SCOPE beachten).

adr
symbolische Adresse (Name) des Feldes

(r)
r = Register mit dem Adresswert des Feldes.

MPNAMLN=
bezeichnet die Länge des unter MPNAMAD angegebenen Namens. Wenn nicht spezifiziert: Längenattribut des Feldes adr oder 54 Byte, wenn MPNAMAD=(r) angegeben wurde.

länge
Länge in Byte.

(r)
r = Register, das länge enthält.

SCOPE=
beschreibt den Geltungsbereich (Teilnehmerkreis) des Memory Pools. Die Angabe dient der Identifizierung des Memory Pools und muss immer in Verbindung mit den Operanden MPNAME bzw. MPNAMAD spezifiziert werden.

LOCAL
Der Memory Pool kann nur von dem Teilnehmer benutzt werden, der den Memory Pool eingerichtet hat.

GROUP
Teilnehmer können alle Tasks unter der Benutzerkennung des Teilnehmers sein, der den Memory Pool eingerichtet hat.

USER_GROUP
Teilnehmer können alle Tasks sein, deren Benutzerkennungen der gleichen Benutzergruppe angehören wie die Benutzerkennung des einrichtenden Teilnehmers.
Der Operandenwert setzt die Existenz von Benutzergruppen voraus und kann daher nur angegeben werden, wenn die Funktionseinheit SRPM des Software-Produkts SECOS im System vorhanden ist. Vor einem Makroaufruf mit SCOPE=USER_GROUP muss deshalb mit dem Makro GETUGR (siehe Handbuch „SECOS“ [14]) geprüft werden, ob SRPM zur Verfügung steht; abhängig vom Ergebnis (Returncode) ist im Programm zu reagieren.

GLOBAL
Teilnehmer können alle im System laufenden Tasks sein.

MPID=
beschreibt die Adresse eines Feldes (Länge = 4 Byte) mit der Kurzkennung für den Memory Pool (siehe auch **ENAMP**). Die Kurzkennung identifiziert den Memory Pool eindeutig. Die Angabe der Kurzkennung beschleunigt die Verarbeitung.

adr
symbolische Adresse (Name) des Feldes mit der Kurzkennung.

(r)
r = Register mit dem Adresswert des Feldes.

ADDR=
bezeichnet die Anfangsadresse des Speicherbereichs, der zugeteilt werden soll. Die Adresse muss auf eine 4 KB-Grenze ausgerichtet sein. Der gesamte Bereich muss im angegebenen Memory Pool liegen.
Voreinstellung: der erste unbenutzte und zusammenhängende Bereich innerhalb des Memory Pools. Der Bereich kann auf eine spezielle Grenze (64KB-/1MB-Grenze) ausgerichtet werden, wenn die Anforderung eine bestimmte Anzahl Speicherseiten überschreitet.

adr
Anfangsadresse.

(r)
r = Register mit der Anfangsadresse.

BSIZE=
bezeichnet die Größe des angeforderten Speicherbereichs in Speicherseiten (4 KB).
Voreinstellung: BSIZE=1; wird auch angenommen, wenn BSIZE=0 spezifiziert wurde.

anzahl
Anzahl der Speicherseiten.

(r)
r = Register, das anzahl enthält.

ALIGN=
gibt die Ausrichtung des gewünschten Speicherbereichs an.

4KB
Der gewünschte Speicherbereich soll auf einer 4-KByte-Grenze beginnen.

HW_PAGE
Der gewünschte Speicherbereich soll auf einer Seitengrenze beginnen, die durch die Hardware vorgegeben ist. In diesem Fall müssen die Operanden BSIZE (in 4-KByte-Einheiten) und ADDR (als Anfangsadresse), sofern verwendet, ein Vielfaches der Hardware-Seitengröße sein.

Hinweis
Die Hardware-Seitengröße kann mit dem Makro **NSIINF INFO=PAGESIZE** ermittelt werden.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

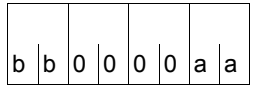
24
Die 24-Bit-Schnittstelle wird generiert; Datenlisten benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB) und beginnen mit dem Standardheader.

Rückinformation und Fehleranzeigen

Nach Funktionsausführung wird die Anfangsadresse des zugewiesenen Speichers in Register R1 übergeben.

R15:



Über die Ausführung des Makros REQMP wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.
aa = X'00': normale Ausführung,
aa = X'04': Funktion nicht ausgeführt.

X'bb'	X'aa'	Erläuterung
X'00'	X'00'	Normale Ausführung. Speicherseiten wurden zugeteilt.
X'18'	X'00'	Normale Ausführung. Innerhalb des angeforderten Bereichs war schon mindestens eine Speicherseite zugeteilt. Die restlichen Speicherseiten wurden zugeteilt.
X'04'	X'04'	Funktion nicht ausgeführt. Aufrufer ist kein Memory-Pool-Teilnehmer (kein ENAMP-Aufruf).
X'14'	X'04'	Funktion nicht ausgeführt. Unzureichender Speicherplatz: – Memory Pool verfügt nicht über so viel zusammenhängenden freien Speicherplatz. – Residenter Speicherplatz: Anforderung überschreitet den Wert des Operanden RESIDENT-PAGES im START- oder LOAD-EXECUTABLEPROGRAM-Kommando – ALIGN=HW_PAGE ist angegeben, aber BSIZE oder ADDR ist kein Vielfaches der Hardware-Seitengröße.
X'18'	X'04'	Funktion nicht ausgeführt. Ungültiger Speicherbereich: – Anfangsadresse oder eine Adresse des angegebenen Speicherbereichs liegt außerhalb des Memory Pools. – Anfangsadresse nicht auf 4-KB-Grenze ausgerichtet.
X'1C'	X'04'	Funktion nicht ausgeführt. Operandenfehler: – unzulässige Adresse des Datenbereichs – fehlerhafter Aufbau des Datenbereichs – unzulässige Adressen für MPNAMAD oder MPID im Datenbereich – Benennung des Memory Pools: – Name enthält unzulässige Zeichen – ungültige Längenangabe (MPNAMLN) – MPNAMLN angegeben, aber MPNAMAD nicht spezifiziert – weder MPNAME noch MPNAMAD noch MPID angegeben – SCOPE angegeben, aber MPNAME/MPNAMAD nicht spezifiziert. – Benennung nicht eindeutig. Es wurde mehr als nur ein Operand spezifiziert (MPNAME/MPNAMAD/MPID). – ungültige SCOPE-Angabe. – ungültige BSIZE-Angabe. – ungültiges Register (R1) angegeben. – SCOPE=USER_GROUP wurde angegeben, obwohl SRPM nicht im System vorhanden ist. – PARMOD=24 in Verbindung mit 31-Bit-Adressierungsmodus (AMODE 31) angegeben. – Die Anforderung von Nicht-Memory-Pool-Seiten wird nicht mehr unterstützt.
X'24'	X'04'	Funktion nicht ausgeführt. Berechtigungsfehler: – Memory Pool ist mit Schreibschutz versehen. – Aufrufer ist nicht berechtigt, Speicherseiten in einem privilegierten oder Klasse-5-MP anzufordern.

31-Bit-Schnittstelle:

- Bei fehlerhafter Ausrichtung oder Initialisierung des Standardheaders werden im Register R15 zusätzlich die Returncodes X'0001FFFF' / X'0003FFFF' / X'0004FFFF' übergeben; siehe Tabelle „Standard-Returncodes“ auf Seite 43.
- Im Standardheader wird kein Returncode übergeben.

RETCO – Contingency-Prozess beenden

Allgemeines

Anwendungsgebiet: Contingency-Verfahren; siehe Seite 111
Makrotyp: S-Typ, MF-Format **1**: Standardform/E-Form/L-Form; siehe Seite 29

Makrobeschreibung

Dieser Makro bewirkt den Ausgang aus einem Contingency-Prozess. Die Kontrolle wird einem Basis- oder Contingency-Prozess gleicher oder niederer Priorität übergeben.
Der **RETCO**-Aufruf darf nicht im Basisprozess gegeben werden (abnormale Programmbeendigung mit Fehlermeldung `ETMEV03`).

Makroaufrufformat und Operandenbeschreibung

RETCO
[MF=L / E]

MF
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Beispiele siehe Abschnitt „Contingency-Prozesse“ auf Seite 111 und die Beschreibung des **POSSIG**-Makros (Seite 754).

RETRN – Rücksprung mit Register laden

Allgemeines

Anwendungsgebiet: Starten, Unterbrechen und Beenden; siehe Seite 72
Makrotyp: O-Typ; siehe Seite 28

Makrobeschreibung

Mit dem Makro **RETRN** kann man gesicherte Registerinhalte zurückladen, von einem Unterprogramm in das übergeordnete Programm zurückspringen und auf Wunsch einen Returncode übergeben.
Man ruft den Makro **RETRN** am Ende eines Unterprogramms auf, wenn man zu dessen Beginn die Register des Hauptprogramms mit dem Makro **SAVE** gesichert hat. Alle Mehrzweckregister außer dem Register R13 können zwischengespeichert und wieder geladen werden. Das Register R13 muss die Adresse des Zwischenspeichers enthalten, der vom Hauptprogramm zur Verfügung gestellt wurde (siehe Makro **SAVE**).

Makroaufrufformat und Operandenbeschreibung

RETRN
[(r1[,r2])] [,T] [,RC=rc / (15)]

r1
gibt ein Register (Mehrzweckregister=MZR) an, das aus dem Zwischenspeicher geladen werden soll. Für r1 dürfen die Zahlen 0 bis 12, 14 oder 15 angegeben werden; die (runden) Klammern können entfallen.

r1,r2
gibt eine fortlaufende Reihe von Registern (MZR) an, die wiedergeladen werden sollen. Der Bereich (r1,r2) darf Register R13 nicht enthalten, er kann sich aber über die Grenze Register R15 / Register R0 erstrecken. z.B. werden mit der Angabe (14,12) sämtliche Register bis auf Register R13, also Register R0 bis R12, R14 und R15 aus dem Zwischenspeicher geladen.

T
ist für die Kompatibilität mit BBS-Übersetzern vorgesehen und dient dazu, die Vorwärtskettung bei Programmschachtelung zu unterbrechen. In Wort 3 des Zwischenspeichers wird im niederwertigsten Bit eine 1 gesetzt, nachdem die Register zurückgeladen sind.

RC=
legt einen Returncode fest, der dem Hauptprogramm in den drei rechtsbündigen Byte von Register R15 übergeben wird.

rc
Der Wert „rc“ muss eine Dezimalzahl bzw. ein absoluter Ausdruck sein.

(15)
gibt an, dass Register R15 einen Returncode in den drei rechtsbündigen Byte enthält. Register R15 wird nicht mit dem ursprünglichen Inhalt aus dem Zwischenspeicher wiedergeladen.

Funktionsweise

Der Makro **RETRN** lädt die angegebenen Register aus einem Zwischenspeicher und führt einen Sprung in das Hauptprogramm aus. Für den Aufruf des Makros **RETRN** wird vorausgesetzt:

– im Unterprogramm wurden zu Beginn die Registerwerte mit dem Makro **SAVE** im Zwischenspeicher gesichert,
– die Adresse des Zwischenspeichers wurde nach Register R13 geladen,
– im Hauptprogramm wurde die Rücksprungadresse nach Register R14 geladen.

Wenn angegeben, übergibt der Makro **RETRN** im Register R15 einen Returncode. Dieser Code wird zwischen dem Haupt- und dem Unterprogramm frei vereinbart. Die Beschreibung des Makros **SAVE** geht auf diesen Sachverhalt ausführlich ein und enthält ein Beispiel.

Hinweis

Für die Registerangaben können auch die Bezeichnungen Rn, (n=Registernummer), verwendet werden.

REVNT – Ereignis empfangen

Allgemeines

Anwendungsgebiet:	Intertaskkommunikation; siehe Seite 76 Kommunikation; siehe Seite 167
Makrotyp:	O-Typ; siehe Seite 28

Makrobeschreibung

Mit dem Makro **REVNT** kann ein Anwender, der an der Intertaskkommunikation (ITC) teilnimmt, eine Nachricht anfordern und auf ihr Eintreffen warten.

Funktionsweise

Jeder ITC-Teilnehmer kann mit dem Makro **REVNT** eine Nachricht anfordern. Er kann dabei festlegen, ob er eine Nachricht von einem beliebigen Teilnehmer annimmt oder ob sie von einem bestimmten Absender stammen muss. Wartet der Teilnehmer auf eine beliebige Nachricht, so kann diese auch von einem Absender kommen, der sich später der ITC angeschlossen hat.
Ist zum Zeitpunkt des **REVNT**-Aufrufs noch keine Nachricht (oder noch keine des gewünschten Absenders) in der Empfangswarteschlange des aufrufenden Teilnehmers, wird die Task unterbrochen. Die unterbrochene Task wird fortgesetzt, wenn die Nachricht eintrifft oder wenn die Wartezeit verstrichen ist, deren Dauer im **REVNT**-Aufruf festgelegt wurde.

Aus der Empfangswarteschlange überträgt das System die Nachricht in das Empfangsfeld des Programms. Sind mehrere Nachrichten in der Warteschlange, dann überträgt es die erste Nachricht bzw. die erste Nachricht des gewünschten Absenders. Im **REVNT**-Aufruf kann der Anwender angeben, ob die Nachricht nach der Übertragung aus der Warteschlange gelöscht werden soll. Lässt er sie nicht löschen, dann kann er sie mit einem weiteren **REVNT**-Aufruf noch einmal übertragen lassen.

Eine Nachricht kann 8 Byte bis 64 KB lang sein (einschließlich 4 Byte Satzlängenfeld). Wenn der Benutzer den Umfang der auszutauschenden Nachrichten beim Programmieren kennt, kann er die Länge der Empfangsfelder danach einrichten. Im Empfangsfeld trägt das System die tatsächliche Länge zusammen mit der Nachricht ein. Kann das Empfangsfeld die Nachricht nicht vollständig aufnehmen, dann überträgt das System nur die ersten 4 Byte der Nachricht, aber im Satzlängenfeld trägt es die vollständige Länge ein.

Der Returncode X'0C' kennzeichnet diesen Fall. Ist die Nachrichtenlänge beim Programmieren unbekannt, dann sollte der Benutzer den Makro **REVNT** mit REL=NO aufrufen. Er erhält die aktuelle Länge, kann das Empfangsfeld diesem Wert anpassen und mit einem

zweiten **REVNT**-Aufruf die volle Nachricht übertragen lassen. Diesmal kann er mit REL=YES die Nachricht löschen lassen, oder er ruft den Makro **RELBF** auf, der die erste Nachricht in der Warteschlange löscht.

ITC-gekoppelte Ereignissteuerung:
Der Wartezustand, in den die Task nach dem **REVNT**-Aufruf versetzt wird, kann durch die Kopplung mit der Ereignissteuerung vermieden werden, wenn die Nachricht von einem beliebigen Absender erwartet wird. Außerdem kann das Warten auf eine ITC-Nachricht zusammengelegt werden mit dem Warten auf ein anderes Ereignis (siehe Abschnitt „Ereignisgesteuerte Verarbeitung (Eventing)“ auf Seite 95).

Die ITC wird dadurch an eine Ereignissteuerung gekoppelt, dass man im **REVNT**-Aufruf als weiteren Operanden die Adresse der Ereignis-Kurzkennung angibt. Diese Adresse muss vorher in einem **ENAEI**-Aufruf festgelegt worden sein. Die Task wird nach dem **REVNT**-Aufruf **nicht** unterbrochen. Der Aufrufer erhält einen eingeschränkten Returncode, der angibt, ob der **REVNT**-Aufruf akzeptiert ist. Angaben über das Eintreffen der Nachricht erhält er durch den Postcode, sobald der **SOLSIG**-Aufruf abgeschlossen ist. Mit dem **SOLSIG**-Aufruf kann jetzt der Anwender zu einem beliebigen Zeitpunkt nach dem **REVNT** (bei Contingency-Angabe auch vorher) ein Ereignis-Signal anfordern. Ist noch keine Nachricht eingetroffen, so tritt die Wartezeit des **SOLSIG**-Aufrufs in Kraft. Die Task wird unterbrochen, falls kein Contingency-Prozess angegeben war. Die Wartezeit des **SOLSIG** wird durch jedes Ereignis beendet, das bei der angegebenen Ereignissteuerung eintritt. Der Post Code gibt an, welcher Klasse das Ereignis angehört.

Hinweise

– Mit dem Aufruf **REVNT**...,WTIME=0 kann die Empfangswarteschlange ohne Unterbrechung geprüft werden.
– Die Begrenzung der Wartezeit (Operand WTIME) soll verhindern, dass sich Teilnehmer gegenseitig blockieren (wenn jeder auf eine Nachricht des anderen wartet).

Bei gekoppeltem **REVNT** zu beachten:

– Die Ereignisklasse im Post Code muss geprüft werden, um festzustellen, ob ein ITC-Ereignis oder ein anderes Ereignis vorliegt.
– Solange ein gekoppelter **REVNT**-Aufruf nicht abgeschlossen ist, darf kein weiterer **REVNT**-Aufruf - ob gekoppelt oder ungekoppelt - gegeben werden; (wird mit Returncode X'18' abgewiesen).
– Solange ein gekoppelter **REVNT**-Aufruf nicht abgeschlossen ist, darf die Ereigniskennung nicht gelöscht werden (**DISEI**-Aufruf). Der **REVNT**-Aufruf kann sonst nicht abgeschlossen werden, und es können Nachrichten verloren gehen (siehe auch den Hinweis im Abschnitt „Intertaskkommunikation (ITC)“ auf Seite 85).

Makroaufrufformat und Operandenbeschreibung
Format 1:

REVNT
empfangsfeld,länge [,WTIME=sekunden] ,REL=YES / NO [,NAME=sendername] [,EIID=adresse]

empfangsfeld
symbolische Adresse des Feldes, in das die Nachricht aus der Empfangswarteschlange übertragen werden soll. Das Feld muss an einer Wortgrenze beginnen. Nach der Übertragung hat es folgenden Inhalt:

Byte	Inhalt
0 - 7	ITC-Name des Teilnehmers, der die Nachricht gesendet hat.
8 9	Länge der vollständigen Nachricht + 4 (für SLF) } Satzlängenfeld (SLF)
10 11	reserviert } Satzlängenfeld (SLF)
12 : n	Nachricht (mindestens 4 Byte, falls das Feld zu klein war, um die vollständige Nachricht aufzunehmen)

Das Empfangsfeld muss mindestens 16 Byte lang sein, die Angabe im Satzlängenfeld muss mindestens den Wert 8 haben.

länge
Dezimalzahl, die die Länge des Empfangsfeldes einschließlich der 8 Byte für den Absender angibt. Der Wert für „länge“ kann zwischen 16 und 65543 liegen. Die Nachricht darf einschließlich Satzlängenfeld von 8 bis 65535 lang sein.

WTIME=sekunden
gibt an, wie lange die Task auf die Nachricht warten soll, wenn sie beim Aufruf von **REVNT** noch nicht vorliegt. Hier sind Zeitangaben von 0 bis 21599 Sekunden zugelassen.
Wird WTIME nicht angegeben, dann wartet die Task standardmäßig 600 Sekunden.

REL=
bestimmt den Verbleib der Nachricht in der Empfangswarteschlange.

YES
Die Nachricht wird nach der Übertragung aus der Empfangswarteschlange gelöscht, auch wenn sie nicht vollständig im Empfangsfeld Platz gefunden hat.

NO
Die Nachricht bleibt in der Empfangswarteschlange.

NAME=sendername
ITC-Name eines ITC-Teilnehmers: gibt an, dass eine Nachricht nur übertragen werden soll, wenn der angegebene Teilnehmer der Absender ist.

EIID=adresse
Dieser Operand muss nur angegeben werden, wenn die ITC mit der Ereignissteuerung gekoppelt werden soll (siehe „Koppelung von ITC und Ereignissteuerung“ auf Seite 81). adresse = symbolische Adresse eines Feldes, das die Kurzkennung der Ereigniskennung enthält. Das Feld ist 4 Byte lang. Durch den Aufruf des Makros **ENAEI** hat das System die Kurzkennung in diesem Feld eingetragen.

Format 2:

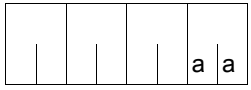
REVNT
(1)

(1)
Register R1 enthält die Adresse eines Operandenfeldes mit folgendem Inhalt:

Byte	Inhalt
0 - 3	Adresse des Empfangsfeldes
4 - 7	Länge des Empfangsfeldes (sedezimal)
8	X'00' : keine Kopplung mit Ereignissteuerung X'01' : die REVNT-Anforderung wird an eine Ereigniskennung gekoppelt
9 - 11	C'YES' für REL=YES X'00' C'NO' für REL=NO
12 - 15	Dauer der Wartezeit in Sekunden
16 - 23	ITC-Name des Senders, ggf. mit Leerzeichen zur Länge 8 aufgefüllt, oder nur Leerzeichen, wenn kein Sender angegeben werden soll.
24 - 27	Adresse der Ereignis-Kurzkennung. Nur anzugeben, wenn die Kennung in Byte 8 gesetzt ist.

Rückinformation und Fehleranzeigen

R15:



Über die Ausführung des Makros REVNT wird im rechtsbündigen Byte des Registers R15 ein Returncode übergeben.

REVNT ohne Kopplung an Ereignissteuerung:

X'aa'	**Erklärung**
X'00'	Die Nachricht wurde vollständig übertragen.
X'04'	Fehler in der Operandenangabe (z.B. Speicher ist nicht Klasse 6 oder nicht zugewiesen). Es wurde keine Nachricht übertragen.
X'08'	Der Aufrufer ist kein ITC-Teilnehmer. Keine Nachricht wurde übertragen.
X'0C'	Das Empfangsfeld ist zu klein für die vollständige Nachricht. Nur der Kopf und die ersten 4 Byte wurden übertragen.
X'10'	Auch während der Wartezeit ist keine Nachricht eingetroffen.
X'18'	Ein früher aufgerufener REVNT mit Kopplung ist noch nicht abgeschlossen. Der jetzige Aufruf wird abgewiesen.

REVNT mit Kopplung an Ereignissteuerung:

X'aa'	**Erklärung**
X'00'	Der REVNT-Aufruf ist akzeptiert.
Bei den folgenden Returncodes führt der REVNT-Aufruf nicht zu einem ITC-Ereignis:	
X'04'	Fehler in Operandenangabe (z.B. Speicher ist nicht Klasse 6 oder nicht zugewiesen). Der REVNT-Aufruf wird abgewiesen.
X'08'	Der Aufrufer ist kein ITC-Teilnehmer. Der REVNT-Aufruf wird abgewiesen.
X'18'	Ein früher aufgerufener REVNT mit Kopplung ist noch nicht abgeschlossen. Der jetzige Aufruf wird abgewiesen.

Bedeutung des Post Codes (nur bei Kopplung an Ereignissteuerung):

Der Post Code ist 4 Byte lang und wird nach dem **SOLSIG**-Aufruf unter einer dort angegebenen Adresse eingetragen. Das linksbündige Byte gibt die Ereignisklasse an (siehe Abschnitt „Ereignisgesteuerte Verarbeitung (Eventing)“ auf Seite 95). Das rechtsbündige Byte enthält den Returncode, der jeweils für die Ereignisklasse gilt. Ein ITC-Ereignis hat die Ereignisklasse X'08':

ITC-Post Code	Erläuterung
X'08000000'	Ein gekoppelter REVNT-Aufruf ist mit dem Empfang einer Nachricht abgeschlossen worden.
X'08000004'	Operandenfehler: Der Speicherplatz für das Empfangsfeld ist nicht mehr zugewiesen (asynchroner Fall der Ereignissteuerung).
X'0800000C'	Das Empfangsfeld ist zu klein für die vollständige Nachricht. Nur der Kopf und die ersten 4 Byte wurden übertragen.
X'08000010'	Die Wartezeit ist abgelaufen, ohne dass eine Nachricht eingetroffen ist.

RPOFEI – POSSIG-Signal senden

Allgemeines

Anwendungsgebiet: (optimierte) Ereignissteuerung; siehe Seite 95
Makrotyp: R-Typ; siehe Seite 28

Forward Eventing (FEV) ist eine optimierte Form der synchronen Ereignissteuerung (synchrones Eventing). FEV vermeidet für wiederholte **POSSIG**- bzw. **SOLSIG**-Aufrufe in einem Programm die wiederholte Validierung der angegebenen Operanden. Stattdessen wird eine Ereignisliste EVENTLST angelegt und z.B. für das Senden von Signalen zu einer Ereigniskennung (POSSIG-Funktion) einmalig ein POSSIG-Eintrag eingetragen. Der Eintrag kann explizit wieder gelöscht werden (**DELFEI**).
Die Task des Aufrufers muss der Ereigniskennung zugeordnet sein (**ENAEI**).

Makrobeschreibung

Der Makro **RPOFEI** nimmt Bezug auf einen POSSIG-Eintrag in der EVENTLST und löst das Senden eines Signals (Ereignisses) an eine Ereigniskennung aus.
Das Signal beendet den Wartezustand der anfordernden Task oder startet dort eine Contingency-Routine, wenn diese in einem **SOLSIG**-Aufruf angegeben wurde.

Makroaufrufformat und Operandenbeschreibung

RPOFEI
REFNUM=(r)

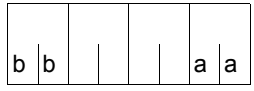
REFNUM=(r)
bezeichnet ein Register, das (direkt) die Referenznummer für den POSSIG-Eintrag enthält.

(r)
r = Register mit der Referenznummer.

Rückinformation und Fehleranzeigen

Während der Makrobearbeitung enthält Register R1 die Referenznummer.
Register R0 wird mit einem internen Funktionscode überschrieben.

R15:



Über die Ausführung des Makros RPOFEI wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'bb'	**X'aa'**	**Erläuterung**
X'00'	X'00'	Normale Ausführung: Das Signal wurde gesendet.
X'04'	X'04'	Keine Aktion: Falsche Referenznummer (POSSIG-Eintrag schon gelöscht?)
X'28'	X'04'	Keine Aktion: Die maximal mögliche Anzahl von Anforderungen in der POSSIG-Warteschlange wurde überschritten.

RSOFEI – POSSIG-Signal (Ereignis) anfordern

Allgemeines

Anwendungsgebiet: (optimierte) Ereignissteuerung; siehe Seite 95
Makrotyp: R-Typ; siehe Seite 28

Forward Eventing (FEV) ist eine optimierte Form der synchronen Ereignissteuerung (synchrones Eventing). FEV vermeidet für wiederholte **SOLSIG**- bzw. **POSSIG**-Aufrufe in einem Programm die wiederholte Validierung der angegebenen Operanden. Stattdessen wird eine Ereignisliste EVENTLST angelegt und z.B. für Signalanforderungen von einer Ereigniskennung (SOLSIG-Funktion) einmalig ein SOLSIG-Eintrag eingetragen. Der Eintrag kann explizit wieder gelöscht werden (**DELFEI**).
Die Task des Aufrufers muss der Ereigniskennung zugeordnet sein (**ENAEI**).

Makrobeschreibung

Der Makro **RSOFEI** nimmt Bezug auf einen SOLSIG-Eintrag in der EVENTLST und fordert ein Signal (Ereignis) von einer Ereigniskennung an. Die Task des Aufrufers wird in den Wartezustand versetzt, wenn das angeforderte Signal noch nicht eingetroffen ist, jedoch längstens bis Ablauf der angegebenen Wartezeit (Makro **DSOFEI**).

Makroaufrufformat und Operandenbeschreibung

RSOFEI
REFNUM=(r)

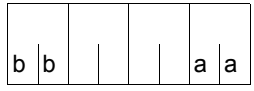
REFNUM=r
bezeichnet ein Register, das (direkt) die Referenznummer für den SOLSIG-Eintrag enthält.

(r)
r = Register mit der Referenznummer.

Rückinformation und Fehleranzeigen

Während der Makrobearbeitung enthält Register R1 die Referenznummer.
Register R0 wird mit einem internen Funktionscode überschrieben.

R15:



Über die Ausführung des Makros RSOFEI wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'bb'	**X'aa'**	**Erläuterung**
X'00'	X'00'	Normale Ausführung: Das Ereignis ist eingetreten. Ein Postcode wurde übertragen, sofern er angegeben war.
X'30'	X'00'	Funktion ausgeführt: Postcode konnte nicht übertragen werden, weil der Empfänger kein Empfangsfeld bereitgestellt hatte.
X'34'	X'00'	Funktion ausgeführt: Postcode wurde nicht übertragen, weil er mit X' 00000000' angegeben wurde.
X'38'	X'00'	Funktion ausgeführt: Postcode wurde rechts abgeschnitten (Empfangsfeld zu klein).
X'3C'	X'00'	Funktion ausgeführt: Postcode wurde linksbündig eingetragen (Empfangsfeld zu groß). Diese Rückinformation tritt nicht auf, wenn im Makro DSOFEI der Operand RPOSTNUM verwendet wird.
X'04'	X'04'	Keine Aktion: Falsche Referenznummer (SOLSIG-Eintrag schon gelöscht?)
X'20'	X'04'	Keine Aktion: Das Ereignis ist innerhalb der Wartezeit nicht eingetreten.
X'28'	X'04'	Keine Aktion: Die Zuordnung zu der Ereigniskennung wurde vor dem Eintreffen des Ereignisses gelöst (Makro DISEI).
X'50'	X'04'	Kein Ereignis verfügbar.

SAVE – Registerinhalte sicherstellen

Allgemeines

Anwendungsgebiet: Starten, Unterbrechen und Beenden; siehe Seite 72
Makrotyp: O-Typ; siehe Seite 28

Makrobeschreibung

Mit dem Makro **SAVE** kann man Registerinhalte zwischenspeichern.

Man ruft den Makro **SAVE** zu Beginn von Neben- oder Unterprogrammen auf, um die Registerinhalte des Hauptprogramms zu sichern. Für den Rücksprung in das Hauptprogramm ruft man den Makro **RETRN** auf, der die gesicherten Inhalte wieder in die Register lädt und den Sprung durchführt. Alle Mehrzweckregister außer dem Register R13 können gesichert werden. Das Register R13 muss die Adresse des Zwischenspeichers enthalten, der vom Hauptprogramm zur Verfügung gestellt werden muss.

Makroaufrufformat und Operandenbeschreibung

SAVE
[(r1[,r2])] [,T] [,entry / *]

r1
gibt ein Register (Mehrzweckregister=MZR) an, das gesichert werden soll. Für r1 dürfen die Zahlen 0 bis 12, 14 oder 15 angegeben werden; die runden Klammern können entfallen.

r1,r2
gibt eine fortlaufende Reihe von Registern (MZR) an, die gesichert werden sollen.
Der Bereich (r1,r2) darf Register 13 nicht enthalten, er kann sich aber über die Grenze Register R15/Register R0 erstrecken.
Es werden z.B. mit der Angabe (14,12) sämtliche Register bis auf Register R13 (also Register R0 bis 12, 14 und 15) gesichert.

Hinweis

Für die Registerangaben kann auch die Bezeichnung Rn (n=Registernummer) verwendet werden.

T
ist für die Kompatibilität mit BBS-Übersetzern vorgesehen und gibt an, dass die Register R14 und 15 gesichert werden sollen. Dieser Operand wird angegeben, wenn das Register r1 oder der Bereich (r1,r2) nicht an die Register R14 und 15 angrenzt.

Hinweis

Beginnt der Bereich (r1,r2) mit dem Register R1 oder R2 und ist der Operand T angegeben, so werden sämtliche Register von R14 bis R2 gesichert, d.h. die Register R0 und evtl. R1 werden nicht aus dem Bereich ausgeschlossen.

entry
ist für die Kompatiblilität mit BBS-Übersetzern vorgesehen und stellt ein Kennzeichen des Makroaufrufs **SAVE** dar. entry darf bis zu 155 Zeichen lang sein, Kommas und Leerzeichen sind nicht erlaubt. Das Kennzeichen wird in die Makroauflösung vor dem ersten auszuführenden Befehl übernommen. Dem Kennzeichen unmittelbar vorangestellt ist ein Byte, das an einer Halbwortgrenze beginnt und die Länge des Kennzeichens beinhaltet.

*****
ist für die Kompatibilität mit BBS-Übersetzern vorgesehen und gibt an, dass als Kennzeichen der Eintrag im Namensfeld des Makroaufrufs verwendet werden soll. Ist das Namensfeld leer, wird der Name der CSECT verwendet, die den Makroaufruf enthält.

Funktionsweise

Der Makro **SAVE** wird - zusammen mit dem Makro **RETRN** - in Unterprogrammen benutzt, um die Registerinhalte des Hauptprogramms zu sichern. Der Makro **SAVE** speichert die Werte ab, der Makro **RETRN** lädt sie vor dem Rücksprung wieder in die Register und übergibt auf Wunsch einen Returncode.
Der Zwischenspeicher für die Registerinhalte muss vom Hauptprogramm mit einer Länge von 18 Worten (72 Byte) definiert werden. Seine Adresse muss in Register R13 geladen werden.

Struktur des Zwischenspeichers

1	Reserviert für COMPILER	
2	Adresse des Zwischenspeichers des vorgeordneten Programms (MZR 13)	vorbehalten für Programmschachtelung
3	Adresse des Zwischenspeichers des anzuspringenden Programms	
4	MZR 14 Rücksprungadresse	gesicherte Registerinhalte
5	MZR 15 Zieladresse	
6	MZR 0	
7	MZR 1 Datenadressen	
8	MZR 2	
9	MZR 3	
10	MZR 4	
11	MZR 5	
12	MZR 6	
13	MZR 7	
14	MZR 8	
15	MZR 9	
16	MZR 10	
17	MZR 11	
18	MZR 12	

Der Zwischenspeicher enthält immer die Registerinhalte des Programms, das ihn definiert hat. Abgespeichert werden sie vom nachgeordneten Unterprogramm, das zu Beginn den Makro **SAVE** aufruft.

Jedem der Register R0 bis R12, R14 und R15 ist ein Wort des Zwischenspeichers fest zugeordnet. Soll ein Register nicht gesichert werden, so übergeht der Makro **SAVE** das zugehörige Speicherwort. Die Anordnung der Registerinhalte im Speicher (R14, R15, R0 bis R12) erlaubt es, sämtliche Register in fortlaufender Reihe abzuspeichern und das Register R13 dabei auszuschließen. Das Register R13 kann nicht vom Makro **SAVE** gesichert werden, da es von **SAVE** zur Adressierung des Zwischenspeichers gebraucht wird.

Wenn das Unterprogramm das Register R13 verändert, muss dieses Register extra gesichert werden. Der Makro **SAVE** kann dazu nicht verwendet werden.

Hinweis zur Programmschachtelung

Ruft ein Unterprogramm selbst ein weiteres auf, das auch mit **SAVE/RETRN** Register sichert, so muss es selbst ebenfalls einen Zwischenspeicher definieren. Bevor es dessen Adresse nach Register R13 lädt, muss es den Inhalt von Register R13 (die Zwischenspeicheradresse des übergeordneten Programms) in Wort 2 des eigenen Zwischenspeichers sichern. Vor der Rückkehr in das übergeordnete Programm muss es das Register R13 von dort wieder laden.
Die Adresse des eigenen Zwischenspeichers kann es im Zwischenspeicher des übergeordneten Programms in Wort 3 hinterlegen.

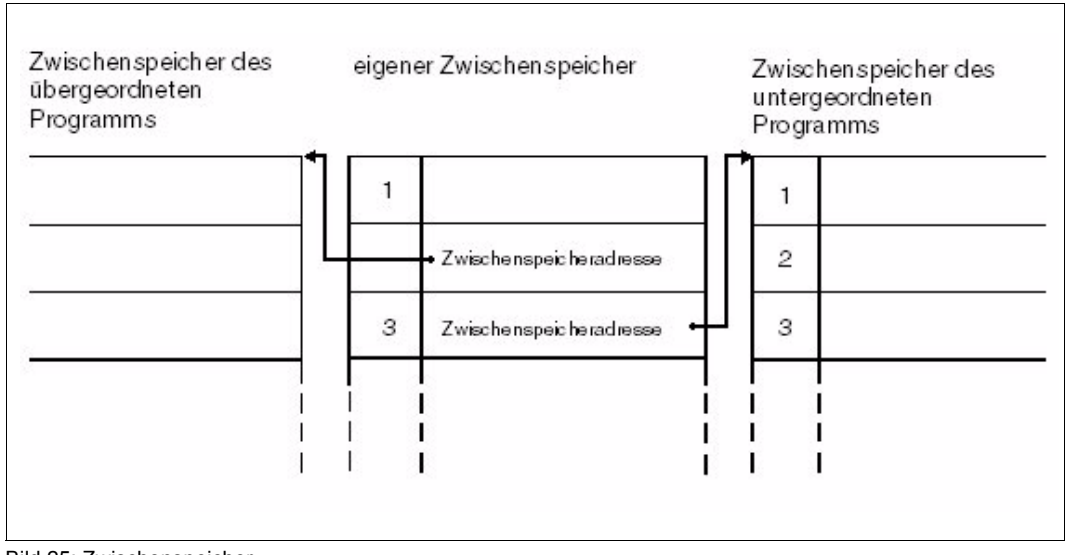


Bild 25: Zwischenspeicher

Die Adresse in Wort 2 wird von dem Programm eingetragen, das den Zwischenspeicher definiert hat.
Die Adresse in Wort 3 wird - nach dessen Aufruf - vom untergeordneten Programm eingetragen, ebenso die Registerinhalte in den Worten 4 bis 18 (Makro **SAVE**).

Registervereinbarungen bei Programmverknüpfung:

Register R13	Adresse des Zwischenspeichers
Register R14	Rücksprungadresse
Register R15	Zieladresse (Einsprungstelle im untergeordneten Programm) bzw. Returncode (siehe Makro **RETRN**)
Register R1	Adresse der Datenadressen (sofern Datenadressen an das untergeordnete Programm übergeben werden).

SEGLD – Segmente laden

Allgemeines

Anwendungsgebiet: Binden und Laden; siehe Seite 47
Makrotyp: O-Typ; siehe Seite 28

Makrobeschreibung

Der Makroaufruf **SEGLD** ermöglicht das automatische Laden eines Segmentes, auch wenn es schon im Speicher steht. Weitere Segmente innerhalb desselben Astes der Überlagerungsstruktur werden automatisch nachgeladen.
Die Fortsetzungsadresse kann angegeben werden.

Makroaufrufformat und Operandenbeschreibung

SEGLD
symbol1[,symbol2]

symbol1
symbolische Adresse innerhalb des zu ladenden Segmentes. Eine 4 Byte lange V-Konstante wird für dieses Symbol erzeugt.

symbol2
symbolische Adresse im aufrufenden oder einem anderen Modul, bei der nach Abschluss des Ladevorganges fortgesetzt werden soll (keine V-Konstante). Wird dieser Operand nicht angegeben, so wird mit dem auf **SEGLD** folgenden Befehl fortgesetzt.

Funktionsweise

Mit dem Makro **SEGLD** (und auch mit dem Makro **CALL**) wird das automatische Laden von Segmenten realisiert (das nicht automatische Laden wird mit dem Makro **LPOV** durchgeführt).
Durch eine Anweisung im Makro **SEGLD** wird der Assembler veranlasst, aus der angegebenen symbolischen Adresse eine V-Konstante zu erzeugen, durch die das zu ladende Segment gekennzeichnet ist. Auf Grund dieser V-Konstanten im Makro **SEGLD** (und des Operanden CONTROL=YES in der Steueranweisung PROGRAM) erzeugt der Binder ein für automatisches Laden notwendiges Überlagerungsmodul, das die V-Konstante mit einer Adresse befriedigt. Zur Ausführungszeit von **SEGLD** wird die Steuerung dem Überlagerungssteuermodul übertragen, das das Segment, das die angegebene symbolische Adres-

se enthält, und alle weiteren Segmente innerhalb des betreffenden Astes der Überlagerungsstruktur in den Speicher lädt. Dies geschieht unabhängig davon, ob die Segmente bereits im Speicher stehen oder nicht.

Nach Abschluss des Ladevorgangs wird - falls der Operand „symbol2“ fehlt - mit dem auf **SEGLD** folgenden Befehl fortgesetzt. Die als Operand „symbol2“ mögliche Adresse kann innerhalb des aufrufenden Moduls liegen oder eine externe Referenz sein. Im zweiten Fall muss der Benutzer die entsprechenden ENTRY- und EXTRN-Anweisungen geben und überdies sicherstellen, dass das Modul, das „symbol2“ enthält, nach Abschluss des Ladevorgangs im Speicher ist (siehe Handbuch „Dienstprogramme“ [27]).

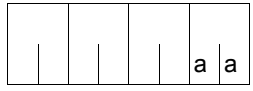
Hinweise zum Makroaufruf

– Beim Entwurf der Überlagerungsstruktur eines Programms, in dem mit automatischem Laden vom Segmenten gearbeitet wird, sollte der zusätzliche Speicherbedarf für den Überlagerungssteuermodul, für ENTAB und für SEGTAB berücksichtigt werden (siehe Handbuch „Dienstprogramme“ [27]).

– Der Makroaufruf **SEGLD** muss in einem Programmbereich liegen, der durch eine USING-Anweisung abgedeckt wird. Da das Register R15 von dem Makro **SEGLD** zum Laden der Segmente verwendet wird, darf es nicht als Basisregister für den Programmteil dienen, in dem der Makroaufruf **SEGLD** liegt.

Rückinformation und Fehleranzeigen

Wenn Fehler während des Ladens eines Überlagerungssegments auftreten, setzt der Ablaufteil das Programm mit dem **SEGLD** folgenden Befehl fort. Tritt kein Fehler auf, so gilt die im Makroaufruf angegebene Fortsetzungsadresse.

R15: 

Über die Ausführung des Makros SEGLD wird im rechtsbündigen Byte des Registers R15. ein Returncode übergeben

X'aa'	**Erläuterung**
X'00'	Das Segment wurde geladen.
X'04'	Falscher Nur-Lese-Modifikationssatz.
X'08'	Falscher Satzcode im Text/Modifikationsblock oder es kommt vor dem 1. Textsatz ein Modifikationssatz.
X'0C'	Es sind zu viele Nur-Lese-Modifikationssätze vorhanden oder Segmentname kann in keinem der Indexsätze für Lademoduln (= Segment) gefunden werden.
X'10'	Es gibt keinen Indexsatz für Lademoduln mit dem Segmentnamen %ROOT.
X'14'	Es ist nicht genügend Speicherplatz vorhanden, um das Segment zu laden.
X'18'	Schlüsselinformation falsch oder Fehler beim Lesen von PAM-Lademoduldatei oder Fehler bei der Ausgabe von Meldungen.
X'1C'	Falscher Segmentname.
X'20'	Fehler beim Laden von AID.
X'24'	Fehler beim Lesen eines C-Elements einer PLAM-Bibliothek.

SELPRGV – Programmversion auswählen

Allgemeines

Anwendungsgebiet: Binden und Laden; siehe Seite 47
Makrotyp: S-Typ, MF-Format **2**: Standardform/C-/D-/E-/L-/M-Form; siehe Seite 29

Zum dynamischen Bindelader DBL siehe auch Handbuch „BLSSERV“ [4].

Makrobeschreibung

Der Makroaufruf **SELPRGV** legt fest, welche Programmversion der DBL verwendet, wenn mehrere Versionen eines Programms geladen werden können. Zum Zeitpunkt der Versionsauswahl ist es nicht notwendig, dass dieses Programm schon geladen ist.

Makroaufrufformat und Operandenbeschreibung

```
SELPRGV

MF=S / C / D / E / L / M
,PRGNAME=<name 1..32>
,PRGNAM@=<var:name 32..32> / (<reg: pointer>)
,PRGVERS=<name 1..24> / *STD
,PRGVER@=<var:name 24..24> / (<reg: pointer>)
,SCOPE=PROGRAM / TASK
,PARAM=<var: pointer> / (reg: pointer>)
,PREFIX=P / p
,MACID=BSL / macid
```

PRGNAME=<name 1..32>
Name des Programms. Aus der Sicht von DBL ist dies der Name einer Ladeeinheit. Der Name darf nur alphanumerische Zeichen enthalten. Angabe nur mit MF=L oder MF=S.

PRGNAM@=<var: name 32..32> / (<reg: pointer>)
Symbolische Adresse oder Register mit der Adresse eines 32 Zeichen langen Feldes, in dem der Programmname steht. Kürzere Namen müssen mit Leerzeichen aufgefüllt werden. Angabe nur mit MF=M.

PRGVERS=<name 1..24>
Programmversion, die der DBL verwenden soll. Angabe nur mit MF=L oder MF=S.

***STD**
Keine Version wird ausgewählt. Der DBL löscht die Programmversion aus seiner Versionstabelle.

PRGVER@=<var: name 24..24> / (<reg: pointer>)
Symbolische Adresse oder Register mit der Adresse eines 24 Zeichen langen Feldes, das die Programmversion enthält. Kürzere Versionsangaben müssen mit Leerzeichen aufgefüllt werden. Angabe nur mit MF=M.

SCOPE=
Geltungsbereich für die Versionsauswahl.

PROGRAM
Die Versionsauswahl ist nur solange gültig, bis eine Programmterminierung erfolgt oder die Versionsauswahl gelöscht wird. Die Versionsauswahl muss deshalb vor bzw. bei jedem Programmaufruf wiederholt werden.

TASK
Die Versionsauswahl ist bis zum Ende der Task gültig oder solange, bis die Versionsauswahl gelöscht wird.

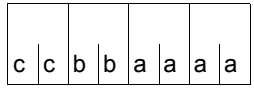
MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. angegebenen Operanden PARAM, PREFIX und MACID siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

Hinweise zum Makroaufruf

– Bei PRGNAM@ und PRGVER@ müssen gültige Klasse-6-Speicher-Adressen angegeben werden.

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros SELPRGV wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Der Makro wurde normal ausgeführt.
X'00'	X'01'	X'0001'	PRGNAME nicht angegeben oder ungültig.
X'00'	X'01'	X'0002'	PRGVERS nicht angegeben oder ungültig.
X'00'	X'01'	X'0003'	Ungültige Angabe bei SCOPE.
X'00'	X'00'	X'0004'	Versionsauswahl kann nicht gelöscht werden, da keine Programmversion ausgewählt war.
X'00'	X'01'	X'0005'	Programmversionstabelle kann nicht erzeugt werden. Versionsauswahl abgewiesen.
X'00'	X'20'	X'0006'	DSSM-Fehler
X'00'	X'20'	X'0300'	Systemfehler
X'00'	X'01'	X'FFFF'	Die Funktion wird nicht mehr oder noch nicht unterstützt.
X'00'	X'03'	X'FFFF'	Die Version der Schnittstelle wird nicht unterstützt.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

SETBF – Pufferlänge für Dialogkommunikation setzen

Allgemeines

Anwendungsgebiet: Verkehr mit Datenstationen; siehe Seite 164
Makrotyp: O-Typ; siehe Seite 28

Makrobeschreibung

Mit dem Makroaufruf **SETBF** kann das Benutzerprogramm die Größe des systeminternen, physikalischen Ein-/Ausgabepuffers für die Kommandos an der Datenstation ändern.
Der **SETBF**-Makro wird im Batch-Betrieb ignoriert.

Makroaufrufformat und Operandenbeschreibung

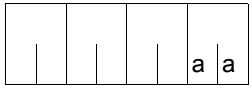
```
SETBF

{ größe [,N] }
{ (1)        }
```

größe
Puffergröße (Anzahl der Zeichen). 80 ≤ größe ≤ 3482.

N
Es soll keine Änderung vorgenommen werden, wenn die angeforderte Puffergröße ebenso groß oder kleiner ist als die bereits bestehende Puffergröße.

(1)
Der Benutzer hat die Größe des gewünschten Puffers in Register R1 geladen, bevor der **SETBF**-Makroaufruf ausgeführt wird. Wird die dem Operanden N entsprechende Funktion verlangt, dann muss das Komplement der Größe in Register R1 geladen werden.

Rückinformation und Fehleranzeigen

R15: 

Über die Ausführung des Makros SETBF wird im rechtsbündigen Byte des Registers R15 ein Returncode übergeben.

X'aa'	**Erläuterung**
X'00'	Der angeforderte Puffer wurde zugeteilt.
X'04'	Ungültige Puffergröße wurde angegeben.
X'08'	Der neue Puffer konnte nicht zugeteilt werden.

SETIC – Intervallzeitgeber setzen

Allgemeines

Anwendungsgebiet:	Starten, Unterbrechen und Beenden; siehe Seite 72 STXIT-Verfahren; siehe Seite 133
Makrotyp:	S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

Makrobeschreibung

Mit dem Makro **SETIC** kann ein Zeitintervall für die CPU-Zeit und/oder die Realzeit definiert und das Ereignis Sommer-/Winterzeitwechsel angezeigt werden. Nach Ablauf des Zeitintervalls wird ein Unterbrechungsereignis „CPU-Zeitintervall abgelaufen“ bzw. „Realzeitintervall abgelaufen“ oder „Sommer-/Winterzeitwechsel“ signalisiert und eine im aufrufenden Programm zugeordnete STXIT-Routine aktiviert (siehe Makro **STXIT**). Ansonsten wird das Programm beendet.

Funktionsweise

Der Ablaufteil setzt Intervallzeitgeber mit den im **SETIC**-Aufruf angegebenen Werten. Sobald diese Werte erreicht sind, wird das Benutzerprogramm unterbrochen. Das System generiert das Ereignis X'20' für „CPU-Zeit-Intervall abgelaufen“ und/oder X'A0' für „Realzeit-Intervall abgelaufen“, und/oder X'20' für „Sommer-/Winterzeitwechsel“, und das Programm verzweigt - falls mit **STXIT** angegeben - zur Unterbrechungsroutine für die entsprechenden Zeitgeber. Wenn keine solche Unterbrechungsroutine durch den **STXIT**-Aufruf angegeben wurde, wird das Programm bei Auftreten einer Zeitgeber-Unterbrechung beendet.
Nachdem die Unterbrechung ausgelöst worden ist, wird der Zeitgeber vom Ablaufteil wieder auf den im **SETIC**-Aufruf angegebenen Wert gesetzt. Die Unterbrechungen finden in den angegebenen Intervallen bzw. zu den angegebenen Tageszeiten so lange statt, bis durch einen weiteren **SETIC**-Aufruf das Zeitintervall geändert oder durch Angabe des Wertes Null ausgeschaltet wird. Mit der Angabe REPEAT=NO kann eine Wiederholung der Intervalle unterdrückt werden.
Wenn sich die Task bei Ablauf des Realzeitintervalls gerade in einem **PASS/VPASS**-Wartezustand befindet, dann wird dieser Wartezustand nicht unterbrochen. Es wird erst nach Beendigung des Wartezustands zur angegebenen **STXIT**-Routine verzweigt.

Makroaufrufformat und Operandenbeschreibung

```
SETIC

[CPUTIM=adr / (r)]

[{ ,REALTIM=adr / (r) }]
 { ,TOD=adr / (r)     }

[,CHWSTIM=YES / NO]
,REPEAT=YES / NO
[,PARMOD=24 / 31]
[,MF=L / (E,..)]
```

CPUTIM=
macht Angaben über das CPU-Zeitintervall (Ereigniscode X'20').

adr
symbolische Adresse des Feldes, das die Angabe für ein CPU-Zeitintervall enthält. Es sind zwei Feldformate möglich:

- Das Feld adr ist ein Wort lang und enthält die Zeitangabe als Binärzahl, die als Anzahl Millisekunden interpretiert wird.
- Das Feld adr ist 6 Byte lang und enthält die Zeitangabe im Format hhmmss - Stunden, Minuten, Sekunden (EBCDIC-Code).

Maximaler Wert für Stunden: 24
Maximaler Wert für Minuten und Sekunden: 59
Wenn die angegebene Zeit abläuft, wird die Steuerung der Routine übergeben, die durch den **STXIT**-Makro definiert wurde (Operand TIMER).

(r)
r = Register mit dem Adresswert des Feldes adr.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB).

REALTIM=
macht Angaben über das Realzeit-Intervall (Ereigniscode X'A0').

adr
symbolische Adresse des Feldes, das die Angabe für ein Realzeitintervall enthält. Es sind zwei Feldformate möglich:
- Das Feld adr ist ein Wort lang und enthält die Zeitangabe als Binärzahl, die als Millisekundenangabe interpretiert wird.
- Das Feld adr ist 6 Byte lang und enthält die Zeitangabe im Format hhmmss - Stunden, Minuten, Sekunden (EBCDIC-Code).

Maximaler Wert für Stunden: 24
Maximaler Wert für Minuten und Sekunden: 59
Die Angabe des Wertes „0“ wird als „24 Stunden“ interpretiert.
Wenn die angegebene Zeit abläuft, wird die Steuerung der Routine übergeben, die im **STXIT**-Makro angegeben wurde (Operand RTIMER).

(r)
r = Register, mit dem Adresswert des Feldes adr.

REPEAT=
gibt an, ob das angegebene Intervall nach Ablauf wieder gesetzt werden soll.

YES
Dasselbe Intervall soll wieder gesetzt werden, nachdem es abgelaufen ist. Dieser Operand wird nur zusammen mit den Operanden CPUTIM, REALTIM oder TOD ausgewertet. Bei Angabe von TOD wird es alle 24 Stunden wiederholt. Bei Angabe von REALTIM werden alle Werte < 50msec auf 50msec gesetzt.
Der Sommer-/Winterzeitwechsel (und umgekehrt) wird immer mit REPEAT=YES realisiert.

NO
Das Intervall wird nicht wieder gesetzt.

TOD=
macht Tageszeitangaben für einen Realzeitgeber, basierend auf einer 24-Stunden-Uhr (Ereigniscode X'A0'). Die Angabe erfolgt im Format hhmmss - Stunden, Minuten, Sekunden (EBCDIC-Code). An der angegebenen Tageszeit wird die Steuerung der im **STXIT**-Makro angegebenen Routine übergeben (Operand RTIMER).

adr
symbolische Adresse eines 6 Byte langen Feldes für die Tageszeitangabe.

(r)
r = Register mit dem Adresswert des Feldes adr.

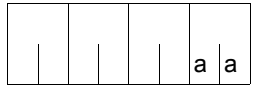
CHWSTIM=
zeigt das Ereignis Sommer-/Winterzeitwechsel (oder umgekehrt) an (Ereigniscode X'C0'). Wenn das Ereignis stattfindet, wird die Steuerung der im **STXIT**-Makro angegebenen Routine übergeben (Operand RTIMER).

YES
Das Ereignis wird angezeigt, d.h. die STXIT-Routine wird - falls definiert - gestartet.

NO
Das Ereignis wird nicht angezeigt.

Rückinformation und Fehleranzeigen

R15:



Über die Ausführung des Makros SETIC wird im Register R15 rechtsbündig ein Returncode übergeben.

X'aa'	Erläuterung
X'00'	Normale Ausführung.
X'04'	Funktion nicht ausgeführt. Ungültige Operanden.
X'08'	Funktion nicht ausgeführt. Ungültige Zeitangabe.

SEVNT – Ereignis senden

Allgemeines

Anwendungsgebiet: Intertaskkommunikation; siehe Seite 76
Kommunikation; siehe Seite 167
Makrotyp: O-Typ; siehe Seite 28

Makrobeschreibung

Mit dem Makro **SEVNT** kann der Anwender, der an der Intertaskkommunikation (ITC) teilnimmt, eine Nachricht an einen anderen ITC-Teilnehmer senden.

Makroaufrufformat und Operandenbeschreibung

SEVNT
{ senderfeld,empfängername (1) }

senderfeld
symbolischer Name des Feldes, das die Nachricht enthält, die gesendet werden soll. Das Feld muss an einer Wortgrenze beginnen und wie ein Satz variabler Länge (mit 4 Byte Satzlängenfeld) aufgebaut sein:

Byte 0-1: Satzlänge in Byte = Länge der Nachricht + 4
(8 ≤ Satzlänge ≤ 65 535)
Byte 2-3: reserviert
Byte 4-n: Nachricht

Die Nachricht (einschließlich Satzlängenfeld) muss mindestens 8 Byte lang sein und darf die Länge von 65535 Byte nicht überschreiten.

empfängername
ITC-Name des Teilnehmers, der die Nachricht empfangen soll.

(1)
gibt an, dass Register R1 die Adresse eines Operandenfeldes enthält. Das Operandenfeld muss auf Wortgrenze ausgerichtet sein und folgenden Inhalt haben:

Byte 0-3: Adresse des Senderfeldes, das den oben beschriebenen Aufbau haben muss.
Byte 4-11: ITC-Name des Empfängers. Ist der Name kürzer als 8 Zeichen, so muss das Feld mit Leerzeichen (X'40') aufgefüllt werden.

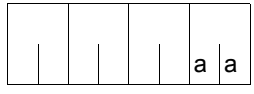
Funktionsweise

Jeder ITC-Teilnehmer kann mit dem Makro **SEVNT** an einen anderen Teilnehmer eine Nachricht senden. Die Nachricht darf zwischen 4 Byte und 64 KB lang sein. Sie wird in die Empfangswarteschlange des empfangenden ITC-Teilnehmers übertragen. Der Makro **SEVNT** wird abgewiesen, wenn nicht genügend Speicherplatz zur Verfügung steht oder wenn die Gesamtlänge der Nachrichten in der Empfangswarteschlange eine vorgegebene Größe (128 KB) überschreitet. Der Absender der Nachricht erfährt nicht automatisch, ob der Empfänger die Nachricht aus der Empfangswarteschlange anfordert und auswertet. Dazu müsste der Empfänger eine Nachricht als Quittung zurücksenden. Die Task mit dem **SEVNT**-Aufruf wird durch das Senden der Nachricht nicht unterbrochen.

Ein ITC-Teilnehmer kann mit aufeinander folgenden Aufrufen von **SEVNT** Nachrichten an verschiedene Empfänger senden.

Rückinformation und Fehleranzeigen

R15:



Über die Ausführung des Makros SEVNT wird im Register R15 rechtsbündig ein Returncode übergeben.

X'aa'	**Erläuterung**
X'00'	Die Nachricht wurde in die Empfangswarteschlange des angegebenen Teilnehmers übertragen.
X'04'	Fehler in der Operandenangabe (z.B. Speicher ist nicht Klasse 6 oder nicht zugewiesen). Die Nachricht wurde nicht übertragen.
X'08'	Der Aufrufer ist kein ITC-Teilnehmer. Die Nachricht wurde nicht übertragen.
X'0C'	Zu wenig Systemspeicher ist verfügbar oder die systeminterne Größe für Empfangswarteschlangen ist bereits überschritten. Die Nachricht wurde nicht übertragen.
X'10'	Der Empfänger ist (noch) kein ITC-Teilnehmer oder der Aufrufer will die Nachricht an sich selber senden. Die Nachricht wurde nicht übertragen.

SHOWMP – Memory Pools anzeigen

Allgemeines

Anwendungsgebiet: Memory Pool Technik; siehe Seite 55
Makrotyp: S-Typ, MF-Format **3**: C-/D-/E-/L-/M-Form; siehe Seite 29

Ein Memory Pool (MP) ist ein Speicherbereich im Klasse-6-Speicher, der von mehreren Anwendern gemeinsam benutzt werden kann. Der Anwender, der den Memory Pool einrichtet, legt eine Größe (Lage), Bezeichnung (Name) und Speicherattribute fest.
Die Größe des Memory Pools und die Belegung der Speicherseiten können mit dem Makro **MINF** abgefragt werden.

Makrobeschreibung

Der Makro **SHOWMP** informiert über Common Memory Pools, die aktuell im System angelegt sind. Lokale Memory Pools werden nicht ausgegeben. Ausgegeben werden Name, Geltungsbereich und die Anzahl der angeschlossenen Tasks. Zusätzlich können auch die TSNs der angeschlossenen Tasks angefordert werden.

Der nicht-privilegierte Benutzer erhält in jedem Fall nur die nicht-privilegierten Memory Pools ausgegeben, an die eine Task seiner Benutzerkennung angeschlossen ist. Bei Auflistung der Sharer-Tasks werden nur Tasks der eigenen Benutzerkennung aufgelistet.

Zur Einschränkung der Informationsmenge bestehen folgende Möglichkeiten:
– Ausgabe für einen bestimmten Namen bzw. Namensraum
– Ausgabe für einen bestimmten Geltungsbereich
– Ausgabe für Memory Pools mit bestimmten Eigenschaften

Zusätzlich kann festgelegt werden, wieviele TSNs bei der Ausgabe der angeschlossenen Tasks maximal aufgelistet werden sollen (Voreinstellung ist 45).

Privilegierte Funktionen

Der privilegierte Anwender (Privileg TSOS bzw. SW-MONITOR-ADMINISTRATION) kann sich über alle Memory Pools informieren. Ihm werden auch alle Sharer-Tasks unabhängig von der Benutzerkennung ausgegeben. Zur Auswahl von privilegierten Memory Pools steht ihm der Operand PRIV_POOL zur Verfügung.

Makroaufrufformat und Operandenbeschreibung

```
SHOWMP

MF=D / E / L / C / M

{,MPNAME=*ALL / '<name 1..54 with_wild>'
 [,MPNAMAD=A(name) / adr / (r) ,MPNAMLN=<integer 1..54> / <var: int:4> / (<reg: integer 1..54>) ] }

,SCOPE=*ANY / *GLOBAL / *GROUP / *USER-GROUP

{,GROUP_USERID=*ANY / *OWN / '<name 1..8>' / <var: name 8..8> / (<reg: A(name 8..8)>)
 ,USER_GROUPID=*ANY / *OWN / '<name 1..8>' / <var: name 8..8> / (<reg: A(name 8..8)>) }

,SELECT=*ALL / *PAR

,PRIV_POOL=*ANY / *YES / *NO

,CONNECT=*ANY / *BY-USER / *BY-TASK

{,USERID=*OWN / '<name 1..8>' / <var: name 8..8> / (<reg: A(name 8..8)>)
 ,TSN=*OWN/ '<name 1..4>' / <var: name 4..4> / (<reg: A(name 4..4)>) }

,INFO=*STD / *ALL

,NUMSHR=45 / <integer 1..4096> / <var: int:4> / (<reg: integer 1..4096>)

,INFO_AREA=adr / (r)

,INFO_LENGTH=1 / <integer 1..1024> / <var: int:4> / (<reg: integer 1..1024>)

[,PARAM=adr / (r)]

,PREFIX=N / p

,MACID=VPI / macid

,XPAND=PARAM / INFA

,EQUATES=YES / NO]
```

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

MPNAME=
Bestimmt den Namen der Memory Pools, die ausgegeben werden sollen (Verbindung mit den Operanden SCOPE und SELECT beachten).
An Stelle dieses Operanden kann in unterschiedlichen MF-Varianten auch der Operand MPNAMAD verwendet werden, siehe die „Programmierhinweise“ auf Seite 836.

***ALL**
Es werden alle Memory Pools ausgegeben.

'<name 1..54 with-wild>'
Nur der angegebene Memory Pool wird ausgegeben. Bei Angabe von Musterzeichen werden alle Memory Pools ausgegeben, deren Name mit der Musterzeichenfolge übereinstimmt. Das Musterzeichensymbol ist *. Das erste Leerzeichen beendet den Namen bzw. die Musterzeichenfolge.

MPNAMAD=
Gibt die Adresse des Feldes mit den Namen der Memory Pools an (Verbindung mit den Operanden SCOPE und SELECT beachten). Das Musterzeichensymbol * ist zulässig. Das erste Leerzeichen beendet den Namen bzw. die Musterzeichenfolge.
An Stelle dieses Operanden kann in unterschiedlichen MF-Varianten auch der Operand MPNAME verwendet werden, siehe die „Programmierhinweise“ auf Seite 836.

A(name)
Adresse des Feldes mit dem Namen der Memory Pools.

adr
Symbolische Adresse (Name) eines Hilfsfeldes, das die Adresse des Feldes mit dem Namen der Memory Pools enthält (nur für MF=M).

(r)
r = Register mit dem Adresswert des Feldes (nur für MF=M).

MPNAMLN=
Beschreibt die Länge des unter MPNAMAD angegebenen Namens. Obligatorische Angabe, wenn MPNAMAD verwendet wird. Sonst wird MPNAMLN ignoriert.

<integer 1..54>
Länge in Byte.

<var: int:4>
Symbolische Adresse (Name) des Feldes, das die Länge enthält (nur für MF=M).

(<reg: integer 1..54>)
reg = Register, das die Länge enthält (nur für MF=M).

SCOPE=
Gibt an, ob nur Memory Pools mit einem bestimmten Geltungsbereich ausgegeben werden sollen.

***ANY**
Die Memory Pools werden unabhängig von ihrem Geltungsbereich ausgegeben.

***GLOBAL / *GROUP / *USER_GROUP**
Die Memory Pools mit dem angegebenen Geltungsbereich werden ausgegeben.

GROUP_USERID=*ANY / *OWN / '<name 1..8>' / <var: name 8..8> / (<reg: A(name 8..8)>)
*Operand wird nur bei SCOPE=*GROUP ausgewertet.*
Es werden nur Memory Pools ausgegeben, die von der angegebenen Benutzerkennung angelegt wurden. Dabei bezeichnet *OWN die Benutzerkennung der aufrufenden Task. Mit *ANY ist die Ausgabe unabhängig von der Benutzerkennung voreingestellt.

'<name 1..8>'
Angabe der Benutzerkennung.

<var: name 8..8>
Symbolische Adresse (Name) des Feldes, das die Benutzerkennung enthält (nur für MF=M).

(<reg:A(name 8..8))
reg = Register, das die Adresse des Feldes mit der Benutzerkennung enthält (nur für MF=M).

USER_GROUPID=*ANY / *OWN / '<name 1..8>' / <var: name 8..8> / (<reg: A(name 8..8)>)
*Operand wird nur bei SCOPE=*USER_GROUP ausgewertet.*
Es werden nur Memory Pools ausgegeben, die von der angegebenen Benutzergruppe angelegt wurden. Dabei bezeichnet *OWN die Benutzergruppe, der die aufrufende Task angehört. Mit *ANY ist die Ausgabe unabhängig von der Benutzergruppe voreingestellt.

'<name 1..8>'
Angabe der Benutzergruppe.

<var: name 8..8>
Symbolische Adresse (Name) des Feldes, das den Namen der Benutzergruppe enthält (nur für MF=M).

(<reg:A(name 8..8))
reg = Register, das die Adresse des Feldes mit dem Namen der Benutzergruppe enthält (nur für MF=M).

SELECT=*ALL / *PAR
Gibt an, ob die durch den Memory-Pool-Namensbereich und SCOPE angegebene Menge von Memory Pools über Auswahlkriterien eingeschränkt werden soll (*PAR) oder nicht (*ALL).

PRIV_POOL=*ANY / *YES / *NO
Gibt bei SELECT=*PAR an, ob nur privilegierte Memory Pools (*YES, nur für nur privilegierte Benutzer zulässig) ausgegeben werden sollen. Bei Angabe von *NO werden keine privilegierten Memory Pools ausgegeben. *ANY zeigt die Memory Pools unabhängig von ihrer Privilegierung an.

CONNECT=
Gibt bei SELECT=*PAR an, ob die Memory Pools in Abhängigkeit von den angeschlossenen Tasks ausgegeben werden sollen. *ANY zeigt die Memory Pools unabhängig von den angeschlossenen Tasks an.

***BY-USER**
Es werden nur Memory Pools ausgegeben, an die eine Task der im nachfolgenden Operanden USERID angegebenen Benutzerkennung angeschlossen ist.

***BY-TASK**
Es werden nur Memory Pools ausgegeben, an die die Task mit der im nachfolgenden Operanden TSN angegebenen TSN angeschlossen ist.

USERID=*OWN / '<name 1..8>' / <var: name 8..8> / (<reg: A(name 8..8)>)
*Operand wird nur für INFO=*ALL ausgewertet.*
Gibt für CONNECT=*BY-USER die Benutzerkennung der angeschlossenen Task an. Mit *OWN ist die Benutzerkennung des Aufrufers voreingestellt. Nur der privilegierte Benutzer kann sich Memory Pools anzeigen lassen, an die Tasks einer anderen Benutzerkennung angeschlossen sind.

'<name 1..8>'
Angabe der Benutzerkennung.

<var: name 8..8>
Symbolische Adresse (Name) des Feldes, das die Benutzerkennung enthält (nur für MF=M).

(<reg:A(name 8..8))
reg = Register, das die Adresse des Feldes mit der Benutzerkennung enthält (nur für MF=M).

TSN=*OWN / '<name 1..4>' / <var: name 4..4> / (<reg: A(name 4..4)>)
Gibt für CONNECT=*BY-TASK die TSN der angeschlossenen Task an. Mit *OWN ist die TSN des Aufrufers voreingestellt. Der nicht-privilegierte Benutzer kann hier nur Tasks der eigenen Benutzerkennung angeben.

'<name 1..4>'
Angabe der TSN.

<var: name 4..4>
Symbolische Adresse (Name) des Feldes, das die TSN enthält (nur für MF=M).

(<reg:A(name 4..4))
reg = Register, das die Adresse des Feldes mit der TSN enthält (nur für MF=M).

INFO=
Bestimmt den Umfang der auszugebenden Informationen.

***STD**
Die Eigenschaften des Memory Pools und die Anzahl der angeschlossenen Tasks werden ausgegeben.

***ALL**
Zusätzlich zur Standardausgabe werden alle angeschlossenen Tasks mit ihrer TSN aufgelistet. Für nicht-privilegierte Benutzer werden nur die TSNs der eigenen Benutzerkennung ausgegeben. Für privilegierte Benutzer werden alle TSNs ausgegeben.

NUMSHR=45 / <integer 1..4096> / <var: int:4> / (<reg: integer 1..4096>)
*Operand wird nur für INFO=*ALL ausgewertet.*
Gibt an, wieviele Tasks maximal aufgelistet werden sollen.

<integer 1..4096>
Angabe der Anzahl der Tasks.

<var: int:4>
Symbolische Adresse (Name) des Feldes, das die Anzahl der Tasks enthält (nur für MF=M).

(<reg: integer 1..4096>)
reg = Register, das die Anzahl der Tasks enthält (nur für MF=M).

INFO_AREA=adr / (r)
Nur für MF=M.
Gibt die Adresse des Bereichs zur Informationsausgabe an. Der Bereich muss aus einer oder mehreren Klasse-6-Speicherseiten bestehen.

adr
Symbolische Adresse (Name) eines Hilfsfeldes, das die Adresse des Bereichs zur Informationsausgabe enthält.

(r)
r = Register mit dem Adresswert des Bereichs zur Informationsausgabe.

INFO_LENGTH=1 / <integer 1..1024> / <var: int:4> / (<reg: integer 1..1024>)
Angabe der Anzahl der Seiten für die Ausgabe.

<integer 1..4096>
Angabe der Anzahl der Seiten.

<var: int:4>
Symbolische Adresse (Name) des Feldes, das die Anzahl der Seiten enthält (nur für MF=M).

(<reg: integer 1..1024>)
reg = Register, das die Anzahl der Seiten enthält (nur für MF=M).

XPAND=
Steuerungs-Operand nur für MF=C und MF=D:
Es wird festgelegt, welche Struktur zu expandieren (erzeugen) ist. Angaben bei diesem Operanden werden bei anderen MF-Werten ignoriert.

PARAM
Das Layout der Parameterliste wird expandiert.

INFA
Das Layout des Informationsbereiches wird expandiert.

EQUATES=
Steuerungs-Operand nur für MF=C und MF=D:
Gibt an, ob bei der Expansion des Parameter- oder Informationsbereichs auch Equates für die Werte der Felder des Parameter- oder Informationsbereichs generiert werden sollen.

YES
Bei der Expansion des Parameter- oder Informationsbereichs werden auch Equates für die Werte der Felder des Parameter- oder Informationsbereichs generiert.

NO
Bei der Expansion des Parameter- oder Informationsbereichs werden keine Equates für die Werte der Felder des Parameter- oder Informationsbereichs generiert.

Programmierhinweise

Es gibt drei Paare von Operanden, die nicht zusammen verwendet werden dürfen und die sich gegenseitig ausschließen: MPNAME und MPNAMAD, GROUP_USERID und USER_GROUPID sowie USERID und TSN. Bei diesen Paaren gilt die Grundregel, dass die (in unterschiedlichen MF-Varianten) zuletzt gemachte Angabe maßgeblich ist und zu den restlichen Operanden passen muss.
Im Detail:

- Werden MPNAME oder MPNAMAD angegeben, so gilt immer die letzte dieser Angaben (d.h. die vom letzten MF=L- oder MF=M-Aufruf).
- Bei SCOPE=*GROUP muss zuletzt GROUP_USERID angegeben werden (nicht USER_GROUPID).
- Bei SCOPE=*USER_GROUP muss zuletzt USER_GROUPID angegeben werden (nicht GROUP_USERID).
- Bei SELECT=*PAR und CONNECT=*BY-USER muss zuletzt der Operand USERID angegeben werden (nicht TSN).
- Bei SELECT=*PAR und CONNECT=*BY-TASK muss zuletzt der Operand TSN angegeben werden (nicht USERID).

Informationen über die Memory Pools werden im Informationsbereich (Operanden INFO_AREA und INFO_LENGTH) bereitgestellt. Daneben gibt es im Parameterbereich zwei weitere Felder mit Ausgabeinformationen:

- Wenn die Funktion erfolgreich war oder teilweise ausgeführt wurde, dann wird die Anzahl Memory Pools, über die Informationen im Informationsbereich ausgegeben wurden, im Feld `&P.NPOL` bereitgestellt.
- Wenn die Größe des Klasse-6-Speichers (Operand INFO_LENGTH) nicht ausreicht (Return-Information: Funktion teilweise ausgeführt. Informationsbereich zu klein), dann wird die benötigte Seitenzahl im Parameterbereich (Feld `&P.INFX`) bereitgestellt. Sie kann für weitere Aufrufe von SHOWMP, z.B. in einer Laufschleife, verwendet werden.

Die Anzahl der Memory Pools kann sich zwischen zwei Aufrufen von SHOWMP ändern!

Layout des Parameterbereichs

```
&P._MDL      DSECT
*   subcode2
&P.NONE      EQU   0                    no further information
&P.NOMP      EQU   1                    no memory pool found
&P.NOCO      EQU   2                    no connection to memory pool
&P.MNER      EQU   3                    error in MPNAME specification
&P.MAER      EQU   4                    invalid MPNAMAD address
&P.MLER      EQU   5                    invalid MPNAMLN value
*                                       specified
&P.SCER      EQU   6                    error in SCOPE specification
&P.SEER      EQU   7                    error in SELECT specification
&P.PPER      EQU   8                    error in PRIV_POOL
*                                       specification
&P.COER      EQU   9                    error in CONNECT
*                                       specification
&P.GUER      EQU   10                   wrong USERID specified for
*                                       SCOPE=*GROUP
&P.GRER      EQU   11                   wrong GROUPID specified for
*                                       SCOPE=*USER-GROUP
&P.CUER      EQU   12                   wrong USERID specified for
*                                       CONNECT=*BY-USER
&P.CTER      EQU   13                   wrong TSN specified for
*                                       CONNECT=*BY-TASK
&P.IFER      EQU   14                   error in INFO specification
&P.NSER      EQU   15                   error in NUMSHR specification
&P.IAER      EQU   16                   invalid INFO_AREA address
&P.ILER      EQU   17                   invalid INFO_LENGTH value
*                                       specified
&P.FHDR  FHDR  MF=(C,&P.),EQUATES=NO    Standardheader

*   main return codes
&P.SUCC       EQU   0                   function processed
*                                       successfully
&P.PART       EQU   1                   function processed only
*                                       partially
&P.PAER       EQU   2                   parameter error
&P.INER       EQU   3                   internal error
&P.PRER       EQU   4                   privilege error
&P.PSAT       EQU   5                   paging area saturation
&P.MSAT       EQU   6                   main memory saturation
&P.USAT       EQU   7                   user space saturation
&P.SSAT       EQU   8                   system space saturation
*
&P.MPSS       DS    AL1                 MPSHOW FLAG
&P.MPNS       EQU   X'80'               *ALL OR MP NAME SPECIFIED
&P.CUSS       EQU   X'40'               *OWN OR USERID SPECIFIED
```

```
&P.TSNS       EQU   X'20'              *OWN OR TSN SPECIFIED
&P.GUSS       EQU   X'10'              *OWN/*ANY OR USERID SPECIFIED
&P.GRPS       EQU   X'08'              *OWN/*ANY OR GROUPID
*                                      SPECIFIED
&P.GUOA       EQU   X'04'              *ANY OR *OWN FOR USERID
*                                      SPECIFIED
&P.GROA       EQU   X'02'              *ANY or *OWN FOR GROUPID
*                                      SPECIFIED
&P.RES1       EQU   X'01'              RESERVED
*
&P.COPE       DS    FL1                SCOPE OF SHOWMP
*   scope of memory pools
&P.ANYS       EQU   0                  any memory pool scope
&P.GRP        EQU   1                  memory pools of scope=group
&P.UGRP       EQU   2                  memory pools of
*                                      scope=user-group
&P.GLB        EQU   3                  memory pools of
*                                      scope=user-global
*
&P.SELE       DS    FL1                SELECTION OF SHOWMP
*   select memory pools
&P.ALLS       EQU   0                  select all memory pool data
&P.PAR        EQU   1                  select specific data
*
&P.CONM       DS    FL1                CONNECTION OF SHOWMP
*   select connected memory pools
&P.ANYC       EQU   0                  select any memory pool
&P.USER       EQU   1                  select memory pools of user
*                                      ID
&P.TASK       EQU   2                  select memory pools of task
*
&P.PRPO       DS    FL1                PRIV-POOL OF SHOWMP
*   privilege of memory pools
&P.ANYP       EQU   0                  select memory pools whether
*                                      privileged or no
&P.YESP       EQU   1                  select only privileged memory
*                                      pools
&P.NOP        EQU   2                  select only non-privileged
*
&P.INFM       DS    FL1                INFO OF SHOWMP
*   type of information
&P.STD        EQU   0                  get data without TSN-list
&P.ALLI       EQU   1                  get data with TSN-list
*
&P.MPNM       DS    CL54               MEMORY POOL NAME
&P.MPNA       DS    A                  ADDRESS OF MEMORY POOL NAME
&P.MPLN       DS    F                  LENGTH OF MPNAME
&P.SCOPE_ID   DS    0XL8               USER- OR GROUP_ID
```

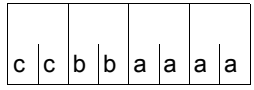
```
&P.GUSI       DS    CL8                 USER-ID BY SCOPE=GROUP
              ORG   &P.SCOPE_ID
&P.GRPI       DS    CL8                 GROUP-ID BY SCOPE=USER-GROUP
              ORG   &P.SCOPE_ID+8
&P.CONNECT_ID DS    OXL8                USER-ID OR TSN
&P.USID       DS    CL8                 USER-ID BY CONNECT=*BY-USER
              ORG   &P.CONNECT_ID
&P.TSN        DS    CL4                 TSN BY CONNECT=*BY-TSN
              ORG   &P.CONNECT_ID+8
&P.NRSH       DS    F                   # OF SHARER
&P.INFP       DS    A                   ADDRESS OF INFO AREA
&P.INFL       DS    F                   LENGTH OF INFO AREA IN PAGES
&P.NPOL       DS    F                   NUMBER OF FOUND MEMORY POOLS
&P.INFX       DS    F                   NEW LENGTH OF INFO AREA (PAGES)
&P.#          EQU   *-&P.FHDR
```

Layout des Informationsbereichs

```
&P.INFA       DSECT
*
*   info area description
&P.NXTM       DS    A                   NEXT MEMORY POOL
&P.MPNA       DS    CL54                MEMORY POOL NAME
&P.MPSC       DS    FL1                 SCOPE OF MEMORY POOL
*   scope of memory pools
&P.GRPO       EQU   1                   memory pools of scope=group
&P.UGRO       EQU   2                   memory pools of
*                                       scope=user-group
&P.GLBO       EQU   3                   memory pools of
*                                       scope=global
*
&P.RES        DS    XL1                 RESERVED BYTE
&P.GUID       DS    CL8                 GROUP- OR USER-ID
&P.NSHR       DS    F                   # OF SHARER TASKS
&P.STSN       DS    CL4                 TSN OF SHARER
&P.INFA#      EQU   *-&P.NXTM
```

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros SHOWMP wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Der Makro wurde normal ausgeführt.
X'01'	X'00'	X'0000'	Normale Ausführung. Kein passender Memory Pool im System
X'02'	X'00'	X'0000'	Normale Ausführung. Spezifizierter Memory Pool vorhanden, aber nicht konnektiert
X'00'	X'00'	X'0001'	Funktion teilweise ausgeführt. Informationsbereich zu klein. Die nötige Seitenzahl wird im Ausgabefeld &P.INFL geliefert.
X'03'	X'01'	X'0002'	Fehler im Operanden MPNAME (auch wenn MPNAMAD auf ein Feld mit fehlerhafter Namensangabe verweist)
X'04'	X'01'	X'0002'	Ungültige Adresse im Operanden MPNAMAD
X'05'	X'01'	X'0002'	Ungültiger Wert im Operanden MPNAMLN
X'06'	X'01'	X'0002'	Fehler im Operanden SCOPE
X'07'	X'01'	X'0002'	Fehler im Operanden SELECT
X'08'	X'01'	X'0002'	Fehler im Operanden PRIV_POOL
X'09'	X'01'	X'0002'	Fehler im Operanden CONNECT
X'0A'	X'01'	X'0002'	Falsche Benutzerkennung im Operanden USERID (SCOPE=*GROUP)
X'0B'	X'01'	X'0002'	Falsche Gruppenkennung im Operanden GROUPID (SCOPE=*USER-GROUP)
X'0C'	X'01'	X'0002'	Falsche Benutzerkennung im Operanden USERID (CONNECT=*BY-USER)
X'0D'	X'01'	X'0002'	Falsche TSN im Operanden TSN (CONNECT=*BY-TASK)
X'0E'	X'01'	X'0002'	Fehler im Operanden INFO
X'0F'	X'01'	X'0002'	Fehler im Operanden NUMSHR
X'10'	X'01'	X'0002'	Ungültige Adresse im Operanden INFO_AREA (auch bei fehlender Angabe bzw. fehlender Seitenausrichtung des Bereichs)
X'11'	X'01'	X'0002'	Ungültige Länge im Operanden INFO_LENGTH
X'00'	X'20'	X'0003'	Interner Fehler
X'00'	X'40'	X'0004'	Privileg für Aufruf nicht vorhanden
X'00'	X'80'	X'0005'	Engpass im Seitenwechselspeicher
X'00'	X'80'	X'0006'	Engpass im Hauptspeicher
X'00'	X'80'	X'0007'	Engpass im Benutzeradressraum

Weitere Returncodes, deren Bedeutung durch Konvention makro-übergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Programmbeispiel

```
         SHOWMP MF=D,XPAND=PARAM
         SHOWMP MF=D,XPAND=INFA
*
SHMEMPO  @ENTR TYP=I,LOCAL=SHMEMPL
SHMEMPL  @PAR  D=YES
SHMEMPC  SHOWMP MF=C,PREFIX=A,XPAND=PARAM
*
SHMPNAME DS    CL8              MPNAME
SHUSERID DS    CL8              USERID FOR GROUP
SHMEMPL  @PAR  LED=YES
...
         LA    R3,SHMEMPC
         USING NVPI_MDL,R3
         LA    R4,1
         MVC   SHMEMPC(NVPI#),SHMEMPD
         MVC   SHMPNAME,MEMPNAME COPY MPNAME
         MVC   SHUSERID,USERID   Assume: MYUID is own USERID
         LA    R9,SHMPNAME       A(MPNAME)
*
         SHOWMP MF=M,MPNAMAD=(R9),MPNAMLN=8,GROUP_USERID=SHUSERID
*
         @CYCL ,
*
         REQM (R4)              REQUEST CLASS 6 PAGES
         @WHEN NZ
         LTR  R15,R15
         @BREAK ,
*
         LR     R5,R1
         SHOWMP MF=M,INFO_AREA=(R5)
*
         SHOWMP MF=E,PARAM=(R3) GET MEMORY POOL DATA
         XR     R15,R15
         @WHEN  EQ
         CLC    NVPIMRET,=X'0000'
         @OR    GT
         CLC    NVPIMRET,=X'0001'
         @BREAK ,
*
         RELM   (R4),(R5)       RELEASE CLASS 6 PAGES
```

```
*
         L      R4,NVPIINFL     GET NEW PAGE NUMBER
         @BEND  ,
*
         @IF    ZE
         LTR    R15,R15
         @AND   EQ
         CLC    NVPIMRET,=X'0000'
         @AND NE
         CLC    NVPINPOL,=F'0'     AVOID CYCLE WITH LOOP COUNT 0
         @THEN ,
*
*        evaluation of returned info data
*
         L      R8,NVPINPOL
         LR     R7,R5           ADDRESS OF INFO AREA
         USING  NVPIINFA,R7
*
         @CYCL (R8)             LOOP OVER FOUND MEMPOOLS
           .
           .
         L      R7,NVPINXTM     NEXT MEMORY POOL
         @BEND ,
*
         @ELSE ,
*
*        error handling
*
         @BEND ,
...
         @END   ,
* DATA
SHMEMPD  SHOWMP MF=L,SCOPE=*GROUP,SELECT=*ALL,INFO=*ALL
*
MEMPNAME DC    CL8'HAUS*'           MPNAME WITH WILDCARD
USERID   DC    CL8'MYUID'           USERID FOR GROUP
```

SOLSIG – Signal anfordern

Allgemeines

Anwendungsgebiet: Ereignissteuerung; siehe Seite 95
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

Bei Verwendung der 24-Bit-Schnittstelle wird für den Post Code nur ein 4 Byte langes Feld generiert. Bei Verwendung der 31-Bit-Schnittstelle können 4 oder 8 Byte Post Code empfangen werden.

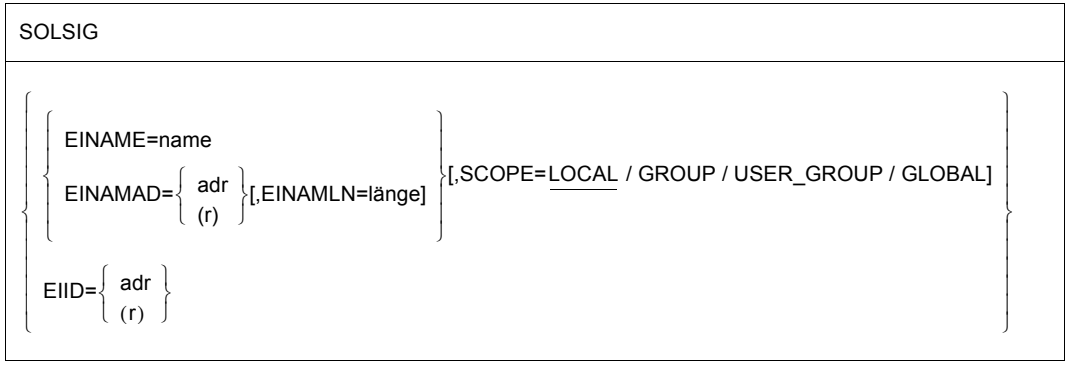
Makrobeschreibung

Der Makroaufruf wird benutzt, um eine „Anforderung für ein Signal“ für eine Ereigniskennung zu geben. Die Ereigniskennung muss zuvor der Task zugeordnet worden sein (**ENAEI**-Makroaufruf). Die Task kann warten, bis ein Ereignis eintritt (synchroner Fall), oder – wenn das Programm fortgesetzt werden soll – ein Contingency-Prozess auf Grund des Ereignisses initiiert wird (asynchroner Fall).
Die Task wird immer fortgesetzt (bzw. der Contingency-Prozess initiiert), wenn eine spezifizierte Zeitperiode überschritten ist - auch dann, wenn das Ereignis nicht eintritt.
Der Contingency-Prozess muss zuvor definiert worden sein (**ENACO**-Makroaufruf).

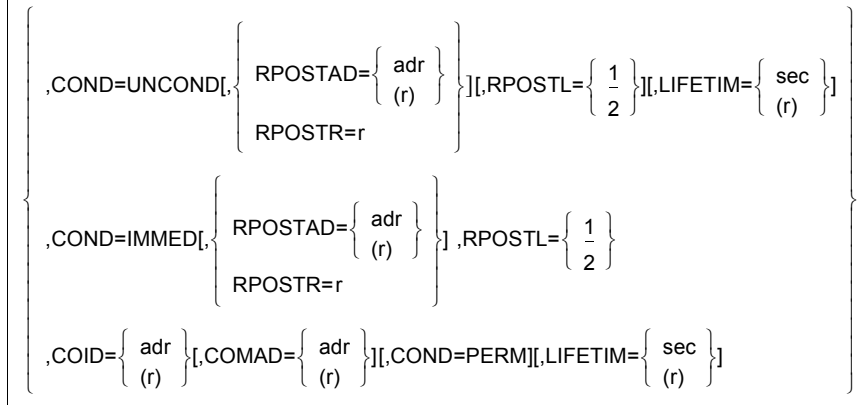
Hinweis

Wenn ein Programm(paket) einen Contingency-Prozess definiert hat, der in SPL geschrieben ist, muss bei **allen ENACO**-, **SOLSIG**- und **POSSIG**-Aufrufen das Register R12 die Adresse des SPL-Program Managers enthalten.

Makroaufrufformat und Operandenbeschreibung



SOLSIG (Fortsetzung)



[,PARMOD=24 / 31]

[,MF=L / (E,..)]

EINAME=name
gibt den Namen der Ereigniskennung an, der das angeforderte Ereignis gemeldet wird. Die Ereigniskennung muss bereits definiert worden sein (**ENAEI**). Der Name der Ereigniskennung ist nur zusammen mit dem Geltungsbereich (SCOPE) eindeutig.

EINAMAD=
bezeichnet den Namen der Ereigniskennung. Diese Angabe ist nur zusammen mit dem Geltungsbereich (SCOPE) eindeutig.

adr
symbolische Adresse des Feldes, das den Namen enthält.

(r)
r = Register, das die Adresse des Feldes enthält.

EINAMLN=länge
gibt die Länge des Namens der Ereigniskennung in Bytes an. Fehlt dieser Operand, so wird das Längenattribut des EINAMAD-Operandenwertes angenommen, wenn EINAMAD=adr angegeben ist; ist EINAMAD=(r) angegeben, so wird die maximale Länge (54 Byte) angenommen.

länge
Länge des Namens der Ereigniskennung.

SCOPE=
beschreibt den Geltungsbereich (Teilnehmerkreis) für die Ereigniskennung.

LOCAL
Die Ereigniskennung wird nur von der Task des Aufrufers benutzt.

GROUP
Teilnehmer sind alle Tasks unter der Benutzerkennung des Aufrufers.

USER_GROUP
Teilnehmer können alle Tasks sein, deren Benutzerkennungen der gleichen Benutzergruppe angehören wie die Benutzerkennung des einrichtenden Teilnehmers.
Der Operandenwert setzt die Existenz von Benutzergruppen voraus und kann daher nur angegeben werden, wenn die Funktionseinheit SRPM des Software-Produkts SECOS im System vorhanden ist. Vor einem Makroaufruf mit SCOPE=USER_GROUP muss deshalb mit dem Makro GETUGR (siehe Handbuch „SECOS“ [14]) geprüft werden, ob SRPM zur Verfügung steht; abhängig vom Ergebnis (Returncode) ist im Programm zu reagieren.

GLOBAL
Teilnehmer sind alle Tasks im System

EIID=
bezeichnet die Kurzkennung der Ereigniskennung. Die Kurzkennung wird dem Benutzer durch den **ENAEI**-Makroaufruf zur Verfügung gestellt. Die Verwendung der Kurzkennung an Stelle des Namens zur Identifizierung der Ereigniskennung beschleunigt die Verarbeitung und ist eindeutig.

adr
symbolische Adresse eines 4 Byte langen Feldes, das die Kurzkennung enthält.

(r)
r = Register, das die Adresse des Feldes enthält.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB).

Operanden für den synchronen Fall

COND=
bestimmt, ob der Aufrufer auf das Eintreten des Ereignisses warten will.

UNCOND
gibt an, dass der Aufrufer bereit ist, auf das Eintreten des Ereignisses zu warten. Die Wartezeit kann mit dem Operanden LIFETIM begrenzt werden.

IMMED
gibt an, dass der Aufrufer nicht auf das Eintreten des Ereigisses warten will. Das Programm soll fortgesetzt werden, auch wenn das Ereignis noch nicht durch einen **POSSIG**-Aufruf der Ereigniskennung gemeldet war. Auskunft hierüber gibt der Sekundäre Indikator X'20' in Register R15 (siehe Returncode).

RPOSTAD=
bezeichnet ein Feld, in das ein Post Code übertragen werden soll. Der Post Code kann 4 oder 8 Byte lang sein. Der Operand RPOSTL legt fest, ob nur das erste Wort oder beide Wörter des Post Codes in das Feld eingetragen werden.
Der Operand RPOSTR führt dieselbe Funktion aus wie RPOSTAD; die Verarbeitung erfolgt aber schneller.
Der Adresswert 0 ist nicht erlaubt.

adr
symbolische Adresse des Feldes, in das der Post Code eingetragen werden soll. Feldlänge = 4 oder 8 Byte.

(r)
r = Register mit dem Adresswert adr.

RPOSTR=r
bezeichnet ein Register, in das der Post Code direkt eingetragen werden soll. Ein Post Code aus zwei Worten bestehend, wird in das angegebene und das nachfolgende Register (der Zahl nach) eingetragen - wenn RPOSTL=2 angegeben wurde.

r
r = Register, in das der Post Code eingetragen werden soll.

RPOSTL=
bezeichnet die Anzahl der Worte des Post Codes, die empfangen werden sollen. Bei Verwendung der 24-Bit-Schnittstelle (PARMOD=24) kann nur RPOSTL=1 angegeben werden.

1
Von dem Post Code soll nur ein Wort (das erste Wort) übertragen werden.

2
Der Post Code soll in voller Länge (2 Worte) übertragen werden.

LIFETIM=
Zeit, während die Task auf das Eintreten des Ereignisses warten soll. Der Returncode gibt an, ob die Anforderung erfüllt oder die Wartezeit überschritten wurde. Der Operand wird ignoriert, wenn COND=IMMED angegeben wurde.

sec
Zeitangabe in Sekunden. $1 \leq sec \leq 43200$
Die Genauigkeit für die Bearbeitung liegt bei +10 Sekunden.
Voreinstellung: 600 sec.

(r)
r = Register, das die Angabe in Sekunden enthält.

Operanden für den asynchronen Fall

COID=
bezeichnet die Kurzkennung des Contingency-Prozesses. Die Kurzkennung wird dem Benutzer durch den **ENACO**-Makroaufruf zur Verfügung gestellt.

adr
symbolische Adresse eines 4 Byte langen Feldes, das die Kurzkennung enthält.

(r)
r = Register, das die Adresse enthält.

COMAD=
bezeichnet eine Contingency-Mitteilung. Eine hier gegebene Contingency-Mitteilung ersetzt eine eventuell bei der Definition der Contingency gegebene Mitteilung (**ENACO**-Makroaufruf).

adr
symbolische Adresse eines Wortes, das eine Contingency-Mitteilung enthält.

(r)
r = Register, das die Adresse enthält.

COND=PERM
Permanenter asynchroner **SOLSIG**:
trifft das durch **SOLSIG** angeforderte Signal innerhalb der Wartezeit (LIFETIM) ein, so wird erneut ein **SOLSIG** abgesetzt, dessen Wartezeit 600 Sekunden beträgt. Trifft während dieser Wartezeit kein Signal mehr ein, so wird kein weiterer **SOLSIG** abesetzt.

LIFETIM=
Zeit, während der das Ereignis eintreten soll. Der Ereignis-Informationscode teilt dem Contingency-Prozess mit, ob die Anforderung innerhalb der gesetzten Zeitperiode erfüllt oder nicht erfüllt wurde.

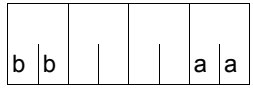
sec
Zeitangabe in Sekunden. $1 \leq sec \leq 43200$
Die Genauigkeit für die Bearbeitung liegt bei +10 Sekunden.
Voreinstellung: 600 sec.

(r)
r = Register, das die Angabe in Sekunden enthält.

Rückinformation und Fehleranzeigen

Während der Makrobearbeitung enthält Register R1 die Adresse der Operandenliste.

R15:



Über die Ausführung des Makros SOLSIG wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'bb'	X'aa'	Erläuterung
X'00'	X'00'	Funktion ausgeführt: – synchroner Fall: Das Ereignis ist eingetreten. Eine Post Code-Information wurde angeboten und übergeben bzw. keine Post Code-Information angeboten noch angefordert. – asynchroner Fall: Die SOLSIG-Anforderung wurde erfolgreich in die Warteschlange eingereiht.
X'30'	X'00'	Funktion ausgeführt: Eine Post Code-Information wurde angeboten (POSSIG), aber seitens des SOLSIG-Aufrufes kein Empfangsfeld (RPOSTAD) bereitgestellt.
X'34'	X'00'	Funktion ausgeführt: Es wurde keine Post Code-Information angeboten (POSSIG), aber seitens des SOLSIG-Aufrufs eine solche gewünscht (RPOSTAD). Post Code Null (X' 000000') wird nicht als Post Code betrachtet.
X'38'	X'00'	Funktion ausgeführt: Post Code ist länger als das angegebene Empfangsfeld. Das 2. Wort (4 Byte rechtsbündig) wurde abgeschnitten.
X'3C'	X'00'	Funktion ausgeführt: Post Code ist kürzer als das angegebene Empfangsfeld. Der Post Code wird linksbündig eingetragen.
X'0C'	X'04'	Keine Aktion: Die vom System erstellte Ereigniskennung ist dem SOLSIG aufrufenden Prozess nicht zugeordnet.
X'10'	X'04'	Keine Aktion: Es wurden ungültige Operanden angegeben.
X'14'	X'04'	Keine Aktion: Ungültiger Name bzw. ungültige Kurzkennung. Es existiert keine Ereigniskennung mit spezifizierter Identifikation.
X'18'	X'04'	Keine Aktion: Maximal erlaubte Anzahl (400) an Contingency-Prozessen pro Basisprozess überschritten.
X'20'	X'04'	Keine Aktion: Das Ereignis ist nicht eingetreten. Entweder COND=IMMED war gesetzt und der „Sende Signal (POSSIG)“-Aufruf war noch nicht gegeben oder die Wartezeit ist abgelaufen.
X'24'	X'04'	Keine Aktion: Ungültige Kurzkennung des Contingency-Prozesses. Es existiert kein Contingency-Prozess mit dieser Identifikation.
X'28'	X'04'	Keine Aktion: Die Ereigniskennung wurde gelöscht, bevor das Ereignis eintreten konnte.

Beispiel: Synchroner Fall

```
  SOLSIG   START
           PRINT NOGEN
  SOLSIG   AMODE ANY
           GPARMOD 31
1                *,MACRO: GPARMOD, VERSION: VER121
           BALR  3,0
           USING *,3
           ENAEI EINAME=EVENT,SCOPE=GLOBAL,EIIDRET=KK ——————————————————  (1)
           GDATE TOD=TIME1
           SOLSIG EIID=KK,COND=UNCOND ———————————————————————————————  (2)
  *** WAITING FOR SIGNAL ***
           GDATE TOD=TIME2
           ST    15,RCFIELD1
           GDATE TOD=TIME3
           SOLSIG EIID=KK,COND=UNCOND,LIFETIM=70 ————————————————————  (3)
  *** WAITING FOR SIGNAL ***
           GDATE TOD=TIME4
           ST    15,RCFIELD2
           GDATE TOD=TIME5
           SOLSIG EIID=KK,COND=IMMED ————————————————————————————————  (4)
           GDATE TOD=TIME6
           ST    15,RCFIELD3
           CHKEI EIID=KK ————————————————————————————————————————————  (5)
           ST    15,RCFIELD4
           DISEI EIID=KK
  DTH1     TERM
  ****   DEFINITIONS   *****
  KK       DS    F
  TIME1    DS    CL8
  TIME2    DS    CL8
  TIME3    DS    CL8
  TIME4    DS    CL8
  TIME5    DS    CL8
  TIME6    DS    CL8
  RCFIELD1 DS    F
  RCFIELD2 DS    F
  RCFIELD3 DS    F
  RCFIELD4 DS    F
           END
```

Ablaufprotokoll:

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,solsig), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,solsig)), -
//        test-support=*aid
%  ASS6011 ASSEMBLY TIME: 453 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 84 MSEC
//end
%  ASS6012 END OF ASSEMBH
/load-executable-program library=macexmp.lib,element-or-symbol=solsig, -
/     test-options=*aid
%  BLS0523 ELEMENT 'SOLSIG', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'SOLSIG', VERSION ' ' OF '<date> <time>' LOADED
/%in dth1;%r
STOPPED AT LABEL: DTH1 , SRC_REF: 287, SOURCE: SOLSIG , PROC: SOLSIG
/%d %@(time1) -> %cl8,%@(time2) -> %cl8 ------------------------------------  (6)
*** TID: 005000D8 *** TSN: 2QSE ***********************************************
**
CURRENT PC: 00000136    CSECT: SOLSIG  ***************************************
**
V'00000158' = SOLSIG   + #'00000158'
00000158 (00000158) 13:14:15
V'00000160' = SOLSIG   + #'00000160'
00000160 (00000160) 13:24:53
/%d %@(rcfield1) -> %x ----------------------------------------------------  (7)
V'00000188' = SOLSIG   + #'00000188'
00000188 (00000188) 20000004                                  ....
/%d %@(time3) -> %cl8,%@(time4) -> %cl8 ------------------------------------  (8)
V'00000168' = SOLSIG   + #'00000168'
00000168 (00000168) 13:24:53
V'00000170' = SOLSIG   + #'00000170'
00000170 (00000170) 13:26:02
/%d %@(rcfield2) -> %x ----------------------------------------------------  (9)
V'0000018C' = SOLSIG   + #'0000018C'
0000018C (0000018C) 20000004                                  ....
/%d %@(time5) -> %cl8,%@(time6) -> %cl8 ----------------------------------- (10)
V'00000178' = SOLSIG   + #'00000178'
00000178 (00000178) 13:26:02
V'00000180' = SOLSIG   + #'00000180'
00000180 (00000180) 13:26:02
```

```
/%d %@(rcfield3) -> %x ──────────────────────────────────────────────── (11)
V'00000190' = SOLSIG   + #'00000190'
00000190 (00000190) 20000004                                    ....
/%d %@(rcfield4) -> %x ──────────────────────────────────────────────── (12)
V'00000194' = SOLSIG   + #'00000194'
00000194 (00000194) 30000000                                    ....
/%r
```

(1) Die Ereigniskennung EVENT wird definiert. Die Adresse der Kurzkennung ist KK.

(2) Ein Signal wird von der Ereignissteuerung angefordert. Der Aufrufer ist bereit, bis zum Ablauf der standardmäßigen Wartezeit (10 Minuten) auf das Eintreffen des Signals zu warten.

(3) Ein zweites Signal wird angefordert. Diesmal beträgt die Wartezeit nur 70 Sekunden.

(4) Ein drittes Signal wird angefordert. Der Aufrufer ist nicht bereit zu warten. Da kein Signal vorliegt (POSSIG-Warteschlange der Ereigniskennung), wird die Task fortgesetzt.

(5) Die Warteschlangen der Ereigniskennung EVENT werden geprüft. Sie sind leer, da die Wartezeiten für alle **SOLSIG**-Aufrufe abgelaufen sind. Anschließend wird die Definition der Ereigniskennung gelöscht.

(6) Uhrzeit vor und nach dem **SOLSIG**-Aufruf mit Standard-Wartezeit: Die Wartezeit betrug 10 Minuten und 38 Sekunden.

(7) Returncode X'20000004' nach dem ersten **SOLSIG**-Aufruf: Das Ereignis ist nicht eingetreten.

(8) Uhrzeit vor und nach dem **SOLSIG**-Aufruf mit 70 Sekunden Wartezeit: Die Wartezeit betrug 69 Sekunden.

(9) Returncode X'20000004' nach dem zweiten **SOLSIG**-Aufruf: Das Ereignis ist nicht eingetreten.

(10) Uhrzeit vor und nach dem **SOLSIG**-Aufruf ohne Wartezeit: Keine Wartezeit.

(11) Returncode X'20000004' nach dem dritten **SOLSIG**-Aufruf: Das Ereignis ist nicht eingetreten.

(12) Ergebnis der Warteschlangenprüfung mit **CHKEI**: Die Warteschlangen enthalten keine Anforderungen.

Weitere **Beispiele** enthält der Abschnitt „Ereignisgesteuerte Verarbeitung (Eventing)“ (Seite 107) und der Abschnitt „Contingency-Prozesse“ (Seite 119).

SRMUINF – Benutzerinformationen aus dem Benutzerkatalog lesen

Allgemeines

Anwendungsgebiet: Abfragen und Zugriff zu Listen und Tabellen; siehe Seite 158
Makrotyp: S-Typ, MF-Format **3**: D-/C-/E-/L-/M-Form; siehe Seite 29

Für jeden Anwender (jede Benutzerkennung) wird von einem Benutzerverwalter ein Eintrag im Benutzerkatalog angelegt. Der Eintrag enthält u.a.:

- Benutzerkennung, Passwortberechtigung
- Angaben zu Systemressourcen, die der Anwender in Anspruch nehmen kann (CPU-Zeit, Speicherplatz,...)
- besondere Rechte des Anwenders (privilegierter Zugriff,...)
- Daten für die Abrechnung (Accounting)
- Versandinformationen (Mail- und E-Mail-Adressen)

Makrobeschreibung

Der Makro **SRMUINF** liest Daten aus dem Benutzerkatalog und überträgt sie in einen vorher spezifizierten Bereich. Je nach Angabe werden die Daten für die Abrechnung (Accounting), die anwenderspezifischen Daten oder der gesamte Eintrag einer Kennung aus dem Benutzerkatalog ausgegeben. Es ist zu unterscheiden:

- Makroaufruf unter der Kennung des nichtprivilegierten Benutzers:
  Es werden nur Daten des eigenen Eintrages ausgegeben.
- Makroaufruf unter der Kennung eines Benutzerverwalters (standardmäßig die Benutzerkennung TSOS oder bei Einsatz des Software-Produkts SECOS eine Benutzerkennung mit dem Privileg USER-ADMINISTRATION):
  - Ausgabe der Daten des eigenen Eintrags
  - Ausgabe der Daten weiterer Anwender (einzeln für jeden beliebigen Anwender oder der Reihe nach für mehrere Anwender)
- Makroaufruf unter der Kennung eines Benutzergruppenverwalters (nur möglich, wenn das Software-Produkt SECOS eingesetzt wird). Es können nur Daten eines lokalen Pubsets ausgegeben werden:
  - Ausgabe der Daten des eigenen Eintrags
  - Ausgabe der Daten weiterer Anwender, die zur Gruppe bzw. einer Untergruppe des Benutzergruppenverwalters gehören

Makroaufrufformat und Operandenbeschreibung

```
SRMUINF

INFO=*USER / *ALL / *ACCOUNT / *POSIX / *EMAIL / *ALL_EMAIL / adr / (r)
[,AREA@=adr / (r)]
[,AREA#=länge / adr / (r)]
,USERID=*OWN / *FIRST / 'userid' / adr / (r)
,PVS=*HOME / 'catid' / adr / (r)
,ACTION=*READ / *READNXT / *READSEQ / adr / (r)
,XPAND={ PARAM
       { OUTPUT,DATA=ALL / USER / ACCOUNT / POSIX / EMAIL / ALL_EMAIL }
,MF=D / C / L / M / E
[,PARAM=adr / (r)]
,PREFIX=S / p
,MACID=RMV / macid
```

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

ACTION=
gibt an, ob zu der angegebenen oder der darauf folgenden Benutzerkennung Daten aus dem Benutzerkatalog gelesen werden sollen.

***READ**
Der Eintrag für die angegebene Benutzerkennung wird aus dem Benutzerkatalog gelesen.

***READNXT**
Es wird der Eintrag aus dem Benutzerkatalog gelesen, der der angegebenen Benutzerkennung folgt.
Dieser Wert kann nur unter der Kennung eines Benutzerverwalters mit dem Systemprivileg „USER-ADMINISTRATION“ angegeben werden (dieses Systemprivileg ist standardmäßig der Benutzerkennung TSOS oder bei Einsatz des Software-Produkts SECOS einer entsprechend privilegierten Benutzerkennung zugeordnet).

***READSEQ**
Es wird der Eintrag aus dem Benutzerkatalog gelesen, der der angegebenen Benutzerkennung folgt. Zusätzlich wird die Benutzerkennung, deren Eintrag gelesen wurde, in die Parameterliste als Operand USERID eingetragen.

adr
symbolische Adresse (Name) des Feldes mit der Information, welcher Eintrag aus dem Benutzerkatalog übergeben werden soll.
Diese Angabe ist nur zusammen mit MF=M erlaubt.

(r)
r = Register mit dem Adresswert, an dem der Operandenwert des Operanden ACTION steht. Diese Angabe ist nur zusammen mit MF=M erlaubt.

AREA@=
bezeichnet die Adresse eines Ausgabebereichs, in den die Daten aus dem Benutzerkatalog übertragen werden.

adr
symbolische Adresse (Name) des Bereichs.

(r)
r = Register mit dem Adresswert, an dem die Ausgabe erfolgen soll. Diese Angabe ist nur zusammen mit MF=M erlaubt.

AREA#=
gibt die Länge des Ausgabebereiches an. Die Mindestlänge ist von der Menge der auszugebenden Informationen abhängig. Die Menge wird durch den Operanden INFO bestimmt. Bei unzureichender Längenangabe wird der Eintrag abgeschnitten (Returncode aa = X'10').

länge
ganzzahliger Wert, der die Länge des Ausgabebereiches in Byte angibt. länge kann die Werte 0, 1, ..., 4096 annehmen.

adr
symbolische Adresse (Name) eines Feldes, das die Länge des Ausgabebereiches enthält. Diese Angabe ist nur zusammen mit MF=M erlaubt.

(r)
r = Register mit dem Adresswert, an dem die Länge des Ausgabebereiches steht. Diese Angabe ist nur zusammen mit MF=M erlaubt.

DATA=
steuert die Beschreibung von spezifischen Ausgabebereichen. Der Operand wird nur bei Angabe von XPAND=OUTPUT ausgewertet.

ALL
Der vollständige Eintrag für die angegebene Benutzerkennung wird beschrieben, jedoch ohne den EMAIL- und POSIX-spezifischen Teil.

USER
Der anwenderspezifische Teil (Anwenderteil ohne Abrechnungsnummern) des Eintrags für die angegebene Benutzerkennung wird beschrieben.

ACCOUNT
Der abrechnungsspezifische Teil (nur Abrechnungsnummern) des Eintrags für die angegebene Benutzerkennung wird beschrieben.

POSIX
Der POSIX-spezifische Teil des Eintrags für die angegebene Benutzerkennung wird beschrieben.

EMAIL
Der EMAIL-spezifische Teil des Eintrags für die angegebene Benutzerkennung wird beschrieben.

ALL_EMAIL
Der vollständige Eintrag mit EMAIL-spezifischem Teil für die angegebene Benutzerkennung wird beschrieben, jedoch ohne den POSIX-spezifischen Teil.

INFO=
gibt die Menge der zu übertragenden Daten aus dem Eintrag des Benutzerkatalogs an.

***USER**
Es werden nur die anwenderspezifischen Daten übertragen.

***ALL**
Es wird der vollständige Eintrag übertragen, jedoch ohne den EMAIL- und POSIX-spezifischen Teil.

***ACCOUNT**
Es werden nur die abrechnungsspezifischen Daten übertragen.

***POSIX**
Es werden die POSIX-spezifischen Daten übertragen. Die selben Daten werden mit dem Kommando SHOW-POSIX-USER-ATTRIBUTES ausgegeben.

***EMAIL**
Es wird der Bereich mit den Empfängeradressen für E-Mails übertragen.

***ALL_EMAIL**
Es wird der vollständige Eintrag und der Bereich mit den Empfängeradressen für E-Mails übertragen, jedoch nicht der POSIX-spezifische Teil.

adr
symbolische Adresse (Name) eines Feldes, das die Information enthält, welche Menge von Daten übertragen werden sollen. Diese Angabe ist nur zusammen mit MF=M erlaubt.

(r)
r = Register mit dem Adresswert, an dem der Operandenwert des Operanden INFO steht. Diese Angabe ist nur zusammen mit MF=M erlaubt.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29 und aus dem Aufrufformat ersichtlich.
Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

PVS=
bezeichnet den PVS (**P**ublic **V**olume **S**et; „Pubset“), von dem ein Eintrag aus dem Benutzerkatalog gelesen werden soll.

***HOME**
Der gewünschte Eintrag ist im Benutzerkatalog des HOME-Pubset enthalten.

'catid'
catid = Katalogkennung des PVS, der den Benutzerkatalog mit dem gewünschten Eintrag enthält.
Maximale Länge der Zeichenkette catid = 4 Zeichen.

adr
symbolische Adresse (Name) eines 4 Byte langen Feldes, das die catid enthält. Kürzere catids sind mit Leerzeichen aufzufüllen. Diese Angabe ist nur zusammen mit MF=M erlaubt.

(r)
r = Register mit dem Adresswert, an dem die 4 Byte lange catid steht. Kürzere catids sind mit Leerzeichen aufzufüllen. Diese Angabe ist nur zusammen mit MF=M erlaubt.

USERID=
bezeichnet die Benutzerkennung, deren Eintrag aus dem angegebenen Benutzerkatalog angefordert wird.

***OWN**
Es wird der Eintrag der eigenen Benutzerkennung (Benutzerkennung des Aufrufers) ausgegeben.

***FIRST**
Es wird der erste Eintrag des angegebenen Benutzerkatalogs ausgegeben.
Der Wert darf nur zusammen mit dem Operanden ACTION=*READNXT und der dazugehörenden Privilegierung aufgerufen werden.

'userid'
userid = Benutzerkennung.
Maximale Länge der Zeichenkette userid = 8 Zeichen.
Welche Benutzerkennungen angegeben werden dürfen, hängt von den Privilegien des Aufrufers ab. Folgende Fälle sind zu unterscheiden:
– Aufrufer ist ein nichtprivilegierter Benutzer:

Zulässig ist nur die eigene Benutzerkennung.

– Aufrufer ist ein Benutzergruppenverwalter (nur möglich bei Einsatz des Software-Produkts SECOS):
  Erlaubt ist jede Benutzerkennung aus der Benutzergruppe des Aufrufers oder einer darunter liegenden Gruppe.
– Aufrufer arbeitet als Benutzerverwalter mit dem Systemprivileg „USER-ADMINISTRATION“ (standardmäßig die Benutzerkennung TSOS oder bei Einsatz des Software-Produkts SECOS die entsprechend privilegierte Benutzerkennung):
  Zulässig ist jede Benutzerkennung auf dem angegebenen PVS.

adr
symbolische Adresse (Name) eines 8 Byte langen Feldes, das die userid enthält. Diese Angabe ist nur zusammen mit MF=M erlaubt.

(r)
r = Register mit dem Adresswert von adr. Diese Angabe ist nur zusammen mit MF=M erlaubt.

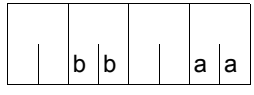
XPAND=
bestimmt, welcher Datenbereich bearbeitet werden soll.

PARAM
Es wird der Aufruf-Datenbereich bearbeitet.

OUTPUT
Es wird die Ausgabe eines speziellen Ausgabebereiches gewünscht. Der Operand DATA muss angegeben werden. Er bestimmt das Ausgabelayout der gewünschten Informationen.

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros SRMUINF wird im Standardheader folgender gegliederter Returncode übergeben (aa=primärer RC, bb=sekundärer RC):

X'bb'	X'aa'	Erläuterung
X'00'	X'00'	Normale Ausführung.
X'01'	X'04'	Operandenfehler oder unzureichende Privilegierung.
X'00'	X'08'	Es existiert kein Eintrag unter dieser Kennung.
X'80'	X'0C'	Auf den Pubset (PVS) kann nicht zugegriffen werden.
X'00'	X'10'	Der Eintrag für die Benutzerkennung wurde nicht vollständig ausgegeben (unzureichende Längenangabe).

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Layout der DSECT (1)

```
           SRMUINF MF=D,PREFIX=A
1          STACK  PRINT
1          PRINT  NOGEN
2               *,##### PREFIX=A, MACID=RMV #####
1          #INTF REFTYPE=REQUEST,INTNAME=SRMUINF,INTCOMP=001
1 ARMVPL     DS    OF          BEGIN of PARAMETERAREA
1          FHDR  MF=(C,ARMV),EQUATES=NO
2          DS    0A
2 ARMVFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
2 ARMVIFID DS    0A              0   INTERFACE IDENTIFIER
2 ARMVFCTU DS    AL2             0   FUNCTION UNIT NUMBER
2 *                                  BIT 15    HEADER FLAG BIT,
2 *                                  MUST BE RESET UNTIL FURTHER NOTICE
2 *                                  BIT 14-12 UNUSED, MUST BE RESET
2 *                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
2 ARMVFCT  DS    AL1             2   FUNCTION NUMBER
2 ARMVFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 ARMVRET  DS    0A              4   GENERAL RETURN CODE
2 ARMVSRET DS    0AL2            4   SUB RETURN CODE
2 ARMVSR2  DS    AL1             4   SUB RETURN CODE 2
2 ARMVSR1  DS    AL1             5   SUB RETURN CODE 1
2 ARMVMRET DS    0AL2            6   MAIN RETURN CODE
2 ARMVMR2  DS    AL1             6   MAIN RETURN CODE 2
2 ARMVMR1  DS    AL1             7   MAIN RETURN CODE 1
2 ARMVFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 *                                       ADDITIONAL MAIN RETURN-CODES:
1 *
1 ARMVMOK  EQU   0                        REQUEST SERVICED
1 ARMVPAER EQU   4                        PARAMETER ERROR
1 ARMVENFO EQU   8                        ENTRY NOT FOUND
1 ARMVPNAC EQU   12                       PVS NOT ACCESSIBLE
1 ARMVRSER EQU   16                       RESOURCE NOT AVAILABLE
1 *
1          DS    XL4                      UNUSED
1 ARMVUSRD DS    CL8                      USERID SPECIFIED
1 ARMVINFO DS    AL1                      INFO-TYPE
1 ARMVIALL EQU   1                          ALL REQUIRED
1 ARMVIUSR EQU   2                          USER INFO REQUIRED
1 ARMVIACI EQU   3                          ACCOUNT INFO REQUIRED
1 ARMVIPOS EQU   4                          POSIX INFO REQUIRED
1 ARMVIEMA EQU   5                          EMAIL INFO REQUIRED
1 ARMVIAEM EQU   6                          ALL&EMAIL INFO REQUIRED
1 ARMVACOD DS    AL1                      ACTION CODE
```

```
1 ARMVACRD EQU   1                        READ
1 ARMVACRN EQU   2                        READ NEXT
1 ARMVPVS  DS    CL4                    PVS CATID
1 ARMVHOME EQU   C'#'                   PVS IS HOME
1          DS    CL4                    FOR CAT-ID EXTENSION
1          DS    XL2                    UNUSED1
1 ARMVARE@ DS    A                      AREA ADDRESS
1 ARMVARE# DS    AL2                    AREA LENGTH
1          DS    XL2                    UNUSED2
1 *
1 ARMVPL#  EQU   *-ARMVPL                P/L LENGTH
```

Layout der DSECT (2)

```
           SRMUINF MF=D,XPAND=OUTPUT,DATA=ALL_EMAIL
1          STACK  PRINT
1          PRINT  NOGEN
2               *,##### PREFIX=S, MACID=RMV #####
1 ************************************************************
1 *        INFORMATION ON JOB                                *
1 ************************************************************
1 SRMVJOB  DS    OF
1 SRMVUSER DS    CL8                  USER-IDENTIFICATION
1 SRMVSVR  DS    CL1                  SEVER INDICATOR
1 SRMVSVRY EQU   1                    - USERID HAS BEEN SEVERED
1 SRMVSVRN EQU   2                    - USERID NOT      SEVERED
1 SRMVPVG  DS    CL1                  USER PRIVILEGE CODE
1 SRMVPVGA EQU   1                    - SYSTEM ADMINISTRATOR
1 SRMVPVGN EQU   2                    - NORMAL USER
1 SRMVPASS DS    CL8                  USER'S PASSWORD
1 SRMVENCR DS    CL1                  PASSWORD-ENCRYPTION
1 SRMVENUN EQU   0                    - UNDEFINED
1 SRMVENNO EQU   1                    - NO ENCRYPTION
1 SRMVESCA EQU   2                    - ENCRYPT-MODE SCA
1 SRMVEOLD EQU   3                    - ENCRYPT-MODE OLD
1 SRMVENOP EQU   4                    - NO PASSWORD
1          DS    XL1                  NOT USED
1 SRMVJOB#  EQU   *-SRMVJOB               LENGTH OF JOB PART
1 ************************************************************
1 *        INFORMATION ON DMS                                *
1 ************************************************************
1 SRMVDMSD DS    OF
1 SRMVSPLI DS    F                    SPACE LIMIT
1 SRMVSPUS DS    F                    SPACE USED
1          DS    XL2                  RESERVED
1 SRMVIPSE DS    CL1                  PUBLIC SPACE EXCESS IND
1 SRMVIPEN EQU   1                    - NOT PERMITTED
```

```
1 SRMVIPET EQU   2                  - TEMPORARY
1 SRMVIPEY EQU   3                  - PERMITTED
1 SRMVTPIG DS    CL1                TPIGNORE  INDICATOR
1 SRMVTPIY EQU   1                  - YES : ERROR MESSAGES IGNORED
1 SRMVTPIN EQU   2                  - NO  : NO TPIGNORE PRIVILEGE
1 SRMVTPIR EQU   3                  - READ: ERROR MSG IGNORED-INPU
1 SRMVTPIB EQU   4                  - BLP : BY-PASS-LABEL
1 SRMVTPIA EQU   5                  - ALL : BLP AND YES PRIVILEGES
1 SRMVDFCT DS    CL4                DEFAULT CAT-ID OF PVS
1          DS    CL4                NOT USED
1 SRMVTSPL DS    F                  TEMP SPACE LIMIT
1 SRMVTSPU DS    F                  TEMP SPACE USED
1 SRMVFILI DS    F                  FILE LIMIT
1 SRMVFILA DS    F                  FILE AMOUNT
1 SRMVJVLI DS    F                  JV LIMIT
1 SRMVJVA  DS    F                  JV AMOUNT
1 SRMVEXHC DS    CL8                EXTENDED HOST CODE
1 SRMVDMTR DS    X                  DMS TUNING RESOURCES
1 SRMVDTRN EQU   1                      NONE
1 SRMVDTRP EQU   2                      PAGEABLE
1 SRMVDTRR EQU   3                      RESIDENT
1 SRMVPAL  DS    X                  PHYSICAL ALLOCATION
1 SRMVPALN EQU   1                      NOT-ALLOWED
1 SRMVPALA EQU   2                      ALLOWED
1          DS    XL2                  UNUSED
1 SRMVUS1L DS    F                  S1 LEVEL USED
1 SRMVUS2L DS    F                  S2 LEVEL USED
1 SRMVDMCL DS    CL8                DEFAULT MANAGEMENT CLASS
1 SRMVDSCL DS    CL8                DEFAULT STORAGE CLASS
1 *                                 PERM SPACE LIMIT
1 SRMVLTOT DS    F                    TOTAL SPACE
1 SRMVLSTD DS    F                    SO LEVEL SPACE
1 SRMVLSHP DS    F                    HIGH PERF SPACE
1 SRMVLSVP DS    F                    VERY HIGH PERF SPACE
1 SRMVLSHA DS    F                    HIGH AVAIL SPACE
1 *                                 TEMP SPACE LIMIT
1 SRMVLTMP DS    F                    TOTAL SPACE
1 SRMVLTHP DS    F                    HIGH PERF SPACE
1 SRMVLTVP DS    F                    VERY HIGH PERF SPACE
1 *                                 WORK SPACE LIMIT
1 SRMVLWRK DS    F                    TOTAL SPACE
1 SRMVLWHP DS    F                    HIGH PERF SPACE
1 SRMVLWVP DS    F                    VERY HIGH PERF SPACE
1 *                                 PERM SPACE USED
1 SRMVUTOT DS    F                    TOTAL SPACE
1 SRMVUSTD DS    F                    SO LEVEL SPACE
1 SRMVUSHP DS    F                    HIGH PERF SPACE
1 SRMVUSVP DS    F                    VERY HIGH PERF SPACE
```

```
1 SRMVUSHA DS    F                       HIGH AVAIL SPACE
1 *                                     TEMP SPACE USED
1 SRMVUTMP DS    F                       TOTAL SPACE
1 SRMVUTHP DS    F                       HIGH PERF SPACE
1 SRMVUTVP DS    F                       VERY HIGH PERF SPACE
1 *                                     WORK SPACE USED
1 SRMVUWRK DS    F                       TOTAL SPACE
1 SRMVUWHP DS    F                       HIGH PERF SPACE
1 SRMVUWVP DS    F                       VERY HIGH PERF SPACE
1 SRMVCYSL DS    F                      CRYPTO SESSION LIMIT
1 SRMVCYSU DS    F                      CRYPTO SESSION USED
1 SRMVNST  DS    X                      NET-STORAGE USAGE
1 SRMVNSTN EQU   1                          NOT-ALLOWED
1 SRMVNSTA EQU   2                          ALLOWED
1          DS    XL3                    UNUSED
1 SRMVNCS  DS    CL8                    NET-CODED-CHAR-SET
1 SRMVDMS# EQU   *-SRMVDMSD              LENGTH OF DMS PART

1 ***************************************************************
1 *        INFORMATION ON TASK/PROGRAM                          *
1 ***************************************************************
1 SRMVTASK DS    0F
1 SRMVUSW  DS    CL4                    USER'S SWITCHES
1 SRMVAIDR DS    CL1                    AID'S READ VALUE
1 SRMVAIDW DS    CL1                    AID'S WRITE VALUE
1 SRMVTPRV DS    CL1                    TESTPRIV INDICATOR
1 SRMVTPRY EQU   1                      - TESTPRIV IS YES
1 SRMVTPRN EQU   2                      - TESTPRIV IS NO
1 SRMVADS2 DS    CL2                    ADDRSPACE VALUE
1 SRMVADIT DS    CL1                    AUDIT INDICATOR
1 SRMVADTY EQU   1                      - AUDIT ALLOWED
1 SRMVADTN EQU   2                      - AUDIT NOT ALLOWED
1 SRMVPSW  DS    CL1                    PSWORD CMD'S RIGHT
1 SRMVPSWY EQU   1                      - PSWORD YES
1 SRMVPSWM EQU   2                      - PSWORD MOD
1 SRMVPSWN EQU   3                      - PSWORD NO
1 SRMVCSMP DS    CL1                    CSTMP-MACRO INDICATOR
1 SRMVCSMY EQU   1                      - CSTMP-MACRO ALLOWED
1 SRMVCSMN EQU   2                      - CSTMP-MACRO NOT ALLOWED
1 SRMVHWAU DS    CL1                    HARDWARE-AUDIT INDICATOR
1 SRMVHWAUY EQU   1                      - HW-AUDIT ALLOWED
1 SRMVHWAUN EQU   2                      - HW-AUDIT NOT ALLOWED
1 SRMVLKAU DS    CL1                    LINKAGE-AUDIT INDICATOR
1 SRMVLKAUY EQU   1                      - LNK-AUDIT ALLOWED
1 SRMVLKAUN EQU   2                      - LNK-AUDIT NOT ALLOWED
1 SRMVPRID DS    CL54                   PROFILE ID
1 SRMVRPAG DS    H                      NB OF RESIDENT PAGES
1 SRMVDTKL DS    CL1                    DEFAULT-MESSAGE-LANGUAGE
```

```
1 SRMVMSGS DS    CL1                    DEFAULT-MESSAGE-SEARCH
1 SRMVSTSK EQU   X'01'                  -TASK
1 SRMVSALL EQU   X'02'                  -*ALL
1 SRMVADRS DS    F                      ADDRSPACE VALUE
1 SRMVRPGS DS    F                      NB OF RESIDENT PAGES
1 SRMVTSK# EQU   *-SRMVTASK               LENGTH OF TASK PART
1 ************************************************************
1 *        INFORMATION ON SPOOL                                *
1 ************************************************************
1 SRMVSPOL DS    0F
1          DS    CL8                    UNUSED
1 SRMVMAIL DS    CL64                   MAILING ADDRESS
1 SRMVSPL# EQU   *-SRMVSPOL               LENGTH OF SPOOL PART
1 *-
1 SRMVUSR# EQU   *-SRMVJOB                LENGTH OF USER PART
1 ************************************************************
1 *        INFORMATION ON ACCOUNTING                           *
1 ************************************************************
1 SRMVACCT DS    0F
1 SRMVNBAC DS    H                      NUMBER OF ACCOUNT-NB
1 SRMVMARC DS    H                      MAXIMUM OF ACCOUNT RECORDS
1 *-
1 SRMVACC# EQU   *-SRMVACCT               LENGTH OF ACC GLOBAL INF.
1 *-
1 *-       INFORMATION ON ACCOUNT ENTRY
1 *-
1 SRMVACCE DS    0F
1 SRMVACT  DS    CL8                    ACCOUNT NUMBER
1 SRMVCPU  DS    F                      CPU  TIME REMAINING
1 SRMVNTL  DS    CL1                    NTL INFORMATION
1 SRMVNTLY EQU   1                      - NTL ALLOWED
1 SRMVNTLN EQU   2                      - NTL NOT ALLOWED
1 SRMVEXP  DS    CL1                    EXPRESS  INFORMATION
1 SRMVEXPY EQU   1                      - EXPRESS  ALLOWED
1 SRMVEXPN EQU   2                      - EXPRESS  NOT ALLOWED
1 SRMVSCLA DS    CL1                    SPOOL-OUT CLASS
1 SRMVPRI  DS    CL1                    HIGHEST  PRIORITY PERMITTED
1 SRMVINHD DS    CL1                    INHIBIT  DEACTIVATION IND
1 SRMVINHY EQU   1                      - INHIBIT DEACTIVATION ALLOW
1 SRMVINHN EQU   2                      - INHIBIT DEACT NOT    ALLOW
1 SRMVTYPL DS    CL1                    UPPER LIMIT OF TASK TYPE
1 SRMVTYST EQU   1                      - STD
1 SRMVTYTP EQU   2                      - TP
1 SRMVTYSY EQU   3                      - SYS
```

```
1 ************************************************************/
1 * INFORMATION FOR POSIX  VERSION V11.2                      */
1 ************************************************************/
1 SRMVPOA  DS    X                       POSIX ACCOUNTNUMBER
1 SRMVPOAY EQU   1                       1: YES POSIX ACCOUNTNUMBER
1 SRMVPOAN EQU   2                       2: NO  POSIX ACCOUNTNUMBER
1 SRMVDAC  DS    X                       DEFAULT ACCOUNTNUMBER
1 SRMVDACY EQU   1                       - YES DEFAULT ACCOUNTNUMBER
1 SRMVDACN EQU   2                       - NO DEFAULT ACCOUNTNUMBER
1 SRMVACE# EQU   *-SRMVACCE                LENGTH OF ONE ACC ENTRY
1 *-
1          DS    CL(59*SRMVACE#)
1 *
1 SRMVACM# EQU   60*SRMVACE#+SRMVACC#      MAX LENGTH OF ALL ACCOUNT
1 *-                                     ENTRIES,GLOBAL INF
SRMVALL#   EQU   *-SRMVJOB                 MAX LENGTH OF A ALL ENTRY
************************************************************
*        INFORMATION ON EMAIL V17.0                         *
************************************************************
SRMVEMA    DS    0F
SRMVEMAL   DS    H                         EMAIL-LENGTH
SRMVEMAI   DS    CL1800                    EMAIL
SRMVEMA#   EQU   *-SRMVEMA                 LENGTH OF EMAIL-INFO
1 *-
1 SRMVETR# EQU   SRMVUSR#+SRMVACM#         MAX LENGTH OF A USER ENTRY
```

Beispiel zur Ausgabe von Benutzerkennung und Default-Pubset

```
SRMUINF  START
SRMUINF  RMODE ANY
SRMUINF  AMODE ANY
         GPARMOD 31
         BALR  3,0
         BCTR  3,0
         BCTR  3,0
         USING SRMUINF,3
         SRMUINF AREA@=SRMAUS,AREA#=SRMVLLLL,INFO=*ALL,MF=M ----------  (1)
         SRMUINF MF=E,PARAM=SRMPL ------------------------------------  (2)
         CLI   SRMVMR1,SRMVMOK          * ERROR IN CALL ?
         BNE   ERROR
         MVC   USERID,SRMVUSER          * CALL OK; OUTPUT USER ID
         MVC   DEFPUB,SRMVDFCT          * AND DEFAULT PUBSET
         WROUT MSGAUSG,0
         B     ENDE
ERROR    EQU   *                        * ERROR IN CALL (RC <> 0);
         UNPK  RC(5),SRMVMR1(2)         * OUTPUT RETURN CODE
         TR    RC(4),TAB
         WROUT MSGFEHL,0
* Definitions *
TAB      DS    CL240
         DC    C'0123456789ABCDEF'
*
* Parameter area SRMUINF macro
         DS    0F
SRMPL    SRMUINF MF=C
         ORG SRMVPL
         SRMUINF MF=L
         ORG
*
* Output area SRMUINF macro
         DS    0F
SRMAUS   SRMUINF MF=C,XPAND=OUTPUT,DATA=ALL
*
* Output area if call OK
MSGAUSG  DS    0H
         DC    Y(MSGAUSGL)
         DC    CL3' '
         DC    C'SRMUINF: User ID: '
USERID   DS    CL8
         DC    C'   , Default pubset: '
DEFPUB   DS    CL4
         DC    C'<'
MSGAUSGL EQU   *-MSGAUSG
*
```

```
* Output area if call contains error
MSGFEHL  DS    0H
         DC    Y(MSGFEHLL)
         DC    CL3' '
         DC    C'Error in SRMUINF: RC = '
RC       DS    CL5
MSGFEHLL EQU   *-MSGFEHL-1
SRMVLLLL DC    Y(SRMVETR#)             * LENGTH OF OUTPUT AREA
         END
               =AL4(SRMAUS)
```

Ablaufprotokoll

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,srmuinf), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,srmuinf))
%  ASS6011 ASSEMBLY TIME: 438 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 151 MSEC
//end
%  ASS6012 END OF ASSEMBH
/start-executable-program library=macexmp.lib,element-or-symbol=srmuinf
%  BLS0523 ELEMENT 'SRMUINF', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'SRMUINF', VERSION ' ' OF '<date> <time>' LOADED
SRMUINF: User ID: QM212      , Default pubset: 2OSG<
```

(1) Datenbereich initialisieren

(2) Lesen der Benutzerinformation

STAMCE – MRSCAT-Einträge lesen

Allgemeines

Anwendungsgebiet: Mehrrechnersysteme; siehe Seite 168
Makrotyp: S-Typ, MF-Format **2**: Standardform/E-/L-/C-/D-/M-Form); siehe Seite 29

Auf jedem Pubset befindet sich ein Dateikatalog TSOSCAT, der über seine Katalogkennung eindeutig identifiziert werden kann. Weiter ist jeder TSOSCAT (und damit jeder Pubset) eindeutig einem System (BS2000 native oder VM2000-Gastsystem) zugeordnet.
Der Home-Pubset enthält im TSOSCAT seinen Eintrag aus dem Katalogverzeichnis MRSCAT, einem Verzeichnis aller im Rechnernetz vorhandenen Pubsets. Generell kann jeder Pubset einen vollständigen MRSCAT enthalten.
Ein Eintrag im MRSCAT enthält u.a.:
– die Katalogkennung eines TSOSCATs
– den BCAM-Namen des Systems, dem dieser TSOSCAT zugeordnet ist
– den Gerätetyp für den zugehörigen Pubset
– Statusinformationen über den Pubset
– pubsetspezifische Operanden (EAM, Cache, Allocator,..)

Makrobeschreibung

Der Makro **STAMCE** übergibt einen Auszug aus einem ausgewählten Eintrag oder aus allen Einträgen des MRS-Katalogs (MRSCAT) in einen Ausgabebereich, der entweder vom Anwender definiert oder vom Makro angelegt wird. Die Auszüge informieren über die im Rechnernetz initialisierten Pubsets bezüglich

– Gerätedaten (Katalogkennung; Gerätetypcode; BCAM-Name des Systems, wenn der Pubset von einem fernen System verwaltet wird; ...).
– Status (Home-Pubset, lokaler Pubset, Pubset nicht zugreifbar, ...).
– Verwendungszweck (Paging, Datenzugriff, ...).
– Eigenschaften (EAM-Operand, Cache-Konfiguration,..)
– Nur für Systemverwaltung oder Operator:
  Anzahl, TSNs und Benutzerkennungen aller Aufträge, die den Pubset belegen (sei es durch geöffnete Dateien oder durch Reservierungsanforderungen (SECURE-Locks), Anzahl und Lage der CMS-Puffer gemäß Voreinstellung und aktueller Einstellung,..)

Ein Einzeleintrag wird über die Katalogkennung des Pubsets identifiziert. Es werden nur die Einträge des MRSCAT auf dem Home-Pubset ausgegeben.

Hinweise

– Sowohl der Datenbereich als auch der Ausgabebereich wurden in ihrem Layout den funktionalen Erweiterungen der einzelnen BS2000-Versionen angepasst. Aus Kompatibilitätsgründen ist es möglich, mit dem **STAMCE**-Makro auch die Layouts älterer Versionen zu erzeugen (Operand VERSION).
– Bei MF=M gilt folgende Einschränkung:
  Werden die Operanden CATID oder AREA angegeben, darf Register R15 nicht als Basisregister für den Datenbereich oder für die durch diese Operanden bestimmten Felder genutzt werden.

Makroaufrufformat und Operandenbeschreibung

```
STAMCE

CATID='␣' / 'catid' / adr / (r)
[,AREA=adr / (r)]
[,LENGTH=länge]
,REF=NO / YES / ALL
[,HOST=*LOCAL / '*LOCAL' / *ALL / '*ALL' / 'bcam-name' / adr]
,SELECT=ALL / ACCESSIBLE / DEF_XCS_CONF / EXCLUSIVE / HSMS_SUPPORTED / INACCESSIBLE /
        LOCAL / LOCAL_ACCESSIBLE / MASTER_CHANGE_ERROR / PAGING / QUIET / REMOTE /
        REMOTE_ACCESSIBLE / SCA / SHARED / SINGLE_FEATURE / SYSTEM_MANAGED /
        UNUSED_VOLSETS / VOLUME_SETS / XCS_CONFIGURATED
[,PUBSET=*ALL / 'catid' / adr]
[,XPAND=PL / *ALL / MCE / OCC]
[,VERSION=5 / 3]
,MF=S / E / L / C / D / M
[,PARAM=adr / (r)]
,PREFIX=D / p
[,MACID=macid]
```

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

AREA=
vereinbart die Adresse eines Ausgabebereiches, in dem **STAMCE** die angeforderten MRSCAT-Einträge übergibt. Wenn der Operand nicht angegeben wird, fordert **STAMCE** für die Ausgabe Klasse-6-Speicher an (in Einheiten von Hauptspeicherseiten zu je 4096 Byte) und versorgt die Felder DMCEAREA und DMCEARLN des Datenbereichs mit der Adresse und der Länge (in Byte) dieses Ausgabebereiches. Für die Rückgabe des vom Makro angeforderten Speicherbereiches ist der Aufrufer zuständig (Makro **RELM**).
Bei MF=S darf AREA nur zusammen mit dem Operanden LENGTH angegeben werden.

adr
symbolische Adresse (Name) eines Feldes zur Aufnahme der angeforderten MRSCAT-Einträge. Das Feld ist auf Wortgrenze auszurichten.

(r)
r = Register mit dem Adresswert adr. Die Angabe ist nur zusammen mit MF=M möglich.

CATID=
legt die Katalogkennungen der Pubsets fest, deren MRSCAT-Einträge ausgegeben werden sollen.

' '
Das in Hochkommas eingeschlossene Leerzeichen ist Voreinstellung.
Eine engere Auswahl der auszugebenden MRSCAT-Einträge kann mit dem Operanden SELECT getroffen werden.

'catid'
catid ist entweder eine explizit angegebene Katalogkennung oder ein Wildcard-Ausdruck zur Auswahl von Katalogkennungen. Die angegebene Zeichenfolge ist stets in Hochkommas einzuschließen.
Abhängig davon, ob die Katalogkennung explizit oder als Wildcard-Ausdruck angegeben wird, gelten für das Format von catid folgende Regeln:

Explizite Angabe der Katalogkennung:
– 'catid' ist entweder eine ein- bis vierstellige Zeichenfolge, bestehend aus den Buchstaben A,...,Z und den Ziffern 0,...,9, wobei die Zeichenfolgen PUB und PUBx (x = beliebiges Zeichen) ausgeschlossen sind
– oder 'catid' ist das Zeichen # (für den Home-Pubset)

Angabe als Wildcard-Ausdruck:
'catid' ist ist eine ein- bis vierstellige Zeichenfolge, die außer Buchstaben und Ziffern noch folgende Platzhalterzeichen enthalten darf:

*	Platzhalter für eine beliebige (auch leere) Zeichenfolge
/	Platzhalter für (genau) ein beliebiges Zeichen
<s1>	Platzhalter für die in den Klammern angegebene Zeichenfolge s1 (maximal zwei Zeichen)
-	Negationszeichen für die sich anschließenden Angaben (nur als erstes Zeichen des Ausdrucks zulässig).

Eine engere Auswahl der auszugebenden MRSCAT-Einträge kann mit dem Operanden SELECT getroffen werden.

adr
symbolische Adresse (Name) eines Feldes, das entweder eine explizit angegebene Katalogkennung oder einen Wildcard-Ausdruck zur Auswahl von Katalogkennungen enthält. In jedem dieser Fälle muss der Aufrufer diese Informationen im jeweils zutreffenden Format hinterlegen:

Explizite Angabe der Katalogkennung:

Im Feld ist eine der beiden folgenden Angaben zu hinterlegen:

– eine Zeichenfolge der Form `[:]catid[␣ / :]`
  Darin steht catid für eine ein- bis vierstellige Zeichenfolge, bestehend aus den Buchstaben A,...,Z und den Ziffern 0,...,9, wobei die Zeichenfolgen PUB und PUBx (x = beliebiges Zeichen) ausgeschlossen sind. Wenn catid kürzer als vier Zeichen ist, muss sie (wie oben dargestellt) am rechten Ende durch ein Leerzeichen (␣; X'40') oder durch einen Doppelpunkt (:) begrenzt werden. Bei vierstelligen Katalogkennungen kann dieser Begrenzer entfallen.

– das Zeichen # (für den Home-Pubset)
  Angabe als Wildcard-Ausdruck:
  Im Feld ist eine Zeichenfolge nach folgendem Muster zu hinterlegen:
  `[:]catid[␣ / :]`
  Darin steht catid für einen Wildcard-Ausdruck, der – ggf. einschließlich des rechten Begrenzungszeichens – bis zu 256 Zeichen lang sein kann und neben Buchstaben und Ziffern noch folgende Platzhalterzeichen enthalten darf:

*	Platzhalter für eine beliebige (auch leere) Zeichenfolge
/	Platzhalter für (genau) ein beliebiges Zeichen
<s1:s2>	Platzhalter für eine Zeichenfolge, die in der lexikografischen Ordnung zwischen den Zeichenfolgen s1 und s2 liegt
<s1,...>	Platzhalter für eine der in den Klammern aufgeführten Zeichenfolgen
-	Negationszeichen für die sich anschließenden Angaben (nur als erstes Zeichen des Ausdrucks zulässig)

  Ist der Wildcard-Ausdruck kürzer als 256 Zeichen, so muss er - wie oben dargestellt - am rechten Ende durch ein Leerzeichen (␣; X'40') oder durch einen Doppelpunkt (:) begrenzt werden. Nur bei Ausdrücken mit 256 Zeichen kann dieser Begrenzer entfallen.

  Eine engere Auswahl der auszugebenden MRSCAT-Einträge kann mit dem Operanden SELECT getroffen werden.

(r)
r = Register mit dem Adresswert adr.
Die Angabe ist nur zusammen mit MF=M möglich.

Hinweis

Werden bei MF=M die Operanden CATID oder AREA angegeben, darf Register R15 nicht als Basisregister für den Datenbereich oder für die durch diese Operanden bestimmten Felder genutzt werden.

HOST=
gibt unter den auf einen Pubset gemeinsam zugriffsberechtigten Systemen dasjenigen an, über dessen Tasks die mit REF=ALL angeforderten Informationen ausgegeben werden sollen.
Der Operand ist nur wirksam, wenn er zusammen mit REF=ALL angegeben wird, andernfalls wird er ignoriert.

***LOCAL**
'*LOCAL'
Die mit REF=ALL angeforderten Informationen werden für lokale Tasks ausgegeben, die den Pubset belegen.

***ALL**
'*ALL'
Wenn der im Operanden CATID angegebene Pubset mehrrechnerbenutzbar und das lokale System der Master des Pubsets ist, dann werden die mit REF=ALL angeforderten Informationen für alle (lokalen und fernen) Tasks ausgegeben, die diesen Pubset belegen; andernfalls nur für die lokalen Tasks.

'bcamname'
bcamname = BCAM-Name des lokalen Systems oder - wenn das lokale System Master des Pubsets ist - eines Slave-Systems.

adr
symbolische Adresse (Name) eines Feldes, das einen der Werte '*LOCAL', '*ALL' oder 'bcamname' enthält.

LENGTH=länge
legt die Länge des Ausgabebereiches für die angeforderten MRSCAT-Einträge fest.
Die implizite Länge eines mit dem Operanden AREA vereinbarten Feldes wird auch dann nicht als Länge des Ausgabebereiches angenommen, wenn der Operand LENGTH nicht angegeben wird.
Bei MF=S darf LENGTH nur zusammen mit AREA angegeben werden.

länge
Länge (in Byte) des Feldes mit der Adresse adr2.

Die Mindestlänge für den Ausgabebereich hängt von der Anzahl der angeforderten MRSCAT-Einträge und dem im Operanden REF angegebenen Informationsumfang ab. Es gelten folgende Werte:

– Ein einzelner MRSCAT-Eintrag wird angefordert:

  Bei REF=NO oder REF=YES muss der Ausgabebereich mindestens so lang sein wie der auszugebende MRSCAT-Eintrag. Das Layout dieses Eintrags wird im Anschluss an die Returncodes als DSECT wiedergegeben.

  Bei REF=ALL wird die Mindestlänge l_{AREA} des Ausgabebereiches durch folgende Formel bestimmt:

  $l_{AREA} = l_{MCE} + n_{OCC} * l_{OCC} + 1$

  Dabei bedeutet:

l_{AREA}	Mindestlänge des Ausgabebereiches (in Byte)
l_{MCE}	Länge des auszugebenden MRSCAT-Eintrags (Layout siehe DSECT im Anschluss an die Beschreibung der Returncodes)
n_{OCC}	Anzahl der Belegungseinträge für den Pubset, auf den der MRSCAT-Eintrag verweist
l_{OCC}	Länge eines Belegungseintrags (Layout siehe DSECT im Anschluss an die Beschreibung der Returncodes)

– Mehrere Einträge werden angefordert:
  Für die Mindestlänge l_{AREA} des Ausgabebereiches gilt folgende Formel:

  $l_{AREA} = n_{MCE} * l_{MCE} + 4$

  Dabei bedeutet:

l_{AREA}	Mindestlänge des Ausgabebereiches (in Byte)
n_{MCE}	Anzahl der auszugebenden MRSCAT-Einträge
l_{MCE}	Länge des auszugebenden MRSCAT-Eintrags (Layout siehe DSECT im Anschluss an die Beschreibung der Returncodes)

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).
Voreinstellung: MACID = MCE bei XPAND=PL
MACID = MCF bei XPAND=MCE
MACID = MCH bei XPAND=OCC

PUBSET=
Angabe nur in Verbindung mit SELECT=VOLUME_SETS zulässig.
Gibt die Catid des SM-Pubsets an, dessen Volume-Set zurückgeliefert wird.

***ALL**
Die Volume-Sets aller SM-Pubsets werden zurückgeliefert.

'catid'
Catid des SM-Pubsets, für dessen Volume-Sets Informationen zurückgeliefert werden sollen.

adr
Symbolische Adresse (Name) eines Feldes, das die Katalogkennung des SM-Pubsets enthält, für dessen Volume-Sets Informationen zurückgeliefert werden sollen.

REF=
bestimmt, ob zusätzlich zu den MRSCAT-Einträgen (also den Informationen, die das Kommando /SHOW-MASTER-CATALOG-ENTRY liefert) weitere Informationen über die angegebenen Pubsets ausgegeben werden sollen.

NO
es sollen keine weiteren Informationen ausgegeben werden.

YES
Für jeden angegebenen Pubset werden zusätzlich zum MRSCAT-Eintrag auch die Pubset-Parameter ausgegeben (siehe Kommando SHOW-PUBSET-PARAMETERS). Es fehlen lediglich die belegenden Tasks; diese werden nur bei REF=ALL ausgegeben.

Diese Angabe wird bei Makroaufrufen nur unter der Kennung der Systemverwaltung (TSOS) ausgewertet, bei allen anderen Benutzern wird sie ignoriert.

ALL
Für einen Pubset werden zusätzlich zu den unter REF=YES beschriebenen Informationen noch die Benutzerkennungen, die TSNs und die TIDs der Tasks ausgegeben, die den Pubset belegen, sowie die SYSIDs der Systeme, auf denen diese Tasks ablaufen. Für jedes System werden die ausgegebenen Einträge sortiert: Zuerst nach Benutzerkennung, dann nach TSN und schließlich nach TID.
REF=ALL ist nur zulässig, wenn im Operanden CATID explizit die Katalogkennung eines Pubsets angegeben wurde. Bei der Angabe eines Wildcard-Ausdrucks oder des (voreingestellten) Wertes '␣' im Operanden CATID wird REF=ALL intern in REF=YES umgewandelt.

Diese Angabe wird bei Makroaufrufen nur unter der Kennung der Systemverwaltung (TSOS) ausgewertet, bei allen anderen Benutzern wird sie ignoriert.

SELECT=
erlaubt bei der Angabe eines Wildcard-Ausdrucks oder des (voreingestellten) Wertes '␣' (Leerzeichen) im Operanden CATID eine engere Auswahl der dadurch vereinbarten Pubsets.

ALL
Die angeforderten Informationen werden für alle im Operanden CATID vereinbarten Pubsets ausgegeben.

ACCESSIBLE
Die angeforderten Informationen werden nur für Pubsets ausgegeben, auf deren Dateikatalog zugegriffen werden kann.

DEF_XCS_CONF
Die angeforderten Informationen werden nur für Pubsets ausgegeben, die als XCS-Pubset definiert sind.

EXCLUSIVE
Die angeforderten Informationen werden nur für Pubsets ausgegeben, die nicht als Shared-Pubset importiert sind.

HSMS_SUPPORTED
Die angeforderten Informationen werden nur für SM-Pubsets ausgegeben, auf die das Attribut HSMS SUPPORTED zutrifft.

INACCESSIBLE
Die angeforderten Informationen werden nur für Pubsets ausgegeben, die nicht importiert sind.

LOCAL
Die angeforderten Informationen werden nur für Pubsets ausgegeben, die lokal importiert sind.

LOCAL_ACCESSIBLE
Die angeforderten Informationen werden nur für Pubsets ausgegeben, die lokal importiert und nicht im Quiet-Status sind.

MASTER_CHANGE_ERROR
Die angeforderten Informationen werden nur für Shared-Pubsets ausgegeben, bei denen ein Master-Wechsel fehlerhaft beendet wurde.

PAGING
Die angeforderten Informationen werden nur für Pubsets mit lokal genutzten Paging-Bereichen ausgegeben.

QUIET
Die angeforderten Informationen werden nur für Pubsets ausgegeben, die im Quiet-Status sind.

REMOTE
Die angeforderten Informationen werden nur für Pubsets ausgegeben, für die nicht das Auswahlkriterium LOCAL zutrifft.

REMOTE_ACCESSIBLE
Die angeforderten Informationen werden nur für Pubsets ausgegeben, die nicht lokal importiert sind, auf deren Katalog aber dennoch zugegriffen werden kann, weil aktuell eine MSCF-Verbindung zu einem fernen System besteht, das das Pubset lokal importiert hat.

SCA
Die angeforderten Informationen werden nur für Pubsets ausgegeben, bei denen das eigene System die Katalogzugriffe über SCA abwickelt.
SCA = Speed Catalog Access. Ersetzt den sequenziellen Zugriff auf den Dateikatalog TSOSCAT durch einen indexsequenziell orientierten (direkten) Dateizugriff.

SHARED
Die angeforderten Informationen werden nur für Pubsets ausgegeben, die als Shared-Pubset importiert sind.

SINGLE_FEATURE
Die angeforderten Informationen werden nur für SF-Pubsets ausgegeben.

SYSTEM_MANAGED
Die angeforderten Informationen werden nur für SM-Pubsets ausgegeben.

UNUSED_VOLSETS
Die angeforderten Informationen werden nur für Volume-Sets ausgegeben, die sich im Zustand DEFINED ONLY befinden.

VOLUME_SETS
Die angeforderten Informationen werden nur für Volume-Sets ausgegeben.
Eine engere Auswahl der auszugebenden Volume-Set-Einträge kann mit dem Operanden PUBSET getroffen werden.

XCS_CONFIGURATED
Die angeforderten Informationen werden nur für Pubsets ausgegeben, die als XCS-Pubset benutzt werden.

VERSION=
legt fest, für welche BS2000-Version das Layout erzeugt und Informationen ausgegeben werden soll.
Wird der Operand VERSION nicht angegeben, werden Layout und Informationen für die BS2000-Version V10.0 erzeugt bzw. ausgegeben.

5
Layout und Informationen für die Versionen ab BS2000/OSD-BC V3.0 werden erzeugt bzw. ausgegeben.

3
Layout und Informationen für die Versionen BS2000/OSD-BC V1.0 und V2.0 werden erzeugt bzw. ausgegeben.

XPAND=
wird nur im Zusammenhang mit MF=C bzw. MF=D ausgewertet und legt fest, für welchen Datenbereich eine CSECT bzw. DSECT erzeugt werden soll.

PL
Es wird eine CSECT/DSECT für den Datenbereich erzeugt. Dieser Wert ist Voreinstellung bei MF=C.

***ALL**
Es werden CSECTs/DSECTs für den Datenbereich und den Ausgabebereich mit Belegungseintrag erzeugt. Voreinstellung bei MF=D.

MCE
Es wird eine CSECT/DSECT für einen MRSCAT-Eintrag erzeugt.

OCC
Es wird eine CSECT/DSECT für die bei REF=ALL zusätzlich ausgegebenen Belegungsinformationen erzeugt.

Rückinformation und Fehleranzeigen

Standard-header:

c	c	b	b	a	a	a	a

Über die Ausführung des Makros STAMCE wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

SUBCODE2 (X'cc') wird in der folgenden Tabelle nicht aufgeführt. Er kann die Werte X'00' und X'01' annehmen. Sie haben die folgenden Bedeutungen:

X'00': Fehler im lokalen System
X'01': Nur bei REF=ALL: Fehler im fernen System

X'bb'	X'aaaa'	Erläuterung
X'00'	X'0000'	Funktion erfolgreich ausgeführt; kein Fehler.
X'20'	X'0310'	Systemfehler: Fehler während der Privilegienprüfung.
X'01'	X'0311'	Operandenfehler: – Sowohl die Katalogkennung als auch eine Adresse mit der Katalogkennung wurden angegeben. – Der Ausgabebereich ist nicht auf Wortgrenze ausgerichtet. – Auf den Datenbereich kann nicht in allen Teilen zugegriffen werden. – Der Ausgabebereich ist nicht verfügbar. – Die Adresse mit der Katalogkennung ist nicht verfügbar oder eine gültige Katalogkennung bzw. ein gültiger Wildcard-Ausdruck konnte im zugewiesenen Speicherbereich nicht gefunden werden.
X'40'	X'0312'	Gesuchter MRSCAT-Eintrag nicht gefunden: – Es gibt keinen MRSCAT-Eintrag mit der angegebenen Katalogkennung. – Es gibt keinen MRSCAT-Eintrag, der den im Wildcard-Ausdruck und/oder im SELECT-Operanden angegebenen Auswahlkriterien entspricht.
X'20'	X'0313'	Fehler bei REQM/RELM: Speicherplatz für die Ausgabe konnte nicht angefordert/freigegeben werden.
X'40'	X'0313'	Fehler bei REQM/RELM: Es ist nicht genügend Klasse-6-Speicherplatz vorhanden.
X'01'	X'0314'	Katalogkennung / Wildcard-Ausdruck ungültig: Die angegebene Katalogkennung bzw. Wildcard-Zeichenfolge entspricht nicht dem geforderten Format.
X'40'	X'0316'	Ausgabebereich zu klein: Der vom Anwender angegebene Ausgabebereich ist zu klein zur Aufnahme der angeforderten Information.
X'40'	X'0317'	Systemfehler: Konflikt mit MRSCAT-Sperre
X'20'	X'0318'	Systemfehler: Synchronisationsfehler
X'40'	X'031A'	MRSCAT noch nicht initialisiert: STAMCE wurde während des STARTUP aufgerufen, bevor der MRSCAT initialisiert wurde.

X'bb'	X'aaaa'	Erläuterung
X'20'	X'031B'	Übertragungsfehler, tritt nur bei REF=ALL auf: Während der HIPLEX-MSCF-Übertragung, Anforderung oder Freigabe des Pubsets trat ein Fehler auf.
X'01'	X'031C'	Ungültiger BCAM-Name, tritt nur bei REF=ALL auf: – Der angegebene BCAM-Name ist dem System nicht bekannt. – Der angegebene BCAM-Name bezeichnet keinen Sharer des Pubsets. – Der Pubset ist nicht mehrrechnerbenutzbar und der BCAM-Name bezeichnet nicht das lokale System – Das lokale System ist nicht der Master des mehrrechnerbenutzbaren Pubsets und der angegebene BCAM-Name bezeichnet nicht das lokale System.
X'01'	X'031F'	Fehlerhafter SELECT-Operand des Datenbereichs: Der Anwender hat das Feld für den SELECT-Operanden im Datenbereich mit einem ungültigen Wert versorgt.
X'01'	X'FFFF'	Interface-Fehler: Falscher Unit- oder Funktions-Code.
X'03'	X'FFFF'	Interface-Fehler: Falsche Versions-Nummer.

Als zusätzliche Returncodes können in den beiden rechten Byte von Register R15 folgende Werte zurückgegeben werden:

X'04A4': Es wurde versucht, einen **STAMCE** einer Version < V9.5 auszuführen. Dabei wurde im Datenbereich ein ungültiger Operationscode entdeckt. Dieser Fehler tritt insbesondere dann auf, wenn die Version des Datenbereichs nicht zur Version des SVC-Aufrufs passt.

X'04A0': Wie oben; darüber hinaus ist das Subsystem JV („Jobvariable“) nicht verfügbar.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Das aufrufende Programm wird beendet, wenn folgende Fehler auftreten:
– Der Datenbereich ist dem Aufrufer nicht zugewiesen.
– Der Datenbereich ist nicht auf Wortgrenze ausgerichtet.
– Der Datenbereich ist gegen Schreibzugriff geschützt.

DSECT für den Datenbereich des Makros (XPAND=PL)

```
          STAMCE MF=D,PREFIX=D,XPAND=PL,VERSION=5
1         #INTF REFTYPE=REQUEST,                                        C
1               INTNAME=STAM,                                          C
1               INTCOMP=5
1         MFCHK    MF=D,                                               C
1               SUPPORT=(C,D,E,L,M,S),                                 C
1               PREFIX=D,                                              C
1               MACID=MCE,                                             C
1               DMACID=MCE,                                            C
1               DNAME=MCEPL,                                           C
1               PARAM=,                                                C
1               SVC=33,                                                C
1               ALIGN=F
2 DMCEPL  DSECT ,
2               *,##### PREFIX=D, MACID=MCE #####
1         STAMLY   MF=D,                                               C
1               PREFIX=D,                                              C
1               MACID=MCE,                                             C
1               PARAM=,                                                C
1               VERSION=5,                                             C
1               XPAND=PL,                                              C
1               FUNCT=1,                                               C
1               CG27=DMCE,                                             C
1               CATID=,                                                C
1               AREA=,                                                 C
1               LENGTH=,                                               C
1               REF=,                                                  C
1               HOST=,                                                 C
1               SELECT=,                                               C
1               PUBSET=
2         #INTF REFTYPE=REQUEST,                                        C
2               INTNAME=STAMLY,                                        C
2               INTCOMP=5
2 ********************
2 *  parameter list  *
2 ********************
2 DMCEFHDR   FHDR  MF=(C,DMCE),EQUATES=NO                 standard header
3 DMCEFHDR DS    0A
3 DMCEFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
3 *
3 DMCEIFID DS    0A              0   INTERFACE IDENTIFIER
3 DMCEFCTU DS    AL2             0   FUNCTION UNIT NUMBER
3 *                                  BIT 15    HEADER FLAG BIT,
3 *                                  MUST BE RESET UNTIL FURTHER NOTICE
3 *                                  BIT 14-12 UNUSED, MUST BE RESET
3 *                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
```

```
3 DMCEFCT  DS    AL1            2   FUNCTION NUMBER
3 DMCEFCTV DS    AL1            3   FUNCTION INTERFACE VERSION NUMBER
3 *
3 DMCERET  DS    0A             4   GENERAL RETURN CODE
3 DMCESRET DS    0AL2           4   SUB RETURN CODE
3 DMCESR2  DS    AL1            4   SUB RETURN CODE 2
3 DMCESR1  DS    AL1            5   SUB RETURN CODE 1
3 DMCEMRET DS    0AL2           6   MAIN RETURN CODE
3 DMCEMR2  DS    AL1            6   MAIN RETURN CODE 2
3 DMCEMR1  DS    AL1            7   MAIN RETURN CODE 1
3 DMCEFHL  EQU   8              8   GENERAL OPERAND LIST HEADER LENGTH
3 *
2 DMCEAREA   DS    A              area address
2 DMCECTAD   DS    A              catid address
2 DMCECTID   DS    CL4            catid
2 DMCEARLN   DS    F              area length
2 DMCERESA   DS    A              reserved (mf=m buffer)
2 DMCEHOST   DS    CL8            bcam name of the host
2 DMCESLCT   DS    C              SELECT value
2 DMCEFLAG   DS    X              flags
2 DMCESMPU   DS    CL4            id of sm pubset
2 DMCEUNUS   DS    XL6            unused
2 DMCE#      EQU   *-DMCEFHDR       length parameter list
2 *
2 * return codes
2 *
2 DMCEOK     EQU   X'0000'        NO_ERROR
2 DMCESRPM   EQU   X'0310'        SRPM_ERROR
2 DMCEOPER   EQU   X'0311'        OPERAND_ERROR_IN_STAM
2 DMCENFND   EQU   X'0312'        MCE_CANNOT_BE_FOUND
2 DMCERQRL   EQU   X'0313'        REQM_RELM_ERROR
2 DMCEICOW   EQU   X'0314'        INVALID_CATID_OR_WILDCARD
2 DMCEATS    EQU   X'0316'        AREA_TOO_SMALL
2 DMCELCKC   EQU   X'0317'        MRSCAT_LOCK_CONFLICT
2 DMCESYER   EQU   X'0318'        SYNCHRONIZATION_ERROR
2 DMCENINI   EQU   X'031A'        MRSCAT_NOT_INITIALIZED
2 DMCETRER   EQU   X'031B'        TRANSMISSION_ERROR
2 DMCEHOIN   EQU   X'031C'        INVALID_HOST_NAME
2 DMCEPAER   EQU   X'031F'        MRS_PARAM_ERROR
2 *
2 * subcode 2
2 *
2 DMCESC2L   EQU  X'00'           LOCAL_ERROR
2 DMCESC2R   EQU  X'01'           REMOTE_ERROR
2 *
2 * last catid
2 *
2 DMCELAST   EQU  X'40404040'
```

```
2 *
2 * last occupation
2 *
2 DMCELOCC   EQU  X'00'
2 *
2 * SELECT values
2 *
2 DMCEALL    EQU   0               all the pubsets
2 DMCEPAGI   EQU   1               paging pubsets
2 DMCELOCA   EQU   2               local pubsets
2 DMCEREMO   EQU   3               remote pubsets
2 DMCEACCE   EQU   4               pubsets accessible
2 DMCELOAC   EQU   5               pubsets local and accessible
2 DMCESHAR   EQU   6               shared pubsets
2 DMCEEXCL   EQU   7               exclusive pubsets
2 DMCEREAC   EQU   8               pubsets remote and accessible
2 DMCESCA    EQU   9               local and accessible pubsets which
2 *                                are connected to SPEEDCAT
2 DMCEXCS    EQU   10              XCS configurated pubsets
2 DMCEHSMS   EQU   11              all HSMS supported sm pubsets
2 DMCESF     EQU   12              all single feature pubsets
2 DMCESM     EQU   13              all system managed pubsets
2 DMCEVOL    EQU   14              all volume sets of a sm pubset
2 DMCEDEF    EQU   15              volume sets of sm pubsets which are
2 *                                defined but not in use
2 DMCEMCHE   EQU   16              pubsets where an error occured during
2 *                                master change processing
2 DMCEINAC   EQU   17              pubsets which are inaccessible
2 DMCEDXCS   EQU   18              defined XCS configurated
2 DMCEQUIT   EQU   19              pubset in quiet status
2 *
2 * flags
2 *
2 DMCEMPVS   EQU  X'80'            set: MPVS mode
2 DMCEREF    EQU  X'40'            set: REF = YES
2 DMCERALL   EQU  X'20'            set: REF = ALL
```

DSECT für den Ausgabebereich des Makros (XPAND=MCE)

```
          STAMCE MF=D,PREFIX=D,XPAND=MCE,VERSION=5
1         #INTF REFTYPE=REQUEST,                                        C
1               INTNAME=STAM,                                          C
1               INTCOMP=5
1         MFCHK    MF=D,                                               C
1               SUPPORT=(C,D,E,L,M,S),                                 C
1               PREFIX=D,                                              C
1               MACID=MCF,                                             C
1               DMACID=MCF,                                            C
1               DNAME=MCFMCE,                                          C
1               PARAM=,                                                C
1               SVC=33,                                                C
1               ALIGN=F
2 DMCFMCE  DSECT ,
2               *,##### PREFIX=D, MACID=MCF #####
1         STAMLY   MF=D,                                               C
1               PREFIX=D,                                              C
1               MACID=MCF,                                             C
1               PARAM=,                                                C
1               VERSION=5,                                             C
1               XPAND=MCE,                                             C
1               FUNCT=1,                                               C
1               CG27=DMCF,                                             C
1               CATID=,                                                C
1               AREA=,                                                 C
1               LENGTH=,                                               C
1               REF=,                                                  C
1               HOST=,                                                 C
1               SELECT=,                                               C
1               PUBSET=
2         #INTF REFTYPE=REQUEST,                                        C
2               INTNAME=STAMLY,                                        C
2               INTCOMP=5
2 DMCFMST    DS    0D
2         MCEDSK  VERSION=5,CG27=DMCF
3         #INTF REFTYPE=REQUEST,                                        C
3               INTNAME=MCEDSK,                                        C
3               INTCOMP=5
3 ******************************
3 *  header of all entry types  *
3 ******************************
3           DS    0D       DW alignement
3 DMCFSHDR  DS    0XL8     header
3 DMCFSCTD  DS    CL4      catid
3 *
3 DMCFSENT  DS    X        entry type set
```

```
3 DMCFSSF    EQU   0        single feature pubset
3 DMCFSSM    EQU   1        system managed pubset
3 DMCFSVOL   EQU   2        volume set
3 *
3            DS    XL3      unused
3 *********************************************
3 *  starting point of the static entry part  *
3 *********************************************
3 DMCFSDSC   DS    0D       starting point of static part
3 ************************************************
3 *  static part of a single feature pubset entry  *
3 ************************************************
3 DMCFSBCA   DS    CL8      BCAM name for RFA
3 DMCFSDEM   DS    0XL2     device mnemonic
3 DMCFSDEV   DS    XL1      device code
3 DMCFSDEF   DS    XL1      device filler
3 DMCFSBNU   DS    XL2      static number of CMS buffer
3 DMCFSBWT   DS    F        batch wait time
3 DMCFSDWT   DS    F        dialog wait time
3 *
3 * static pubset status values (1)
3 *
3 DMCFSSTA   DS    X        static status byte (1)
3 DMCFSAUT   EQU   X'80'    set  : autoquiet selected
3 DMCFSBDF   EQU   X'40'    set  : static buffer defined
3 DMCFSBCL   EQU   X'20'    set  : static buffer resident (cl.3)
3 *                         reset: static buffer non-res. (cl.4)
3 DMCFSSH    EQU   X'10'    set  : pubset shared (next import)
3 *                         reset: pubset exclusive (next import)
3 DMCFSAC    EQU   X'08'    set  : pubset with controlled use
3 DMCFSUVA   EQU   X'04'    set  : physical allocation by users allowed
3 DMCFSFIM   EQU   X'02'    set  : continue IMPORT if cache can't be
3 *                                connected
3 *                         reset: abort IMPORT if cache can't be
3 *                                connected
3 DMCFSXCS   EQU   X'01'    set  : XCS pubset at startup
3 *
3 DMCFSSPC   DS    X        SPEEDCAT set
3 DMCFSNSP   EQU   0        no automatic start
3 DMCFSSSP   EQU   1        SPEEDCAT task
3 DMCFSUSP   EQU   2        SPEEDCAT own task
3 DMCFSNSS   EQU   4        no, non start SPEEDCAT
3 *
3 * static pubset status values (2)
3 *
3 DMCFSST2   DS    X        static status byte (2)
3 DMCFSRIM   EQU   X'80'    set  : remote import by command only
3 *                         reset: remote import by connection
```

```
3 DMCFSCCT   EQU   X'40'    set  : TSOSCAT is to be converted to V10
3 *                              format during next EXPORT
3            DS    XL1      unused
3 DMCFSUID   DS    CL8      userid allowed to access pubset
3 *
3 * static cache values
3 *
3 DMCFSCBS   DS    F        cache size
3 *
3 DMCFSCBY   DS    X        byte for bit values
3 DMCFSCBU   EQU   X'80'    set  : cache size unit is KB
3 *                         reset: cache size unit is MB
3 DMCFSCST   EQU   X'40'    set  : size tolerance
3 *
3 DMCFSCM    DS    X        cache medium set
3 DMCFSCNC   EQU   0        no cache
3 DMCFSCDC   EQU   1        controller
3 DMCFSCES   EQU   2        expanded storage
3 DMCFSCGS   EQU   3        global storage
3 DMCFSCMM   EQU   4        main memory
3 *
3 DMCFSCSZ   DS    X        segment size set
3 DMCFSC4    EQU   0        4 KB
3 DMCFSC8    EQU   1        8 KB
3 DMCFSC16   EQU   2        16 KB
3 DMCFSC32   EQU   3        32 KB
3 *
3 DMCFSCFS   DS    X        file selection
3 DMCFSBUS   EQU   0        by user
3 DMCFSALL   EQU   1        all
3 DMCFSAUS   EQU   2        autoselected
3 *
3 DMCFSGDS   DS    X        GS data security set
3 DMCFSGNS   EQU   0        no security
3 DMCFSGCO   EQU   1        connect
3 *
3 DMCFSGU1   DS    XL1      GS unit=1 or 2
3 DMCFSGU2   DS    XL1      unused
3 *
3 DMCFSGDB   DS    X        GS double recording by buffer set
3 DMCFSGST   EQU   0        std
3 DMCFSGNY   EQU   1        mono
3 DMCFSGNN   EQU   2        any
3 DMCFSGYE   EQU   3        yes
3 *
3 DMCFSGFO   DS    X        force out set
3 DMCFSGNF   EQU   0        no force out
3 DMCFSGLF   EQU   1        at low filling
```

```
3 DMCFSGHF   EQU   2        at high filling
3 *
3 DMCFSCFE   DS    X        DC prefetch set
3 DMCFSCFN   EQU   0        no prefetch
3 DMCFSCFL   EQU   1        low
3 DMCFSCFH   EQU   2        high
3 *
3 * static attach/detach pubset
3 *
3 DMCFSMN    DS    XL2      attach pubset with dev#
3 *
3 * static allocator values
3 *
3 DMCFSAL1   DS    F        residual space at sat level 1
3 DMCFSAL2   DS    F        residual space at sat level 2
3 DMCFSAL3   DS    F        residual space at sat level 3
3 DMCFSAL4   DS    F        residual space at sat level 4
3 DMCFSAL5   DS    F        residual space at sat level 5
3 DMCFSAPA   DS    F        predet primary alloc amount
3 DMCFSASA   DS    F        predet secondary alloc amount
3 DMCFSADL   DS    F        sec alloc doubling limit
3 DMCFSAZP   DS    F        residual space for ZIP startup
3 *
3 * static eam values
3 *
3 DMCFSEMA   DS    F        maximal size of file SYSEAM
3 DMCFSEMI   DS    F        minimal size of file SYSEAM
3 DMCFSESA   DS    F        secondary allocation of file SYSEAM
3 DMCFSEMS   DS    F        virtual memory size of file SYSEAM
3 *
3 DMCFS#     EQU   *-DMCFSBCA    length of the static part of a single
3 *                              feature pubset entry
3            ORG   DMCFSDSC      starting point of static part
3 ************************************************
3 *  static part of a system managed pubset entry  *
3 ************************************************
3 DMCFHBCA   DS    CL8      BCAM name for RFA
3 DMCFHDEM   DS    0XL2     device mnemonic of volres of ctl volset
3 DMCFHDEV   DS    XL1      device code
3 DMCFHDEF   DS    XL1      device filler
3 DMCFHBNU   DS    XL2      static number of CMS buffer
3 DMCFHBWT   DS    F        batch wait time
3 DMCFHDWT   DS    F        dialog wait time
3 *
3 * static pubset status values (1)
3 *
3 DMCFHSTA   DS    X        static status byte (1)
3 DMCFHAUT   EQU   X'80'    set  : autoquiet selected
```

```
3 DMCFHBDF   EQU   X'40'    set  : static buffer defined
3 DMCFHBCL   EQU   X'20'    set  : static buffer resident (cl.3)
3 *                         reset: static buffer non-res. (cl.4)
3 DMCFHSH    EQU   X'10'    set  : pubset shared (next import)
3 *                         reset: pubset exclusive (next import)
3 DMCFHAC    EQU   X'08'    set  : pubset with controlled use
3 DMCFHHSM   EQU   X'04'    set  : pubset is HSMS supported
3 DMCFHFIM   EQU   X'02'    set  : continue IMPORT if cache can't be
3 *                                connected
3 *                         reset: abort IMPORT if cache can't be
3 *                                connected
3 DMCFHXCS   EQU   X'01'    set  : XCS pubset at startup
3 *
3            DS    XL1      unused
3 *
3 * static pubset status values (2)
3 *
3 DMCFHST2   DS    X        static status byte (2)
3 DMCFHRIM   EQU   X'80'    set  : remote import by command only
3 *                         reset: remote import by connection
3            DS    XL1      unused
3 DMCFHUID   DS    CL8      userid allowed to access pubset
3 *
3 * static cache values
3 *
3            DS    XL4      unused
3 DMCFHCBY   DS    X        byte for bit values
3 DMCFHCST   EQU   X'40'    set  : size tolerance
3 *
3 DMCFHVID   DS    CL4      catid of ctl volset
3 *
3 DMCFHDFF   DS    X        default file format set
3 DMCFHDST   EQU   0        std
3 DMCFHPAM   EQU   1        pamkey format
3 DMCFHNK2   EQU   2        NK2 format
3 DMCFHNK4   EQU   3        NK4 format
3 *
3            DS    XL4      unused
3 *
3 * static attach/detach pubset
3 *
3 DMCFHMN    DS    XL2      attach pubset with dev#
3 *
3 * static allocator values
3 *
3            DS    XL20     unused
3 DMCFHAPA   DS    F        predet primary alloc amount
3 DMCFHASA   DS    F        predet secondary alloc amount
```

```
3 DMCFHADL   DS    F        sec alloc doubling limit
3            DS    XL4      unused
3 *
3 * static EAM values
3 *
3 DMCFHEMA   DS    F        maximal size of file SYSEAM
3 DMCFHEMI   DS    F        minimal size of file SYSEAM
3 DMCFHESA   DS    F        secondary allocation of file SYSEAM
3 DMCFHEMS   DS    F        virtual memory size of file SYSEAM
3 *
3 DMCFH#     EQU   *-DMCFHBCA    length of the static part of a system
3 *                              managed pubset entry
3            ORG   DMCFSDSC      starting point of static part
3 ***********************************
3 *  static part of a pubset entry  *
3 ***********************************
3 DMCFFBCA   DS    CL8      BCAM name for RFA
3 DMCFFDEM   DS    0XL2     device mnemonic
3 DMCFFDEV   DS    XL1      device code
3 DMCFFDEF   DS    XL1      device filler
3 DMCFFBNU   DS    XL2      static number of CMS buffer
3 DMCFFBWT   DS    F        batch wait time
3 DMCFFDWT   DS    F        dialog wait time
3 *
3 * static pubset status values (1)
3 *
3 DMCFFSTA   DS    X        static status byte (1)
3 DMCFFAUT   EQU   X'80'    set  : autoquiet selected
3 DMCFFBDF   EQU   X'40'    set  : static buffer defined
3 DMCFFBCL   EQU   X'20'    set  : static buffer resident (cl.3)
3 *                         reset: static buffer non-res. (cl.4)
3 DMCFFSH    EQU   X'10'    set  : pubset shared (next import)
3 *                         reset: pubset exclusive (next import)
3 DMCFFAC    EQU   X'08'    set  : pubset with controlled use
3 DMCFFUVA   EQU   X'04'    set  : physical allocation by users allowed
3 DMCFFFIM   EQU   X'02'    set  : continue IMPORT if cache can't be
3 *                                connected
3 *                         reset: abort IMPORT if cache can't be
3 *                                connected
3 DMCFFXCS   EQU   X'01'    set  : XCS pubset at startup
3            DS    XL1      unused
3 *
3 * static pubset status values (2)
3 *
3 DMCFFST2   DS    X        static status byte (2)
3 DMCFFRIM   EQU   X'80'    set  : remote import by command only
3 *                         reset: remote import by connection
3            DS    XL1      unused
```

```
3 DMCFFUID   DS    CL8      userid allowed to access pubset
3 *
3 * static cache values
3 *
3            DS    XL4      unused
3 DMCFFCBY   DS    X        byte for bit values
3 DMCFFCST   EQU   X'40'    set  : size tolerance
3 *
3            DS    XL9      unused
3 *
3 * static attach/detach pubset
3 *
3 DMCFFMN    DS    XL2      attach pubset with dev#
3 *
3 * static allocator values
3 *
3            DS    XL20     unused
3 DMCFFAPA   DS    F        predet primary alloc amount
3 DMCFFASA   DS    F        predet secondary alloc amount
3 DMCFFADL   DS    F        sec alloc doubling limit
3            DS    XL4      unused
3 *
3 * static eam values
3 *
3 DMCFFEMA   DS    F        maximal size of file SYSEAM
3 DMCFFEMI   DS    F        minimal size of file SYSEAM
3 DMCFFESA   DS    F        secondary allocation of file SYSEAM
3 DMCFFEMS   DS    F        virtual memory size of file SYSEAM
3 *
3 DMCFF#     EQU   *-DMCFFBCA    length of the static part of a pubset
3 *                              entry
3            ORG   DMCFSDSC      starting point of static part
3 ************************************
3 *  static part of a volume set entry  *
3 ************************************
3 DMCFBPID   DS    CL4      corresponding pubset id
3            DS    XL4      unused
3 DMCFBDEM   DS    OXL2     device mnemonic
3 DMCFBDEV   DS    XL1      device code
3 DMCFBDEF   DS    XL1      device filler
3 *
3 * static performance attributes
3 *
3 DMCFBVSU   DS    X        volset usage set
3 DMCFBVST   EQU   0        standard volset
3 DMCFBWRK   EQU   1        work volset
3 DMCFBHSM   EQU   2        HSMS controlled volset
3 *
```

```
3 DMCFBAVA   DS    X        availability set
3 DMCFBAST   EQU   0        standard availability
3 DMCFBHIG   EQU   1        high availability
3 *
3 DMCFBPER   DS    X        performance profile
3 DMCFBPST   EQU   X'80'    standard performance
3 DMCFBHIH   EQU   X'40'    high performance
3 DMCFBVHI   EQU   X'20'    very high performance
3 *
3 DMCFBCRE   DS    X        write consistency set
3 DMCFBBYC   EQU   0        by close
3 DMCFBIMM   EQU   1        immediate
3 *
3            DS    XL1      unused
3 *
3 DMCFBNFA   DS    X        new file allocation
3 DMCFBNNR   EQU   0        not restricted
3 DMCFBNPO   EQU   1        physical only
3 DMCFBNNA   EQU   2        not allowed
3 *
3 DMCFBVAC   DS    X        volset access
3 DMCFBVNR   EQU   0        not restricted
3 DMCFBVAO   EQU   1        administrator only
3 *
3 DMCFBVSS   DS    X        volset status
3 DMCFBVSD   EQU   0        normal use
3 DMCFBVDO   EQU   1        defined only
3 DMCFBVIH   EQU   2        in hold
3 DMCFBVDF   EQU   3        defect
3 *
3            DS    XL2      unused
3 *
3 * static volset status values
3 *
3 DMCFBSTA   DS    X        static status byte
3 DMCFBCVS   EQU   X'04'    set: volset is ctl volset of a sm pubset
3            DS    XL11     unused
3 *
3 * static cache values
3 *
3 DMCFBCBS   DS    F        cache size
3 *
3 DMCFBCBY   DS    X        byte for bit values
3 DMCFBCBU   EQU   X'80'    set  : cache size unit is KB
3 *                         reset: cache size unit is MB
3 *
3 DMCFBCM    DS    X        cache medium set
3 DMCFBCNC   EQU   0        no cache
```

```
3 DMCFBCDC   EQU   1        controller
3 DMCFBCES   EQU   2        expanded storage
3 DMCFBCGS   EQU   3        global storage
3 DMCFBCMM   EQU   4        main memory
3 *
3 DMCFBCSZ   DS    X        segment size set
3 DMCFBC4    EQU   0        4 KB
3 DMCFBC8    EQU   1        8 KB
3 DMCFBC16   EQU   2        16 KB
3 DMCFBC32   EQU   3        32 KB
3 *
3 DMCFBCFS   DS    X        file selection
3 DMCFBBUS   EQU   0        by user
3 DMCFBALL   EQU   1        all
3 DMCFBAUS   EQU   2        autoselected
3 *
3 DMCFBGDS   DS    X        GS data security set
3 DMCFBGNS   EQU   0        no security
3 DMCFBGCO   EQU   1        connect
3 *
3 DMCFBGU1   DS    XL1      GS unit=1 or 2
3 DMCFBGU2   DS    XL1      unused
3 *
3 DMCFBGDB   DS    X        GS double recording by buffer set
3 DMCFBGST   EQU   0        standard
3 DMCFBGNY   EQU   1        mono
3 DMCFBGNN   EQU   2        any
3 DMCFBGYE   EQU   3        dual
3 *
3 DMCFBGFO   DS    X        force out set
3 DMCFBGNF   EQU   0        no force out
3 DMCFBGLF   EQU   1        at low filling
3 DMCFBGHF   EQU   2        at high filling
3 *
3 DMCFBCFE   DS    X        DC prefetch set
3 DMCFBCFN   EQU   0        no prefetch
3 DMCFBCFL   EQU   1        low
3 DMCFBCFH   EQU   2        high
3 *
3 * static allocator values
3 *
3 DMCFBAL1   DS    F        residual space at sat level 1
3 DMCFBAL2   DS    F        residual space at sat level 2
3 DMCFBAL3   DS    F        residual space at sat level 3
3 DMCFBAL4   DS    F        residual space at sat level 4
3 DMCFBAL5   DS    F        residual space at sat level 5
3            DS    XL12     unused
3 DMCFBAZP   DS    F        residual space for ZIP startup
```

```
3 *
3            DS    XL16     unused
3 *
3 DMCFB#     EQU   *-DMCFBPID     length of the static part of a volume
3 *                               set entry
3            ORG   DMCFSDSC+DMCFS#      length of a static MRSCAT entry
2 ********************************************
2 *  starting point of the dynamic entry part  *
2 ********************************************
2 DMCFD      DS    0D       starting point of dynamic part
2 ************************************************
2 *  dynamic part of a single feature pubset entry  *
2 ************************************************
2 DMCFDOC#   DS    F        counter for occupations
2 *
2 * dynamic pubset status values (1)
2 *
2 DMCFDSTA   DS    X        dynamic status byte (1)
2 DMCFDLOC   EQU   X'80'    set: local          reset: remote
2 DMCFDHOM   EQU   X'40'    set: home           reset: imported
2 DMCFDSH    EQU   X'20'    set: shared         reset: exclusive
2 DMCFDIMC   EQU   X'10'    set: import in process
2 DMCFDEXC   EQU   X'08'    set: export in process
2 DMCFDMAS   EQU   X'04'    set: master         reset: slave
2 DMCFDINA   EQU   X'02'    set: inaccessible   reset: not inacc
2 DMCFDQUI   EQU   X'01'    set: quiet
2 *
2 * dynamic pubset status values (2)
2 *
2 DMCFDST2   DS    X        dynamic status byte (2)
2 DMCFDUVA   EQU   X'10'    set: physical allocation by users allowed
2 DMCFDAC    EQU   X'08'    set: pubset with controlled use
2 DMCFDMCP   EQU   X'04'    set: master change in process
2 DMCFDPAG   EQU   X'02'    set: paging pubset
2 DMCFDERI   EQU   X'01'    set: eram inhibit
2 *
2 DMCFDSES   DS    X        pubset session number
2 *
2 DMCFDFLA   DS    X        CMS flags
2 DMCFDBDF   EQU   X'80'    set:   CMS buffers defined
2 DMCFDBCL   EQU   X'40'    set:   CMS buffers resident (class 3)
2 *                         reset: CMS buffers not resident (class 4)
2 DMCFDSPC   EQU   X'20'    set: speedcat is running
2 DMCFDELC   EQU   X'10'    set: Extra_large_catalog
2 *
2 DMCFDBNU   DS    XL2      number of CMS buffers
2 *
2 DMCFDATT   DS    X        attribute
```

```
2 DMCFDLOB   EQU   X'40'    set: large_objects
2 *                                files/volumes with more than 32 GB
2 DMCFDLFA   EQU   X'20'    set: large_files_allowed
2 DMCFDRAI   EQU   X'10'    set: pubset with RAID volumes
2 DMCFDGSV   EQU   X'08'    set: gs volumes
2 DMCFDDRV   EQU   X'02'    set: high availability by DRV
2 DMCFDKEY   EQU   X'01'    set: key pubset
2 *
2 DMCFDXCN   DS    CL8      XCS name
2 DMCFDHOS   DS    CL8      host name       :* MSCF host name
2 *
2 DMCFDPUB   DS    X        pubset set
2 DMCFD2KN   EQU   0        NK2 (2K native)
2 DMCFD4KN   EQU   1        NK4 (4K native)
2 DMCFD4KO   EQU   2        NK2, allocation unit multiple of 4K
2 *                         (4K oriented)
2 * dynamic cache values
2 *
2 DMCFDCSZ   DS    F        size of cache buffer
2 *
2 DMCFDCB8   DS    X        byte for bit values
2 DMCFDCBU   EQU   X'80'    set  : cache size unit is KB
2 *                         reset: cache size unit is MB
2 DMCFDCDS   EQU   X'40'    set  : data security ensured
2 DMCFDCDB   EQU   X'20'    set  : double recording by buffer
2 DMCFDCDD   EQU   X'10'    set  : cache deactivated
2 DMCFDCIH   EQU   X'08'    set  : cache in hold
2 DMCFDCCU   EQU   X'04'    set  : cache used
2 DMCFDCSF   EQU   X'02'    set  : save file failed
2 *
2 DMCFDCM    DS    X        cache medium set
2 DMCFDCNC   EQU   0        no cache
2 DMCFDCDC   EQU   1        controller
2 DMCFDCES   EQU   2        expanded storage
2 DMCFDCGS   EQU   3        global storage
2 DMCFDCMM   EQU   4        main memory
2 *
2 DMCFDCS    DS    X        segment size set
2 DMCFDC4    EQU   0        4 KB
2 DMCFDC8    EQU   1        8 KB
2 DMCFDC16   EQU   2        16 KB
2 DMCFDC32   EQU   3        32 KB
2 *
2 DMCFDCU1   DS    XL1      GS unit=1 or 2
2 DMCFDCU2   DS    XL1      unused
2 *
2 DMCFDCFO   DS    X        force out set
2 DMCFDCNF   EQU   0        no force out
```

```
2 DMCFDCIP   EQU   1        at low filling
2 DMCFDCIN   EQU   2        at high filling
2 *
2 DMCFDCFE   DS    X        prefetch set
2 DMCFDCFN   EQU   0        no prefetch
2 DMCFDCFL   EQU   1        low
2 DMCFDCFH   EQU   2        high
2 *
2 DMCFDCFS   DS    X        file selection
2 DMCFDBUS   EQU   0        by user
2 DMCFDALL   EQU   1        all
2 DMCFDAUS   EQU   2        auto select
2 *
2 DMCFDCAS   DS    H        size of allocation unit (# of half pages)
2 DMCFDMTL   DS    H        maximal I/O transfer length
2 DMCFDUID   DS    CL8      userid allowed to access pubset
2 *
2            DS    XL2      unused
2 *
2 * dynamic allocator values
2 *
2 DMCFDAL5   DS    F        residual space at sat level 5
2 DMCFDAL4   DS    F        residual space at sat level 4
2 DMCFDAL3   DS    F        residual space at sat level 3
2 DMCFDAL2   DS    F        residual space at sat level 2
2 DMCFDAL1   DS    F        residual space at sat level 1
2 DMCFDAPA   DS    F        predet primary alloc amount
2 DMCFDASA   DS    F        predet secondary alloc amount
2 DMCFDADL   DS    F        sec alloc doubling limit
2 DMCFDAZP   DS    F        residual space for ZIP startup
2 *
2 * dynamic EAM values
2 *
2 DMCFDEMA   DS    F        minimal size of file SYSEAM
2 DMCFDEMI   DS    F        maximal size of file SYSEAM
2 DMCFDESA   DS    F        secondary allocation of file SYSEAM
2 DMCFDEMS   DS    F        virtual memory size of file SYSEAM
2 *
2 DMCFDREF   DS    XL4      counter of occupations (duplicate)
2 *
2 DMCF#      EQU   *-DMCFMST      length of the STAM single
2 *                               feature pubset entry
2            ORG   DMCFD    starting point of dynamic part
2 ************************************************
2 *  dynamic part of a system managed pubset entry  *
2 ************************************************
2 DMCFKOC#   DS    F        counter for occupations
2 *
```

```
2 * dynamic pubset status values (1)
2 *
2 DMCFKSTA   DS    X        dynamic status byte (1)
2 DMCFKLOC   EQU   X'80'    set: local          reset: remote
2 DMCFKHOM   EQU   X'40'    set: home           reset: imported
2 DMCFKSH    EQU   X'20'    set: shared         reset: exclusive
2 DMCFKIMC   EQU   X'10'    set: import in process
2 DMCFKEXC   EQU   X'08'    set: export in process
2 DMCFKMAS   EQU   X'04'    set: master         reset: slave
2 DMCFKINA   EQU   X'02'    set: inaccessible   reset: not inacc
2 DMCFKQUI   EQU   X'01'    set: quiet
2 *
2 * dynamic pubset status values (2)
2 *
2 DMCFKST2   DS    X        dynamic status byte (2)
2 DMCFKAC    EQU   X'08'    set: pubset with controlled use
2 DMCFKMCP   EQU   X'04'    set: master change in process
2 DMCFKPAG   EQU   X'02'    set: paging pubset
2 DMCFKERI   EQU   X'01'    set: eram inhibit
2 *
2 DMCFKSES   DS    X        session number
2 *
2 DMCFKFLA   DS    X        CMS flags
2 DMCFKBDF   EQU   X'80'    set:   CMS buffers defined
2 DMCFKBCL   EQU   X'40'    set:   CMS buffers resident (class 3)
2 *                         reset: CMS buffers not resident (class 4)
2 DMCFKBNU   DS    XL2      number of CMS buffers
2 *
2 * dynamic sm pubset status values
2 *
2 DMCFKSMS   DS    X        special status bytes for sm pubsets
2 DMCFKGEN   EQU   X'80'    set: pubset is in generation
2            ORG   DMCFKSMS
2 DMCFKATT   DS    X        attribute
2 DMCFKLOB   EQU   X'40'    set: large_objects
2 *                                files/volumes with more than 32 GB
2 DMCFKLFA   EQU   X'20'    set: large_files_allowed
2 DMCFKXCN   DS    CL8      XCS name
2 DMCFKHOS   DS    CL8      host name       :* MSCF host name
2 *
2 DMCFKDFF   DS    X        default file format set
2 DMCFKPAM   EQU   0        pamkey format
2 DMCFKNO2   EQU   1        NK2 format
2 DMCFKNO4   EQU   2        NK4 format
2 *
2 * dynamic performance attributes
2 *
2 DMCFKPER   DS    X        performance profile
```

```
2 DMCFKSTD   EQU   X'80'    standard performance
2 DMCFKHIG   EQU   X'40'    high performance
2 DMCFKVHI   EQU   X'20'    very high performance
2 *
2 DMCFKWRC   DS    X        write consistency
2 DMCFKBC    EQU   X'80'    by close
2 DMCFKIMM   EQU   X'40'    immediate
2 *
2 DMCFKAVA   DS    X        availability
2 DMCFKAST   EQU   X'80'    standard availability
2 DMCFKAHI   EQU   X'40'    high availability
2 *
2            DS    XL1      unused
2 *
2 DMCFKFMT   DS    X        format profile
2 DMCFKK     EQU   X'80'    system managed pubset with K volsets
2 DMCFKNK2   EQU   X'40'    system managed pubset with NK2 volsets
2 DMCFKFN4   EQU   X'20'    system managed pubset with NK4 volsets
2 *
2 DMCFKUSA   DS    X        volume set usage
2 DMCFKUST   EQU   X'80'    sm pubset with standard volsets
2 DMCFKWRK   EQU   X'40'    sm pubset with work volsets
2 DMCFKHSS   EQU   X'20'    sm pubset with HSMS controlled volsets
2 *
2            DS    XL2      unused
2 DMCFKNOV   DS    F        number of volsets
2            DS    XL1      unused
2 DMCFKMTL   DS    H        maximal I/O transfer length
2 DMCFKUID   DS    CL8      userid allowed to access the pubset
2            DS    XL4      unused
2 *
2 * dynamic allocator values
2 *
2            DS    XL20     unused
2 DMCFKAPA   DS    F        predet primary alloc amount
2 DMCFKASA   DS    F        predet secondary alloc amount
2 DMCFKADL   DS    F        sec alloc doubling limit
2            DS    XL4      unused
2 *
2 * dynamic EAM values
2 *
2 DMCFKEMA   DS    F        maximal size of file SYSEAM
2 DMCFKEMI   DS    F        minimal size of file SYSEAM
2 DMCFKESA   DS    F        secondary allocation of file SYSEAM
2 DMCFKEMS   DS    F        virtual memory size of file SYSEAM
2 *
2 DMCFKREF   DS    XL4      counter of occupations (duplicate)
2 *
```

```
2 DMCFK#     EQU   *-DMCFMST      length of the STAM system
2 *                               managed pubset entry
2            ORG   DMCFD    starting point of dynamic part
2 ************************************
2 *  dynamic part of a pubset entry  *
2 ************************************
2 DMCFGOC#   DS    F        counter for occupations
2 *
2 * dynamic pubset status values (1)
2 *
2 DMCFGSTA   DS    X        dynamic status byte (1)
2 DMCFGLOC   EQU   X'80'    set: local          reset: remote
2 DMCFGHOM   EQU   X'40'    set: home           reset: imported
2 DMCFGSH    EQU   X'20'    set: shared         reset: exclusive
2 DMCFGIMC   EQU   X'10'    set: import in process
2 DMCFGEXC   EQU   X'08'    set: export in process
2 DMCFGMAS   EQU   X'04'    set: master         reset: slave
2 DMCFGINA   EQU   X'02'    set: inaccessible   reset: not inacc
2 DMCFGQUI   EQU   X'01'    set: quiet
2 *
2 * dynamic pubset status values (2)
2 *
2 DMCFGST2   DS    X        dynamic status byte (2)
2 DMCFGAC    EQU   X'08'    set: pubset with controlled use
2 DMCFGMCP   EQU   X'04'    set: master change in process
2 DMCFGPAG   EQU   X'02'    set: paging pubset
2 DMCFGERI   EQU   X'01'    set: eram inhibit
2 *
2 DMCFGSES   DS    X        session number
2 *
2 DMCFGFLA   DS    X        CMS flags
2 DMCFGBDF   EQU   X'80'    set:   CMS buffers defined
2 DMCFGBCL   EQU   X'40'    set:   CMS buffers resident (class 3)
2 *                         reset: CMS buffers not resident (class 4)
2 DMCFGBNU   DS    XL2      number of CMS buffers
2 DMCFGATT   DS    X        attribute
2 DMCFGLOB   EQU   X'40'    set: large_objects
2 *                                files/volumes with more than 32 GB
2 DMCFGLFA   EQU   X'20'    set: large_files_allowed
2 DMCFGXCN   DS    CL8      XCS name
2 DMCFGHOS   DS    CL8      host name       :* MSCF host name
2            DS    XL14     unused
2 DMCFGMTL   DS    H        maximal I/O transfer length
2 DMCFGUID   DS    CL8      userid allowed to access pubset
2            DS    XL4      unused
2 *
2 * dynamic allocator values
2 *
```

```
2            DS    XL20     unused
2 DMCFGAPA   DS    F        predet primary alloc amount
2 DMCFGASA   DS    F        predet secondary alloc amount
2 DMCFGADL   DS    F        sec alloc doubling limit
2            DS    XL4      unused
2 *
2 * dynamic EAM values
2 *
2 DMCFGEMA   DS    F        minimal size of file SYSEAM
2 DMCFGEMI   DS    F        maximal size of file SYSEAM
2 DMCFGESA   DS    F        secondary allocation of file SYSEAM
2 DMCFGEMS   DS    F        virtual memory size of file SYSEAM
2 *
2 DMCFGREF   DS    XL4      counter of occupations (duplicate)
2 *
2 DMCFG#     EQU   *-DMCFMST      length of the STAM pubset entry
2 *
2            ORG   DMCFD    starting point of dynamic part
2 ****************************************
2 *  dynamic part of a volume set entry  *
2 ****************************************
2 DMCFEOC#   DS    F        counter for occupations
2 *
2 * dynamic volume set status values (1)
2 *
2 DMCFESTA   DS    X        dynamic status bytes (1)
2 DMCFECON   EQU   X'80'    set: volume set is connected
2 *
2 * dynamic volume set status values (2)
2 *
2 DMCFEST2   DS    X        dynamic status bytes (2)
2 DMCFEMCP   EQU   X'04'    set: master change in process
2 DMCFEERI   EQU   X'01'    set: eram inhibit
2 *
2            DS    XL4      unused
2 *
2 DMCFEATT   DS    X        attribute
2 DMCFERAI   EQU   X'10'    set: volset with RAID volumes
2 DMCFEGSV   EQU   X'08'    set: gs volumes
2 DMCFEDRV   EQU   X'02'    set: high availability by DRV
2 DMCFEKEY   EQU   X'01'    set: key volset
2 *
2            DS    XL16     unused
2 *
2 DMCFEVOL   DS    X        volume set format set
2 DMCFE2KN   EQU   0        NK2 (2K native)
2 DMCFE4KN   EQU   1        NK4 (4K native)
2 DMCFE4KO   EQU   2        NK2, allocation unit multiple of 4K
```

```
2 *                          (4K oriented)
2 * dynamic cache values
2 *
2 DMCFECSZ   DS    F         size of cache buffer
2 *
2 DMCFECB8   DS    X         byte for bit values
2 DMCFECBU   EQU   X'80'     set  : cache size unit is KB
2 *                          reset: cache size unit is MB
2 DMCFECDS   EQU   X'40'     set  : data security ensured
2 DMCFECDB   EQU   X'20'     set  : double recording by buffer
2 DMCFECDD   EQU   X'10'     set  : cache deactivated
2 DMCFECIH   EQU   X'08'     set  : cache in hold
2 DMCFECCU   EQU   X'04'     set  : cache used
2 DMCFECSF   EQU   X'02'     set  : save file failed
2 *
2 DMCFECM    DS    X         cache medium set
2 DMCFECNC   EQU   0         no cache
2 DMCFECDC   EQU   1         controller
2 DMCFECES   EQU   2         expanded storage
2 DMCFECGS   EQU   3         global storage
2 DMCFECMM   EQU   4         main memory
2 *
2 DMCFECS    DS    X         segment size set
2 DMCFEC4    EQU   0         4 KB
2 DMCFEC8    EQU   1         8 KB
2 DMCFEC16   EQU   2         16 KB
2 DMCFEC32   EQU   3         32 KB
2 *
2 DMCFECU1   DS    XL1       GS unit_1
2 DMCFECU2   DS    XL1       GS unit_2
2 *
2 DMCFECFO   DS    X         force out set
2 DMCFECNF   EQU   0         no force out
2 DMCFECIP   EQU   1         at low filling
2 DMCFECIN   EQU   2         at high filling
2 *
2 DMCFECFE   DS    X         prefetch set
2 DMCFECFN   EQU   0         no prefetch
2 DMCFECFL   EQU   1         low
2 DMCFECFH   EQU   2         high
2 *
2 DMCFECFS   DS    X         file selection
2 DMCFEBUS   EQU   0         by user
2 DMCFEALL   EQU   1         all
2 DMCFEAUS   EQU   2         auto select
2 *
2 DMCFECAS   DS    H         size of allocation unit (# of half pages)
2 DMCFEMTL   DS    H         maximal I/O transfer length
```

```
2            DS    XL10     unused
2 *
2 * dynamic allocator values
2 *
2 DMCFEAL5   DS    F        residual space at sat level 5
2 DMCFEAL4   DS    F        residual space at sat level 4
2 DMCFEAL3   DS    F        residual space at sat level 3
2 DMCFEAL2   DS    F        residual space at sat level 2
2 DMCFEAL1   DS    F        residual space at sat level 1
2            DS    XL12     unused
2 DMCFEAZP   DS    F        residual space for ZIP startup
2 *
2            DS    XL16     unused
2 DMCFEREF   DS    XL4      counter of occupations (duplicate)
2 *
2 DMCFE#     EQU   *-DMCFMST      length of the STAM volume
2 *                               set entry
```

DSECT für den Ausgabebereich des Makros (XPAND=OCC)

```
           STAMCE MF=D,PREFIX=D,XPAND=OCC,VERSION=5
1          #INTF REFTYPE=REQUEST,                                        C
1                INTNAME=STAM,                                         C
1                INTCOMP=5
1          MFCHK    MF=D,                                              C
1                SUPPORT=(C,D,E,L,M,S),                                C
1                PREFIX=D,                                             C
1                MACID=MCH,                                            C
1                DMACID=MCH,                                           C
1                DNAME=MCHOCC,                                         C
1                PARAM=,                                               C
1                SVC=33,                                               C
1                ALIGN=F
2 DMCHOCC  DSECT ,
2                *,##### PREFIX=D, MACID=MCH #####
1          STAMLY   MF=D,                                              C
1                PREFIX=D,                                             C
1                MACID=MCH,                                            C
1                PARAM=,                                               C
1                VERSION=5,                                            C
1                XPAND=OCC,                                            C
1                FUNCT=1,                                              C
1                CG27=DMCH,                                            C
1                CATID=,                                               C
1                AREA=,                                                C
1                LENGTH=,                                              C
1                REF=,                                                 C
1                HOST=,                                                C
1                SELECT=,                                              C
1                PUBSET=
2          #INTF REFTYPE=REQUEST,                                        C
2                INTNAME=STAMLY,                                       C
2                INTCOMP=5
2 DMCHOST    DS    OF
2 DMCHSYS    DS    X        sysid
2 DMCHUNUS   DS    XL3      unused
2 DMCHUSID   DS    CL8      userid
2 DMCHTSN    DS    CL4      tsn
2 DMCHTID    DS    F        tid
2 DMCH#      EQU   *-DMCHOST     length of the occupation entry
```

Beispiel

Im folgenden Programm wird der Makro **STAMCE** aufgerufen. Dabei werden von den ersten 25 der von **STAMCE** übergebenen MRSCAT-Einträgen die Katalogkennung und der BCAM-Name des Systems auf den Bildschirm ausgegeben.

```
STAMCE   START
         PRINT NOGEN
         BALR   10,0
         USING  *,10
         USING  DSTAM3,6  ------------------------------------------------  (1)
         USING  DSTAM4,7  ------------------------------------------------  (2)
         STAMCE MF=E,PARAM=STAM3,VERSION=5  ------------------------------  (3)
         LR     7,1  -------------------------------------------------------  (4)
         L      6,DMCEAREA  ------------------------------------------------  (5)
         L      5,=F'25'
         WROUT  HEADER,WROUTERR ------------------------------------------  (6)
SHOW     CLC    DMCFSCTD,=AL4(DMCELAST)------------------------------------  (7)
         BE     WROUTERR
         MVC    CATID,DMCFSCTD---------------------------------------------  (8)
         MVC    PROCESS,DMCFFBCA
         CLI    PROCESS,X'00'
         BNE    WROUT2
         MVC    PROCESS,=CL8' '
WROUT2   WROUT  OUTREC,WROUTERR
         LA     6,DMCFG#(6)
         BCT    5,SHOW
WROUTERR TERM
*** Definitions
STAM3    STAMCE CATID=' ',MF=L,VERSION=5
HEADER   DC     Y(HEADERE-HEADER)
         DC     CL2'  '
         DC     CL1' '
         DC     C'CATID  PROCESSOR '
HEADERE  EQU    *
OUTREC   DC     Y(OUTRECE-OUTREC)
         DC     CL2'  '
         DC     CL1' '
CATID    DS     CL4
         DC     CL3'   '
PROCESS  DS     CL8
OUTRECE  EQU    *
         LTORG
         DS     0F
DSTAM3   STAMCE MF=D,PREFIX=D,XPAND=MCE,VERSION=5
DSTAM4   STAMCE MF=D,PREFIX=D,XPAND=PL,VERSION=5
         END
```

Ablaufprotokoll:

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,stamce), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,stamce))
%  ASS6011 ASSEMBLY TIME: 210 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 27 MSEC
//end
%  ASS6012 END OF ASSEMBH
/start-executable-program library=macexmp.lib,element-or-symbol=stamce
%  BLS0523 ELEMENT 'STAMCE', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'STAMCE', VERSION ' ' OF '<date> <time>' LOADED
CATID  PROCESSOR ————————————————————————————————————————————————————  (9)
A      N89H04
AAK3   D015B219
AAK4   D015B219
AAN3   D015B219
AA4N
AKEY   HELIOS2
ALB2
ANG3   ANGELA2
AP13   STARTB2
BAB2   BABETTE2
BAB3   BABETTE2
BECK
BEDS   SOPHIE2
BSAD   D015B007
BS41
BUEB   D015B011
BUR3   D017ZE39
BUR4   D017ZE39
BUR5
.
.
.
B202
B203   D015B019
```

(1) Dem Assembler wird Register R6 als Basisadressregister zur Adressierung der DSECT für den Ausgabebereich des Makros **STAMCE** zugewiesen. Diese DSECT wird an der symbolischen Adresse DSTAM3 durch einen **STAMCE**-Aufruf mit MF=D und XPAND=MCE erzeugt.

(2) Dem Assembler wird Register R7 als Basisadressregister zur Adressierung der DSECT für den Ausgabebereich des Makros **STAMCE** zugewiesen. Diese DSECT wird an der symbolischen Adresse DSTAM4 durch einen **STAMCE**-Aufruf mit MF=D und XPAND=PL erzeugt.

(3) Der Makro **STAMCE** wird in seiner E-Form aufgerufen. Der zugehörige Datenbereich wird an der symbolischen Adresse STAM3 durch einen **STAMCE**-Aufruf mit MF=L erzeugt, wobei

CATID=' ' (Leerzeichen, für alle MRSCAT-Einträge)

eingetragen wird.
Da für AREA kein Wert angegeben ist, wird die AREA-Adresse implizit im Feld DMCEAREA des Datenbereichs des Makros **STAMCE** abgelegt (siehe DSTAM4: STAMCE MF=D, PREFIX=D, XPAND=PL, VERSION=5).

(4) In Register R7 wird die Anfangsadresse des Datenbereichs des Makros **STAMCE** geladen. Dies ist für die Adressierung des Ausgabebereichs über DMCEAREA (zeigt auf die Adresse des Ausgabebereichs) nötig.

(5) Register R6 wird mit der Adresse des Ausgabebereichs geladen.

(6) Ausgabe der Überschriftszeile.

(7) Prüfung, ob das Ende der gelieferten MRSCAT-Einträge erreicht ist.

(8) In einer Schleife werden die ersten 25 MRSCAT-Einträge ausgewertet, die von **STAMCE** im Ausgabebereich hinterlegt worden sind: für jeden Eintrag werden die Katalogkennung (Feld DMCFSCTD der DSECT) und der BCAM-Name des Systems (Feld DMCFFBCA der DSECT) - bzw. Leerzeichen, falls kein BCAM-Name eingetragen ist - in einen Ausgabesatz übertragen und auf dem Bildschirm ausgegeben. Anschließend wird die DSECT um die Länge eines Eintrags (DMCFG#) verschoben.

(9) Ausgabe der ersten 25 MRSCAT-Einträge (gekürzt dargestellt).

STXIT – Ausgang für Unterbrechungsereignis spezifizieren

Allgemeines

Anwendungsgebiete: STXIT-Verfahren; siehe Seite 133
Starten, Unterbrechen und Beenden; siehe Seite 72
Kommunikation; siehe Seite 167
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

Während eines Programmlaufs auftretende Ereignisse beeinflussen den weiteren Programmablauf, wie z.B.

– ungültiger Operationscode, ungültiger SVC
– Datenfehler, Überlauf
– Ende Programmlaufzeit, Break-, Escape-Unterbrechungen oder INFORM-PROGRAM-Kommando,
– Adressfehler (siehe Tabelle 14 auf Seite 912)

Das STXIT-Verfahren bietet mit den Makros

STXIT STXIT-Prozess definieren
EXIT STXIT-Prozess beenden
LEVCO Verarbeitungsebene (Prozesspriorität) des STXIT-Prozesses ändern
CONTXT Zugriff zum unterbrochenen Prozess oder zum Basisprozess ermöglichen
SETIC Zeitgeberintervall setzen (CPU-Zeit oder Realzeit)

dem Anwender die Möglichkeit, derartige Programmunterbrechungen mit eigenen (STXIT-) Routinen selbst zu behandeln und damit evtl. einer vorzeitigen Programmbeendigung vorzubeugen. Eine STXIT-Routine ist ein Programmabschitt in einem Haupt- oder Unterprogramm, der im Falle einer bestimmten Programmunterbrechung vom Ablaufteil von BS2000 als selbstständiger Prozess (eigener Registersatz (PCB), eigene Prozesspriorität) aktiviert wird. In dem Programmabschnitt reagiert der Anwender auf das von ihm angenommene Unterbrechungsereignis.

Makrobeschreibung

Mit dem Makro **STXIT** ordnet der Anwender die STXIT-Routinen seines Programms den auftretenden Unterbrechungsereignissen (Tabelle 14 auf Seite 912) zu.
Die angebenen Zuordnungen werden intern in einen STXIT-Verwaltungsblock eingetragen.

Zwei Modi des **STXIT**-Aufrufes müssen unterschieden werden:

a) **STXIT**-Aufrufe ohne die Operanden STXDNEW/STXDID:
Für den ersten **STXIT**-Aufruf wird ein STXIT-Verwaltungsblock angelegt. In diesen werden die spezifizierten Zuordnungen „STXIT-Ereignisklasse - STXIT-Routine“ eingetragen. Jeder weitere Aufruf modifiziert oder ergänzt die in diesem Verwaltungsblock eingetragenen Zuordnungen (Update des Verwaltungsblockes). Einer STXIT-Ereignisklasse kann somit in einem Programm oder Programmsystem immer nur eine STXIT-Routine zugeordnet sein. Bei Modifizierung einer Zuordnung „STXIT-Ereignisklasse - STXIT-Routine“ ist nur die zuletzt vorgenommene wirksam.

b) **STXIT**-Aufruf mit den Operanden STXDNEW/STXDID:
Für jeden **STXIT**-Aufruf mit STXDNEW wird ein eigener STXIT-Verwaltungsblock mit den angegebenen Zuordnungen angelegt. Die Verwaltungsblöcke sind miteinander verkettet, und die STXIT-Routinen für die gleiche STXIT-Ereignisklasse werden nach einem vorgegebenen Reihenfolgenprinzip aktiviert. Dem Aufrufer wird eine Kennung für den Verwaltungsblock übergeben. Mittels dieser Kennung kann er in einem nachfolgenden **STXIT**-Aufruf (mit STXDID=...) die in dem betreffenden Verwaltungsblock eingetragenen Zuordnungen ergänzen, modifizieren oder aufheben.

In einem Programmsystem können beide Formen des **STXIT**-Aufrufs nebeneinander verwendet werden. Maximal können 100 STXIT-Verwaltungsblöcke für ein Programmsystem angelegt werden. Die Verarbeitungsebene eines STXIT-Prozesses kann während seines Ablaufs geändert werden (Makro **LEVCO**).
Eine STXIT-Routine wird mit **EXIT** oder **TERM** beendet. Im Aufruf **EXIT** kann angegeben werden, ob weitere, der STXIT-Ereignisklasse zugeordnete STXIT-Routinen gestartet werden sollen. Gleichen STXIT-Ereignisklassen zugeordnete STXIT-Prozesse können sich gegenseitig unterbrechen (geschachtelter Ablauf).

Hinweise

– Der Ereignisklasse „SVC“ kann in einem Programm (-system) nur eine STXIT-Routine zugeordnet werden. Die Zuordnung muss im ersten **STXIT**-Aufruf erfolgen oder sich auf den zuerst angelegten STXIT-Verwaltungsblock beziehen (die SVC-Nummer wird beim Auftreten des Ereignisses in Register R4 des STXIT-Prozesses übergeben).

– Für die STXIT-Ereignisklasse „Intervallzeitgeber CPU-Zeit“ bzw. „Intervallzeitgeber Realzeit“ ist zu beachten, dass zu einem Zeitpunkt immer nur ein Zeitintervall für die CPU-Zeit und/oder für die Realzeit gesetzt ist (siehe letzter Aufruf des Makros **SETIC**).

Makroaufrufformat und Operandenbeschreibung

```
STXIT

[ { STXDNEW=adr / (r) } ]
  { STXDID=adr / (r)   }

,MODE=DEFUNCD / DEFMODE / INTMODE
[,STXMSG=adr / (r)]
[,TERM=list]
[,TIMER=list]
[,ERROR=list]
[,ABEND=list]
[,PROCHK=list]
[,RUNOUT=list]
[,RTIMER=list]
[,ESCPBRK=list]
[,HWERROR=list]
[,SVC=list]
[,SVCLIST=adr / (r) / CLOSE]
[,CONTXTL=adr / (r)]
[,INTR=list]
[,INTRBUF=adr / (r) / CLOSE]
,TERMRUN=STD / FORCED
[,MIGRATE=list]
[,PARMOD=24 / 31]
[,MF=L / (E,..)]
```

Die im Operandenteil stehende Variable `list` entspricht folgendem Ausdruck:

```
list ≙ { (adr[,zahl])  }
       { ((r)[,zahl])  }
       { (CLOSE)       }
```

Dabei haben die Operandenwerte folgende Bedeutung:

adr
symbolische Adresse (Name) einer STXIT-Routine.

(r)
r = Register mit dem Adresswert adr.

zahl
maximale Anzahl der an einer Schachtelung beteiligten STXIT-Routinen.
$0 \leq$ zahl ≤ 127. Voreinstellung: zahl = 0.

CLOSE
Die Zuordnung zwischen der STXIT-Ereignisklasse und der STXIT-Routine wird aufgehoben.

STXDNEW=
Ein (neuer) STXIT-Verwaltungsblock wird angelegt. In den Verwaltungsblock werden die Zuordnungen STXIT-Ereignisklasse zu STXIT-Routine eingetragen. Dem Aufrufer wird vom System eine Kennung für den Verwaltungsblock übergeben. Mittels dieser Kennung kann der Aufrufer in nachfolgenden **STXIT**-Aufrufen die festgelegten Zuordnungen modifizieren.

adr
symbolische Adresse (Name) eines Feldes, in das die Kennung eingetragen wird.
Feldlänge = 4 Byte.

(r)
r = Register mit dem Adresswert adr.

STXDID=
Die in einem vorausgegangenen **STXIT**-Aufruf festgelegten Zuordnungen sollen modifiziert werden. Der entsprechende STXIT-Verwaltungsblock wird über die zurückgegebene Kennung angesprochen.

adr
symbolische Adresse (Name) des Feldes mit der Kennung für den Verwaltungsblock.
Feldlänge = 4 Byte.

(r)
r = Register mit dem Adresswert adr.

MODE=
bestimmt den Adressierungsmodus für die STXIT-Routine.
Der Adressierungsmodus wird den STXIT-Ereignisklassen zugeordnet, die im gleichen **STXIT**-Makroaufruf definiert werden (Wirkung: ereignisklassen-spezifisch). Durch Aufruf mehrerer **STXIT**-Makros mit unterschiedlichen Ereignisklassen kann also der Adressierungsmodus separat für jede Ereignisklasse eingestellt werden.

DEFUNCD
Die STXIT-Routine wird in demselben Adressierungsmodus gestartet, der zum Zeitpunkt des **STXIT**-Aufrufs eingeschaltet war. Es wird nicht geprüft, ob zum Zeitpunkt der Unterbrechung der gleiche Adressierungsmodus eingeschaltet ist.

DEFMODE
Die STXIT-Routine wird in demselben Adressierungsmodus gestartet, der zum Zeitpunkt des **STXIT**-Aufrufs eingeschaltet war. Die STXIT-Routine wird jedoch nicht aktiviert, wenn der Adressierungsmodus zum Zeitpunkt des **STXIT**-Aufrufs ungleich dem Adressierungsmodus zum Zeitpunkt des Ereigniseintritts ist.

INTMODE
Die STXIT-Routine wird in demselben Adressierungsmodus gestartet, der zum Zeitpunkt der Unterbrechung eingeschaltet ist.

STXMSG=
Bestimmt die Adresse eines 4 Byte langen Feldes, dass die STXIT-Meldung enthält. Diese Meldung wird dem STXIT-Contingency-Prozess übergeben (im Register R1).

adr
symbolische Adresse (Name) des Meldungsfeldes.

(r)
r = Register mit dem Adresswert adr.

TERM=
beschreibt die Adresse der STXIT-Routine für die STXIT-Ereignisklasse „Programmbeendigung“ (Programmbeendigung durch synchrone Ereignisse).

list
siehe oben: Beschreibung der Variable `list`.

TIMER=
beschreibt die Adresse der STXIT-Routine für die STXIT-Ereignisklasse „Intervallzeitgeber CPU-Zeit“.

list
siehe oben: Beschreibung der Variable `list`.

ERROR=
beschreibt die Adresse der STXIT-Routine für die STXIT-Ereignisklasse „nicht behebbarer Programmfehler“.

list
siehe oben: Beschreibung der Variable `list`.

ABEND=
beschreibt die Adresse der STXIT-Routine für die STXIT-Ereignisklasse „ABEND“ (Programmbeendigung durch asynchrone Ereignisse).

list
siehe oben: Beschreibung der Variable `list`.

PROCHK=
beschreibt die Adresse der STXIT-Routine für die STXIT-Ereignisklasse „Programmfehler“.

list
siehe oben: Beschreibung der Variable `list`.

RUNOUT=
beschreibt die Adresse der STXIT-Routine für die STXIT-Ereignisklasse „Ende Programmlaufzeit“.

list
siehe oben: Beschreibung der Variable `list`.

RTIMER=
beschreibt die Adresse der STXIT-Routine für die STXIT-Ereignisklasse „Intervallzeitgeber Realzeit“.

list
siehe oben: Beschreibung der Variable `list`.

ESCPBRK=
beschreibt die Adresse der STXIT-Routine für die STXIT-Ereignisklasse „ESCPBRK“. Zusätzlich zum Ereigniscode in Register R3 wird der Routine der Funktionstastencode im rechten Byte von Register R4 zur Verfügung gestellt (Funktionstastencodes siehe Handbuch „TIAM“ [16] oder die Tabelle im Anhang ab Seite 1193). Diese Funktionalität wird nur zur Verfügung gestellt, wenn der TIAM-Partner ein Terminal ist und keine Applikation (z.B. OMNIS).

list
siehe oben: Beschreibung der Variable `list`.

HWERROR=
beschreibt die Adresse der STXIT-Routine für die STXIT-Ereignisklasse „Hardware Error“.

list
siehe oben: Beschreibung der Variable `list`.

SVC=
beschreibt die Adresse der STXIT-Routine für die STXIT-Ereignisklasse „SVC-Unterbrechung“. Zusätzlich zum Ereigniscode in Register R3 wird der Routine die SVC-Nummer im rechten Byte von Register R4 zur Verfügung gestellt. Der überwachte SVC wird nicht ausgeführt. Der unterbrochene PCB läuft nach EXIT hinter dem SVC weiter.

list
siehe oben: Beschreibung der Variable `list`.

SVCLIST=
beschreibt die Adresse eines Feldes mit SVC-Nummern.
Diese SVC-Nummern sind die Elemente der STXIT-Ereignisklasse.
Aufbau:

Byte 0:	Anzahl der SVC-Einträge > 0
Byte 1:	SVC-Nummer (sedezimal)
:	:
Byte n:	SVC-Nummer (sedezimal)

adr
symbolische Adresse (Name) des Feldes mit SVC-Nummern.

(r)
r = Register mit dem Adresswert adr.

CLOSE
Die Zuordnung zwischen der STXIT-Ereignisklasse „SVC“ und der STXIT-Routine wird aufgehoben.

CONTXTL=
beschreibt die Adresse eines 4 Byte langen Feldes (auf Wortgrenze ausgerichtet), in dem die Länge des Datenaustauschfeldes abgelegt wird. Dieses wird vom Makro **CONTXT** mit den Operanden SAVE und LAYOUT=FCONTXT für den Datenaustausch mit dem PCB des angegebenen Prozesses benötigt. Es gibt unterschiedliche Längen für /390-Server und x86-Server.

adr
symbolische Adresse (Name) des Längenfeldes

(r)
r = Register mit dem Adresswert adr

INTR=
beschreibt die Adresse der STXIT-Routine für die STXIT-Ereignisklasse „Mitteilung an das Programm“.

list
siehe oben: Beschreibung der Variable `list`.

INTRBUF=
beschreibt die Adresse eines Feldes für eine Mitteilung, die mit dem Kommando INFORM-PROGRAM gesendet wird. Das Feld muss 64 Byte groß sein.
Textlänge < Feldlänge: Ende der Mitteilung = X'00'.
Textlänge > Feldlänge: Mitteilung wird abgeschnitten.
Textlänge = 0: X'00' wird in das erste Byte des Feldes eingetragen.

adr
symbolische Adresse (Name) des Feldes für eine Mitteilung.

(r)
r = Register mit dem Adresswert adr.

CLOSE
Die „INTR-STXIT-Routine“ hat kein Empfangsfeld für eine Mitteilung definiert. Eine Mitteilung wird ignoriert.

TERMRUN=
beschreibt den Ablaufmodus der TERM/ABEND STXIT-Routine.

STD
Mit STD angemeldete STXIT-Routinen laufen nach dem LIFO- und dem Prioritätsprinzip der Einreihung in die Warteschlange ab.

FORCED
Mit FORCED angemeldete STXIT-Routinen laufen immer und als letzte der STXIT-Routinen ab. Die Reihenfolge untereinander wird durch das LIFO-Prinzip bestimmt.

MIGRATE=
beschreibt die Adresse der STXIT-Routine für die STXIT-Ereignisklasse „Live Migration“.

list
siehe oben: Beschreibung der Variable `list`.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB).

STXIT-Ereignisklasse, STXIT-Operand und zugeordnete Ereignisse

In der folgenden Tabelle ist dargestellt:
– welche Unterbrechungsereignisse eine STXIT-Ereignisklasse beinhaltet,
– nach welchem Prinzip (FIFO/LIFO) die einer Ereignisklasse zugeordneten STXIT-Routinen aktiviert werden (Warteschlangen-Einreihung),
– die maximal angebbare Schachtelungstiefe und
– der einem Unterbrechungsereignis zugeordnete Ereigniscode.

STXIT-Ereignisklasse	STXIT-Operand	Unterbrechungsergebnis	Ereigniscode in R3	Einreihung in Warteschlange	max. Schachtelungstiefe
Programmfehler	PROCHK	unzulässiger SVC	X'04'	LIFO	127
		unzulässiger Operationscode	X'58'		
		Datenfehler	X'60'		
		Exponentenüberlauf	X'64'		
		Divisionsfehler oder neg. Quadratwurzel	X'68'		
		Mantisse = 0	X'6C'		
		Exponentenunterlauf	X'70'		
		Dezimalüberlauf	X'74'		
		Festpunktüberlauf	X'78'		
Intervallzeitgeber CPU-Zeit	TIMER	„SETIC-Intervall“ abgelaufen für CPU-Zeit	X'20'	FIFO	127
Intervallzeitgeber Realzeit	RTIMER	„SETIC-Intervall“ abgelaufen für Realzeit	X'A0'	FIFO	127
		Sommer-/Winterzeitumstellung	X'C0'		
Ende Programmlaufzeit	RUNOUT	CPU-Zeitgrenze für die Task bzw. das Programm überschritten	X'80'	FIFO	0
nicht behebbarer Programmfehler	ERROR	privilegierter SVC	X'08'	LIFO	127
		Zugriff auf nicht vorhandene Speicherseite	X'48'		
		privilegierte Operation	X'54'		
		Adressenfehler (z.B. Ausrichtungsfehler, falsches Register)	X'5C'		

Tabelle 14: STXIT-Ereignisklassen und zugehörige Unterbrechungsereignisse (Teil 1 von 2)

STXIT-Ereignisklasse	STXIT-Operand	Unterbrechungsergebnis	Ereig-niscode in R3	Einreihung in Warte-schlange	max. Schachte-lungstiefe
nicht behebbarer Programmfehler (Fortsetzung)	ERROR	XA-Fehler bei SVC-Aufruf (im 31-Bit-Mode den 24-Bit-Datenbereich benutzt)	X'9C'	LIFO	127
		Realtimer (Condition Error)	X'A4'		
		Ausrichtungsfehler des Da-tenbereichs bei SVC-Aufruf	X'AC'		
		Validierungsfehler	X'B0'		
		Ungültige UNIT-Nr. im Standardheader	X'C4'		
Mitteilung an das Programm	INTR	INFORM-PROGRAM-Kommando	X'44'	LIFO	127
ESCPBRK	ESCPBRK	BREAK/ESCAPE (über Tasten)	X'84'	LIFO	127
Programmbeen-digung durch asynchrone Ereignisse	ABEND	vom System erkannter Feh-ler, z.B. Fehler im System, Leitungverlust	X'88'	LIFO	0
		START-PROGRAM, LOAD-PROGRAM, ABEND, EXIT-JOB, CANCEL-JOB	X'8C'		
		Adress-Übersetzungsfehler wegen Hardwarefehler	X'94'		
		Hardwarefehler (CPU)	X'A8'		
		Erzwungenes Entladen eines Subsystems (Systembetreu.)	X'B8'		
		nicht behebbarer DMS-Feh-ler	X'BC'		
Programmbeen-digung durch synchrone Ereignisse	TERM	TERM-SVC aus einem TU-Programm	X'90'	LIFO	0
		Programmbeendigung durch CMD-/LGOFF-Makro	X'98'		
SVC-Unter-brechung	SVC	Aufruf eines angegebenen SVC' s	X'50'	LIFO	127
Hardwarefehler	HWERROR	Ein-/Ausgabefehler bei Data-In-Virtual-Technik	X'28'	LIFO	0
Live Migration	MIGRATE	Live Migration	X'D0'	FIFO	127

Tabelle 14: STXIT-Ereignisklassen und zugehörige Unterbrechungsereignisse (Teil 2 von 2)

Hinweise zum Makroaufruf

– FIFO = First In First Out; LIFO = Last In First Out

– Operand ESCPBRK: Dieses STXIT-Ereignis betrifft nur Programmunterbrechungen über Funktionstasten. Es umfasst nicht Programmunterbrechungen auf Grund eines HOLD-PROCEDURE- bzw. ESCAPE- oder eines HOLD-PROGRAM- bzw. BREAK-Kommandos.

– Operand INTRBUF: bei Schachtelungstiefe > 0 wird der Meldungstext durch ein nachfolgendes INFORM-PROGRAM-Kommando überschrieben.

– Operand SVC und SVCLST: die Angabe des SVC 128 (STXIT-Gruppe) als Unterbrechungsereignis kann zu einer Schleife führen.

– Operand SVC und SVCLST: Der überwachte SVC wird nicht ausgeführt und der unterbrochene PCB läuft nach EXIT hinter dem SVC weiter.

– ungültige Adresse im **STXIT**-Aufruf: das Ereignis „Adressfehler“ wird gemeldet und eine in einem vorherigen **STXIT**-Aufruf zugeordnete STXIT-Routine aktiviert oder das Programm wird mit „Adressfehler“ beendet.

– Der Aufruf des Makros **TERM** in einem Programmsystem führt zur Aktivierung aller der STXIT-Ereignisklasse TERM zugeordneten STXIT-Routinen. Ein erneuter **TERM**-Aufruf - auch in einer STXIT-Routine (an Stelle von **EXIT**) - führt wunschgemäß zur sofortigen Programmbeendigung.

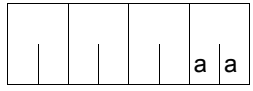
– Eine STXIT-Routine der Ereignisklasse ABEND oder TERM kann mit der Dialogtesthilfe AID getestet werden.
Ausnahme: STXIT-Routine, die durch ein CANCEL-JOB-Kommando aktiviert wurde.

Rückinformation und Fehleranzeigen

In Register R3 des STXIT-Prozesses steht das Unterbrechungsgewicht (Ereigniscode).

In Register R4 des STXIT-Prozesses wird der Funktionstastencode übergeben, wenn ein Unterbrechungsereignis der Klasse ESCPBRK aufgetreten ist, (für die Funktionstastencodes siehe Handbuch „TIAM“ [16] oder im Anhang ab Seite 1193). Diese Funktionalität wird nur zur Verfügung gestellt, wenn der TIAM-Partner ein Terminal ist und keine Applikation (z.B. OMNIS).
Beim Autreten eines Unterbrechungsereignisses der Klasse SVC wird die SVC-Nummer im Register R4 des STXIT-Prozesses übergeben.

R15:



Über die Ausführung des Makros STXIT wird im rechtsbündigen Byte des Register R15 ein Returncode übergeben.

X'aa'	**Erläuterung**
X'00'	Normale Ausführung.
X'04'	Funktion nicht ausgeführt. Ungültige Operanden.
X'08'	Funktion nicht ausgeführt. Ungültige Kennung für den STXIT-Steuerblock (Operand STXDID).
X'0C'	Funktion nicht ausgeführt. Kein Arbeitsspeicherplatz für STXIT-Steuerblock.
X'10'	Funktion nicht ausgeführt. Maximal 100 STXIT-Steuerblöcke können für ein Programmsystem angelegt werden
X'14'	Funktion nicht ausgeführt. Die STXIT-Ereignisklasse SVC ist nur für den zuerst angelegten STXIT-Verwaltungsblock erlaubt.
X'1C'	Funktion wird ausgeführt. STXIT-Kontrollblock kann nicht aktualisiert werden, weil der Speicherschlüssel im PCB nicht mit dem im Kontrollblock übereinstimmt.

SUSPEND – Prozess in Wartezustand versetzen

Allgemeines

Anwendungsgebiet:	Contingency-Verfahren; siehe Seite 111 STXIT-Verfahren; siehe Seite 133 Ereignissteuerung; siehe Seite 95
Makrotyp:	S-Typ, MF-Format **1**: (Standardform/E-Form/L-Form) siehe Seite 29

Makrobeschreibung

Der **SUSPEND**-Makro versetzt den aufrufenden Basis- oder Contingency-Prozess in einen Wartezustand, bis ein (STXIT-)Contingency-Prozess startet.
Vorsicht: Wird kein (STXIT-)Contingency-Prozess gestartet, so verbleibt der **SUSPEND**-gebende Prozess im Wartezustand!

Wird der **SUSPEND** in Verbindung mit der Ereignissteuerung verwendet (z.B. asynchroner **SOLSIG**-Aufruf mit nachfolgendem **SUSPEND**-Aufruf), liegt es in der Verantwortung des Anwenders, dafür zu sorgen, dass noch ein **POSSIG**-Signal eintrifft, um mit dem dadurch gestarteten Contingency-Prozesses den Wartezustand zu beenden.

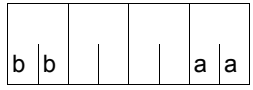
Makroaufrufformat und Operandenbeschreibung

SUSPEND
[MF=L / E]

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29 . Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Rückinformation und Fehleranzeigen

R15:



Über die Ausführung des Makros SUSPEND wird ein gegliederter Returncode (aa=primärer RC, bb=sekundärer RC) im Register R15 übergeben.

X'aa'	**X'bb'**	**Erläuterung**
X'00'	X'00'	Der aufrufende Prozess wurde in den Wartezustand versetzt.
X'04'	X'04'	Der Makro ist nur in TU-Programmen erlaubt. Keine Aktion.
X'10'	X'04'	Ungültige Operanden wurden angegeben. Keine Aktion.

SWITCH – Benutzer- und Auftragsschalter abfragen und verändern

Allgemeines

Anwendungsgebiete:	Benutzer- und Auftragsschalter; siehe Seite 73 Kommunikation; siehe Seite 167
Makrotyp:	S-Typ, MF-Format **3**: C-/D-/L-/E-/M-Form; siehe Seite 29

- Der Makro **SWITCH** vereinigt die Funktionalitäten der Makros GETSW, GETUS, SETSW und SETUS.

Jeder Benutzerkennung stehen 32 **Benutzerschalter** zur Verfügung. Die Schalter sind im Benutzerkatalog abgelegt. Es werden nur die Benutzerschalter des Benutzerkatalogs des Home-Pubsets verwendet.
Die Benutzerschalter sind von 0 bis 31 durchnummeriert. Nach dem Einrichten einer Benutzerkennung sind alle 32 Schalter ausgeschaltet („OFF“). Danach behalten sie die Stellung, die ihnen der Anwender gibt.
Jeder Schalter kann einzeln ein- bzw. ausgeschaltet oder invertiert werden. Benutzerschalter sind permanente Schalter, d.h. sie behalten ihre Stellung auch nach EXIT-JOB.

Das Betriebssystem stellt jedem Auftrag 32 **Auftragsschalter** zur Verfügung. Die Auftragsschalter sind von 0 bis 31 durchnummeriert und in Listen des TCB abgelegt. Anders als bei den Benutzerschaltern sind Auftragsschalter zu Beginn eines Auftrags immer ausgeschaltet. Der Anwender muss die Bedeutung der Schalterstellungen für sein Programm selbst festlegen.
Jeder Schalter kann einzeln ein- bzw. ausgeschaltet oder invertiert werden. Die Auftragsschalter sind temporäre Schalter, d.h. sie behalten ihren Wert nur bis zum Ende des Auftrages (bis EXIT-JOB). Es ist zu beachten, dass auch einige Systemkomponenten und Dienstprogramme die Auftragsschalter benutzen (siehe auch Abschnitt „Benutzer- und Auftragsschalter“ auf Seite 73). Bei Ausführung des Kommandos SET-JOB-STEP werden die Schalter 16 bis 31 ausgeschaltet.

Makrobeschreibung

Mit dem Makro **SWITCH** kann der Anwender die Benutzerschalter, die seiner Benutzerkennung zugeordnet sind, und die Auftragsschalter, die seinen Aufträgen zugeordnet sind, ein- bzw. ausschalten, invertieren und abfragen.

Makroaufrufformat und Operandenbeschreibung

```
SWITCH

[MODE=TASK / USER]
,USERID=*OWN / adr
[,SWITCH=adr / (nr, ...)]
,ACTION=*READ / *WRITE / *ON / *OFF / *INVERT / adr
,CONST=YES / NO
,MF=D / C / L / E / M
[,PARAM=adr / (r)]
,PREFIX=J / p
,MACID=CSS / macid
```

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

ACTION=
gibt an, welche Funktion ausgeführt werden soll.

***READ**
Alle Benutzer- oder Auftragsschalter werden abgefragt. Der Operand SWITCH wird nicht ausgewertet.
Die Information, welcher Schalter ein- bzw. ausgeschaltet ist, wird nach dem MF=E-Aufruf in einem 4 Byte-Feld des Datenbereichs (Voreinstellung: JCSSSW) hinterlegt. Dabei entspricht das Bit 2^n dem Schalter n. Ist das Bit 2^n auf „1“ gesetzt, ist der Schalter n eingeschaltet. Ist es auf „0“ gesetzt, ist der Schalter ausgeschaltet.

***WRITE**
Alle Schalter, die im Operanden SWITCH angegeben sind, werden eingeschaltet. Nicht angegebene Schalter werden ausgeschaltet.

***ON**
Alle Schalter, die im Operanden SWITCH angegeben sind, werden eingeschaltet. Nicht angegebene Schalter bleiben unverändert.

***OFF**
Alle Schalter, die im Operanden SWITCH angegeben sind, werden ausgeschaltet. Nicht angegebene Schalter bleiben unverändert.

***INVERT**
Alle Schalter, die im Operanden SWITCH angegeben sind, werden invertiert. Nicht angegebene Schalter bleiben unverändert.

adr
symbolische Adresse (Name) eines 1 Byte langen Feldes, das die auszuführende Funktion in folgender Form enthält:

```
0  Schalter werden abgefragt
1  Schalter werden ein- bzw. ausgeschaltet
2  Schalter werden eingeschaltet
3  Schalter werden ausgeschaltet
4  Schalter werden invertiert
```

Bei MF=L ist dieser Operandenwert nicht erlaubt.

CONST=
gibt an, ob Equates generiert werden sollen oder nicht.

YES
Equates werden generiert.

NO
Es werden keine Equates generiert.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29 . Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

Bei der E-Form des Makroaufrufes wird die Adresse des Datenbereichs im Operanden PARAM abgelegt. Voreinstellung: JCSS$PL

MODE=
legt fest, ob die angegebene Funktion für Benutzer- oder für Auftragsschalter ausgeführt werden soll.
Die Angabe des Operanden ist außer bei MF=M Pflicht.

TASK
Es werden die Auftragsschalter abgefragt oder verändert.
Der Operand USERID wird nicht ausgewertet.

USER
Es werden die Benutzerschalter der angegebenen Kennung abgefragt oder verändert.

SWITCH=
benennt die Benutzer- bzw. Auftragsschalter, deren Stellung abgefragt oder geändert werden soll.
Wird der Operand SWITCH nicht angegeben, wird die voreingestellte Bitmaske X'00000000' verwendet.

(nr, ...)
nr = Nummer eines Schalters (0 .. 31).
Die mit diesem Operandenwert angegebene Liste von Schaltern benennt diejenigen Schalter, deren Stellung abgefragt oder geändert werden soll, bzw. unterteilt die Gesamtheit der Schalter für eine Änderung bei ACTION=*WRITE.

Auch wenn nur ein Schalter angegeben wird, müssen die Klammern gesetzt werden.

adr
symbolische Adresse (Name) eines 4 Byte langen Feldes (Bitmaske), in welchem jedes Bit einem Benutzer- bzw. Auftragsschalter entspricht:

Bit 2^0 ≙ Schalter 0, Bit 2^1 ≙ Schalter 1, ..., Bit 2^{31} ≙ Schalter 31

Diese Bitmaske gibt an, welche Schalter verändert werden sollen. Die Art der Änderung wird durch den Operanden ACTION bestimmt.
Ist das jeweilige Bit auf „1“ gesetzt, wird der entsprechende Schalter ein- bzw. ausgeschaltet oder invertiert. Ist das Bit auf „0“ gesetzt, wird der Schalter nicht verändert, außer bei ACTION=*WRITE: die mit „0“ belegten Schalter werden ausgeschaltet.

Bei MF=L ist die Angabe dieses Operandenwertes nicht erlaubt.

USERID=
gibt die Benutzerkennung an, für die die Benutzerschalter abgefragt oder verändert werden sollen.
Bei Auftragsschaltern (MODE=TASK) wird dieser Operand nicht ausgewertet.

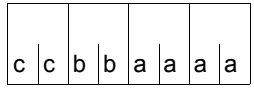
***OWN**
Die Benutzerschalter der eigenen Kennung sollen abgefragt oder verändert werden.

adr
symbolische Adresse (Name) eines 8 Byte langen Feldes, das linksbündig (ggf. mit nachfolgenden Leerzeichen) die gewünschte Benutzerkennung als Zeichenkette enthält.

*Bei ACTION≠*READ ist die Angabe einer fremden Benutzerkennung nur unter der privilegierten Benutzerkennung TSOS bzw. mit dem Systemprivileg „USER-ADMINISTRATION“ möglich. Bei ACTION=*READ können die Schalter einer fremden Benutzerkennung auch ohne Privileg abgefragt werden.*

Rückinformationen und Fehleranzeige

Standard-header:



Über die Ausführung des Makros SWITCH wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Funktion erfolgreich ausgeführt.
X'02'	X'00'	X'0001'	Warnung: Die Funktion wurde ausgeführt, aber im Umfeld der auftragsübergreifenden Benutzerschalter ist ein interner Fehler aufgetreten. Dieser Returncode kann nur bei Benutzerschaltern (MODE=USER) auftreten.
X'00'	X'01'	X'0002'	Operandenfehler.
X'00'	X'40'	X'0008'	Die angegebene Benutzerkennung existiert nicht.
X'00'	X'82'	X'000C'	Die angegebene Benutzerkennung ist gesperrt.
X'00'	X'82'	X'0010'	Die benötigte Zugriffsberechtigung fehlt.
	X'20'	X'0020'	(Verschiedene) interne Fehler.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Layout der DSECT für MODE=TASK

```
           SWITCH MF=D,MODE=TASK
1 *--------------------- START OF SWITCH -----------------------------*
1          MFCHK MF=D,PREFIX=J,MACID=CSS,PARAM=,                        C
1                SVC=42,                                                C
1                DMACID=CSS,SUPPORT=(D,L,C,M,E)
2 JCSS     DSECT ,
2                *,##### PREFIX=J, MACID=CSS #####
1 *
1          #INTF REFTYPE=REQUEST,INTNAME=SWITCH,INTCOMP=001
1 JCSS$PL  DS    OF          BEGIN OF PARAMETERAREA
1          FHDR  MF=(C,JCSS),EQUATES=YES
2          DS    0A
2 JCSSFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
2 JCSSIFID DS    0A              0   INTERFACE IDENTIFIER
2 JCSSFCTU DS    AL2             0   FUNCTION UNIT NUMBER
2 *                                  BIT 15    HEADER FLAG BIT,
2 *                                  MUST BE RESET UNTIL FURTHER NOTICE
2 *                                  BIT 14-12 UNUSED, MUST BE RESET
2 *                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
2 JCSSFCT  DS    AL1             2   FUNCTION NUMBER
2 JCSSFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 JCSSRET  DS    0A              4   GENERAL RETURN CODE
2 *
2 * GENERAL_RETURN_CODE CLEARED (X'00000000') MEANS
2 * REQUEST SUCCESSFUL PROCESSED AND NO ADDITIONAL INFORMATION
2 *
2 JCSSSRET DS    0AL2            4   SUB RETURN CODE
2 JCSSSR2  DS    AL1             4   SUB RETURN CODE 2
2 * ALWAYS CLEARED (X'00') IF MAIN_RETURN_CODE IS X'FFFF'
2 * Standard subcode2 values as defined by convention:
2 JCSSR2OK EQU   X'00'               All correct, no additional info
2 JCSSR2NA EQU   X'01'               Successful, no action was necessary
2 JCSSR2WA EQU   X'02'               Warning, particular situation
2 JCSSSR1  DS    AL1             5   SUB RETURN CODE 1
2 *
2 * GENERAL INDICATION OF ERROR CLASSES
2 *
2 * CLASS A    X'00'           FUNCTION WAS SUCCESSFULLY PROCESSED
2 * CLASS B    X'01' - X'1F'   PARAMETER SYNTAX ERROR
2 * CLASS C    X'20'           INTERNAL ERROR IN CALLED FUNCTION
2 * CLASS D    X'40' - X'7F'   NO CLASS SPECIFIC REACTION POSSIBLE
2 * CLASS E    X'80' - X'82'   WAIT AND RETRY
2 *
2 JCSSRFSP EQU   X'00'               FUNCTION SUCCESSFULLY PROCESSED
```

```
2 JCSSRPER EQU   X'01'               PARAMETER SYNTAX ERROR
2 *  3 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'01' - X'1F'
2 JCSSRFNS EQU   X'01'               CALLED FUNCTION NOT SUPPORTED
2 JCSSRFNA EQU   X'02'               CALLED FUNCTION NOT AVAILABLE
2 JCSSRVNA EQU   X'03'               INTERFACE VERSION NOT SUPPORTED
2 *
2 JCSSRAER EQU   X'04'               ALIGNMENT ERROR
2 JCSSRIER EQU   X'20'               INTERNAL ERROR
2 JCSSRCAR EQU   X'40'               CORRECT AND RETRY
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'40' - X'7F'
2 JCSSRECR EQU   X'41'               SUBSYSTEM (SS) MUST BE CREATED
2 *                                  EXPLICITELY BY CREATE-SS
2 JCSSRECN EQU   X'42'               SS MUST BE EXPLICITELY CONNECTED
2 *
2 JCSSRWAR EQU   X'80'               WAIT FOR A SHORT TIME AND RETRY
2 JCSSRWLR EQU   X'81'                    "     LONG       "
2 JCSSRWUR EQU   X'82'               WAIT TIME IS UNCALCULABLY LONG
2 *                                  BUT RETRY IS POSSIBLE
2 *  2 GLOBALLY DEFINED ISL ERROR CODES IN CLASS X'80' - X'82'
2 JCSSRTNA EQU   X'81'               SS TEMPORARILY NOT AVAILABLE
2 JCSSRDH  EQU   X'82'               SS IN DELETE / HOLD
2 *
2 JCSSMRET DS    0AL2           6    MAIN RETURN CODE
2 JCSSMR2  DS    AL1            6    MAIN RETURN CODE 2
2 JCSSMR1  DS    AL1            7    MAIN RETURN CODE 1
2 *
2 * SPECIAL LAYOUT OF LINKAGE_MAIN_RETURN_CODE (YYYY IN X'00XXYYYY')
2 *
2 JCSSRLNK EQU   X'FFFF'              LINKAGE ERROR / REQ. NOT PROCESSED
2 JCSSFHL  EQU   8              8    GENERAL OPERAND LIST HEADER LENGTH
2 *
1 *
1 * ****************************************************************
1 * *** END OF STANDARD HEADER - START SPECIAL SWITCH PARAMETERLIST
1 * ****************************************************************
1 *
1 JCSSHDR  EQU   X'009A0B01',4  std header task switch (TU)
1 *
1 *
1 *            *****  SET OF RETURN CODES  *****
1 *   OUT OF THE SYSTEM-WIDE DEFINED RETURN-CODES, THE FOLLOWING MAY
1 *   BE EXPECTED (CONFER INCLUDE FHDRI):
1 *      00 01 FFFF       SPECIFIED FUNCTION IS NOT SUPPORTED
1 *      00 03 FFFF       SPECIFIED VERSION IS NOT SUPPORTED
1 *      00 04 FFFF       ALIGNMENT ERROR
1 *
1 *   ADDITIONAL SPECIAL RETURNCODES ARE DEFINED :
1 *
```

```
1 *      00 00 0000       NORMAL EXECUTION
1 *      02 00 0001       EXECUTION, BUT ERROR IN WHENQ PROCESSING
1 *      00 01 0002       PARAMETER ERROR
1 *      00 40 0008       USERID NOT FOUND
1 *      00 82 000C       USERID SEVERED
1 *      00 82 0010       NO PRIVILEGED
1 *      XX 20 0020       SYSTEM ERROR
1 *
1 *   MAIN-RETURNCODES
1 JCSSOK   EQU   X'0000'   execution ok
1 JCSSWHQE EQU   X'0001'   execution with warning
1 JCSSPARE EQU   X'0002'   parameter error
1 JCSSUNFE EQU   X'0008'   userid not found
1 JCSSUSEE EQU   X'000C'   userid severed
1 JCSSNPRE EQU   X'0010'   no privileged
1 JCSSIERR EQU   X'0020'   internal error
1 *
1 *
1 *   DATEN-BEREICH
1 JCSSACT  DS    XL1         ACTION
1 JCSSREA  EQU   0              = *READ
1 JCSSWRT  EQU   1              = *WRITE
1 JCSSON   EQU   2              = *ON
1 JCSSOFF  EQU   3              = *OFF
1 JCSSINV  EQU   4              = *INVERT
1 *
1 JCSSRES  DS    XL3         FILLER
1 *
1 JCSSSW   DS    F           SWITCHES 31-0
1 *
1          ORG   JCSSSW
1 JCSSSW3  DS    XL1         SWITCHES 31 - 24
1 JCSSS31  EQU   X'80'           = SWITCH 31
1 JCSSS30  EQU   X'40'           = SWITCH 30
1 JCSSS29  EQU   X'20'           = SWITCH 29
1 JCSSS28  EQU   X'10'           = SWITCH 28
1 JCSSS27  EQU   X'08'           = SWITCH 27
1 JCSSS26  EQU   X'04'           = SWITCH 26
1 JCSSS25  EQU   X'02'           = SWITCH 25
1 JCSSS24  EQU   X'01'           = SWITCH 24
1 *
1 JCSSSW2  DS    XL1         SWITCHES 23 - 16
1 JCSSS23  EQU   X'80'           = SWITCH 23
1 JCSSS22  EQU   X'40'           = SWITCH 22
1 JCSSS21  EQU   X'20'           = SWITCH 21
1 JCSSS20  EQU   X'10'           = SWITCH 20
1 JCSSS19  EQU   X'08'           = SWITCH 19
1 JCSSS18  EQU   X'04'           = SWITCH 18
```

```
1 JCSSS17  EQU   X'02'          = SWITCH 17
1 JCSSS16  EQU   X'01'          = SWITCH 16
1 *
1 JCSSSW1  DS    XL1         SWITCHES 15 -  8
1 JCSSS15  EQU   X'80'          = SWITCH 15
1 JCSSS14  EQU   X'40'          = SWITCH 14
1 JCSSS13  EQU   X'20'          = SWITCH 13
1 JCSSS12  EQU   X'10'          = SWITCH 12
1 JCSSS11  EQU   X'08'          = SWITCH 11
1 JCSSS10  EQU   X'04'          = SWITCH 10
1 JCSSS9   EQU   X'02'          = SWITCH  9
1 JCSSS8   EQU   X'01'          = SWITCH  8
1 *
1 JCSSSW0  DS    XL1         SWITCHES  7 -  0
1 JCSSS7   EQU   X'80'          = SWITCH  7
1 JCSSS6   EQU   X'40'          = SWITCH  6
1 JCSSS5   EQU   X'20'          = SWITCH  5
1 JCSSS4   EQU   X'10'          = SWITCH  4
1 JCSSS3   EQU   X'08'          = SWITCH  3
1 JCSSS2   EQU   X'04'          = SWITCH  2
1 JCSSS1   EQU   X'02'          = SWITCH  1
1 JCSSS0   EQU   X'01'          = SWITCH  0
1 *
1 JCSSUID  DS    CL8         USERID
1 *
1 JCSS#    EQU   *-JCSS$PL   LENGTH OF PARAMETERAREA
1 *---------------------- END OF SWITCH --------------------------------*
```

Beispiel

```
SWITCH   START
         PRINT NOGEN
SWITCH   AMODE ANY
         BALR  R3,0
         USING *,R3
         MVC   TASKSW,INITSW
         SWITCH MF=M,ACTION=*READ
         SWITCH MF=E,MODE=TASK,PARAM=TASKSW ———————————————————— (2)
SW1      CLC   JCSSMRET,=Y(JCSSOK)
         BNE   ERROR
*
         TM    JCSSSW0,JCSSS1 ————————————————————————————————— (3)
SW2      BZ    END
         SWITCH MF=M,ACTION=*INVERT,SWITCH=(2,3)
         SWITCH MF=E,MODE=TASK,PARAM=TASKSW
         SWITCH MF=M,ACTION=*READ
         SWITCH MF=E,MODE=TASK,PARAM=TASKSW ———————————————————— (3)
SW3      CLC   JCSSMRET,=Y(JCSSOK)
         BNE   ERROR
         B     END
*
ERROR    EQU   *
******   ... ERROR HANDLING ... **************
         B     END
END      TERM
R3       EQU   3
INITSW   SWITCH MF=L,MODE=TASK
TASKSW   SWITCH MF=C,MODE=TASK
         END
```

Ablaufprotokoll:

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,switch), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,switch)), -
//        test-support=*aid
%  ASS6011 ASSEMBLY TIME: 395 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 84 MSEC
//end
%  ASS6012 END OF ASSEMBH
```

```
/mod-job-sw on=(1,2,3,4,5) ──────────────────────────────────────────── (1)
/load-program *m(macexmp.lib,switch),test-options=*aid,run-mod=*adv
/%in sw1
/%in sw2
/%in sw3
/%r
STOPPED AT LABEL: SW1 , SRC_REF: 20, SOURCE: SWITCH , PROC: SWITCH
/%d jcsssw %x;%r ────────────────────────────────────────────────────── (2)
*** TID: 009301BB *** TSN: 6WWQ ****************************************
CURRENT PC: 00000012    CSECT: SWITCH  ***********************************
V‘00000090‘ = JCSSSW   + #‘00000000‘
00000090 (00000000) 0000003E                                  ....
STOPPED AT LABEL: SW2 , SRC_REF: 24, SOURCE: SWITCH , PROC: SWITCH
/%d jcsssw0 %x;%r ────────────────────────────────────────────────────── (3)
CURRENT PC: 00000020    CSECT: SWITCH  ***********************************
V‘00000093‘ = JCSSSW0  + #‘00000000‘
00000093 (00000000) 3E                                        .
STOPPED AT LABEL: SW3 , SRC_REF: 52, SOURCE: SWITCH , PROC: SWITCH
/%d jcsssw %x;%r ─────────────────────────────────────────────────────── (4)
CURRENT PC: 0000003E    CSECT: SWITCH  ***********************************
V‘00000090‘ = JCSSSW   + #‘00000000‘
00000090 (00000000) 00000032                                  ....
```

(1) Vor der Programmausführung werden zur Demonstration die Schalter 1 bis 5 eingeschaltet.

(2) Die Auftragsschalter werden gelesen, die Ausgabe erfolgt in das Feld JCSSSW: X'0000003E' = $2^5 + 2^4 + 2^3 + 2^2 + 2^1$ bedeutet, dass die Schalter 1, 2, 3, 4 und 5 angeschaltet sind, alle anderen Schalter sind ausgeschaltet.

(3) Es wird abgefragt, ob Schalter 1 angeschaltet ist: Das Feld JCSSSW0 ist mit X'3E' belegt. Der logische Vergleich mit der Bitmaske ergibt nicht Null. Die Abarbeitung des Programms wird fortgesetzt.

(4) Nach der Invertierung der Schalter 2 und 3 werden alle Schalter gelesen, die Ausgabe erfolgt in das Feld JCSSSW: X'00000032' = $2^5 + 2^4 + 2^1$ bedeutet, dass die Schalter 1, 4 und 5 angeschaltet sind, alle anderen Schalter sind ausgeschaltet.

SYSFL – Systemdateien zuordnen

Allgemeines

Anwendungsgebiet: Systemdateien; siehe Seite 159
Makrotyp: S-Typ, MF-Format **1**: Standardform/E-/L-/C-/D-Form); siehe Seite 29

Die (Standard-) Dateinamen SYSDTA, SYSLST, SYSLST01 bis SYSLST99 und SYSOUT bezeichnen vom Betriebssystem benutzte Dateien zur Daten- bzw. Kommandoeingabe an das Betriebssystem oder zur Datenausgabe durch das Betriebssystem. Diese Dateien werden jeweils durch die Task erzeugt und bezeichnen anfänglich (primär) vorgegebene Ein- bzw. Ausgabebereiche.
Der Anwender kann die primäre Zuordnung aufheben und den (Standard-) Dateinamen eigene (katalogisierte) Dateien zuordnen. Einige der Standardnamen können auch gleichgesetzt werden. Die zugeordnete Datei (rechts vom Gleichheitszeichen) übernimmt dann die Funktion der (System-)Datei (links vom Gleichheitszeichen).
Mit dem Makro **SYSTA** kann die aktuelle Zuweisung für Systemdateien ausgegeben werden.

Die für die Ein-/Ausgabe zur Verfügung stehenden Systemdateien sind im allgemeinen Teil „Systemdateien“ auf Seite 159 beschrieben.

Makrobeschreibung

Mit dem Makroaufruf **SYSFL** wird über den Makroanschluss des Kommandosprachübersetzers MCLP das entsprechende Kommando gegeben, ohne dass der Programm-Modus verlassen wird (siehe Abschnitt „Makroaufrufe an den Kommandosprachübersetzer“ auf Seite 45).

Mit dem Makro **SYSFL** kann der Benutzer die Zuordnung der (System-)Dateien SYSDTA, SYSLST, SYSLST01 bis SYSLST99 und SYSOUT ändern.
Für die (System-)Datei SYSLST können Angaben über das Ausgabegerät und das Ausgabeformat gemacht werden.
Mit dem Makro **SYSFL** kann der Benutzer außerdem eine Bindemoduldatei (TASKLIB) für den dynamischen Bindelader angeben.
Meldungen bezüglich der Kommandobearbeitung werden auf SYSOUT ausgegeben und auch in einen Bereich des aufrufenden Programms übertragen.

Makroaufrufformate und Operandenbeschreibungen

Die folgenden Tabellen zeigen eine Zusammenstellung der verschiedenen Aufrufformate für den Makro **SYSFL**. Danach werden die einzelnen Formate und ihre Operanden ausführlich beschrieben.

SYSFL
[,PARMOD=24 / 31] [,MF=D / (E,..) / C]

Zur Beschreibung der Operanden PARMOD und MF siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29.

Zuordnung von **SYSDTA** (Beschreibung siehe Seite 933):

SYSFL
{ 'SYSDTA=pfadname / #dateiname / (SYSCMD) / (PRIMARY)' 'SYSDTA=pfadname1(element), VERSION=*STD / vers, TYPE=*STD / type' } [,adr / (r)] [,PARMOD=24 / 31] [,MF=L]

Zuordnung von **SYSOUT** (Beschreibung siehe Seite 935):

SYSFL
'SYSOUT=pfadname / (pfadname,EXTEND) / *DUMMY / (PRIMARY)' [,adr / (r)] [,PARMOD=24 / 31] [,MF=L]

Zuordnung von **SYSLST** (Beschreibung siehe Seite 936):

SYSFL
'SYSLST=pfadname / (pfadname,EXTEND) / #dateiname / (SYSCMD) / (PRIMARY)' [,adr / (r)] [,PARMOD=24 / 31] [,MF=L]

Ausgabe von **SYSLST** auf Drucker (Beschreibung siehe Seite 938):

SYSFL
'FILE=SYSLST, PRINTER=136 / 160 [,HREC=m] [,FORM=code] [,LOOP=vfb]] [,COPIES=anzahl1 / ([anzahl1],anzahl12)] [,CHARS=(z1[,z2][,z3][,z4])] ,CONTROL=NO / PHYS [,IMAGE=xxxx] [,SHIFT=spalten] [,DIS=zz]' [,adr / (r)] [,PARMOD=24 / 31] [,MF=L]

Zuordnung von **SYSLSTn** (Beschreibung siehe Seite 939):

SYSFL
'SYSLSTn=pfadname / (pfadname,EXTEND) / *DUMMY / (PRIMARY) / *SYSLSTn' [,adr / (r)] [,PARMOD=24 / 31] [,MF=L]

Zuordnung für den Dynamischen Bindelader **DBL** (Beschreibung siehe Seite 940):

SYSFL
'TASKLIB=pfadname / (NO)' [,adr / (r)] [,PARMOD=24 / 31] [,MF=L]

Hinweise

- Die Operanden müssen in Apostrophe eingeschlossen sein.
- Im Gegensatz zu den System-Ausgabedateien werden die zugeordneten katalogisierten Dateien nicht automatisch auf Drucker ausgegeben. Der Benutzer kann diese Dateien mit dem Kommando PRINT-DOCUMENT ausdrucken lassen.
- Werden bei der Angabe von (SYSCMD) oder (PRIMARY) die Klammern weggelassen, so werden diese Bezeichnungen als Dateinamen gewertet.
- Das Feld für das SYSOUT-Protokoll ist auf Wortgrenze auszurichten. Der Ausgabesatz beginnt mit dem 4 Byte langen Satzlängenfeld, das in Byte 0-1 die Satzlänge enthält. Nach dem Satzlängenfeld folgt der Ausgabetext.
  Aufbau des Ausgabebereichs:

```
         DS  0F
AUSGABE  DC  Y(AENDE-AUSGABE)  Ausgabebereich
SLF      DS  0CL4              4 Byte Satzlängenfeld
SL       DS  CL2               2 Byte Satzlänge des Ausgabesatzes
         DS  CL2               2 Byte reserviert
TEXT     DS  CL300             Ausgabetext
AENDE    EQU *
```

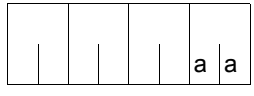
Hinweise zu temporären Dateien

- Das System kann ohne temporäre Dateien arbeiten.
- Temporäre Dateien sind taskbezogen und werden mit Taskbeendigung gelöscht.
- Temporäre Dateien können vom Typ BTAM, SAM, ISAM oder PAM sein.

Rückinformation und Fehleranzeigen

Meldungen bezüglich der Kommando-Bearbeitung sind Bestandteil des SYSOUT-Protokolls, das auf Wunsch dem Programm übergeben wird (Operand adr).

R15:



Über die Ausführung des Makros SYSFL wird im rechtsbündigen Byte des Registers R15 ein Returncode übergeben.

X'aa'	Erläuterung
X'00'	Normale Beendigung.
X'04'	Ungenügender Speicherbereich ist vorhanden, die Anforderung wurde nicht bearbeitet.
X'08'	Fehler im Datenbereich (Adressbereich)
X'0C'	Der letzte Ausgabesatz, der in den Benutzerbereich gebracht wurde, ist abgeschnitten.
X'10'	Makroaufruf/Kommando Fehler (das Kommando gab dem MCLP einen Fehler zurück).

Zuordnung von SYSDTA

```
SYSFL

{'SYSDTA=pfadname / #dateiname / (SYSCMD) / (PRIMARY)'
 'SYSDTA=pfadname1(element), VERSION=*STD / vers, TYPE=*STD / type'}

[,adr / (r)]
[,PARMOD=24 / 31]
[,MF=L]
```

SYSDTA=

pfadname
pfadname bedeutet: [:catid:][$userid.]dateiname

catid
Katalogkennung des Pubsets, auf dem die Datei gespeichert ist.
Voreinstellung ist die catid, die der userid im Benutzerkatalog zugeordnet wurde.

userid
Benutzerkennung, der die Datei zugeordnet ist.
Voreinstellung ist die userid aus dem SET-LOGON-PARAMETERS-Kommando.

dateiname
Name einer katalogisierten Datei (auch Dateigeneration).
Die Datei muss eine SAM- oder ISAM-Datei mit variabler Satzlänge sein. Bei einer ISAM-Datei ist außerdem erforderlich, dass der Schlüssel in Spalte 5 beginnt und eine Länge von 8 Byte hat (KEYPOS=5, KEYLEN=8, siehe Handbuch „DVS Makros“ [7]). Die Angabe einer Dateigruppe ist nur für Banddateien zulässig (nicht zu verwechseln mit Dateigenerationsgruppen).

#dateiname
Name einer temporären Datei.
= Zeichen, das durch den Systemparameter TEMPFILE als Präfix des Dateinamens temporärer Dateien festgelegt wird.
Das Zeichen können Sie mit dem Kommando SHOW-SYSTEM-PARAMETERS oder dem Makro NSIOPT ermitteln.
dateiname = beliebiger (Datei-)Name; Länge ≤ 30 Zeichen.
Siehe *Hinweise zu temporären Dateien* auf Seite 932.

(SYSCMD)
Die (System-)Datei SYSCMD wird der (System-)Datei SYSDTA gleichgesetzt, d.h. SYSCMD übernimmt zusätzlich die Funktion von SYSDTA. Von der (System-)Datei SYSCMD können dann nicht nur Kommandos, sondern auch Daten gelesen werden.

(PRIMARY)
SYSDTA wird auf die primäre Zuordnung zurückgelegt.

SYSDTA=pfadname1(element)
pfadname1 bedeutet: [:catid:][$userid.]bibliothek

catid	siehe SYSDTA=pfadname
userid	siehe SYSDTA=pfadname
bibliothek	Name einer PLAM-Bibliothek. Der Ausdruck bibliothek(element) darf maximal 41 Zeichen lang sein.
element	Name eines Elements der Bibliothek. Folgende Zeichen sind zugelassen: – alphabetische Zeichen A...Z – Sonderzeichen $, #, -, @ – numerische Zeichen 0...9 Das erste Zeichen muss alphabetisch sein. Das letzte Zeichen darf kein Bindestrich sein.

VERSION=
Ergänzung des Elementnamens durch die Versionsangabe.

***STD**
Es wird die höchste Version genommen.

vers
Versionsbezeichnung (max. 10 Zeichen).

TYPE=
Typ des Elements (1 Buchstabe).

***STD**
Elementtyp = S.

typ
Zeichen aus der Menge (D,S,M).

adr
symbolische Adresse (Name) eines Feldes für das SYSOUT-Protokoll. Aufbau siehe Seite 932.

(r)
r = Register mit dem Adresswert adr.

Zuordnung von SYSOUT

SYSFL
'SYSOUT=pfadname / (pfadname,EXTEND) / *DUMMY / (PRIMARY)' [,adr / (r)] [,PARMOD=24 / 31] [,MF=L]

SYSOUT=

pfadname
pfadname bedeutet: [:catid:][$userid.]dateiname

catid
Katalogkennung des Pubsets, auf dem die Datei gespeichert ist.
Voreinstellung ist die catid, die der userid im Benutzerkatalog zugeordnet wurde.

userid
Benutzerkennung, der die Datei zugeordnet ist.
Voreinstellung ist die userid aus dem SET-LOGON-PARAMETERS-Kommando.

dateiname
Name einer Datei bzw. Dateigeneration.
Die Datei wird als SAM-Datei auf gemeinschaftlichen Datenträgern eingerichtet, ihre Größe wird durch den Systemparameter SSMAPRI bestimmt, ihre SECONDARY-ALLOCATION durch den Systemparameter SSMASEC.
Die Datei kann aber auch auf privaten Datenträgern stehen. Der Benutzer muss das zuvor in einem CREATE-FILE-Kommando angeben.

Es empfiehlt sich, die voraussichtliche Größe der Datei „dateiname“ abzuschätzen und einen entsprechenden PRIMARY-ALLOCATION-Operanden im CREATE-FILE-Kommando zu geben, damit zu viele Speicheranforderungen vermieden werden.

(pfadname,EXTEND)
SYSOUT wird die Datei „dateiname“ zugeordnet. Datensätze werden ab Dateiende eingetragen.

***DUMMY**
SYSOUT wird eine Pseudodatei zugeordnet. Die Datensätze werden nicht gespeichert.

(PRIMARY)
für SYSOUT gilt wieder die primäre Zuordnung.

adr
symbolische Adresse (Name) eines Feldes für das SYSOUT-Protokoll. Aufbau siehe Seite 932.

(r)
r = Register mit dem Adresswert adr.

Zuordnung von SYSLST

SYSFL
'SYSLST=pfadname / (pfadname,EXTEND) / #dateiname / (SYSCMD) / (PRIMARY)'
[,adr / (r)]
[,PARMOD=24 / 31]
[,MF=L]

SYSLST=

pfadname
pfadname bedeutet: [:catid:][$userid.]dateiname

catid
Katalogkennung des Pubsets, auf dem die Datei gespeichert ist.
Voreinstellung ist die catid, die der userid im Benutzerkatalog zugeordnet wurde.

userid
Benutzerkennung, der die Datei zugeordnet ist.
Voreinstellung ist die userid aus dem SET-LOGON-PARAMETERS-Kommando.

dateiname
Name einer Datei oder Dateigeneration.
SYSLST wird dieser Datei bzw. Dateigeneration zugewiesen. Die Datei wird als SAM-Datei auf einem gemeinschaftlichen Datenträger eingerichtet, ihre Größe wird durch den Systemparameter SSMAPRI bestimmt, ihre SECONDARY-ALLOCATION durch den Systemparameter SSMASEC. Die Datei kann aber auch auf privaten Datenträgern stehen. Das muss der Benutzer zuvor in einem CREATE-FILE-Kommando angeben; ein Multifile-Band darf nicht verwendet werden.

Es empfiehlt sich, die voraussichtliche Größe der Datei „dateiname“ abzuschätzen und einen entsprechenden PRIMARY-ALLOCATION-Operanden im CREATE-FILE-Kommando zu geben, damit zu viele Speicheranforderungen vermieden werden.

Ist während der SYSLST-Ausgabe auf eine Plattendatei kein weiterer Speicherplatz verfügbar, so fordert das System ein Band an. Die Datei wird automatisch auf dieses Band kopiert und anschließend auf der Platte gelöscht. Die SYSLST-Ausgabe wird dann in die Banddatei fortgesetzt.

#dateiname
Name einer temporären Datei.
= Zeichen, das durch den Systemparameter TEMPFILE als Präfix für die Dateinamen temporärer Dateien festgelegt wird.
Das Zeichen können Sie mit dem Kommando SHOW-SYSTEM-PARAMETERS oder dem Makro NSIOPT ermitteln.
dateiname = beliebiger (Datei-)Name; Länge ≤ 30 Zeichen.
Siehe *Hinweise zu temporären Dateien* auf Seite 932.

***DUMMY**
SYSLST wird eine Pseudodatei zugeordnet,
Erläuterung siehe Kommando ADD-FILE-LINK, Handbuch „Kommandos“ [19].
Die Datensätze werden nicht gespeichert.

(pfadname,EXTEND)
SYSLST wird die Datei „dateiname“ zugeordnet; die Datensätze werden ab Dateiende eingetragen.

(PRIMARY)
für SYSLST gilt wieder die primäre Zuordnung.

adr
symbolische Adresse (Name) eines Feldes für das SYSOUT-Protokoll. Aufbau siehe Seite 932.

(r)
r = Register mit dem Adresswert adr.

Ausgabe von SYSLST auf Drucker

SYSFL
'FILE=SYSLST, PRINTER=136 / 160 [,HREC=m] [,FORM=code] [,LOOP=vfb]] [,COPIES=anzahl1 / ([anzahl1],anzahl12)] [,CHARS=(z1[,z2][,z3][,z4])] ,CONTROL=NO / PHYS [,IMAGE=xxxx] [,SHIFT=spalten] [,DIS=zz]' [1] [,adr / (r)] [,PARMOD=24 / 31] [,MF=L]

[1] Die Operanden PRINTER, HREC, COPIES, CHARS, CONTROL, IMAGE, SHIFT und DIA beziehen sich auf den PRNT-Makro von SPOOL V2.7 und sind in den aktuellen Handbüchern nicht mehr beschrieben.

FILE=SYSLST
beschreibt mit den nachfolgenden Operanden die Ausgabeform der (System-)Datei SYSLST.

adr
symbolische Adresse (Name) eines Feldes für das SYSOUT-Protokoll.
Aufbau siehe Seite 932.

(r)
r = Register mit dem Adresswert adr.

Die Operanden FORM und LOOP beschreiben die Form der Druckausgabe und sind beim **PRNTDOC**-Makro beschrieben (siehe Handbuch „SPOOL & Print - Makros und Exits“ [23]).

Zuordnung von SYSLSTn

SYSFL
'SYSLSTn=pfadnam' / (pfadname,EXTEND) / *DUMMY / (PRIMARY) / *SYSLSTm' [,adr / (r)] [,PARMOD=24 / 31] [,MF=L]

SYSLSTn=
n = zweistellige Zahl aus der Menge (01,02,...,99).
Die Dateien SYSLSTn sind nur wirksam, wenn ihnen (katalogisierte) SAM-Dateien zugeordnet werden. Primär ist ihnen die Datei zugeordnet, die der (System-)Datei SYSLST zum gleichen Zeitpunkt zugeordnet ist.

pfadname
pfadname bedeutet: [:catid:][$userid.]dateiname

catid
Katalogkennung des Pubsets, auf dem die Datei gespeichert ist.
Voreinstellung ist die catid, die der userid im Benutzerkatalog zugeordnet wurde.

userid
Benutzerkennung, der die Datei zugeordnet ist.
Voreinstellung ist die userid aus dem SET-LOGON-PARAMETERS-Kommando.

dateiname
Name einer Datei bzw. Dateigeneration. Die Datei wird SYSLSTn zugeordnet und als SAM-Datei auf gemeinschaftlichen Datenträgern eingerichtet.
Es empfiehlt sich, die voraussichtliche Größe der Datei „dateiname“ abzuschätzen und den Operanden PRIMARY-ALLOCATION im CREATE-FILE-Kommando entsprechend zu spezifizieren.

***DUMMY**
SYSLSTn wird eine Pseudodatei zugeordnet. Die Datensätze werden nicht gespeichert.

(pfadname,EXTEND)
SYSLST wird die Datei „dateiname“ zugeordnet; Datensätze werden ab Dateiende eingetragen.

(PRIMARY)
primäre Zuordnung.

***SYSLSTm**
m = zweistellige Zahl aus der Menge (01,02,...,99); m ≠ n
Die (System-)Dateien SYSLSTn können auch untereinander zugeordnet werden. Dabei ist zu beachten:

– wechselseitige Zuordnung ist nicht erlaubt

Beispiel für falsche Zuordnung

```
SYSFL     SYSLSTn  = *SYSLSTm
SYSFL     SYSLSTm  = *SYSLSTn
```

– die Zuordnung muss letztendlich auf eine katalogisierte oder eine Pseudodatei führen

Beispiel

```
SYSFL     SYSLSTn  = dateiname
SYSFL     SYSLSTm  = *SYSLSTn
SYSFL     SYSLSTp  = *SYSLSTm
```

adr
symbolische Adresse (Name) eines Feldes für das SYSOUT-Protokoll. Aufbau siehe Seite 932.

(r)
r = Register mit dem Adresswert adr.

Zuordnung für den Dynamischen Bindelader DBL

SYSFL
'TASKLIB=pfadname / (NO)' [,adr / (r)] [,PARMOD=24 / 31] [,MF=L]

TASKLIB=
bezeichnet eine Bindemoduldatei, die vom Bindelader DBL durchsucht wird, wenn
– beim Laden eines Programms eine Bindemoduldatei (-bibliothek) nicht angegeben wurde und/oder
– Externverweise noch befriedigt werden müssen.
Voreinstellung: TASKLIB=(PRIMARY) d.h. es wird die (Anwender-) Datei TASKLIB durchsucht oder, wenn diese nicht existiert, die Datei $TSOS.TASKLIB.

pfadname
pfadname bedeutet: [:catid:][$userid.]dateiname

catid
Katalogkennung des Pubsets, auf dem die Datei gespeichert ist. Voreinstellung ist die catid, die der userid im Benutzerkatalog zugeordnet wurde.

userid
Benutzerkennung, der die Datei zugeordnet ist.
Voreinstellung ist die userid aus dem SET-LOGON-PARAMETERS-Kommando.

dateiname
Name einer Bindemoduldatei.
Beim Binden eines Programms wird diese vom Bindelader DBL vor der (Anwender)-Datei TASKLIB oder der (System-)Datei $TSOS.TASKLIB nach dem Bindemodul durchsucht.

Hinweise
- „dateiname“ darf nicht der Name einer Dateigeneration sein.
- Prozedurschachtelung: TASKLIB erhält nach dem Kommando END-PROCEDURE oder EXIT-PROCEDURE die Zuordnung, die vor Aufruf der Prozedur gültig war.

(NO)
Die Zuordnung wird rückgängig gemacht. Es gilt TASKLIB=(PRIMARY) bzw. bei Schachtelung von Prozeduren die Zuweisung, die vor Aufruf der jeweiligen Prozedur bestand.

adr
symbolische Adresse (Name) eines Feldes für das SYSOUT-Protokoll. Aufbau siehe Seite 932.

(r)
r = Register mit dem Adresswert adr.

SYSTA – Systemdatei- und TASKLIB-Zuweisung ausgeben

Allgemeines

Anwendungsgebiet: Systemdateien; siehe Seite 159
Makrotyp: S-Typ, MF-Format **1**: Standardform/E-/L-/C-/D-Form); siehe Seite 29

Die (Standard-)Dateinamen SYSDTA, SYSCMD, SYSLST, SYSLST01, SYSLST02 bis SYSLST99 und SYSOUT bezeichnen vom Betriebssystem benutzte Dateien zur Daten- bzw. Kommandoeingabe an das Betriebssystem oder zur Datenausgabe durch das Betriebssystem. Diese Dateien werden jeweils durch die Task kreiert und bezeichnen anfänglich (primär) vorgegebene Ein- bzw. Ausgabebereiche.
Mit dem Makro **SYSFL** kann der Anwender die primäre Zuordnung aufheben und den (Standard-) Dateinamen eigene (katalogisierte) Dateien zuordnen. Einige der Standardnamen können auch gleichgesetzt werden.

Makrobeschreibung

Mit dem Makroaufruf **SYSTA** wird über den Makroanschluss des Kommandosprachübersetzers MCLP das Kommando SHOW-SYSTEM-FILE-ASSIGNMENTS gegeben, ohne dass der Programm-Modus verlassen wird (siehe Abschnitt „Makroaufrufe an den Kommandosprachübersetzer“ auf Seite 45).
Meldungen bezüglich der Kommandobearbeitung werden auf SYSOUT ausgegeben und auch in einen Bereich des aufrufenden Programms übertragen. Mit dem Kommando SHOW-SYSTEM-FILE-ASSIGNMENTS erhält der Benutzer Informationen über die Zuordnung der Systemdateien und der Bindemoduldatei (TASKLIB) für den dynamischen Bindelader.

Makroaufrufformat und Operandenbeschreibung

SYSTA
'([SYSCMD][,SYSDTA][,SYSOUT][,SYSLST][,SYSLST01][,SYSLST02]...[,SYSLST99][,TASKLIB])'
[,adr / (r)]
,SYSOUT=YES / NO
,DIALOG=NO / YES
[,PARMOD=24 / 31]
[,MF=L / (E,..) / D / C]

SYSCMD / SYSDTA / SYSOUT / SYSLST / SYSLST01 / SYSLST02 / .. / SYSLST99 / TASKLIB
Mehrere Systemdateien einschließlich der TASKLIB können angegeben werden. Die Klammern können weggelassen werden, wenn nur eine Systemdatei angegeben wird. Wenn die Operanden weggelassen werden, so erhält der Benutzer Information über alle Systemdateien und die TASKLIB. Diese Operanden müssen in Apostrophe eingeschlossen werden.

adr
symbolische Adresse (Name) eines Feldes für das SYSOUT-Protokoll.

(r)
r = Register mit dem Adresswert adr.

Das Feld für das SYSOUT-Protokoll ist auf Wortgrenze auszurichten. Der Ausgabesatz beginnt mit dem 4 Byte langen Satzlängenfeld, das in Byte 0-1 die Satzlänge enthält. Nach dem Satzlängenfeld folgt der Ausgabetext.
Aufbau des Ausgabebereichs:

```
         DS  0F
AUSGABE  DC  Y(AENDE-AUSGABE)  Ausgabebereich
SLF      DS  0CL4              4 Byte Satzlängenfeld
SL       DS  CL2               2 Byte Satzlänge des Satzes
         DS  CL2               2 Byte reserviert
TEXT     DS  CL300             Ausgabetext
AENDE    EQU *
```

SYSOUT=
beschreibt, ob das Protokoll auch nach SYSOUT ausgegeben werden soll.

YES
Ausgabe erfolgt nach SYSOUT.

NO
Es erfolgt keine Ausgabe nach SYSOUT.
In diesem Fall muss das Empfangsfeld adr spezifiziert sein.

DIALOG=
beschreibt, ob ein Fehlerdialog beim Erkennen von Syntaxfehlern geführt werden soll.

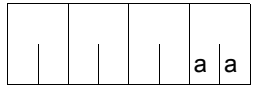
NO
Es wird kein Fehlerdialog geführt.

YES
Es soll ein Fehlerdialog geführt werden.

Rückinformation und Fehleranzeigen

Meldungen bezüglich der Kommando-Bearbeitung sind Bestandteil des SYSOUT-Protokolls, das ebenfalls in den Ausgabebereich übertragen wird (Operand adr).

R15:



Über die Ausführung des Makros SYSTA bis zur Kommandoausführung wird im rechtsbündigen Byte des Registers R15 ein Returncode übergeben.

X'aa'	**Erläuterung**
X'00'	Normale Beendigung
X'04'	Ungenügender Speicherbereich ist vorhanden, die Anforderung wurde nicht bearbeitet.
X'08'	Fehler im Datenbereich (Adressbereich)
X'0C'	Der letzte Ausgabesatz, der in den Benutzerbereich gebracht wurde, ist abgeschnitten.
X'10'	Makroaufruf/Kommando Fehler (das Kommando gab dem MCLP einen Fehler zurück).

TCHNG – Datenstationseigenschaften ändern

Anwendungsgebiet: Verkehr mit Datenstation; siehe Seite 164
Makrotyp: O-Typ; siehe Seite 28

- Stand der Beschreibung: TIAM V13.2A

Makrobeschreibung

Mit dem Makroaufruf **TCHNG** können Eigenschaften der logischen Datenstation durch das Benutzerprogramm geändert werden.
Die Wirkung des Aufrufs **TCHNG** bleibt bis zur Beendigung des betreffenden Benutzerprogramms oder bis zum nächsten **TCHNG**-Aufruf erhalten und gilt nur für die Aus- und Eingaben dieses Programms auf die Datenstation mit den Makros **RDATA**, **WROUT** und **WRTRD**.

Makroaufrufformat und Operandenbeschreibung

```
TCHNG
{ EDOPT=DYN
{ EDOPT=STAT{ ,MODE=LINE ,OHCOPY={N|Y} ,OHOM={N|Y}
            { ,OINFO={N|Y} ,ONOPOSN={N|Y}
            { ,OBELL={N|Y} ,IGETBS={N|Y}
            { ,ILCASE={N|Y} ,IGETFC={N|Y}
            { ,IGETIC={N|Y} ,ICFD={N|Y}
            { ,MODE=FORM ,IGETBS={N|Y} ,ILCASE={Y|N}
,OFLOW=SYS / USER
,SUB=OUT / OUTIN
,INFOLIN=NO / YES
,CLEAR=YES / NO
```

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

CLEAR=
bestimmt, ob beim Wechsel des Ausgabemodus der Bildschirm gelöscht werden soll.

YES
Beim Wechsel des Ausgabemodus, z.B.
von LINE zu FORM
von FORM zu LINE, PHYS, COMP
von PHYS zu LINE, FORM, COMP
von COMP zu FORM

soll der Bildschirm gelöscht werden. Erfolgt der Moduswechsel nicht nach einer Eingabe, wartet das System t Sekunden vor dem Löschen, damit der Benutzer seine letzte Ausgabe lesen kann (siehe Kommando MODIFY-TERMINAL-OPTIONS OVERFLOW-CONTROL=...).

NO
Beim Wechsel des Ausgabemodus wird der Bildschirm nicht gelöscht. Die nachfolgende Nachricht wird ohne Wartezeit ausgegeben.

Hinweis
Im Benutzerprogramm muss festgelegt sein, dass der Bildschirminhalt keine Funktionen des neueingestellten Modus beeinträchtigt, z.B. durch vormodifizierte Felder.

EDOPT=
gibt an, welche Edit-Options-Werte ausgewertet werden sollen.

DYN
Bei nachfolgenden Aufrufen der Makros **RDATA**, **WROUT** und **WRTRD** sollen die dort angegebenen Edit-Options-Werte ausgewertet werden (Voreinstellung durch das System).

STAT
Bei nachfolgenden Aufrufen der Makros **RDATA**, **WROUT** und **WRTRD** sollen die im **TCHNG**-Makro angegebenen Edit-Options-Werte für MODE, OBELL, OHCOPY, OHOM, OINFO, ONOPOSN, IGETBS, ILCASE, IGETFC, IGETIC und ICFD ausgewertet werden. Diese Festlegung gilt bis zu einer erneuten Änderung durch **TCHNG** oder bis Programmende. Sie ist jedoch ohne Einfluss auf Systemeingaben und Systemausgaben. EDOPT=STAT wird bei Verwendung von **VTSUCB** ignoriert.

INFOLIN=
bestimmt die Abbildung ankommender informativer Nachrichten an Datenstationen.

NO
Ankommende informative Nachrichten sollen an Datenstationen ohne Systemzeile wie normale Line-Mode-Nachrichten abgebildet werden.

YES
An Datenstationen ohne hardwaremäßige Systemzeile sollen ankommende informative Nachrichten in der letzten Bildschirmzeile abgebildet werden, wenn am Bildschirm gerade eine formatierte oder physikalische Nachricht abgebildet ist (siehe Makroaufruf **WROUT**).

OFLOW=
gibt die Art der Überlaufkontrolle an.

SYS
Bei langen Ausgaben des Benutzerprogramms soll das System eine Überlaufkontrolle durchführen, um einen Informationsüberlauf an der Datenstation zu verhindern. Die Art der Systemüberlaufkontrolle bestimmt der Datenstationsbenutzer durch das Kommando MODIFY-TERMINAL-OPTIONS (Voreinstellung und Wirkung siehe Handbuch „Kommandos“ [19]).

USER
Das System soll keine Vorkehrungen treffen, einen Informationsüberlauf bei langen Ausgaben des Benutzerprogramms zu verhindern. Eine Überlaufkontrolle kann so individuell vom Benutzerprogramm durchgeführt werden.

Hinweis
Meldungen des Betriebssystems unterliegen nach wie vor der vom System voreingestellten oder mit dem Kommando MODIFY-TERMINAL-OPTIONS festgelegten Überlaufkontrolle.

SUB=
bestimmt, wann das definierte Ersatzzeichen Zeichen ersetzen soll.

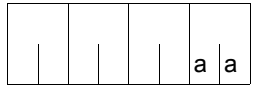
OUT
Das definierte Ersatzzeichen soll bei Line-Mode-Ausgaben unerlaubte Zeichen ersetzen.

OUTIN
Das definierte Ersatzzeichen soll
– bei Line-Mode-Ausgaben unerlaubte Zeichen ersetzen
– bei Line-Mode-Eingaben dem Benutzerprogramm das logische Steuerzeichen SUB übergeben (siehe Makroaufruf **VTCSET**).

MODE=
OHCOPY=
OHOM=
OBELL=
ONOPOSN=
OINFO=
IGETBS=
ILCASE=
IGETFC=
IGETIC=
ICFD=
Zur Bedeutung dieser Operanden siehe Makro **WRTRD** bzw. Makro **WROUT**.

Rückinformation und Fehleranzeigen

R15:



Über die Ausführung des Makros TCHNG wird im rechtsbündigen Byte des Registers R15 ein Returncode übergeben.

X'aa'	**Erläuterung**
X'00'	Normale Beendigung
X'04'	Nicht behebbarer Fehler
X'08'	Operandenfehler
X'0C'	Der Aufrufer ist nicht im Teilnehmerbetrieb.
X'10'	Operandenfehler
X'14'	Falsche Edit-Options-Angabe

TERM – Programm und Prozedurabschnitt beenden

Allgemeines

Anwendungsgebiet: Starten, Unterbrechen und Beenden; siehe Seite 72
Testhilfe; siehe Seite 166
Makrotyp: S-Typ, MF-Format **1**: Standardform/E-Form/L-Form; siehe Seite 29

Makrobeschreibung

Der **TERM**-Makro führt folgende Funktionen durch:
– Programm beenden (Standardwert)
– Programm und Prozedurabschnitt beenden (Operand UNIT=STEP)
– Speicherabzug ausgeben (Operand DUMP)
– Rückkehrcode an programmüberwachende Jobvariable übergeben (Operand URETCD)

Vor Programmbeendigung werden alle laufenden Ein-/Ausgabe-Operationen beendet.

Makroaufrufformat und Operandenbeschreibung

TERM
UNIT=PRGR / STEP ,DUMP=N / Y ,MODE=**N**ORMAL / **A**BNORMAL [,URETCD=code / adr / (r)] ,MF=S / (E,..) / L

UNIT=
bestimmt, ob bei der Beendigung des Programms nach Betriebsarten unterschieden werden soll.

PRGR
Das Programm wird beendet.

STEP
beendet das Programm, wobei unterschieden wird, in welcher Betriebsart das Programm gelaufen ist.

– Dialogbetrieb:
  Wurde das Programm in einer Nicht-S-Prozedur aufgerufen, so verzweigt das System zusätzlich zum nächsten SET-JOB-STEP-, EXIT-JOB-, END-PROCEDURE- oder CANCEL-PROCEDURE-Kommando. Wurde das Programm in einer S-Prozedur aufgerufen, so leitet das System zusätzlich die SDF-P-Fehlerbehandlung ein.

– Batch-Betrieb (ENTER-Datei):
  Das System verzweigt zusätzlich zum nächsten SET-JOB-STEP- oder EXIT-JOB-Kommando.

Hinweis
Folgende Angaben werden empfohlen:
UNIT=PRGR mit MODE=NORMAL
UNIT=STEP mit MODE=ABNORMAL

DUMP=
entscheidet, ob ein Speicherauszug ausgegeben werden soll.

N
Ein Speicherauszug wird nicht ausgegeben.

Y
gibt einen Speicherauszug aus, sofern kein Kommando MODIFY-TEST-OPTIONS mit DUMP=NO gegeben wurde.

MODE=
bestimmt die Art der Beendigung des Programms.

NORMAL
Das Programm soll normal beendet werden.
Für Anwender von Jobvariablen:
Ist eine programmüberwachende Jobvariable definiert, so wird ihre Zustandsanzeige auf C'$T␣' gesetzt.

ABNORMAL
Das Programm soll abnormal beendet werden. Die Meldung

`.... ABNORMAL PROGRAM TERMINATION (&00)` wird ausgegeben.
`(&00)= NRT0001`, wenn UNIT=PRGR angegeben wurde
`(&00)= NRT0101`, wenn UNIT=STEP angegeben wurde

Für Anwender von Jobvariablen:
Ist eine programmüberwachende Jobvariable definiert, so wird ihre Zustandsanzeige auf C'$A␣' gesetzt. Siehe *Hinweis* oben.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29 . Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Der folgende Operand steht nur dem Anwender mit dem Software-Produkt JV (siehe Handbuch „JV“ [22]) zur Verfügung:

URETCD=code
gibt einen 1 bis 4 Byte langen alphanumerischen Wert, dezimal (C'cccc') oder sedezimal (X'xxxxxxxx') an. Dieser Wert wird als Rückkehrcode vom Programm an die programmüberwachende Jobvariable übergeben (linksbündig, Byte 4 bis 7).

Hinweis

Fehlt der Operand, so wird der Wert C'␣␣␣␣' in die programmüberwachende Jobvariable übergeben.
Der Operand wird ignoriert, wenn keine programmüberwachende Jobvariable definiert wurde.

adr
relative virtuelle Adresse eines 4 Zeichen langen alphanumerischen Wertes. Dieser Wert wird als Rückkehrcode in die das Programm überwachende Jobvariable (Byte 4-7) übergeben.

(r)
r = Register, in dem ein 4 Zeichen langer alphanumerischer Wert steht. Dieser Wert wird als Rückkehrcode in die das Programm überwachende Jobvariable (Byte 4-7) übergeben. Ist URETCD=(r) angegeben, so wird Register R0 zerstört.

Funktionsweise

Wenn der Makroaufruf ausgeführt wird, tritt Folgendes auf:

- Alle dem Programm zugewiesenen und noch geöffneten Dateien werden geschlossen.
- Der Speicherplatz für das Programm wird freigegeben.
- Eine ggf. definierte STXIT-Routine für die Ereignisklasse TERM wird aktiviert.
- AIDSYS wird aufgerufen mit dem Ereignis „TERM“.
- In der Geräteliste werden die Bytes 8 - 30 für jedes freigegebene Gerät gelöscht. In der Operationsliste wird das erste Byte auf X'FF' gesetzt. Im Eintrag der Programmtabelle wird die Programmanfangsadresse auf 0 gesetzt (4 Byte).
- Anschließend wechselt das System in den Kommando-Modus.

Hinweise zum Makroaufruf

– Ist im URETCD-Operanden eine ungültige Adresse angegeben, so wird der Operand ignoriert.

– Register R1 enthält die Adresse des Datenbereichs.
Ist die Adresse des Datenbereichs ungültig oder wurden fehlerhafte Operanden eingegeben, so wird
TERM UNIT=STEP,MODE=ABNORMAL,DUMP=Y
ausgeführt und folgende Fehlermeldung ausgegeben:
`%.... ABNORMAL PROGRAM TERMINATION NRT0601`

– Der Aufruf des Makros **TERM** mit dem Operanden DUMP=Y veranlasst die Meldung
`PROCESSING INTERRUPTED AT...`
Die Ausgabe des Speicherabzugs hängt vom Wert des Operanden DUMP im Kommando MODIFY-TEST-OPTIONS ab. Bei DUMP=STD (Standardwert) veranlasst **TERM** eine der folgenden Meldungen:

  – im Dialogbetrieb:
  `DUMP DESIRED ? REPLY (Y=YES, N=NO)`
  Die Ausgabe des Dumps hängt von der Antwort ab.

  – im Batch-Betrieb und in Prozeduren:
  `SYSTEM REGULATIONS PROHIBIT DUMP`
  Der Dump wird nicht ausgegeben.

  Der Speicherabzug wird in nicht aufbereiteter Form als PAM-Datei auf Platte ausgegeben, siehe Makro **CDUMP2**. Die Datei, die den Speicherabzug enthält, wird unter der Kennung des Benutzers eingerichtet, der ihn angefordert hat. Sobald der Speicherabzug erstellt ist, werden die Meldung
  `DUMP WRITTEN,FILENAME=$userid.DUMP.tsn.i`
  und die TITLE-Zeile des Dumps ausgegeben.
  Mit dem Wert „i“ werden die Speicherabzüge durchnummeriert, wenn pro TSN mehrere angefordert werden. Die Datei kann mit dem Aufbereitungsprogramm DAMP ausgewertet werden.

TINF – Taskattribute lesen und modifizieren

Allgemeines

Anwendungsgebiet: Starten, Unterbrechen und Beenden; siehe Seite 72
Makrotyp: S-Typ, MF-Format **1**: Standardform/E-/L-Form; siehe Seite 29

Makrobeschreibung

Mit dem Makro **TINF** kann der Anwender
– die Runpriorität der Task
– das Taskattribut
– die Operanden für ein Deaktivierungsverbot der Task
lesen oder modifizieren (gemäß der im Benutzerkatalog festgelegten Werte).

Außerdem ist es möglich, dass sich die aufrufende Task an eine Affinitäts-Taskgruppe anschließt oder sich davon abmeldet.
Tasks, die häufig auf gemeinsame Daten schreibend zugreifen, werden als zueinander affin bezeichnet. Diese gemeinsamen Daten können sowohl im Systemadressraum (der von allen Tasks gemeinsam benutzt wird) als auch im Benutzeradressraum innerhalb von Common Memory Pools liegen. Solche Tasks können in einer Affinitäts-Taskgruppe zusammengefasst werden.
Dabei kann eine Task immer nur einer (oder keiner) Taskgruppe zugeordnet werden. Die Funktionalität der affinen Taskgruppen wird von dem Subsystem TANGRAM realisiert (siehe dazu auch im Handbuch „Systembetreuung“ [10]).

Makroaufrufformat und Operandenbeschreibung

TINF
ACCESS=R / W [,DEACT=Y / N] [,DWTR=Y / N] [,DSSR=Y / N] [,TPRYAD=adr / (r)] [,TTYPAD=adr / (r)] [,TGAFF=Y / N,TGIDAD=adr / (r)] [,PROCNAD=adr / (r)] [,PARMOD=24 / 31] [,MF=L / (E,..)]

ACCESS=
steuert die Lese/Schreib-Funktion der Operanden TPRYAD und TTYPAD.

R
Die Daten werden in die angegebenen Felder übertragen.

W
Die Daten werden von den angegebenen Feldern in den TCB übertragen.

DEACT=
gibt an, ob die Task in folgenden Fällen deaktiviert werden kann:
– Die System-Auslastung (CPU, Speicher, Paging-Rate) ist sehr hoch.
– Die Task nimmt bestimmte Wartezustände ein oder verweilt darin (siehe Operand DWTR). Ist der Operand DWTR ebenfalls angegeben, hat seine Einstellung Vorrang.
– Die Task steht auf Grund ihrer bisher verbrauchten Systemleistungen zur Deaktivierung an (siehe Operand DSSR). Ist der Operand DSSR ebenfalls angegeben, hat seine Einstellung Vorrang.

Y
Die Task kann deaktiviert werden.

N
Die Task soll nicht deaktiviert werden. Eine Deaktivierung aus anderen Gründen (z.B. Makro VPASS mit mehr als 500 ms Wartezeit) ist weiter möglich.

DWTR=
gibt an, ob die Task in folgenden Fällen deaktiviert werden kann:
– Sie nimmt bestimmte, länger andauernde Wartezustände ein, z.B. aufgrund eines PASS
– Sie verweilt länger in einem aktiven Wartezustand, z.B. nach einem SOLSIG mit COND=UNCOD oder nach einem MSG7X/TYPIO mit REPLY.

Y
Die Task kann deaktiviert werden.

N
Die Task soll nicht deaktiviert werden.

DSSR=
gibt an, ob die Task deaktiviert werden kann, wenn sie eine bestimmte Menge an Systemleistungen erhalten hat (z.B CPU-Zeit).

Y
Die Task kann deaktiviert werden.

N
Die Task soll nicht deaktiviert werden.

TPRYAD=
bezeichnet die Runpriorität der Task.

adr
Adresse eines 1 Byte langen Feldes mit dem Wert für die Runpriorität.
Für ACCESS=R wird die Runpriorität der Task aus dem TCB in das angegebene Feld übertragen. Für ACCESS=W wird die Runpriorität aus dem angegebenen Feld in den TCB übertragen.

(r)
r = Register mit dem Adresswert von adr.

TTYPAD=
bezeichnet das Taskattribut. Werte für das Taskattribut:

```
TTYPTP   EQU X'81'   Transaktionsauftrag
TTYPIACT EQU X'40'   Dialogauftrag
TTYPPB   EQU X'20'   Batch-Auftrag
```

Für ACCESS=R wird die Auftragsart in das angegebene Feld übertragen. Für ACCESS=W wird das Taskattribut aus dem angegebenen Feld in den TCB übertragen.

adr
Adresse eines 1 Byte langen Feldes mit dem Wert für das Taskattribut.

(r)
r = Register mit dem Adresswert von adr.

TGAFF=
gibt an, ob sich die aufrufende Task bei einer Affinitäts-Taskgruppe an- oder abmelden will. Dieser Operand muss zusammen mit TGIDAD angegeben werden.

Y
Die Task meldet sich zu der Taskgruppe an, deren Taskgruppen-ID in dem durch TGIDAD adressierten Feld steht, oder zu einer neu einzurichtenden Taskgruppe, wenn das Feld mit Null belegt ist. Wird eine Taskgruppe eingerichtet, wird die neue Taskgruppen-ID im durch TGIDAD adressierten Feld zurückgegeben.

N
Die Task meldet sich von der Gruppe ab, deren Taskgruppen-ID im durch TGIDAD adressierten Feld steht. Ist die abzumeldende Task die letzte ihrer Taskgruppe, wird diese Taskgruppe gelöscht und die Taskgruppen-ID im durch TGIDAD adressierten Feld auf Null gesetzt.

TGIDAD=
bezeichnet die Adresse der Taskgruppen-ID für die aufrufende Task.
Die Angabe dieses Operanden ist nur zusammen mit TGAFF zulässig.

adr
symbolische Adresse eines 4 Byte langen Feldes, das die Taskgruppen-ID enthält.

(r)
r = Register mit dem Adresswert von adr.

PROCNAD=
gibt folgende Informationen aus:
– Ist die aufrufende Task einer Affinitäts-Taskgruppe zugeordnet?
– Wenn ja, wie viele CPUs sind momentan dieser Task zugeordnet?
– Wie viele CPUs werden zurzeit vom System genutzt?

adr
symbolische Adresse eines 4 Byte langen Feldes.
Die ersten 2 Byte enthalten die Anzahl der CPUs, auf denen die Task momentan laufen darf.
Die zweiten 2 Byte enthalten die Anzahl der CPUs, die zurzeit überhaupt zur Verfügung stehen.
Ist die aufrufende Task keiner Taskgruppe zugeordnet oder ist das Subsystem TANGRAM nicht geladen, stimmen die Zahlen in beiden Teilfeldern überein.

(r)
r = Register mit dem Adresswert von adr.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29 . Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

Hinweise für Affinitäts-Taskgruppen

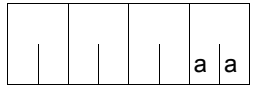
– Zum Zeitpunkt des Anmeldens einer Task an eine Taskgruppe muss bereits ein Memory Pool eingerichtet sein, sodass es möglich ist, das mit TGIDAD bezeichnete Feld in einen Bereich zu legen, der allen Tasks der Anwendung zugänglich ist. Dieses Feld muss mit Null initialisiert werden. Die erste Task richtet dann beim Anmelden die Gruppe ein, alle anderen Tasks schließen sich nur noch der bestehenden Gruppe an.

– Die Synchronisation der Anmeldevorgänge wird von BS2000 übernommen.
Die (zeitlich gesehen) zweite Anmeldung wird erst dann bearbeitet, wenn die erste Anmeldung abgeschlossen ist und die neue Gruppen-ID zurückgeschrieben wurde.

Rückinformation und Fehleranzeigen

Während der Makrobearbeitung enthält Register R1 die Adresse des Datenbereichs.

R15:



Über die Ausführung des Makros TINF wird im rechtsbündigen Byte des Registers R15 ein Returncode übergeben.

X'aa'	**Erläuterung**
X'00'	Der Aufruf wurde erfolgreich durchgeführt.
X'04'	Es wurden ungültige Operanden angegeben. Mindestens eine geforderte Funktion wurde nicht ausgeführt.
X'08'	Das mit TGIDAD bezeichnetet Feld enthält keine für die Task gültige Gruppen-ID.
X'10'	Für mindestens eine Funktion ist keine Berechtigung vorhanden.
X'0C'	Die Task, die sich zu einer Taskgruppe anmelden will, gehört bereits einer anderen Taskgruppe an.
X'14'	Die abzumeldende Task (Operand TGAFF=N) gehört keiner Gruppe an.
X'18' [1]	Es dürfen max. 65535 Taskgruppen angemeldet werden. Dieses Limit ist überschritten.
X'1C' [1]	Interner TANGRAM-Fehler. Funktion wurde unvollständig ausgeführt.
X'20' [1]	Interne TANGRAM-Administrationsfunktion (Subsystem TANGBAS) nicht verfügbar.

[1] Beim Auftreten dieser Returncodes können die Anwendungen (z.B. das Hochfahren einer Datenbank) fortgesetzt werden

Beispiel

Der Makro **TINF** überträgt Runpriorität, Taskattribut und Informationen über die Taskgruppe in je einen Bereich des Anwenderprogramms:

```
TINF     START
         PRINT NOGEN
TINF     AMODE ANY
         BALR  3,0
         USING *,3
CHKOUT1  TINF  TPRYAD=PRI,TTYPAD=TYPE,PROCNAD=INFO,                    *
               TGAFF=N,TGIDAD=AFFINI,PARMOD=31  ------------------------  (1)
CHKIN    TINF  TGAFF=Y,TGIDAD=AFFINI,PARMOD=31  ------------------------  (2)
CHKOUT2  TINF  TGAFF=N,TGIDAD=FALSE,PARMOD=31  ------------------------  (3)
CHKOUT3  TINF  TGAFF=N,TGIDAD=AFFINI,PARMOD=31  -----------------------  (4)
END      TERM
*
****  Definitions  ****
PRI      DS    L1
TYPE     DS    L1
INFO     DS    F
AFFINI   DC    X'00000000'
FALSE    DC    X'08150815'
         END
```

Ablaufprotokoll:

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,tinf), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,tinf)), -
//        test-support=*aid
%  ASS6011 ASSEMBLY TIME: 283 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 79 MSEC
//end
%  ASS6012 END OF ASSEMBH
/load-executable-program library=macexmp.lib,element-or-symbol=tinf, -
/     test-options=*aid
%  BLS0523 ELEMENT 'TINF', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'TINF', VERSION ' ' OF '<date> <time>' LOADED
/%in chkin
/%in chkout2
/%in chkout3
/%in end
/%r
STOPPED AT LABEL: CHKIN , SRC_REF: 40, SOURCE: TINF , PROC: TINF
/%d %15 %xl, pri %xl1, type %xl1, affini %xl;%r ————————————————————  (1)
*** TID: 00AF0265 *** TSN: 6WPP ********************************************
CURRENT PC: 00000024    CSECT: TINF  ***************************************
%15             = 00000014   ....
V‘0000007A‘ = PRI      + #‘00000000‘
0000007A (00000000) D2                                      K
V‘0000007B‘ = TYPE     + #‘00000000‘
0000007B (00000000) 40
V‘00000080‘ = AFFINI   + #‘00000000‘
00000080 (00000000) 00000000                                ....
STOPPED AT LABEL: CHKOUT2 , SRC_REF: 59, SOURCE: TINF , PROC: TINF
/%d %15 %xl, affini %xl;%r ——————————————————————————————————————  (2)
CURRENT PC: 00000038    CSECT: TINF  ***************************************
%15             = 00000000   ....
V‘00000080‘ = AFFINI   + #‘00000000‘
00000080 (00000000) E3C70021                                TG..
STOPPED AT LABEL: CHKOUT3 , SRC_REF: 78, SOURCE: TINF , PROC: TINF
/%d %15 %xl, false %xl;%r ————————————————————————————————————————  (3)
CURRENT PC: 0000004C    CSECT: TINF  ***************************************
%15             = 00000008   ....
V‘00000084‘ = FALSE    + #‘00000000‘
```

```
00000084 (00000000) 08150815                                    ....
STOPPED AT LABEL: END , SRC_REF: 96, SOURCE: TINF , PROC: TINF
/%d %15 %xl, affini %xl;%r ──────────────────────────────────────────  (4)
CURRENT PC: 00000060    CSECT: TINF  ************************************
%15             = 00000000   ....
V‘00000080‘ = AFFINI   + #‘00000000‘
00000080 (00000000) 00000000                                    ....
```

(1) Die Task soll von der Affinitäts-Taskgruppe mit der im Feld AFFINI stehenden Taskgruppen-ID (X'00000000') abgemeldet werden:
Register 15 enthält den Returncode X'00000014', d.h. die abzumeldende Task gehörte keiner Taskgruppe an.
Die Runpriorität der Task beträgt X'D2' = 210.
Die Task ist ein Dialogauftrag (X'40').
Der Inhalt des Feldes AFFINI bleibt unverändert.

(2) Die Task meldet sich zu einer neu einzurichtenden Affinitäts-Taskgruppe an (Feld AFFINI enthält als Input X'00000000').
Register 15 enthält den Returncode X'00000000', d.h. der **TINF** wurde ohne Fehler ausgeführt.
Die Taskgruppen-ID der neu eingerichteten Affinitäts-Taskgruppe wird im Feld AFFINI zurückgegeben. Feld AFFINI enthält jetzt X'E3C70021'.

(3) Die Task soll von der Affinitäts-Taskgruppe mit der im Feld FALSE stehenden Taskgruppen-ID (X'08150815') abgemeldet werden:
Register 15 enthält den Returncode X'00000008', d.h. im mit TGIDAD bezeichneten Feld steht keine für diese Task gültige Taskgruppen-ID.
Der Inhalt des Feldes FALSE bleibt unverändert.

(4) Die Task soll von der Affinitäts-Taskgruppe mit der im Feld AFFINI stehenden Taskgruppen-ID (X'E3C70021') abgemeldet werden:
Register 15 enthält den Returncode X'00000000', d.h. der **TINF** wurde ohne Fehler ausgeführt.
Da die Task die letzte (weil einzige) Task ihrer Taskgruppe war, wird diese Taskgruppe gelöscht und die Taskgruppen-ID im durch TGIDAD adressierten Feld auf Null gesetzt. Feld AFFINI enthält wieder X'00000000'.

TMODE – Auftragsattribute abfragen

Allgemeines

Anwendungsgebiete: Abfragen und Zugriff zu Listen und Tabellen; siehe Seite 158
Kommunikation; siehe Seite 167
Makrotyp: O-Typ; siehe Seite 28

Zur Beschreibung der Ausgabefelder siehe Makro **DTMODE**. Die symbolischen Feldnamen bei Generierung der 24-Bit-Schnittstelle beginnen mit dem Präfix TSK statt TMOD.

Makrobeschreibung

Durch den Aufruf des Makros **TMODE** erhält das Benutzerprogramm Information über den Auftrag (Job), unter dem es abläuft. Das Benutzerprogramm stellt einen Bereich zur Verfügung, in dem die Information gespeichert werden soll.
Das Aufrufformat ist von der gewünschten Schnittstelle abhängig.

Hinweis

Die Benutzerkennung, unter der das Benutzerprogramm läuft, kann auch mit dem Makro **RDUID** abgefragt werden. Die Privilegierung des Auftrags, unter dem es läuft, lässt sich auch mit dem Makro **CHKPRV** abfragen.

Makroaufrufformate und Operandenbeschreibungen

Format 1: Aufruf der 24-Bit-Schnittstelle

```
[name] TMODE

[ { bereich[,länge] } ]
  { D               }

[,PARMOD=24]
```

name
Name der DSECT, wenn der Operand D angegeben wird.
Voreinstellung: name = TSKINF0.

bereich
bereich = Name eines Bereichs, in dem die Information über den Auftrag abgespeichert wird. Der Name D ist nicht zulässig.

länge
Länge des Bereichs in Byte. Wenn dieser Operand nicht angegeben wird, wird das Längenattribut von bereich verwendet. Wenn in jedem dieser beiden Fälle die Längenangabe kleiner ist als die Länge der angebotenen Information, dann werden nur so viele Daten zur Verfügung gestellt, wie in diesen Bereich passen.
Die Länge der angebotenen Informationen steht als symbolische Konstante L@TSKINF(EQU-Wert) zur Verfügung.

D
Eine Dummy Section (DSECT) zu der Ausgabeliste wird generiert.

PARMOD=
steuert die Makroauflösung. Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

keine Angabe
Die Operandenwerte können auch in einem Datenbereich übergeben werden, deren Adresse Register R1 enthalten muss. Aufbau der Ope-randenliste:

Byte	Operand
0	muss auf X' 00' gesetzt sein; wichtig, da nur dann die ordnungsgemäße Makroausführung gewährleistet ist. Wenn nicht angegeben, wird u.U. ein falscher Returncode übergeben.
1 - 3	Adresse des Bereichs, in dem die Information über den Auftrag gespeichert werden soll.
4 - 5	Länge dieses Bereichs (hier kann die symbolische Konstante L@TSKINF angegeben werden).

Format 2: Aufruf der 31-Bit-Schnittstelle

TMODE
[PARMOD=31] [,PARLIST=adr / (r)]

PARMOD=
steuert die Makroauflösung. Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

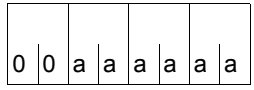
PARLIST=
bezeichnet die Adresse eines Bereichs zur Aufnahme der Auftragsdaten. Der Bereich ist auf Wortgrenze auszurichten und muss mit dem Standardheader beginnen.
Der Makro **DTMODE** erzeugt eine Beschreibung des Ein-/Ausgabebereichs als DSECT oder als Datenabschnitt; der Standardheader ist initialisiert.

adr
symbolische Adresse (Name) des Bereichs.

(r)
r = Register mit dem Adresswert adr.

Rückinformation und Fehleranzeigen

R15:



Über die Ausführung des Makros TMODE wird im Register R15 ein Returncode übergeben.

X'aaaaaa'	Erläuterung
X'000000'	Funktion ausgeführt.
X'000004'	Operandenfehler.
X'00000C'	Systemfehler.
X'01FFFF'	Falsche Angabe für UNIT/FUNCTION im Standardheader.
X'03FFFF'	Falsche Angabe für VERSION im Standardheader.

Beispiel

Mit dem Makro **TMODE** sollen einige Auftragsattribute, wie Typ der Datensichtstation, Größe des Puffers der Datensichtstation, TSN des Auftrags, Benutzerkennung usw. abgefragt werden. Die Ausführung erfolgt im 31-Bit-Adressierungsmodus, der Ausgabebereich wird mit dem Makro **DTMODE** generiert (der Standardheader ist initialisiert). Für die Ausgabe mit dem Makro **WROUT** werden die Ausgabefelder geeignet aufbereitet.

```
  TMODE    START
           PRINT NOGEN
  TMODE    AMODE ANY
           BALR  3,0
           USING *,3
           TMODE PARMOD=31,PARLIST=TMODPL ————————————————————————————  (1)
           UNPK  HFIELD(3),TMODTYPE(2)
           MVC   TASKT+2(2),HFIELD
           TR    TASKT+2(2),CODTAB
           UNPK  HFIELD(5),TMODBUFS(3)
           MVC   TASKB+2(4),HFIELD
           TR    TASKB+2(4),CODTAB
           UNPK  HFIELD(3),TMODPRI(2)
           MVC   TASKP+2(2),HFIELD
           TR    TASKP+2(2),CODTAB
           MVC   TSN,TMODTSN
           MVC   USERID,TMODUSER
           MVC   ACC,TMODACCT
           MVC   JOB,TMODNAME
           WROUT AUSB,TERM,MODE=LINE,PARMOD=31 ———————————————————————  (2)
2                *,@DCEO      999    921011   53531004
  TERM     TERM
           DS    0F
```

```
AUSB     DC    Y(AUSBE-AUSB)
         DS    CL3
         DC    X'15'
         DC    C'TERMINAL TYPE: '
TASKT    DC    C'X''00'''
         DC    X'15'
         DC    C'BUFFER TERMINAL: '
TASKB    DC    C'X''0000'''
         DC    X'15'
         DC    C'RUN PRIORITY: '
TASKP    DC    C'X''00'''
         DC    X'15'
         DC    C'TSN: '
TSN      DS    CL4
         DC    X'15'
         DC    C'USER ID: '
USERID   DS    CL8
         DC    X'15'
         DC    C'ACCOUNT NUMBER: '
ACC      DS    CL8
         DC    X'15'
         DC    C'JOB NAME: '
JOB      DS    CL8
         DC    X'15'
AUSBE    EQU   *
HFIELD   DS    CL5
CODTAB   DS    CL240
         DC    C'0123456789'
         DC    C'ABCDEF'
         DS    0F
         DTMODE DSECT=NO ------------------------------------------------  (3)
         END
```

Ablaufprotokoll:

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,tmode), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,tmode))
%  ASS6011 ASSEMBLY TIME: 336 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 86 MSEC
//end
%  ASS6012 END OF ASSTRAN
/start-executable-program library=macexmp.lib,element-or-symbol=tmode
%  BLS0523 ELEMENT 'TMODE', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'TMODE', VERSION ' ' OF '<date> <time>' LOADED
TERMINAL TYPE: X'35'  ------------------------------------------------------  (4)
BUFFER TERMINAL: X'07F8'
RUN PRIORITY: X'D2'
TSN: 2QSE
USER ID: QM212
ACCOUNT NUMBER: 89002
JOB NAME: MACTEST
```

(1) Aufruf des Makros **TMODE**. Die 31-Bit-Schnittstelle wird generiert; der Ausgabebereich wird mit dem Makro **DTMODE** erzeugt.

(2) Ausgabe der abgefragten Werte mit **WROUT** im LINE-Modus. Einige Werte mussten zuvor für die Ausgabe aufbereitet werden (entpacken und in abdruckbare Zeichen umwandeln).

(3) Der Ausgabebereich wird mit dem Makro **DTMODE** erzeugt. Der Standardheader ist initialisiert.

(4) Ausgabe im LINE-Modus:
Typ der Datenstation = X'35'
Puffergröße = 2040 Byte
Runpriorität = 210
usw.

TSPRIO – Runprioritäten ausgeben

Allgemeines

Anwendungsgebiet: Abfragen und Zugriff zu Listen und Tabellen; siehe Seite 158
Makrotyp: O-Typ; siehe Seite 28

Makrobeschreibung

Mit dem Makro **TSPRIO** werden die Unter- und Obergrenzen für feste und variable Prioritäten ausgegeben.

Makroaufrufformat

TSPRIO

```
          PRINT  GEN
          TSPRIO
EPRIFIXU  EQU     30 Obergrenze feste Priorität
EPRIFIXL  EQU    127 Untergrenze feste Priorität
EPRIVARU  EQU    128 Obergrenze variable Priorität
EPRIVARL  EQU    255 Untergrenze variable Priorität
```

TSTAT – Datenstationseigenschaften abfragen

Allgemeines

Anwendungsgebiete: Verkehr mit Datenstationen; siehe Seite 164
Abfragen und Zugriff auf Listen und Tabellen; siehe Seite 158
Makrotyp: S-Typ, MF-Format **1**:
31-Bit-Schnittstelle: Standardform/L-/E-/C-/D-Form;
siehe Seite 29

- Stand der Beschreibung: TIAM V13.2A

Makrobeschreibung

Mit dem Makro **TSTAT** kann man im Teilnehmerbetrieb Informationen über die Datenstation anfordern. Die Informationen beziehen sich auf den generierten Gerätetyp.

Makroaufrufformat und Operandenbeschreibung

TSTAT
TCHAR / PHDIM / LIDIM / **VDT**YP / EDOPT / OFLOW / STNAM / PRNAM / ALL / MONCS / PERPH / BASIC ,bereich [,länge] [,MF=C / (C,pre) / (E,..) / (D,pre) / D / L]

TCHAR
fragt nach dem physikalischen Typ der Datenstation.
Es wird der Typ geliefert, unter dem die Datenstatoin im PDN generiert wurde.

PHDIM
fragt nach den physikalischen Eigenschaften der Datenstation (Zeilenmodus).

LIDIM
fragt nach den logischen Eigenschaften der Datenstation (Zeilenmodus).

VDTYP
fragt nach dem logischen Typ der Datenstation.

EDOPT
fragt nach den statischen Edit-Options.

OFLOW
fragt nach der Art der Steuerung bei Bildschirmüberlauf.

STNAM
fragt nach dem Stationsnamen.

PRNAM
fragt nach dem Prozessornamen.

ALL
Sämtliche Informationen werden ausgegeben.

MONCS
fragt nach Informationen über den Monitor und über die Zeichensätze der Datenstation.

PERPH
fragt nach Informationen über die angeschlossene Peripherie

BASIC
fragt nach Grundinformationen über die Datenstation

bereich
symbolische Adresse eines Bereichs, in dem die angeforderte Information abgespeichert wird. Der Bereich muss auf Halbwortgrenze ausgerichtet sein.

länge
gibt die Länge des Bereichs an.
– für ALL : 64 Byte
– für MONCS : mindestens 14 Byte
– für BASIC : 640 Byte
– für andere Optionen : 8 Byte
Fehlt diese Angabe wird das Längenattribut „bereich“ verwendet. Ist die Bereichslänge bei den Operanden ALL, MONCS oder BASIC kleiner als die angebotene Terminalinformation, werden nur so viele Teilinformationen zur Verfügung gestellt, wie in den Bereich passen.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29 . Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form und D-Form des Makroaufrufs kann ein Präfix (pre = 1..3 Buchstaben), wie im Aufrufformat dargestellt, angegeben werden. Voreinstellung: pre = TST

Die Operanden TCHAR, PHDIM, LIDIM, VDTYP, EDOPT, OFLOW, STNAM, PRNAM, ALL, MONCS, PERPH und BASIC entsprechen denen, die beim Makroaufruf **DCSTA** beschrieben sind.

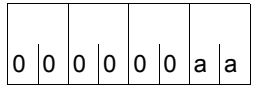
Funktionsweise

Der Bereich, der die Informationen aufnehmen soll, kann vom aufrufenden Programm selbst definiert werden, es kann ihn aber auch durch den Aufruf des Makros DCSTA C,... definieren lassen (siehe Beispiel 2).
Definiert das Programm den Empfangsbereich selbst, kann es die Adressierung mit einer DSECT durchführen, die durch den Aufruf DCSTA D,... generiert wird.

Beschreibung der übergebenen Information: siehe Makro **DCSTA**.

Rückinformation und Fehleranzeigen

R15:



Über die Ausführung des Makros TSTAT wird im Register R15 ein Returncode übergeben.

X'aa'	**Erläuterung**
X'00'	Normale Beendigung
X'04'	Nicht behebbarer Fehler
X'08'	Operandenfehler
X'0C'	Es ist keine Datenstation vorhanden.
X'10'	Länge des Empfangsfeldes zu klein. Es wurde nur ein Teil der Information geliefert, wenn der Operand ALL, MONCS oder BASIC angegeben wurde. Bei allen anderen Operanden wird nichts übertragen.
X'14'	Die gewünschte Information ist (teilweise) nicht verfügbar.

Beispiel 1

Im Beispiel wird der logische Typ der Datenstation und die Steuerungsart bei Bildschirmüberlauf abgefragt. Die durch den Makro **TSTAT** versorgten Empfangsfelder (LOG und UEL) enthalten diese Informationen in sedezimaler Darstellung. Um sie auswerten zu können, wird der Makro **DCSTA** benötigt. Mit dem Aufruf DCSTA C,... werden Speicherbereiche mit symbolischen Adressen generiert, die beim Aufruf des Makros **TSTAT** als Empfangsfeld angegeben werden können.
In der **DCSTA**-Makrobeschreibung (Seite 391) sind alle durch den Makro generierten symbolischen Namen und ihre Bedeutung aufgelistet. So kann z.B. im Feld `<PREFIX>OFLOW` durch Auswertung der Bitleiste die Art der Überlaufsteuerung gelesen werden.

```
TSTAT1   START
         PRINT NOGEN
         BALR  3,0
         USING *,3
         TSTAT VDTYP,LOG,8 ------------------------------------------  (1)
         TSTAT OFLOW,UEL,8 ------------------------------------------  (2)
DTH1     TERM
LOG      DS    CL8
UEL      DS    CL8
         END
```

Ablaufprotokoll:

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,tstat1), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,tstat1)), -
//        test-support=*aid
%  ASS6011 ASSEMBLY TIME: 245 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 78 MSEC
//end
%  ASS6012 END OF ASSEMBH
/load-executable-program library=macexmp.lib,element-or-symbol=tstat1, -
/     test-options=*aid
%  BLS0523 ELEMENT 'TSTAT1', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'TSTAT1', VERSION ' ' OF '<date> <time>' LOADED
/%in dth1<%d log %x, uel %x>
/%r
*** TID: 005000D8 *** TSN: 2QSE ******************************************
**
```

```
CURRENT PC: 00000022    CSECT: TSTAT1  ************************************
**
V'0000003E' = LOG      + #'00000000'  ------------------------------------  (3)
0000003E (00000000) 5B010000 00000000                        $.......
V'00000046' = UEL      + #'00000000'  ------------------------------------  (4)
00000046 (00000000) 02060000 00000000                        ........
```

(1) Der logische Typ der Datenstation wird abgefragt.

(2) Die Art der Steuerung bei Bildschirmüberlauf wird abgefragt.

(3) Der Prozess läuft an einer Datensichtstation (Bit 2^6 gesetzt), an der Zeilenmodus, Formatmodus und physikalischer Modus erlaubt sind (Bits 2^0, 2^1 und 2^3 gesetzt).

(4) Das System (Bit 2^5=0) steuert den Überlauf. Wenn der Bildschirm voll ist, fordert das System an der Datenstation eine Quittung an, bevor es ihn überschreibt (Bit 2^1=1).

Beispiel 2

```
  TSTAT2   START
           PRINT NOGEN
           BALR  3,0
           USING *,3
           TSTAT TCHAR,STATCHAR,8 ------------------------------------------  (1)
           TSTAT PHDIM,PHYSAREA,8 ------------------------------------------  (2)
           TSTAT LIDIM,FILIDIM,8 -------------------------------------------  (3)
  DTH1     TERM
           PRINT GEN
           DCSTA C,TYPE=TCHAR ----------------------------------------------  (4)
1 STATCHAR DS    0XL8
1 *
1 *              DEFINE TERMINAL CHARACTERISTICS FIELDS
1 *
1 STASTTCH DS    0XL8                    TERMINAL CHARACTERISTICS AREA
1 STAMNTCH DS    0XL8                    MINIMUM TERMINAL CHARICS. AREA
1 *
1 STAPTTYP DC    AL1(0)                  PARTNERTYPE
1 STADVTYP DC    AL1(0)                  DEVICE TYPE
1 STATCHR2 DC    AL1(0)                  TERMINAL CHARACTERISTICS BYTE 2
1 STATCHR3 DC    AL1(0)                  TERMINAL CHARACTERISTICS BYTE 3
1 STATCHR4 DC    AL1(0)                  TERM. CHARACTERISTIC BYTE 4 901
1 STATCHRS DC    AL1(0)                  TERM. CHAR FROM STATION     920
1 STACTRLU DC    AL1(0)                  CONTROL UNIT FOR PRINTER    701
1 STACHCAD DC    AL1(0)                  CENTRAL HARDCOPY ADDRESS
1 *
1 *              DEFINE PARTNER TYPES (PTTYP)
1 *
1 STADCAMP EQU   X'00'                 PARTNER IS A PROGRAM
1 STADCAMT EQU   X'01'                 PARTNER IS A TERMINAL
1 *
1          DCDEVCH STA
2 *
2 *              DEFINE DEVICE TYPES (DVTYP)
2 *
2 STAD8103 EQU   X'02'                 TELETYPE 8103
2 STAD8150 EQU   X'04'                 VIDEO TERMINAL 8150
2 STAD8153 EQU   X'05'     *NO VTSU*   VIDEO TERMINAL 8153
2 STADHOST EQU   X'08'                 INTELLIGENT PARTNER
2 STAD8151 EQU   X'15'                 VIDEO TERMINAL 8151
2 STAD8152 EQU   X'16'                 VIDEO TERMINAL 8152
2 STAD8110 EQU   X'17'                 SS-
8110                          00530000
2 STAD6154 EQU   X'18'     *NO VTSU*   VIDEO 8161 54 CHAR PER LINE
2 STAD6164 EQU   X'19'     *NO VTSU*   VIDEO 8161 64 CHAR PER LINE
2 STAD6180 EQU   X'1A'     *NO VTSU*   VIDEO 8161 80 CHAR PER LINE
```

```
2 STAD8161 EQU   X'1A'     *NO VTSU*   VIDEO 8161
2 STAD8121 EQU   X'1C'                 PRINTER STATION 8121
2 STADPT80 EQU   X'1D'     *AS 8103*   TELETYPE PT80
2 STAD1000 EQU   X'1E'     *AS 8103*   TELETYPE T1000
2 STADT100 EQU   X'23'     *AS 8103*   TELETYPE T100
2 STAD100E EQU   X'26'     *AS 8103*   FS100-E
2 STAD8122 EQU   X'2B'                 PRINTER STATION 8122
2 STAD8162 EQU   X'2C'                 VIDEO 8162
2 STAD8160 EQU   X'2D'                 VIDEO 8160
2 STAD8124 EQU   X'2E'                 PRINTER STATION 8124
2 STAD8167 EQU   X'2F'     *AS 8160*   VIDEO 8167
2 STADAP   EQU   X'30'     *AS HOST*   AP-STATION
2 STAD9750 EQU   X'35'                 VIDEO 9750 OR 9749
2 STAD9003 EQU   X'36'                 PRINTER STATION 9003
2 STAD9770 EQU   X'39'     *AS 8151*   DS 9770
2 STAD9002 EQU   X'3B'                 PRINTER STATION 9002
2 STAD3974 EQU   X'3D'                 VIDEO TERMINAL 3974
2 STAD9751 EQU   X'3F'     *AS 8160*   DSS 9751
2 STAD9752 EQU   X'40'     *AS 9750*   DSS 9752
2 STAD9753 EQU   X'41'     *AS 9750*   DSS 9753
2 STAD9001 EQU   X'42'                 PRINTER 9001
2 STAD9731 EQU   X'43'     *AS 3974*   GRAFIC STATION 9731
2 STAD9004 EQU   X'45'                 PRINTER 9004
2 STAD9754 EQU   X'4C'     *AS 8160*   VIDEO 9754
2 STAD9755 EQU   X'4E'                 DSS-9755
2 STAD9763 EQU   X'4F'                 DSS-9763
2 STADBTXF EQU   X'55'     *AS HOST*   BTX-STATION T-3000 (FELDVERS.)
2 STADBTXE EQU   X'56'     *AS HOST*   BTX-EDITIER-STATION (DIENST)
2 STADBTXA EQU   X'57'     *AS HOST*   BTX-ABFRAGE-STATION (DIENST)
2 STADUTC  EQU   X'5A'                 UTC FUER TELETEX
2 STAD9012 EQU   X'5B'                 PRINTER 9012
2 STAD9013 EQU   X'5C'                 PRINTER 9013
2 STAD3270 EQU   X'5E'                 DSS-3270
2 STAD0131 EQU   X'65'                 PRINTER 9001-31
2 STAD0189 EQU   X'66'                 PRINTER 9001-8931
2 STAD9022 EQU   X'68'                 PRINTER 9022
2 STAD1118 EQU   X'6B'                 PRINTER 9011-18
2 STAD1119 EQU   X'6C'                 PRINTER 9011-19
2 STAD3287 EQU   X'6E'                 PRINTER 3287
2 STADPCL  EQU   X'70'                 PRINTERS PCL
2 STAD9021 EQU   X'70'                 PRINTERS 9021 / 9022-200, HP LJ
2 STAD9014 EQU   X'72'                 PRINTER 9014
2 STAD9026 EQU   X'73'                 PRINTER 9026 (HDLC,COMP.9025)
2 STADTN08 EQU   X'74'                 Telnet without overflow 8-bit
2 STADTOV8 EQU   X'75'                 Telnet with overflow 8-bit
2 STADFE   EQU   X'78'                 FRONT-END TERMINAL (FHS-DOORS)
2 *
2 *             DEFINE TERMINAL CHARACTERISTICS BYTE 2 (TCHR2) BITS
```

```
2 *
2 STATC2EX EQU   8                       SECONDARY CHARACTER SET
2 STATC2LC EQU   32                      LOWER CASE
2 STATC2DT EQU   64                      GERM KEYB WITH GERM NAT CHAR
2 STATC2DF EQU   128                     BYTE 2 DEFINED
2 *
2 *              DEFINE TERMINAL CHARACTERISTICS BYTE 3 (TCHR3) BITS
2 *
2 STATC3H1 EQU   1                       HARDCOPY BIT 1  (LOCAL)
2 STATC3H2 EQU   2                       HARDCOPY BIT 2  (CENTRAL)
2 STATC3HC EQU   3                       HARDCOPY BITS
2 STATC3IC EQU   4                       IDENTITY CARD READER
2 STATC3FD EQU   8                       FLOPPY DISK
2 STATC3AP EQU   16                      APL CAPABILITY
2 STATC3GF EQU   32                      GRAPHICS
2 STATC3DZ EQU   64                      DEZENTRAL FORMATING
2 STATC3DF EQU   128                     BYTE 3 DEFINED
2 *
2 *              DEFINE TERMINAL CHARACTERISTICS BYTE 4 (TCHR4) BITS
2 *
2 STATC4CO EQU   1                       4 COLOURS(ITALIC/HALFBRIGHT)
2 STATC4ZF EQU   2                       NEW ZAT AND FAT POSSIBLE
2 STATC4ST EQU   4                       STATUS QUERY POSSIBLE
2 STATC4HI EQU   8                       HARDWARE INFOLINE AVAILABLE
2 STATC4C8 EQU   16                      8 COLOURS
2 STATC4HP EQU   32                      HP LASER JET II
2 STATC4DF EQU   128                     BYTE 4 DEFINED
2 *
2 *              DEFINE TERM CHAR FROM STATION   BYTE   (TCHRS) BITS
2 *
2 STATCSDT EQU   1                       GERMAN KEYBOARD
2 STATCSHC EQU   2                       LOCAL HARDCOPY PRINTER
2 STATCSIC EQU   4                       ID-CARD READER
2 STATCDOR EQU   8                       DOORS capability (reserved)
2 STATCDSK EQU   16                      DESK capability
2 STATCECC EQU   32                      ENCRYPTION capability (res)
2 STATCPER EQU   64                      Permanent ENCRYPTION reques
2 STATCSDF EQU   128                     TERM CHAR FROM STAT RECEIVED
2                *,DCDEVCH    200    960821
1 *
1                *,DCSTA      201    970513
  PHYSAREA DCSTA C,TYPE=PHDIM ------------------------------------------  (5)
1 PHYSAREA DS    0XL8
1 *
1 *              DEFINE PHYSICAL TERMINAL ATTRIBUTES FIELDS
1 *
1 STASTPV  DS    0XL8                    PHYSICAL TERMINAL ATTR. AREA
1 STAMNPV  DS    0XL8                    MINIMUM PHYS. TERM. ATTR. AREA
```

```
1 *
1 STALLEN  DC    H'0'                    PHYSICAL LINE LENGTH
1 STANOLIN DC    H'0'                    PHYSICAL NUMBER OF LINES
1 STAMAXDB DC    H'0'                    MAX. PHYSICAL DEVICE BUFFER
1          DC    2AL1(0)                 RESERVED FOR FUTURE DEVELOPMENT
1                *,DCSTA      201    970513
           DCSTA C,FI,TYPE=LIDIM ————————————————————————————————————  (6)
1 FILIDIM  DS    0XL8
1 *
1 *              DEFINE VIRTUAL TERMINAL ATTRIBUTES FIELDS
1 *
1 FISTLV   DS    0XL8                    VIRTUAL TERMINAL ATTR. AREA
1 FIMNLV   DS    0XL8                    MINIMUM VIRTUAL TERM ATTR AREA
1 *
1 FILLLEN  DC    H'0'                    VIRTUAL LINE LENGTH
1 FILNOLN  DC    H'0'                    VIRTUAL NUMBER OF LINES
1 FILMAXB  DC    H'0'                    MAXIMUM VIRTUAL DEVICE BUFFER
1          DC    2AL1(0)                 RESERVED FOR FUTURE DEVELOPMENT
1                *,DCSTA      201    970513
           END
```

(1) Der physikalische Typ der Datenstation wird abgefragt. Der Empfangsbereich, der durch den Aufruf **DCSTA C,...** generiert wird, hat standardmäßig den Namen STATCHAR.

(2) Die physikalischen Eigenschaften der Datenstation werden abgefragt. PHYSAREA ist der vom Benutzer gewählte Name des Empfangsbereichs.

(3) Die logischen Eigenschaften der Datenstation werden abgefragt. Bei dem Namen des Empfangsbereichs FILIDIM ist der Standard-Präfix „STA“ durch den Präfix „FI“ ersetzt.

(4) Der Empfangsbereich für die Abfrage des physikalischen Typs wird generiert, zusammen mit den symbolischen Konstanten zur Abfrage von Bitwerten.

(5) Der Empfangsbereich für die Abfrage der physikalischen Eigenschaften wird generiert.

(6) Der Empfangsbereich für die Abfrage der logischen Eigenschaften wird generiert. Der Präfix der Feldnamen soll „FI“ lauten (Standard „STA“).

Ablaufprotokoll

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,tstat2), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,tstat2)), -
//        test-support=*aid
%  ASS6011 ASSEMBLY TIME: 389 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 112 MSEC
//end
%  ASS6012 END OF ASSEMBH
/load-executable-program library=macexmp.lib,element-or-symbol=tstat2, -
/     test-options=*aid
%  BLS0523 ELEMENT 'TSTAT2', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'TSTAT2', VERSION ' ' OF '<date> <time>' LOADED
/%in dth1<%d statchar %xl8, physarea %xl8, filidim %xl8>
/%r
*** TID: 005000D8 *** TSN: 2QSE ******************************************
**
CURRENT PC: 00000036    CSECT: TSTAT2  ************************************
**
V'00000052' = STATCHAR + #'00000000'
00000052 (00000000) 0135A081 88000000                      ...ah... ---   (7)
V'0000005A' = PHYSAREA + #'00000000'
0000005A (00000000) 00500018 17FF0000                      .&...~..  ---  (8)
V'00000062' = FILIDIM  + #'00000000'
00000062 (00000000) 00500018 07800000                      .&......  ---  (9)
```

(7) Der Prozess läuft an einer Datensichtstation 9755.

(8) X'0050' Die physikalische Zeilenlänge beträgt 80 Zeichen.
X'0018' Die physikalische Zeilenanzahl beträgt 24 Zeilen.
X'17FF' Der physikalische Gerätepuffer fasst 6143 Zeichen.

(9) X'0050' Die logische Zeilenlänge beträgt 80 Zeichen (Zeilenmodus).
X'0018' Die logische Zeilenanzahl beträgt 24 Zeilen (Zeilenmodus).
X'0780' Der logische Zeichenpuffer fasst 1920 Zeichen
(= 24 Zeilen multipliziert mit 80 Spalten).

TYPIO – Mitteilung an Konsole ausgeben

Allgemeines

Anwendungsgebiete: Verkehr mit Datenstationen; siehe Seite 164
Meldungswesen; siehe Seite 165
Kommunikation; siehe Seite 167

Makrotyp: S-Typ, MF-Format **1**: (Standardform/E-Form/L-Form)
siehe Seite 29

Physikalische und logische Konsolen und ihre Verwendung in BS2000 sind ausführlich im Handbuch „Systembetreuung“ [10] beschrieben.

Makrobeschreibung

Mit dem Makro **TYPIO** kann eine Nachricht an der Konsole ausgegeben und eine Antwort von dort übernommen werden.

Makroaufrufformat und Operandenbeschreibung

```
TYPIO

MSG= { adr1
     { (r1)
     { (basis1,[index1],[dist1])

[,REPLY=(länge, { adr2
                { (r2)
                { (basis2,[index2],[dist2]) )]

,SHORT=NO / YES

[,UCDEST= { 'destcode'
          { adr3
          { (r3)
          { (basis3,[index3].[dist3]) ]

,MF=S / L / (E, { adr4
                { (1)
                { (r4) )
```

MSG=
beschreibt die Adresse eines Ausgabebereichs für die Ausgabenachricht (Satz variabler Länge). Nachrichtenlänge ≤ 230 Byte. Nachrichten mit einer Länge von 231 bis 251 Byte werden abgeschnitten; noch längere Nachrichten werden nicht übertragen.

adr1
symbolische Adresse (Name) des Ausgabebereiches.

(r1)
r1 = Register mit dem Wert der Adresse adr1.

(...)
Indirekte Adressierung für adr1.
basis1 = Basisregister 1; index1 = Indexregister 1; dist1 = Distanz 1

REPLY=
eine Antwort wird erwartet. Der Operand beschreibt die Adresse des Eingabebereiches für die Antwort (Satz variabler Länge).
Antwortlänge ≤ 72 Byte.

länge
Länge des Eingabebereiches (erwartete Antwortlänge + 4);
4 ≤ länge ≤ 76.
- Längere Antworten werden abgeschnitten.
- Kürzere Antworten werden linksbündig eingetragen und ein Byte mit X'00' angehängt.
- länge < 4 bzw. länge > 251 ergibt Längenfehler (RC = X'0C').
- 77 ≤ länge ≤ 251 ergibt Längenfehler (RC = X'04') und wird wie länge = 76 behandelt.

adr2
symbolische Adresse (Name) des Eingabebereiches

(r2)
r2 = Register mit dem Wert der Adresse adr2

(...)
Indirekte Adressierung für adr2.
basis2 = Basisregister 2; index2 = Indexregister 2; dist2 = Distanz 2

SHORT=
gibt an, ob die Nachricht mit ungekürztem Meldungsvorspann oder mit gekürztem Vorspann ausgegeben werden soll (nur wirksam bei Ausgabe auf physikalische Konsolen).

NO
Die Meldung wird ungekürzt ausgegeben.

YES
Der Meldungsvorspann wird in der Form `%xxxx_` ausgegeben

Dabei bedeuten:
% = Meldung ohne Antwort
xxxx = Quellenkennzeichen/Name/TSN

SHORT=YES ist nur erlaubt, wenn REPLY nicht spezifiziert wurde.

UCDEST=
UCON-Ausgabeort.

Das Ziel (Ausgabeort) der Nachricht kann wie folgt angegeben werden:
– mnemotechnischer Konsolname für eine bestimmte physikalische Konsole,
– Berechtigungsschlüssel (Routingcode) für Konsolen und berechtigte Benutzerprogramme, denen ein bestimmtes Aufgabengebiet zugeordnet ist,
– Berechtigungsname für ein berechtigtes Benutzerprogramm.

Ohne Angabe von UCDEST geht die Ausgabe an das im Systemparameter MSGDEST angegebene Ziel. Bei ungültiger UCDEST-Angabe geht die Ausgabe an die Hauptkonsole.

'destcode'
folgende Angaben sind möglich (Angaben müssen in Apostrophe eingeschlossen werden):

– destcode = (mn)
  wobei mn = 2 Zeichen langer mnemotechnischer Konsolname

– destcode = <x
  wobei x = Berechtigungsschlüssel (< muss mit angegeben werden)

– destcode = Name des berechtigten Benutzerprogramms (4 Zeichen).

adr3
symbolische Adresse (Name) eines Feldes (Wort) mit der Angabe für destcode.

(r3)
r3 = Register mit der Angabe für destcode

(...)
Indirekte Adressierung für adr3.
basis3 = Basisregister 3; index3 = Indexregister 3; dist3 = Distanz 3

Hinweis
Alle Einträge müssen linksbündig eingetragen werden.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der E-Form des Makroaufrufs wird die Adresse des Datenbereichs während der Ausführung in Register R1 geschrieben.

Hinweise zum Makroaufruf

– Bei indirekter Adressierung (Operanden MSG, REPLY, UCDEST) Kommas immer mitschreiben.

– Aufbau des Ausgabe- bzw. Eingabebereiches (Operanden MSG bzw. REPLY):

  Byte 0-1: Satzlängenfeld (Länge der Nachricht/Antwort + 4); bei REPLY wird dieses Feld durch **TYPIO** versorgt.
  Byte 2-3: reserviert
  Byte 4-n: Nachrichten-/Antworttext

```
Beispiel für einen Ausgabebereich:
nachricht  DC   Y(nend-nachricht)
           DS   CL2
           DC   C'nachrichtentext'
nend       EQU  *
```

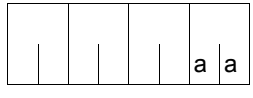
```
Beispiel für einen Eingabebereich:
antwort    DS   0CL54
slfeld     DS   CL2
           DS   CL2
atext      DS   CL50
```

– Der Makro ist in ablaufinvarianten Programmen verwendbar, wenn nur Registerangaben benutzt und L- und E-Form getrennt aufgerufen werden.

– Während der Ausführung des Makros **TYPIO** werden die Inhalte der Register R0 und R1 überschrieben: Der Inhalt von Register R0 wird mit binären Nullen überschrieben. In Register R1 wird die Anfangsadresse des Datenbereichs geschrieben (bei der E-Form des Makroaufrufs). Beide Register sollten daher nicht für die Speicherung anderer Werte genutzt werden.

Rückinformation und Fehleranzeige

R15:



Über die Ausführung des Makros TYPIO wird im rechtsbündigen Byte des Registers R15 ein Returncode übergeben; die restlichen Byte sind gelöscht.

X'aa'	**Erläuterung**
X'00'	Normale Ausführung
X'04'	TYPIO ausgeführt. Nachrichten- oder Antwortlänge abgeschnitten Nachricht: Länge 230 bis 251 angegeben Antwort: Länge 77 bis 251 angegeben
X'08'	Adressfehler: Adressierung erfolgte über die Register R0 oder R1 oder eine Adresse (adr1- adr3) liegt zumindest teilweise außerhalb des Benutzerbereichs.
X'0C'	Längenfehler: Nachrichtenlänge <= 0 oder größer 251 Byte.
X'10'	Nachrichtenausgabe derzeit nicht möglich (kann z.B. während Erzeugung oder Beendigung der aufrufenden Task auftreten).

UNBIND – Objekte entladen und entbinden

Allgemeines

Anwendungsgebiet: Binden und Laden; siehe Seite 47
Makrotyp: S-Typ, MF-Format **2**: Standardform/C-/D-/L-/E-/M-Form
siehe Seite 29

Zum dynamischen Bindelader DBL siehe auch Handbuch „BLSSERV“ [4].

Makrobeschreibung

Der Makroaufruf **UNBIND** gibt während des Programmlaufs den Speicherplatz frei, der von nicht mehr benötigten Objekten belegt ist. Das Objekt kann ein Kontext, eine Ladeeinheit, ein LLM oder ein Bindemodul (OM) sein. Die Symbole in den entladenen Objekten sind dann nicht mehr verfügbar. Der belegte Speicherplatz wird nur seitenweise (in Einheiten von 4 KB) freigegeben. Der Speicherplatz wird dem Memory Management nur zurückgegeben, falls keine anderen Module auf derselben Seite Speicherplatz beanspruchen. Ansonsten vermerkt sich der DBL die freien Bereiche und verwendet sie bei der nächsten Gelegenheit.

Wahlfrei können Programmabschnitte (CSECTs) und Einsprungstellen (ENTRYs) in dem Objekt entbunden werden, d.h. Externverweise zu diesen Symbolen behandelt dann der DBL als nicht befriedigte Externverweise. Entbinden ist nur innerhalb eines Kontextes möglich und auch nur dann, wenn beim Laden LDINFO=REF (beim Makro **BIND**) bzw. LOAD-INFORMATION=*REFERENCES (beim Kommando LOAD-PROGRAM) angegeben wurde.

Makroaufrufformat und Operandenbeschreibung

```
UNBIND

[{ CONTEXT=name      }]
 { CONTXT@=adr / (r) }
[,{ UNIT=name        }]
 { UNIT@=adr / (r)   }
[,{ MODULE=name      }]
 { MODULE@=adr / (r) }
[,{ PGMVERS=*STD / version }]
 { PGMVER@=adr / (r)       }
,UNLINK=NO / YES
,MSG=*DBLOPT / INFORMATION / WARNING / ERROR / NONE
,MF=S / C / D / E / L / M
[,PARAM=adr / (r)]
,PREFIX=P / p
[,LABEL=name]
```

CONTEXT=name
gibt den Namen des Kontextes an, der entladen wird, oder in dem Objekte entladen werden. Objekte können sein:

– eine mit UNIT festgelegte Ladeeinheit oder
– ein mit MODULE festgelegtes Modul.

„name“ kann maximal 32 Zeichen lang sein und darf nicht mit dem Zeichen „$“ oder „#“ beginnen.
Fehlt der Operand, wird als Standardwert „LOCAL#DEFAULT“ angenommen.

CONTXT@=
Angabe nur mit MF=M.
Gibt die Adresse eines Feldes an, das den Namen des Kontextes enthält, der entladen wird.

adr
Adresse des Feldes, das den Namen enthält.

(r)
r = Register mit dem Adresswert adr.

UNIT=name
gibt den Namen der Ladeeinheit an, die entladen wird, oder in das ein mit MODULE festgelegtes Modul entladen wird. name ist der Name der Ladeeinheit, der zum Zeitpunkt des Ladens im Makroaufruf **BIND** angegeben wurde. Dies kann sein:
– der mit dem Operanden UNIT@ oder UNIT vereinbarte Name oder
– der mit dem Operanden SYMBOL@ oder SYMBOL vereinbarte Name, falls UNIT@ oder UNIT nicht angegeben wurde.

name kann maximal 32 Zeichen lang sein.

UNIT@=
Angabe nur mit MF=M.
Gibt die Adresse eines Feldes an, das den Namen der Ladeeinheit enthält, die entladen wird.

adr
Adresse des Feldes, das den Namen enthält.

(r)
r = Register mit dem Adresswert adr.

MODULE=name
gibt den Namen des Moduls an, der entladen wird. Der Modul kann ein LLM oder ein OM sein. Für ein LLM muss der interne Name angegeben werden. name kann maximal 32 Zeichen lang sein.

MODULE@=
Angabe nur mit MF=M.
Gibt die Adresse eines Feldes an, das den Namen des Moduls enthält, das entladen wird.

adr
Adresse des Feldes, das den Namen enthält.

(r)
r = Register mit dem Adresswert adr.

UNLINK=
gibt an, ob in dem entladenen Objekt Programmabschnitte (CSECTs) und Einsprungstellen (ENTRYs) entbunden werden oder nicht. Externverweise von entbundenen Symbolen behandelt dann der DBL als nicht befriedigte Externverweise. Entbinden ist nur möglich, wenn zum Zeitpunkt des Ladens im Makroaufruf **BIND** der Operand LDINFO=REF angegeben wurde.
Symbole in einem Kontext, die sich auf einen anderen Kontext beziehen, können nicht entbunden werden. Entbinden ist nur im selben Kontext möglich.

NO
Symbole werden nicht entbunden.

YES
Symbole werden entbunden.

PGMVERS=
Gibt die Programmversion an, die zu entladen ist.

***STD**
bedeutet, dass beim Entladen keine Version berücksichtigt wird.

version
Die Versionsangabe darf maximal 24 Zeichen lang sein.
Wenn der DBL diese Programmversion nicht findet, dann wird das angegebene Programm nicht entladen.

PGMVER@=
Angabe nur mit MF=M.
Gibt die Adresse eines Feldes an, das die Programmversion enthält.

adr
Adresse des Feldes, das den Namen enthält.

(r)
r = Register mit dem Adresswert adr.

MSG=
Legt die niedrigste Meldungsklasse fest, ab der Meldungen ausgegeben werden.

***DBLOPT**
Der Parameterwert wird aus dem letzten Aufruf des Kommandos MODIFY-DBL-DEFAULTS übernommen. Falls für den betreffenden Parameter mit MODIFY-DBL-DEFAULTS noch kein Wert festgelegt wurde, gilt MSG=INFORMATION.

INFORMATION
Die Meldungen aller Meldungsklassen werden ausgegeben.

WARNING
Nur Meldungen der Meldungsklasse WARNING und ERROR werden ausgegeben. Nicht ausgegeben werden Meldungen der Meldungsklasse INFORMATION.

ERROR
Nur Meldungen der Meldungsklasse ERROR werden ausgegeben.

NONE
Keine Meldungen werden ausgegeben.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. angegebenen Operanden PARAM und PREFIX siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.
Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

LABEL=name
Angabe nur mit MF=M
Name der Struktur, d.h. der DSECT, die den Datenbereich des **UNBIND**-Makros beschreibt. Der Operand muss angegeben werden, wenn keine gültige USING-Anweisung für die Definition des Basisadressregisters für die DSECT des Datenbereichs angegeben ist. Der Operand LABEL muss zusammen mit dem Operanden PARAM angegeben werden. Beide Operanden werden benutzt, um eine gültige USING-Anweisung zu gewinnen.

Als Name kann angegeben werden:

1. Der Name der im Namensfeld eines vorhergehenden Makroaufrufs `name UNBIND MF=D` angegeben wurde.
2. Der Name „xPBUNDS“, wenn bisher noch kein Name name angegeben wurde. Dabei ist „x“ der Wert des Operanden PREFIX eines vorhergehenden Makroaufrufs `UNBIND MF=D,PREFIX=x`
   Der Standardwert von „x“ ist „P“.
3. Der Name der längeren DSECT, die den Datenbereich des **UNBIND**-Makros enthält, wenn zuvor der Makroaufruf `UNBIND MF=C` angegeben wurde.

Hinweise zum Makroaufruf

– Die Operanden CONTEXT, UNIT und MODULE können zusammen angegeben werden, um den Suchvorgang des DBL zu beschleunigen oder um Namenskonflikte bei gleichen Namen von Ladeeinheiten oder gleichen Modulnamen zu beheben. Standardmäßig wählt der DBL immer den ersten Namen, den er findet.

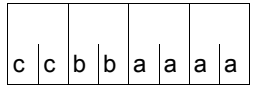
– Die verzögerte Befriedigung von Externverweisen (DELAY) und das Entbinden (UNLINK) sind auf einen Kontext beschränkt. Referenzen, die von Symbolen in verschiedenen Kontexten befriedigt wurden, können nicht entbunden werden.

– Mit **UNBIND** können nur Objekte entladen werden, die mit dem BIND-Makro oder mit den Kommandos LOAD- und START-EXECUTABLE-PROGRAM (bzw. LOAD- und START-PROGRAM) geladen wurden. Shared Code, der mit dem Makro **ASHARE** in einen Common Memory Pool geladen wurde, kann nicht mit **UNBIND**, sondern nur mit **DSHARE** entladen werden.

– Ein Modul, das einer List-Name-Unit angehört, kann nicht unabhängig von der List-Name-Unit entladen werden.

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros UNBIND wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Der Makro wurde normal ausgeführt.
X'0C'	X'01'	X'0018'	Ein reserviertes Feld des Datenbereichs ist nicht mit Nullen vorbelegt.
X'0C'	X'01'	X'0100'	Unzulässige Kombination von Operanden im Datenbereich. Für die Werte von CONTXT, UNIT und MODULE müssen Leerzeichen eingetragen werden.
X'0C'	X'01'	X'015C'	Der angegebene Kontext ist nicht vorhanden.
X'0C'	X'01'	X'0170'	Die angegebene Ladeeinheit ist nicht vorhanden.
X'0C'	X'01'	X'0174'	Das angegebene Modul ist nicht vorhanden.
X'0C'	X'01'	X'0178'	Ein Modul, das einer List-Name-Unit angehört, kann nicht unabhängig von der List-Name-Unit entladen werden.
X'0C'	X'01'	X'0198'	Unzulässiger Kontextname. Das erste Zeichen ist kein Buchstabe.
X'0C'	X'20'	X'0204'	Inkonsistenzen in den Memory Management Tabellen des DBL (Systemfehler).
X'0C'	X'20'	X'0208'	Inkonsistenzen in den Tabellen des DBL (Systemfehler).
X'0C'	X'20'	X'0300'	Fehler bei der Verarbeitung von RETMEM (Systemfehler).
X'00'	X'01'	X'FFFF'	Die Funktion wird nicht mehr oder noch nicht unterstützt.
X'00'	X'03'	X'FFFF'	Die Version der Schnittstelle wird nicht unterstützt.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Beispiel

Während des Programmlaufs von UNBIND1 wird mit Hilfe des Makros **BIND** ein zweiter Programmabschnitt BIND3 nachgeladen. BIND3 steht als Bindemodul in der Bibliothek MACEXMP.LIB. Beide Programmabschnitte sollen im 31-Bit-Adressierungsmodus ablaufen. UNBIND1 soll unterhalb und BIND3 oberhalb der 16MB-Grenze geladen werden. Nach Aufruf des Makros **BIND** soll zuerst BIND3 ablaufen. Nach dem Ablauf von BIND3 soll in UNBIND1 zurückverzweigt und das Modul BIND3 mit Hilfe des Makros **UNBIND** entladen werden.

```
UNBIND1  START
UNBIND1  AMODE 31 ------------------------------------------------------------ (1)
UNBIND1  RMODE 24
         BALR  3,0
         USING *,3
         USING BINDDS,6 --------------------------------------------------------- (2)
         USING UNBDS,7 ---------------------------------------------------------- (3)
         ST    3,AREA11
         UNPK  AREAH,AREA1
         MVC   AREAA(8),AREAH
WROUT1   WROUT OUT,ERROR,PARMOD=31 ------------------------------------------- (4)
BACK     LA    12,UNBIND
BIND     BIND  MF=E,PARAM=BINDPAR ----------------------------------------------- (5)
         LA    6,BINDPAR
         CLC   XBINRET,=X'00000000' --------------------------------------------- (6)
         BE    UNBIND
         MVC   OUT+5(28),='BIND ERROR!                '
         WROUT OUT,ERROR,PARMOD=31 ---------------------------------------------- (7)
         B     ERROR
UNBIND   UNBIND MF=E,PARAM=UNBPAR --------------------------------------------- (8)
         LA    7,UNBPAR
         CLC   YUNBRET,=X'00000000' --------------------------------------------- (9)
         BE    MVC
         MVC   OUT+5(28),='UNBIND ERROR!              '
         WROUT OUT,ERROR,PARMOD=31 ---------------------------------------------- (10)
         B     ERROR
MVC      MVC   OUT+5(28),='UNBIND PROCESSED           '
         WROUT OUT,ERROR,PARMOD=31 ---------------------------------------------- (11)
         MVC   OUT+5(28),='RETURN TO UNBIND1          '
         WROUT OUT,ERROR,PARMOD=31
ERROR    TERM
*****************
OUT      DC    Y(OUTE-OUT)
         DS    CL3
         DC    C'UNBIND1: BASE REG.= '
AREAA    DS    CL8
OUTE     EQU   *
AREA     DS    0F
```

```
AREA1    DS    0CL5
AREA11   DS    CL4
AREA12   DC    C'0'
         DS    0F
AREAH    DS    CL9
BINDPAR BIND MF=L,SYMBOL=BIND3,SYMBLAD=BIND3@,BRANCH=YES,PROGMOD=ANY,* - (5)
               LIBLINK=PLAMLIB
UNBPAR   UNBIND MF=L,MODULE=BIND3 ------------------------------------- (8)
BIND3@   DS    A
BINDDS   BIND  MF=D,PREFIX=X ------------------------------------------ (12)
UNBDS    UNBIND MF=D,PREFIX=Y ------------------------------------------ (13)
         END


BIND3    CSECT ---------------------------------------------------------- (14)
         PRINT NOGEN
BIND3    AMODE ANY ------------------------------------------------------ (15)
BIND3    RMODE ANY
         BALR  4,0
         USING *,4
         ST    4,AREA11
         UNPK  AREAH,AREA1
         MVC   AREAA(8),AREAH
         WROUT OUT,ERROR,PARMOD=31 -------------------------------------- (16)
         BR    12
ERROR   TERM
*****************
OUT      DC    Y(OUTE-OUT)
         DS    CL3
         DC    C'BIND3:   BASE REG.= '
AREAA    DS    CL8
OUTE     EQU   *
AREA     DS    0F
AREA1    DS    0CL5
AREA11   DS    CL4
AREA12   DC    C'0'
AREAH    DS    CL9
         END
```

(1) Für den Programmabschnitt UNBIND1 wird das Attribut AMODE=31 vereinbart. Mit RMODE=24 wird UNBIND1 immer unterhalb der 16MB-Grenze geladen.

(2) Register 6 wird dem Assembler als Basisadressregister zur Adressierung der DSECT für die Operandenliste des **BIND**-Makros zugewiesen, die an der symbolischen Adresse BINDDS durch einen **BIND**-Aufruf mit MF=D erzeugt wird.

(3) Register 7 wird dem Assembler als Basisadressregister zur Adressierung der DSECT für die Operandenliste des **UNBIND**-Makros zugewiesen, die an der symbolischen Adresse UNBDS durch einen **UNBIND**-Aufruf mit MF=D erzeugt wird.

(4) Der Inhalt des Basisregisters von UNBIND1 wird zur Darstellung des Adressierungsmodus und der Ladeadresse ausgegeben.

(5) An der symbolischen Adresse BIND wird der Makro **BIND** in seiner E-Form aufgerufen. An dieser Stelle im Programm wird daher nur der Befehlsteil erzeugt. Die zugehörige Operandenliste wird an der symbolischen Adresse BINDPAR durch einen **BIND**-Aufruf mit MF=L angelegt. Die dort angegebenen Operandenwerte veranlassen den **BIND**-Makro, bei der Programmausführung

- die CSECT BIND3 (SYMBOL=BIND3) aus der mit dem Linknamen PLAMLIB zugewiesenen Bibliothek (LIBLINK=PLAMLIB) nachzuladen,
- die Startadresse von BIND3 im Feld BIND3@ zu hinterlegen (SYMBLAD=BIND3@),
- für BIND3 den 31-Bit-Adressierungsmodus einzustellen (PROGMOD=ANY),
- nach dem Laden von BIND3 den Programmlauf in BIND3 fortzusetzen (BRANCH=YES).

(6) Nach der Ausführung des **BIND**-Makros wird geprüft, ob das Feld XBINRET des Standardheaders den Returncode X'00000000' enthält, der eine fehlerfreie Makroausführung anzeigt. Der Name XBINRET stammt aus der DSECT, die unter der symbolischen Adresse BINDDS durch einen **BIND**-Aufruf mit MF=D und PREFIX=X erzeugt wurde (siehe 12.). Diese DSECT beschreibt den Aufbau der Operandenliste des **BIND**-Makros. Die symbolischen Namen der DSECT können zur Adressierung innerhalb der Operandenliste verwendet werden, nachdem das zugeordnete Basisadressregister (hier Register 6) mit der Anfangsadresse der Operandenliste (hier BINDPAR) geladen worden ist.

(7) Wenn der **BIND**-Makro nicht fehlerfrei ausgeführt wurde, wird eine Fehlermeldung über SYSOUT ausgegeben und der Programmlauf von UNBIND1 wird beendet.

(8) An der symbolischen Adresse UNBIND wird der Makro **UNBIND** in seiner E-Form aufgerufen. An dieser Stelle im Programm wird daher nur der Befehlsteil erzeugt. Die zugehörige Operandenliste wird an der symbolischen Adresse UNBPAR durch einen **UNBIND**-Aufruf mit MF=L angelegt. Der dort angegebene Operand MODULE=BIND3 veranlasst den **UNBIND**-Makro, das Modul BIND3 zu entladen.

(9) Nach der Ausführung des **UNBIND**-Makros wird geprüft, ob das Feld YUNBRET des Standardheaders den Returncode X'00000000' enthält, der eine fehlerfreie Makroausführung anzeigt. Der Name YUNBRET stammt aus der DSECT, die unter der symbolischen Adresse UNBDS durch einen UNBIND-Aufruf mit MF=D und PREFIX=Y erzeugt wurde (siehe (13)). Diese DSECT beschreibt den Aufbau der Operandenliste des **UNBIND**-Makros. Die symbolischen Namen der DSECT können zur Adressierung innerhalb der Operandenliste verwendet werden, nachdem das zugeordnete Basisadressregister (hier Register 7) mit der Anfangsadresse der Operandenliste (hier UNBPAR) geladen worden ist.

(10) Wenn der **UNBIND**-Makro nicht fehlerfrei ausgeführt wurde, wird eine Fehlermeldung über SYSOUT ausgegeben und der Programmlauf von UNBIND1 wird beendet.

(11) Meldungen nach SYSOUT informieren darüber, dass die Programmausführung in UNBIND1 fortgesetzt und das Modul BIND3 entladen wurde.

(12) Der Makroaufruf **BIND** mit MF=D erzeugt eine DSECT, die den Aufbau der Operandenliste des **BIND**-Makros beschreibt. Der Operand PREFIX=X bewirkt, dass alle symbolischen Namen in dieser DSECT (Feldnamen und Equates) mit dem Buchstaben X beginnen.

(13) Der Makroaufruf **UNBIND** mit MF=D erzeugt eine DSECT, die den Aufbau der Operandenliste des **UNBIND**-Makros beschreibt. Der Operand PREFIX=Y bewirkt, dass alle symbolischen Namen in dieser DSECT (Feldnamen und Equates) mit dem Buchstaben Y beginnen.

(14) Die CSECT-Anweisung definiert den Programmabschnitt BIND3.

(15) AMODE=ANY zeigt dem Betriebssystem an, dass BIND3 sowohl im 24- als auch im 31-Bit-Adressierungsmodus ablaufen kann.

(16) Der Inhalt des Basisregisters von BIND3 wird zur Darstellung des Adressierungsmodus und der Ladeadresse ausgegeben.

Ablaufprotokoll

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,unbind1), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,unbind1))
%  ASS6011 ASSEMBLY TIME: 585 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 200 MSEC
```

```
//compile source=*library-element(macexmp.lib,bind3), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,bind3))
%  ASS6011 ASSEMBLY TIME: 169 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 83 MSEC
//end
%  ASS6012 END OF ASSEMBH
/add-file-link link-name=plamlib,file-name=macexmp.lib ——————————— (17)
/start-executable-program library=macexmp.lib,element-or-symbol=unbind1  (18)
%  BLS0523 ELEMENT 'UNBIND1', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'UNBIND1', VERSION ' ' OF '<date> <time>' LOADED
UNBIND1: BASE REG.= 80000002 ——————————————————————————————————— (19)
BIND3  : BASE REG.= 81000002 ——————————————————————————————————— (20)
UNBIND PROCESSED
RETURN TO UNBIND1 ————————————————————————————————————————————— (21)
```

(17) Der im BIND-Aufruf (05) verwendete Dateikettungsname wird zugewiesen.

(18) Der DBL wird aufgerufen, um das Programm zu binden, zu laden und zu starten.

(19) Der Inhalt des Basisregisters von UNBIND1 wird ausgegeben. 31-Bit-Adressierung ist eingestellt (Bit 2^{31} = 1); die Ladeadresse liegt unterhalb der 16MB-Grenze.

(20) Der DBL hat die CSECT BIND3 nachgeladen. Der Inhalt des Basisregisters von BIND3 wird ausgegeben. 31-Bit-Adressierung ist eingestellt (Bit 2^{31} = 1); die Ladeadresse liegt oberhalb der 16MB-Grenze.

(21) Nach der Rückkehr aus BIND3 wird der Programmlauf in UNBIND1 fortgesetzt und das Modul BIND3 entladen.

VMGINF – Informationen über den VM2000-Betrieb ausgeben

Allgemeines

Anwendungsgebiet: Abfragen und Zugriff zu Listen und Tabellen; siehe Seite 158
Makrotyp: S-Typ, MF-Format **3**: D-/C-/E-/L-Form; siehe Seite 29

Das virtuelle Maschinensystem VM2000 ermöglicht den gleichzeitigen Betrieb unterschiedlicher, voneinander völlig abgeschotteter Systemumgebungen auf *einem* Server, mit einer dem „Native-Betrieb“ vergleichbaren Performance. VM2000 erhöht somit die Einsatzvielfalt und die Auslastung eines Servers. Für weitere Informationen siehe Handbuch „VM2000“ [17].

Makrobeschreibung

Der Makro **VMGINF** gibt dem Aufrufer Auskunft darüber, ob BS2000 unter VM2000 läuft (Feld `<PREFIX><MACID>VIND` des Datenbereichs). Ist das der Fall, erhält der Aufrufer folgende Informationen über das VM2000-System, auf dem BS2000 aktuell läuft bzw. auf dem der IPL für BS2000 ausgeführt wurde:

– Den VM-Index und den VM-Namen der virtuellen Maschine.
– Die Auskunft, ob das System ein Monitorsystem ist.
– Die SYSID (VM-CONFIGURATION-ID) des Monitorsystems.
– Bestimmte VM-Privilegien und Zustände des Gastsystems
– Die Version von VM2000
– Den BCAM-Namen des Monitorsystems
– Die Version des Monitorsystems
– Xen-Domänenkennung (x86-Server)

Makroaufrufformat und Operandenbeschreibung

VMGINF
SRVUNIT = *STD / *INITIAL / *CURRENT ,MF=D / C / E / L [,PARAM=adr / (r)] ,PREFIX=V / p ,MACID=MGI / macid

SRVUNIT=
Spezifiziert die Server Unit, deren Daten ausgegeben werden sollen.

***STD**
derzeit gültige Standard-Einstellung der Server Unit.

***INITIAL**
Server-Unit, auf der der IPL für BS2000 ausgeführt wurde.

***CURRENT**
Server Unit, auf der BS2000 aktuell läuft.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29 . Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form oder D-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

Nach Aufruf des Makros muss zuerst abgefragt werden, ob das System unter VM2000 läuft, bevor weitere Ausgabeinformationen ausgewertet werden.

Layout des Datenbereichs

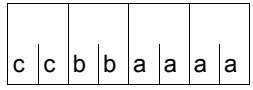
```
           VMGINF MF=D
1          MFTST MF=D,PREFIX=V,MACID=MGI,ALIGN=F,                          C
1                DMACID=MGI,SUPPORT=(E,D,C,M,L)
2 VMGI     DSECT ,
2                *,##### PREFIX=V, MACID=MGI #####
1 VMGIUNIT    EQU   137                    unit number
1 *
1 VMGIFCO4    EQU   4                      function number
1 *
1 VMGIVRO2    EQU   2                      version number
1 *
1 *   parameterarea description
1 VMGIPA   DS    0F                  begin of parameterarea    _INOUT
1 VMGIHDR  FHDR  MF=(C,VMGI),EQUATES=NO                    Standardheader
2 VMGIHDR  DS    0A
2 VMGIFHE  DS    0XL8           0    GENERAL PARAMETER AREA HEADER
2 *
2 VMGIIFID DS    0A             0    INTERFACE IDENTIFIER
2 VMGIFCTU DS    AL2            0    FUNCTION UNIT NUMBER
2 *                                  BIT 15    HEADER FLAG BIT,
2 *                                  MUST BE RESET UNTIL FURTHER NOTICE
2 *                                  BIT 14-12 UNUSED, MUST BE RESET
2 *                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
2 VMGIFCT  DS    AL1            2    FUNCTION NUMBER
2 VMGIFCTV DS    AL1            3    FUNCTION INTERFACE VERSION NUMBER
2 *
2 VMGIRET  DS    0A             4    GENERAL RETURN CODE
2 VMGISRET DS    0AL2           4    SUB RETURN CODE
2 VMGISR2  DS    AL1            4    SUB RETURN CODE 2
2 VMGISR1  DS    AL1            5    SUB RETURN CODE 1
2 VMGIMRET DS    0AL2           6    MAIN RETURN CODE
2 VMGIMR2  DS    AL1            6    MAIN RETURN CODE 2
2 VMGIMR1  DS    AL1            7    MAIN RETURN CODE 1
2 VMGIFHL  EQU   8              8    GENERAL OPERAND LIST HEADER LENGTH
2 *
1 *   main return codes
1 VMGIMSCC    EQU   0                      function executed
1 VMGIMPAR    EQU   1                      parameter error
1 VMGIMINT    EQU   2                      internal error
1 VMGIMTIM    EQU   7                      timeout error
1 VMGIMANA    EQU   64                     VM2000 agent not available
1 VMGIMXNA    EQU   65                     xend not available
1 *
```

```
1 VMGIVIND    DS    FL1                 VM2000 indicator
1 *   VM2000 indicator set
1 VMGIVM2R    EQU   232                 VM2000 running
1 VMGIVM2N    EQU   213                 VM2000 not running
1 *
1 VMGISIND    DS    AL1                 status indicator
1 VMGIV2MO    EQU   X'80'               monitor system
1 VMGILTSY    EQU   X'40'               local time syn via adjust
1 *                                     time
1 VMGIMPAD    EQU   X'20'               MP grade adjustment possible
1 VMGIPIOM    EQU   X'10'               VM-PRIV: DYNAMIC-IOREC
1 VMGIPGIO    EQU   X'08'               VM-PRIV: VMGLOB-IOREC
1 VMGIUNU0    EQU   X'04'               unused
1 VMGIGS2N    EQU   X'02'               GS-UNIT2 not in use
1 VMGIGS1N    EQU   X'01'               GS-UNIT1 not in use
1 VMGIVCID    DS    X                   VM configuration ID
1 *
1 VMGIVM_IDENT DS    0XL9                 VM identification
1 VMGIVMIX    DS    X                   VM index
1 VMGIVMNM    DS    CL8                 VM name
1 *
1 VMGIV1IN    DS    AL1                 valid indicator 1
1 VMGIV1O1    EQU   X'80'               :S: _OUT_11 valid
1 VMGIV1O2    EQU   X'40'               :S: _OUT_12 valid
1 VMGIV1O3    EQU   X'20'               :S: _OUT_13 valid
1 VMGIV1O4    EQU   X'10'               :S: _OUT_14 valid
1 VMGIUNU1    EQU   X'0F'               unused
1 VMGIV2IN    DS    AL1                 valid indicator 2
1 VMGIUNU3    EQU   X'FF'               unused
1 VMGIVVRS    DS    CL6                 VM2000 version (Vxx.xx)
1 *                                     _OUT_11
1 VMGISIN2    DS    AL1                 status indicator 2
1 VMGIPIDA    EQU   X'80'               VM-PRIV: IMPL-DEV-ASSIGN
1 *                                     _OUT_12
1 VMGIPIOP    EQU   X'40'               VM-PRIV: IO-PRIORITY _OUT_12
1 VMGIPIOR    EQU   X'20'               VM-PRIV: IO-RESET _OUT_12
1 VMGICHMF    EQU   X'10'               CHN-MON-FCL ACTIVE FOR VM
1 *                                     _OUT_12
1 VMGIMBCA    EQU   X'08'               BCAM ACTIVE IN MONITOR
1 *                                     _OUT_12
1 VMGIPASA    EQU   X'04'               VM-PRIV: AUTO-SNAP-ASSIGN
1 *                                     _OUT_12
1 VMGIRSCS    EQU   X'02'               RSC SUPPORTED _OUT_14
1 VMGIUNU2    EQU   X'01'               unused
1 VMGIMBCN    DS    CL8                 monitor BCAM name _OUT_12
1 VMGIMOVS    DS    CL10                monitor OSD version _OUT_12
1 VMGIDOID    DS    XL8                 Xen domid _OUT_13
1 VMGIRES1    DS    XL43                reserverd
```

```
1 VMGISUI     DS    FL1                   server unit indicator
1 *   Server unit indicator set
1 VMGISUIS    EQU   0                     standard
1 VMGISUII    EQU   1                     initial
1 VMGISUIC    EQU   2                     current
1 *
1 VMGICID     DS    FL1                   caller identifier (internal
1 *                                       use)
1 *   Caller identifier set
1 VMGICIDS    EQU   0                     system
1 VMGICIDU    EQU   1                     user
1 *
1 VMGI#       EQU   *-VMGIHDR
```

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros VMGINF wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'01'	X'0001'	Parameterfehler
X'00'	X'20'	X'0002'	Interner Fehler
X'0C'	X'40'	X'0007'	Zeitüberschreitung
X'0C'	X'82'	X'0040'	VM2000-Agent nicht verfügbar
X'0C'	X'82'	X'0041'	XEND nicht verfügbar

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Die Namen der Ausgabefelder sind dem Datenbereich zu entnehmen.

VPASS – Warten

Allgemeines

Anwendungsgebiet: Starten, Unterbrechen und Beenden; siehe Seite 72
Makrotyp: R-Typ; siehe Seite 28

Makrobeschreibung

Der Makro **VPASS** setzt die Anwendertask für eine bestimmte Zeit in den Wartezustand. Die Wartezeit kann relativ zur CPU-Geschwindigkeit angegeben werden.

Makroaufrufformat und Operandenbeschreibung

VPASS
anzahl / (1) ,MSEC=N / Y ,CPUDEP=N / Y

anzahl
anzahl = Anzahl der Sekunden oder Millisekunden, während denen die Task im Wartezustand verbleibt. Angabe als Dezimal- oder Sedezimalzahl (X'....').

(1)
gibt an, dass Register R1 mit der gewünschten Anzahl von Sekunden oder Millisekunden geladen ist.

Wertebereiche:
- Angabe in Sekunden:
  - direkte Angabe: 0 ≤ anzahl ≤ 4095 (=X'0FFF')
  - Register R1: 0 ≤ anzahl ≤ 21599
- Angabe in Millisekunden: 1 ≤ anzahl ≤ 999 (=X'3E7')

MSEC=
beschreibt die Zeiteinheit für anzahl.

N
Angabe in Sekunden (sec).

Y
Angabe in Millisekunden (ms).

CPUDEP=
gibt an, ob die Zeitangabe entsprechend der CPU-Geschwindigkeit relativiert werden soll.

N
Die Zeitangabe wird nicht relativiert.

Y
Bei 1 MOPS-Servern (MOPS = Million Operations per Second) wird die Zeitangabe nicht verändert.
Bei schnelleren Servern wird die angegebene Wartezeit relativ zur CPU-Geschwindigkeit verringert; bei langsameren Servern entsprechend erhöht.

Die Relativierung der Zeitangabe ist nur bei kleinen Wartezeiten sinnvoll, da nur dann die Zahl der durchlaufenen Befehle hinreichend genau abgeschätzt werden kann.

Hinweise zum Makroaufruf

- Zeitangaben ≤ 500 ms führen nicht zur Deaktivierung der Task. Die Task verbleibt im aktiven Wartezustand.
- Die Angabe anzahl = 0 sec entspricht (aus Kompatibilitätsgründen) einer Wartezeit von 500 ms.

Beispiel siehe Abschnitt „Contingency-Prozesse“ auf Seite 111.

VSVI1 – Binde- und Ladeinformation ausgeben

Allgemeines

Anwendungsgebiet: Binden und Laden; siehe Seite 47
Makrotyp: S-Typ, MF-Format **2**: Standardform/C-/D-/L-/E-/M-Form
siehe Seite 29

Zum dynamischen Bindelader DBL siehe auch Handbuch „BLSSERV“ [4].

Makrobeschreibung

Der Makroaufruf **VSVI1** informiert den Benutzer über Einträge in den Tabellen des DBL. Der DBL greift dabei auf folgende Kontexte zu:
– Benutzerkontexte und/oder Systemkontexte
Systemkontexte werden nur unter TSOS ausgegeben.
– Kontexte von Common Memory Pools zu, in denen Shared Code abgelegt ist und an die der Benutzer angeschlossen ist.

Folgende Informationen können abgefragt werden:

– Eine Liste mit Namen der Kontexte (SELECT=CTXLIST)
– Der Umfang des in einem Kontext geladenen Codes und der Umfang der dazugehörigen Binde- und Ladeinformationen (SELECT=CTXSIZE)
– Eine Liste mit Namen, Ladeadressen, Längen, Typen, Attributen und Kontexten von CSECTs, ENTRYs und COMMON-Bereichen (SELECT=ALLLIST)
– Eine Liste mit Namen, Ladeadressen, Längen, Typen, Attributen und Kontexten aller CSECTs und COMMON-Bereiche (SELECT=MODLIST)
– Ein Satz mit Name, Ladeadresse, Länge, Typ, Attribut, Kontext, Version und HSI-Code eines *einzelnen* Programmabschnitts (CSECT), ENTRYs oder COMMON-Bereichs (SELECT=BYNAME)
– Ein Satz mit Name, Ladeadresse, Länge, Typ, Attributen und Kontext eines durch eine Adresse angegebenen Programmabschnitts (CSECT) oder COMMON-Bereichs (SELECT=BYADDR)
– Eine Liste der ILEs, die zu einem oder mehreren Kontexten gehören

Die Einzelinformationen (Name, Ladeadresse, Länge, Attribut, Typ, Kontext und HSI-Code) können unabhängig voneinander ausgewählt werden. Es ist auch möglich, nur die Länge der gewünschten Ausgabeinformation anzufordern.
Die Beschreibung der Ausgabeinformation erfolgt im Anschluss an die Operandenbeschreibung (siehe Seite 1011).

Makroaufrufformat und Operandenbeschreibung

VSVI1

SELECT=CTXLIST / CTXSIZE / ALLLIST / MODLIST / BYNAME / BYADDR / ILELIST

,OUTADDR=adr / (r) / label

,OUTLEN=integer

,ADDRESS=**Y**ES / **N**O

,CONTEXT=**Y**ES / **N**O

,CTXPRIV=ANY / YES / NO

,CTXSEL=ALL / LOCAL / GLOBAL / POOL / SSLOCAL

,HSI=**N**O / **Y**ES

[, { INNAME=name | INNAME@=adr / (r) | INADDR=adr / (r) / label }]

[, { INCTX=name | INCTX@=adr / (r) }]

,INSTRUCT=adr

,INTVERS=BLSP2 / SRV001 / SRV002 / SRV003

,LEN=**Y**ES / **N**O

,NAME=**Y**ES / **N**O

,RUNMOD=**S**TD / **A**DV

,SCOPE=ALL / USER_GROUP / GLOBAL / GROUP

,SIZONLY=**N**O / **Y**ES

,SYMTYP = **ANY** / CSECT / ISL / NOTISL

,TYPE=**Y**ES / **N**O

,VERSION=**N**O / **Y**ES

,MF=S / C / D / E / L / M

[,PARAM=adr / (r)]

,PREFIX=P / p

[,LABEL=name]

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

ADDRESS=
legt fest, ob in den ausgegebenen Informationen die Ladeadressen enthalten sind.

YES
Die Ladeadressen werden ausgegeben.

NO
Die Ladeadressen werden nicht ausgegeben.

CONTEXT=
legt fest, ob in den ausgegebenen Informationen die Namen der Kontexte enthalten sind.

YES
Die Namen werden ausgegeben.

NO
Die Namen werden nicht ausgegeben.

CTXPRIV=
legt fest, für welche Zugriffsberechtigung Kontexte durchsucht werden (nur zusammen mit RUNMOD=ADV).

– Für nichtprivilegierte Benutzer (nicht TSOS) wird die Angabe ignoriert und intern auf CTXPRIV=NO gesetzt.

– Für privilegierte Benutzer (TSOS) gilt:
Falls RUNMODE=ADV ist und der Parameter CTXPRIV nicht angegeben wurde, setzt der DBL CTXPRIV=ANY.
Bei RUNMODE=STD ignoriert DBL den Eingabewert für CTXPRIV und setzt CTXPRIV=YES.

ANY
Sowohl privilegierte wie nichtprivilegierte Kontexte werden durchsucht. Für nichtprivilegierte Benutzer werden nur nichtprivilegierte Kontexte durchsucht.

YES
Nur privilegierte Kontexte werden durchsucht. Diese Angabe ist nur für privilegierte Benutzer erlaubt.

NO
Nur nicht privilegierte Kontexte werden durchsucht.

CTXSEL=
legt fest, für welchen Geltungsbereich Kontexte durchsucht werden (nur zusammen mit RUNMOD=ADV).

– Für nichtprivilegierte Benutzer (nicht TSOS) ist nur die Angabe CTXSEL=LOCAL oder CTXSEL=POOL erlaubt. Alle anderen Angaben werden wie CTXSEL=LOCAL behandelt.

– Für privilegierte Benutzer (TSOS) gilt:
  Falls RUNMODE=ADV ist und der Parameter CTXSEL nicht angegeben wurde, setzt der DBL CTXSEL=ALL.
  Bei RUNMODE=STD ignoriert DBL den Eingabewert für CTXSEL und setzt CTXSEL=GLOBAL.

  ALL
  Kontexte mit dem Geltungsbereich SYSTEM und USER werden durchsucht. Kontexte mit dem Geltungsbereich POOL und SSLOCAL werden nicht berücksichtigt.

  GLOBAL
  Nur Kontexte mit dem Geltungsbereich SYSTEM werden durchsucht.

  LOCAL
  Kontexte mit dem Geltungsbereich USER werden durchsucht.
  Außerdem wird bei RUNMODE=ADV der vorgeladene Teil des Kontextes von Subsystemen durchsucht, falls folgende Bedingungen zutreffen:
  – Die Task, in der der Makro VSVI1 aufgerufen wird, ist mit dem Subsystem verbunden
  – Das Subsystem besitzt das Attribut MEMORY-CLASS=*BY-SLICE

  POOL
  Kontexte von Memory Pools, in die mit dem Makro **ASHARE** Shared Code geladen wurde, werden durchsucht. Die Menge der Memory-Pool-Kontexte kann durch Angabe des Geltungsbereiches der Memory Pools eingeschränkt werden (Operand SCOPE).

  SSLOCAL
  Nur Kontexte lokaler Subsysteme werden durchsucht.

HSI=
legt fest, ob in der Ausgabe die Informationen über die Hardware-Software-Schnittstelle enthalten sind (nur zusammen mit RUNMOD=ADV).

  NO
  Der HSI-Code wird nicht ausgegeben.

  YES
  Der HSI-Code und die HSI-Compiler-Information werden ausgegeben. Der Operandenwert ist nur für die Ausgabe von Symbolinformationen relevant und er darf nur zusammen mit RUNMOD=ADV angegeben werden.

INNAME=name
gibt den Namen eines Programmabschnitts (CSECT), ENTRY oder COMMON-Bereichs an, dessen Name, Ladeadresse, Länge und Attribute ausgegeben werden.
name darf maximal 32 Zeichen lang sein.
Der Operand INNAME muss zusammen mit dem Operanden SELECT=BYNAME angegeben werden.

INNAME@=
Angabe nur mit MF=M.
Gibt die Adresse eines Feldes an, das den Namen eines Programmabschnitts (CSECT), ENTRY oder COMMON-Bereichs enthält.

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.

(r)
r = Register mit der gesuchten Feldadresse.

INADDR=
gibt eine Adresse an, für die die entsprechenden Informationen (Name, Ladeadresse, Länge und Attribute) ausgegeben werden.
Der Operand INADDR muss zusammen mit dem Operanden SELECT=BYADDR angegeben werden.

adr
Adresse eines Feldes, das die gesuchte Programmadresse enthält.
Angabe nur mit MF=M.

(r)
r = Register mit der gesuchten Programmadresse. Angabe nur mit MF=M.

label
Programmadresse. Sie kann als symbolische Adresse oder als Konstante (X'...') angegeben werden. Angabe nur mit MF=S oder MF=L.

INCTX=name
legt einen Kontext fest, der durchsucht wird. name darf maximal 32 Zeichen lang sein. Fehlt der Operand, werden die Kontexte entsprechend den Angaben der Operanden CTXSEL und CTXPRIV durchsucht, bei Angabe von CTXSEL=POOL in Abhängigkeit vom Operanden SCOPE. Der Operand INCTX wird ignoriert, wenn gleichzeitig SELECT=CTXLIST angegeben wurde.

INCTX@=
Angabe nur mit MF=M.
Gibt die Adresse eines Feldes an, das den Namen des Kontextes enthält, der durchsucht wird.

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.

(r)
r = Register mit der gesuchten Feldadresse.

INSTRUCT=adr
Dieser Operand ist bei Angabe von INTVERS=SRVxxx und xxx ≥ 001 verfügbar. Er muss zusammen mit dem Operanden SELECT=BYNAME angegeben werden.
Symbolische Adresse eines Bereichs, der einen EEN-Namen (Extended External Name) enthält. Der Bereich besteht aus zwei 4 Byte langen Feldern. Das erste dieser Felder enthält die Länge des EEN-Namens, und das zweite enthält dessen Adresse.

INTVERS=
Der Operand legt die Version der Makro-Schnittstelle VSVI1 fest.

BLSP2
Voreinstellung. Entspricht der Makro-Version 3.

SRV001
Entspricht der Makro-Version 4. Diese Version wird ab BLSSERV V2.0 unterstützt.

SRV002
Entspricht der Makro-Version 5. Diese Version wird ab BLSSERV V2.3B unterstützt.

SRV003
Entspricht der Makro-Version 6. Diese Version wird ab BLSSERV V2.5A unterstützt.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. angegebenen Operanden PREFIX und PARAM siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

LABEL=name
Angabe nur mit MF=M
Name der Struktur, d.h. der DSECT, die den Datenbereich des **VSVI1**-Makros beschreibt. Der Operand muss angegeben werden, wenn keine gültige USING-Anweisung für die Definition des Basisadressregisters für die DSECT des Datenbereichs angegeben ist.
Der Operand LABEL muss zusammen mit dem Operanden PARAM angegeben werden. Beide Operanden werden benutzt, um eine gültige USING-Anweisung zu gewinnen.

Als Name kann angegeben werden:

1. Der Name der im Namensfeld eines vorhergehenden Makroaufrufs `name VSVI1 MF=D` angegeben wurde.
2. Der Name „xVSVIDS“, wenn bisher noch kein Name name angegeben wurde. Dabei ist „x“ der Wert des Operanden PREFIX eines vorhergehenden Makroaufrufs
   `VSVI1 MF=D, PREFIX=x`
   Der Standardwert von „x“ ist „P“.
3. Der Name der längeren DSECT, die den Datenbereich des **VSVI1**-Makros enthält, wenn zuvor der Makroaufruf `VSVI1 MF=C` angegeben wurde.

LEN=
legt fest, ob in den ausgegebenen Informationen die Längen enthalten sind.

YES
Die Längen werden ausgegeben.

NO
Die Längen werden nicht ausgegeben.

NAME=
legt fest, ob in den ausgegebenen Informationen die Namen von CSECTs, ENTRYs und COMMON-Bereichen enthalten sind.

YES
Die Namen werden ausgegeben.

NO
Die Namen werden nicht ausgegeben.

OUTADDR=
gibt die Adresse eines Feldes an, in das der DBL die Informationen übertragen soll.

adr
Adresse eines Hilfsfeldes, das die gesuchte Feldadresse enthält.
Angabe nur mit MF=M.

(r)
r = Register mit der gesuchten Feldadresse. Angabe nur mit MF=M.

label
Symbolische Adresse des Feldes. Angabe nur mit MF=S oder MF=L.

OUTLEN=integer
gibt die Länge des Ausgabefeldes (in Byte) an. Die Mindestlänge ist 4 Byte.

RUNMOD=
legt den Betriebsmodus fest, in dem der **VSVI1**-Makro abgearbeitet wird.

STD
Der Makro wird in der Standardform abgearbeitet. Die ausgegebene Information ist dabei voll kompatibel zu der Information, die vom alten Makro VSVI der BS2000-Version 9.5 ausgegeben wurde.
In der Standardform verarbeitet der Makro **VSVI1** bei CSECTs, ENTRYs und COMMON-Bereichen nur 8 Zeichen lange Namen und bei Kontexten nur maximal 16 Zeichen lange Namen. Dies kann zu Konflikten führen, wenn z. B. Informationen über ein geladenes LLM ausgegeben werden sollen, da längere Namen gekürzt werden.

ADV
Der Makro wird in der erweiterten Form abgearbeitet. In der erweiterten Form verarbeitet der Makro **VSVI1** maximal 32 Zeichen lange Namen.
Folgende Operanden sind nur zusammen mit RUNMOD=ADV erlaubt:
SELECT=ILELIST, CTXSEL, SCOPE, CTXPRIV, HSI, VERSION.

SCOPE=
gibt an, welche Memory-Pool-Kontexte berücksichtigt werden, wenn CTXSEL=POOL angegeben wurde (nur zusammen mit RUNMOD=ADV).

ALL
Alle Kontexte von Memory Pools werden berücksichtigt.

GROUP
Nur Kontexte von Common Memory Pools mit dem Geltungsbereich GROUP werden berücksichtigt.

USER_GROUP
Nur Kontexte von Common Memory Pools mit dem Geltungsbereich USER_GROUP werden berücksichtigt.

GLOBAL
Nur Kontexte von Common Memory Pools mit dem Geltungsbereich GLOBAL werden berücksichtigt.

SELECT=
legt die Art der Information fest, die ausgegeben wird.

CTXLIST
Eine Liste mit Namen der Kontexte wird ausgegeben (Benutzerkontexte der Task und/oder Systemkontexte und/oder Memory-Pool-Kontexte und/oder lokale Subsystemkontexte).

CTXSIZE
Diese Angabe ist nur bei RUNMOD=ADV erlaubt.

Der Umfang des einem Kontext geladenen Codes und der Umfang der dazugehörigen Binde- und Ladeinformationen wird ausgegeben. Die Angabe erfolgt in Byte und wird auf ein Vielfaches von 4 KByte aufgerundet. Der Name des Kontexts muss mit INCTX oder INCTX@ festgelegt werden.

ALLLIST
Eine Liste mit Namen, Ladeadressen, Längen und Attributen aller CSECTs, ENTRYs und COMMON-Bereiche wird ausgegeben.

MODLIST
Eine Liste mit Namen, Ladeadressen, Längen und Attributen aller CSECTs und COMMON-Bereiche wird ausgegeben.

BYNAME
Ein Satz mit Name, Ladeadresse, Länge und Attributen eines einzelnen Programmabschnitts (CSECT), ENTRY oder COMMON-Bereichs wird ausgegeben. Der Name des Symbols muss mit dem Operanden INNAME oder INSTRUCT (und INTVERS=SRVxxx mit xxx ≥ 001) festgelegt werden.

BYADDR
Ein Satz mit Name, Ladeadresse, Länge und Attributen eines einzelnen Programmabschnitts (CSECT) oder COMMON-Bereichs wird ausgegeben. Die Adresse des Symbols muss mit dem Operanden INADDR festgelegt werden.

ILELIST
Eine Liste mit Informationen über ILEs im angegebenen Kontext wird ausgegeben. SELECT=ILELIST darf nur zusammen mit RUNMOD=ADV angegeben werden.

SIZONLY=
legt fest, ob die gewünschte Information oder nur die Länge der gewünschten Information ausgegeben wird.

NO
Die Information wird in das Ausgabefeld übertragen.

YES
Nur die Länge der Information wird in das Ausgabefeld übertragen.

SYMTYP=
Dieser Operand ist bei Angabe von INTVERS=SRVxxx und xxx ≥ 003 verfügbar. Er ist nur zusammen mit dem Operanden SELECT=BYNAME sinnvoll und wird andernfalls ignoriert. Er gibt den Typ des mit INNAME, INNAME@ oder INSTRUCT definierten Symbols an, für das Information angefordert wird.

ANY
Der Typ des gesuchten Symbols ist irrelevant.

CSECT
Es wird nur nach CSECTs mit dem angegebenen Namen gesucht.

ISL
Diese Angabe ist nur für privilegierte Benutzer relevant. Außerdem ist sie nur sinnvoll, wenn der CP-Kontext (siehe INCTX/INCTX@) oder privilegierte Kontexte (CTXSEL=ALL/GLOBAL und CTXPRIV=ANY/ALL) durchsucht werden.

Es wird nur nach ISL ENTRYs mit dem angegebenen Namen gesucht.

NOTISL
Es wird nur nach Symbolen mit dem angegebenen Namen gesucht, die CSECTs oder ENTRYs, jedoch nicht ISL sind.

TYPE=
legt fest, ob in den ausgegebenen Informationen die Typen (CSECT/ENTRY/COMMON) enthalten sind.

YES
Die Typen werden ausgegeben.

NO
Die Typen werden nicht ausgegeben.

VERSION=
legt fest, ob die Ausgabe Informationen über die Programmversion enthält (nur zusammen mit RUNMOD=ADV).

NO
Die Programmversion wird nicht ausgegeben.

YES
Die Programmversion wird ausgegeben. Der Operandenwert ist nur für die Ausgabe von Symbolinformationen relevant und er darf nur zusammen mit RUNMOD=ADV angegeben werden.

Hinweise zum Makroaufruf

- Für die Ausgabe von Informationen muss mindestens einer der Operanden NAME, ADDRESS, LEN, TYPE, CONTEXT, VERSION oder HSI angegeben werden.
- Keiner der Operanden NAME, ADDRESS, LEN, TYPE, CONTEXT, VERSION oder HSI muss angegeben werden, wenn der Benutzer nur prüfen will, ob
  - ein angegebener Name (Operand INNAME/INSTRUCT) der Name eines Symbols ist (Operand SELECT=BYNAME),
  - eine angegebene Adresse (Operand INADDR) verfügbar ist (Operand SELECT=BYADDR),
  - ein angegebener Kontext (Operand INCTX) verfügbar ist (Operand SELECT=ALLLIST oder SELECT=MODLIST).

  In diesen Fällen ist nur der Returncode von Bedeutung. Kann der Name, die Adresse oder der Kontext nicht gefunden werden, wird folgender Returncode übergeben:
  X'0440003C' Name des Symbols nicht gefunden,
  X'04400038' Adresse nicht gefunden,
  X'04400040' Kontext nicht gefunden.

  Wird der Name, die Adresse oder der Kontext gefunden, übergibt der DBL den Returncode X'00000000'.

- Die Länge des Ausgabebereichs, in das der DBL die Information übertragen soll, muss mit dem Operanden OUTLEN festgelegt werden. Bei zu kleiner Längenangabe wird die Information abgeschnitten auf die Länge, die im Operanden OUTLEN angegeben wurde. Ist die angegebene Länge kleiner als die Länge der kleinsten Teilinformation, wird keine Information ausgegeben. In beiden Fällen übergibt der DBL einen Returncode.
- Enthält ein Modul mehrere CSECTs, wählt der DBL die Information in Abhängigkeit vom Operanden SELECT wie folgt aus:
  – bei SELECT=BYADDR bezieht sich die Information auf die CSECT, die die mit INADDR angegebene Adresse enthält.
  – bei SELECT=BYNAME und wenn INNAME einen CSECT-Namen bezeichnet, bezieht sich die Information auf diese CSECT.
  – bei SELECT=ALLLIST wird für jede CSECT ihre eigene Information ausgegeben.
- Grundsätzlich führt der **VSVI1**-Makro im Fehlerfall die Verarbeitung so weit wie möglich durch. Wenn z.B. die Operanden festlegen, dass mehr als ein Kontext durchsucht werden muss, und ein Fehler während des Durchsuchens eines Kontextes auftritt, setzt der Makro **VSVI1** mit dem Durchsuchen des nächsten Kontextes fort. Die Fehlerursache wird vom DBL in einem Returncode übergeben. Dieser Returncode bezieht sich immer auf den zuletzt aufgetretenen Fehler.
- Wenn SELECT=BYNAME angegeben ist, muss einer der beiden Operanden INNAME oder INSTRUCT angegeben werden. Sie dürfen jedoch nicht gemeinsam angegeben werden.
- Bei SELECT=BYNAME durchsucht der DBL den Kontext nur solange, bis das erste Symbol des angegebenen Namens (Operand INNAME oder INSTRUCT) gefunden wird. Sind mehrere Symbole mit gleichen Namen vorhanden, wird nur die Information über das erste gefundene Symbol ausgegeben.
- Wird nur die Länge der Information gewünscht (Operand SIZONLY=YES), nimmt der DBL als Länge die Länge der Information bis zum Leereintrag.
- Wenn der bei OUTADDR angegebene Speicherbereich nur lesbar ist oder wenn bei OUTLEN Null angegeben wurde, wird die Verarbeitung mit einem USER-DUMP abgebrochen.
- Wird eine Kontextliste angefordert (SELECT=CTXLIST), so können die Operanden NAME, ADDRESS, LEN, TYPE, CONTEXT, VERSION, HSI und SIZONLY mit NO belegt werden.
- Bei der Abfrage von Informationen über Kontexte von Common Memory Pools (CTXSEL=POOL) ist es möglich, dass der Returncode X'08400048' wegen konkurrierender Zugriffe auf einen Common Memory Pool zurückgegeben wird.

- Für nichtprivilegierte Benutzer (nicht $TSOS) gilt:
  - Die benutzerspezifischen Werte für die Operanden werden ignoriert und vom DBL intern mit CTXPRIV=NO belegt.
  - Für den Operanden CTXSEL werden andere Werte als POOL ignoriert und vom DBL intern mit CTXSEL=LOCAL belegt.
  - Wird VSVI1 mit SELECT=BYNAME, aber ohne INCTX aufgerufen, sucht der DBL zuerst im privaten Klasse-6-Speicher und in Common Memory Pools. Findet der DBL das angegebene Symbol darin nicht, so versucht er, eine Verbindung zu nichtprivilegierten Subsystemen herzustellen.

Mögliche Operandenkombinationen

Die folgende Tabelle zeigt, welche Operanden in Abhängigkeit vom Operanden SELECT Pflicht oder erlaubt sind:

SELECT=	**INADDR**	**INNAME/ INSTRUCT**	**INCTX**	**OUTADDR**	**OUTLEN**	**CTXSEL**	**CTXPRIV**	**SCOPE**	**SYMTYP**
CTXLIST	-	-	-	P	P	E	E	E	I
CTXSIZE	-	-	P	P	P	-	-	-	I
ALLLIST	-	-	E	P	P	E	E	E	I
MODLIST	-	-	E	P	P	E	E	E	I
ILELIST	-	-	E	P	P	E	E	E	I
BYADDR	P	-	E	P	P	E	E	E	I
BYNAME	-	P	E	P	P	E	E	E	E

P Pflichtparameter
E Erlaubter Operand
I Ignorierter Operand
- Der Operand ist für diese SELECT-Angabe nicht sinnvoll.

Ausgabeinformation bei RUNMOD=STD

Im Betriebsmodus RUNMOD=STD wird der Makro in der Standardform abgearbeitet. Die Ausgabeinformation ist dabei voll kompatibel zu der Information, die vom alten Makro VSVI der BS2000-Version 9.5 ausgegeben wurde.

In der Standardform verarbeitet der Makro bei CSECTs, ENTRYs und COMMON-Bereichen nur maximal 8 Zeichen lange Namen bzw. bei Kontexten nur maximal 16 Zeichen lange Namen. Längere Namen von CSECTs, ENTRYs und COMMON-Bereichen werden auf 8 Zeichen, längere Namen von Kontexten auf 16 Zeichen gekürzt.

Die Bezugsgröße für eine ausgegebene Information ist ein Eintrag in den DBL-Tabellen. Der Eintrag hat eine feste Länge von 36 Byte.

Aufbau der Einträge

Byte	Länge	Feld	Feldeintrag	Codierung/Bemerkung
0- 7	8	Name des Symbols	linksbündig mit nachfolgenden Leerzeichen	Zeichenkonstante
8-11	4	Ladeadresse	-	Sedezimalkonstante
12-15	4	Länge	-	Sedezimalkonstante
16	1	Typ	-	X'F0' ≙ CSECT X'F1' ≙ ENTRY X'F3' ≙ COMMON
17	1	Attribute	pro Attribut 1 oder 2 Bit	2^7 ≙ INVISIBILITY 2^6 und 2^5 ≙ AMODE 00 ≙ AMODE=32 01 ≙ AMODE=31 10 ≙ AMODE=24 11 ≙ AMODE=ANY 2^4 ≙ RESIDENT 2^3 ≙ PAGE 2^2 ≙ READ-ONLY
18-19	2		X'0000'	Ausrichtung des folgenden Felds
20-35	16	Name des Kontextes	linksbündig mit nachfolgenden Leerzeichen	Zeichenkonstante

Wenn nur die Länge der Information gewünscht wird (Operand SIZONLY=YES) belegt diese die ersten 4 Byte des Ausgabefeldes.

Mehrere Einträge werden in der gleichen Reihenfolge hintereinander ausgegeben. Dem letzten Eintrag folgt ein Leereintrag. Ein Pseudoeintrag wird ausgegeben, wenn die angegebene Programmadresse nicht aufgefunden wird (Operand SELECT=BYADDR).

Feld	Leereintrag		Pseudoeintrag
	Länge	Feldeintrag	
Name der CSECT/COMMON/ENTRY	8	X'40.....40'	C'ABSOLUTE'
Ladeadresse	4	X'0.....0'	X'0.....0'
Länge	4	X'F.....F'	X'0.....0'
Typ	1	X'C5'	X'00'
Attribut	1	X'00'	X'00'
Name des Kontexts	16	X'40.....40'	X'40.....40'

Hinweise

– Die Länge des Ausgabefeldes ergibt sich aus der Addition der einzelnen Einträge. Bei zu kleiner Längenangabe wird die Information abgeschnitten. Die Information wird nicht übertragen, wenn die Mindestlänge für einen geforderterten Eintrag unterschritten wird. Mindestlänge = Länge eines Eintrags abzüglich der Länge nicht auszugebender Teile (z.B. NAME=NO).

– Für einen ENTRY sind nur die Attribute INVISIBILITY und AMODE relevant; für einen COMMON-Bereich nur PAGE und AMODE.

– Ein Programmabschnitt oder ENTRY hat das Attribut INVISIBILITY, wenn er beim Binden oder späteren Laden maskiert wurde.

– Bei Angabe von SELECT=CTXLIST wird eine Liste von 16 Byte langen Kontextnamen ausgegeben, gefolgt von 16 Leerzeichen (X'40') als Endekennzeichen.

  Format:

CTX1␣␣␣␣␣␣␣␣␣␣␣␣
CTX2␣␣␣␣␣␣␣␣␣␣␣␣
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣

– Bei Angabe von SELECT=CTXSIZE besteht die Ausgabeinformation aus zwei Worten. Das erste Wort enthält den Umfang des Codes im Kontext, das zweite Wort enthält den Umfang der Binde- und Ladeinformationen zu diesem Kontext. Die Angabe erfolgt in Byte und wird auf ein Vielfaches von 4 KB aufgerundet.

Ausgabeinformation bei RUNMOD=ADV

Im Betriebsmodus RUNMOD=ADV wird der Makro in der erweiterten Form abgearbeitet. In der erweiterten Form verarbeitet der Makro maximal 32 Zeichen lange Namen und es können zusätzlich ILE-Informationen, HSI-Code und HSI-Compiler-Informationen angefordert werden. Die Bezugsgröße für eine angegebene Information ist ein Eintrag in den DBL-Tabellen. Der Eintrag hat eine variable Länge, die abhängig ist von der Länge der Namen und der Art der gewünschten Information.

Aufbau der Einträge

Byte	Länge	Feld	Feldeintrag	Codierung / Bemerkung
0 - 3	4	Ladeadresse	—	Sedezimalkonstante
4 - 7	4	Länge	—	Sedezimalkonstante
8	1	Typ	—	X'F0' = CSECT X'F1' = ENTRY X'F2' = COMMON
9	1	Attribute	pro Attribut 1 oder 2 Bit	2^7 ≙ INVISIBILITY 2^6 und 2^5 ≙ AMODE 00 ≙ AMODE=32 01 ≙ AMODE=31 10 ≙ AMODE=24 11 ≙ AMODE=ANY 2^4 ≙ RESIDENT 2^3 ≙ PAGE 2^2 ≙ READ-ONLY
10 - 11	2		X'0000'	Ausrichtung des folgenden Feldes
12	1	HSI-Code	—	X'01' = 7500 (/390) X'09' = X86
13	1	HSI-Compiler-Information	—	abhängig vom Compiler; BLS zeichnet diese Information nur auf und gibt sie hier aus.
14	1	Länge n des Symbolnamens	—	Sedezimalkonstante
15	n	Name des Symbols	—	Zeichenkonstante
15+n	1	Länge m der Version	—	Sedezimalkonstante
15+(n+1)	m	Version des Programmes, zu dem das Symbol gehört	—	Zeichenkonstante
15+(n+1) +m	1	Länge l des Kontextnamens	—	Sedezimalkonstante
15+(n+1) +(m+1)	l	Name des Kontextes	—	Zeichenkonstante

Die Gesamtlänge eines Eintrages kann wie folgt berechnet werden:

Eintragslänge = 17 + n + m + l

Wenn nur die Länge der Information gewünscht wird (Operand SIZONLY=YES) belegt diese die ersten 4 Byte des Ausgabefeldes.

Leereinträge und Pseudoeinträge

Mehrere Einträge werden in der gleichen Reihenfolge hintereinander ausgegeben. Dem letzten Eintrag folgt ein Leereintrag. Ein Pseudoeintrag wird ausgegeben, wenn die angegebene Programmadresse nicht aufgefunden wird (Operand SELECT=BYADDR).

Feld	**Leereintrag**		**Pseudoeintrag**
	Länge	**Feldeintrag**	
Ladeadresse	4	X'0.....0'	X'0.....0'
Länge	4	X'F.....F'	X'0.....0'
Typ	1	X'C5'	X'00'
Attribut	1	X'00'	X'00'
Länge n des Symbolnamens	1	X'08'	X'08'
Name des Symbols	8	X'40.....40'	C'ABSOLUTE'
Länge m des Kontextnamens	1	X'00'	X'00'

Hinweise

– Bei Angabe von SELECT=CTXLIST wird eine Liste von Kontextnamen variabler Länge ausgegeben. Die Listenelemente enthalten im ersten Byte die Länge des Kontextnamens und in den folgenden Bytes den Kontextnamen. Die Liste wird mit einem Listenelement abgeschlossen, das 32 Leerzeichen (X'40') enthält.

  Format:

0D	LOCAL#DEFAULT
04	CTX1
04	CTX2
20	␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣

– Bei Angabe von SELECT=CTXSIZE besteht die Ausgabeinformation aus zwei Worten. Das erste Wort enthält den Umfang des Codes im Kontext, das zweite Wort enthält den Umfang der Binde- und Ladeinformationen zu diesem Kontext. Die Angabe erfolgt in Byte und wird auf ein Vielfaches von 4 KByte aufgerundet.

– Ruft ein Benutzer (ungleich $TSOS) den Makro **VSVI1** mit SELECT=BYNAME, aber ohne Angabe von INCTX auf und findet der DBL dieses Symbol dann weder im Klasse-6-Speicher dieses Benutzers noch Shared Code in Memory Pools, so versucht der DBL, eine Verbindung zu den nichtprivilegierten Subsystemen des DSSM aufzubauen.

– Bei Angabe von SELECT=ILELIST wird für jedes ILE ein formatierter Listeneintrag ausgegeben, in dem auf Anforderung auch der Kontextname enthalten sein kann. Bei Angabe von CONTEXT=NO wird der Kontextname nicht ausgegeben. Ein Eintrag für ein ILE-Symbol hat folgendes Format:

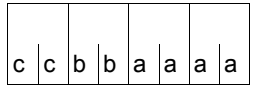
Byte	Länge	Feldname	Bedeutung und/oder Werte
0	1	STATE	X'01' = ACTIVE X'02' = NOT_ACTIVE
1	1	CONTROL	X'01' = SYSTEM X'02' = USER
2	1	HSI_CODE	X'01' = 390 X'09' = X86
3	1	RESERVED1	reserviert (muss mit X'00' belegt sein)
4	4	LOAD_ADDR	Adresse der IL-Routine
8	4	SERVER_ADDR	Adresse des ILE-Servers
12	2	REF_DISPL	Distanz des Externverweises auf den Server innerhalb der IL-Routine
14	32	NAME	Name des ILE-Symbols
46	2	RESERVED2	reserviert (muss mit X'0000' belegt sein)
48	32	CONTEXT	Name des Kontextes, zu dem der ILE gehört

Das Format des Eintrages entspricht ungefähr dem Format, das auch beim Makroaufruf **ILEMIT** erzeugt wird.

Dem letzten ILE-Eintrag folgt ein Leereintrag, in dem alle Felder mit Ausnahme des NAME-Feldes mit binären Nullen belegt sind. Das NAME-Feld enthält Leerzeichen. Ein Leereintrag enthält kein CONTEXT-Feld.

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros VSVI1 wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Der Makro wurde normal ausgeführt.
X'0C'	X'01'	X'0018'	Ein reserviertes Feld des Datenbereichs ist nicht mit Nullen vorbelegt.
X'0C'	X'01'	X'0020'	Die angegebene Länge des Ausgabefeldes (Operand OUTLEN) ist kleiner als die tatsächliche Länge des Feldes.
X'0C'	X'01'	X'0024'	Das Ausgabefeld (Operand OUTADDR) ist nicht auf Halbwortgrenze ausgerichtet, nur lesbar oder nicht allokiert.
X'0C'	X'01'	X'0028'	Unzulässige Angabe für den Operanden SELECT.
X'0C'	X'01'	X'002C'	Unzulässige Angabe für einen Operanden. Dies kann sein: – HSI oder VERSION wurde zusammen mit RUNMOD=STD angegeben, – (syntaktisch) falscher Name bei INNAME, INSTRUCT oder INCTX angegeben, – privilegierter Kontext angegeben und der Benutzer ist nicht privilegiert, – ein nichtpriv. Benutzer hat im Betriebsmodus RUNMOD=STD einen unzulässigen Kontextnamen angegeben. Bei RUNMOD=STD dürfen von einem nichtpriv. Benutzer nur Informationen von Kontexten angefordert werden, deren Namen aus Leerzeichen (X'40') oder aus der Zeichenkette „LOCAL#DEFAULT“ besteht. – SELECT=BYADDR wurde ohne oder mit ungültiger INADDR angegeben (z.B. INADDR=X'FFFFFFFF'), – ein nichtpriv. Benutzer hat SYMTYP mit einem Wert ungleich ANY und `SELECT=BYNAME,CTXSEL=LOC,RUNMOD=ADV` angegeben.
X'0C'	X'01'	X'002D'	Unzulässige Angabe für den Operanden SYMTYP.
X'0C'	X'01'	X'0030'	Der Aufrufer hat keinen der Informationsoperanden (NAME, ADDRESS, LEN, TYPE, CONTEXT, VERSION, HSI) angegeben und SELECT ist ungleich BY_NAME oder BY_ADDR. Dieser Returncode wird auch übergeben, wenn: – die Operanden NAME, ADDRESS, LEN, TYPE, HSI, CONTEXT und VERSION alle mit NO belegt sind und SIZONLY=YES wurde angegeben. – die Operanden NAME, ADDRESS, LEN, TYPE, HSI, CONTEXT und VERSION alle mit NO belegt sind, SIZONLY=NO,SELECT=MODLIST oder SELECT=ALLLIST wurden angegeben, und INCTX wurde nicht angegeben.

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'08'	X'40'	X'0034'	Die angegebene Länge des Ausgabebereichs im Operanden OUTLEN ist kleiner als die Gesamtlänge der angeforderten Informationen. Ausgabe unvollständig.
X'0C'	X'01'	X'0034'	Die angegebene Länge des Ausgabebereichs im Operanden OUTLEN ist zu klein, um die angeforderte kleinste Teilinformation zu übertragen. Keine Ausgabe.
X'04'	X'40'	X'0038'	Die im Operanden INADDR angegebene Adresse gehört zu keinem der schon geladenen Module.
X'04'	X'40'	X'003C'	Der im Operanden INNAME oder INSTRUCT angegebene Name bezeichnet keinen der schon geladenen Module.
X'04'	X'40'	X'0040'	Der im Operanden INCTX angegebene Name für einen Kontext kann nicht gefunden werden oder die Task, aus der der VSVI1-Aufruf erfolgt, ist nicht mit dem Subsystem mit dem angegebenen Kontextnamen verbunden.
X'0C'	X'20'	X'0044'	Interner Fehler bei der Funktionsausführung. Keine Ausgabe möglich.
X'08'	X'40'	X'0048'	Ein oder mehrere Kontexte mit globalem Geltungsbereich werden von einer anderen Task benutzt. Ausgabe unvollständig.
X'04'	X'40'	X'004C'	Der Name des Symbols ist größer als 8 Zeichen und RUNMOD=STD ist angegeben. Der Name wurde auf 8 Zeichen gekürzt.
X'04'	X'40'	X'0050'	Der angegebene Kontext ist zwar vorhanden, aber leer, da alle Objekte entladen wurden.
X'04'	X'40'	X'0070'	Die Task ist nicht an den Memory Pool angeschlossen.
X'0C'	X'20'	X'0198'	Fehler bei Speicheranforderung (kein Speicher mehr verfügbar).
X'0C'	X'40'	X'0204'	Interner Fehler im Memory Management.
X'0C'	X'40'	X'0208'	Interner Fehler im Data Manager.
X'0C'	X'40'	X'020C'	Interner Fehler in der symbolischen Informationstabellen.
X'0C'	X'20'	X'0300'	Fehler bei $REQM, $RELM (Systemfehler).
X'00'	X'01'	X'FFFF'	Die Funktion wird nicht mehr oder noch nicht unterstützt.
X'00'	X'03'	X'FFFF'	Die Version der Schnittstelle wird nicht unterstützt.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Beispiel

Während des Programmlaufs von PROGA wird mit Hilfe des Makros **BIND** ein zweiter Programmabschnitt PROGB nachgeladen. PROGB steht als Bindemodul in der Bibliothek MACEXMP.LIB. Vor und nach dem Nachladen von PROGB wird der Makro **VSVI1** aufgerufen, um Binde- und Ladeinformation aus den DBL-Tabellen in einen Ausgabebereich zu übergeben. Beide Programmabschnitte sollen im 31-Bit-Adressierungsmodus ablaufen. PROGA soll unterhalb und PROGB oberhalb der 16MB-Grenze geladen werden. In PROGB ist ein ENTRY vereinbart. Nach Aufruf des Makros **BIND** soll zuerst PROGB ablaufen. Nach dem Ablauf von PROGB soll in PROGA zurückverzweigt werden.

Ausdruck des Quellprogramms

```
PROGA    START
PROGA    AMODE 31 ------------------------------------------------------------ (1)
PROGA    RMODE 24
         BALR  3,0
         USING *,3
         USING BINDDS,6 ------------------------------------------------------ (2)
         USING VSVI1DS,7 ----------------------------------------------------- (3)
         ST    3,AREA11
         UNPK  AREAH,AREA1
         MVC   AREAA(8),AREAH
WROUT1   WROUT OUT,ERROR,PARMOD=31 ---------------------------------------- (4)
         MVI   ADR1,X'D1' ---------------------------------------------------- (5)
         MVC   ADR1+1(L'ADR1-1),ADR1
         VSVI1 MF=E,PARAM=VSVI1PAR ------------------------------------------- (6)
         LA    7,VSVI1PAR
         CLC   YVSVRET,=X'00000000' ------------------------------------------ (7)
         BNE   VSVIERR ------------------------------------------------------- (8)
BACK     LA    12,VSVI
BIND     BIND  MF=E,PARAM=BINDPAR -------------------------------------------- (9)
         LA    6,BINDPAR
         CLC   XBINRET,=X'00000000' ------------------------------------------ (10)
         BE    VSVI
         MVC   OUT+5(28),='BIND ERROR!                '
         WROUT OUT,ERROR,PARMOD=31 ------------------------------------------- (11)
         B     ERROR
VSVI     VSVI1 MF=E,PARAM=VSVI1PAR ------------------------------------------- (12)
         CLC   YVSVRET,=X'00000000' ------------------------------------------ (13)
         BE    MVC
VSVIERR  MVC   OUT+5(28),='VSVI1 ERROR!               '
         WROUT OUT,ERROR,PARMOD=31 ------------------------------------------- (14)
         B     ERROR
MVC      MVC   OUT+5(28),='VSVI1 PROCESSED            '
         WROUT OUT,ERROR,PARMOD=31 ------------------------------------------- (15)
```

```
         MVC   OUT+5(28),='RETURN TO PROGA            '
         WROUT OUT,ERROR,PARMOD=31
ERROR    TERM
*****************
         DS    0F
ADR1     DS    CL180 ------------------------------------------------------ (16)
OUT      DC    Y(OUTE-OUT)
         DS    CL3
         DC    C'PROGA: BASE REG.= '
AREAA    DS    CL8
OUTE     EQU   *
AREA     DS    0F
AREA1    DS    0CL5
AREA11   DS    CL4
AREA12   DC    C'0'
         DS    0F
AREAH    DS    CL9
BINDPAR  BIND  MF=L,SYMBOL=PROGB,SYMBLAD=PROGB@,BRANCH=YES,PROGMOD=ANY,*
               LIBLINK=PLAMLIB ---------------------------------------------- (9)
VSVI1PAR VSVI1 MF=L,SELECT=ALLLIST,CTXSEL=ALL,OUTADDR=ADR1,OUTLEN=180 ---- (6)
PROGB@   DS    A
BINDDS   BIND  MF=D,PREFIX=X ------------------------------------------------ (17)
VSVI1DS  VSVI1 MF=D,PREFIX=Y ------------------------------------------------ (18)
         END
PROGB    CSECT PAGE ------------------------------------------------------- (19)
PROGB    AMODE ANY
PROGB    RMODE ANY
         ENTRY ENTR ------------------------------------------------------- (20)
ENTR     BALR  4,0
         USING *,4
         ST    4,AREA11
         UNPK  AREAH,AREA1
         MVC   AREAA(8),AREAH
         WROUT OUT,ERROR,PARMOD=31 ---------------------------------------- (21)
         BR    12
ERROR    TERM
*****************
OUT      DC    Y(OUTE-OUT)
         DS    CL3
         DC    C'PROGB: BASE REG.= '
AREAA    DS    CL8
OUTE     EQU   *
AREA     DS    0F
AREA1    DS    0CL5
AREA11   DS    CL4
AREA12   DC    C'0'
AREAH    DS    CL9
         END
```

(1) Für den Programmabschnitt PROGA wird das Attribut AMODE=31 vereinbart. Mit RMODE=24 wird PROGA immer unterhalb der 16MB-Grenze geladen.

(2) Register 6 wird dem Assembler als Basisadressregister zur Adressierung der DSECT für die Operandenliste des **BIND**-Makros zugewiesen, die an der symbolischen Adresse BINDDS durch einen **BIND**-Aufruf mit MF=D erzeugt wird.

(3) Register 7 wird dem Assembler als Basisadressregister zur Adressierung der DSECT für die Operandenliste des **VSVI1**-Makros zugewiesen, die an der symbolischen Adresse VSVI1DS durch einen **VSVI1**-Aufruf mit MF=D erzeugt wird.

(4) Der Inhalt des Basisregisters von PROGA wird zur Darstellung des Adressierungsmodus und der Ladeadresse ausgegeben.

(5) Das Ausgabefeld für den **VSVI1**-Makro wird mit C'J' vorbesetzt.

(6) Der Makro **VSVI1** wird in seiner E-Form aufgerufen. An dieser Stelle im Programm wird daher nur der Befehlsteil erzeugt. Die zugehörige Operandenliste wird an der symbolischen Adresse VSVI1PAR durch einen **VSVI1**-Aufruf mit MF=L angelegt. Die dort angegebenen Operandenwerte veranlassen den **VSVI1**-Makro eine Liste mit Namen, Ladeadressen, Längen und Attributen aller CSECTs, ENTRYs und COMMON-Bereichen vor dem Nachladen von PROGB auszugeben.

(7) Nach der Ausführung des **VSVI1**-Makros wird geprüft, ob das Feld YVSVRET des Standardheaders den Returncode X'00000000' enthält, der eine fehlerfreie Makroausführung anzeigt. Der Name YVSVRET stammt aus der DSECT, die unter der symbolischen Adresse VSVI1DS durch einen VSVI1-Aufruf mit MF=D und PREFIX=Y erzeugt wurde (siehe 18). Diese DSECT beschreibt den Aufbau der Operandenliste des **VSVI1**-Makros. Die symbolischen Namen der DSECT können zur Adressierung innerhalb der Operandenliste verwendet werden, nachdem das zugeordnete Basisadressregister (hier Register 7) mit der Anfangsadresse der Operandenliste (hier VSVI1PAR) geladen worden ist.

(8) Wenn der **VSVI1**-Makro nicht fehlerfrei ausgeführt wurde, wird zum Fehlerausgang VSVIERR verzweigt, eine Fehlermeldung über SYSOUT ausgegeben und der Programmlauf von PROGA beendet.

(9) An der symbolischen Adresse BIND wird der Makro **BIND** in seiner E-Form aufgerufen. An dieser Stelle im Programm wird daher nur der Befehlsteil erzeugt. Die zugehörige Operandenliste wird an der symbolischen Adresse BINDPAR durch einen **BIND**-Aufruf mit MF=L angelegt. Die dort angegebenen Operandenwerte veranlassen den **BIND**-Makro, bei der Programmausführung

- die CSECT PROGB (SYMBOL=PROGB) aus der mit dem Linknamen PLAMLIB zugewiesenen Bibliothek (LIBLINK=PLAMLIB) nachzuladen,
- die Startadresse von PROGB im Feld PROGB@ zu hinterlegen (SYMBLAD=PROGB@),

– für PROGB den 31-Bit-Adressierungsmodus einzustellen (PROGMOD=ANY),

– nach dem Laden von PROGB den Programmlauf in PROGB fortzusetzen (BRANCH=YES).

(10) Nach der Ausführung des **BIND**-Makros wird geprüft, ob das Feld XBINRET des Standardheaders den Returncode X'00000000' enthält, der eine fehlerfreie Makroausführung anzeigt. Der Name XBINRET stammt aus der DSECT, die unter der symbolischen Adresse BINDDS durch einen BIND-Aufruf mit MF=D und PREFIX=X erzeugt wurde (siehe (17)). Diese DSECT beschreibt den Aufbau der Operandenliste des **BIND**-Makros. Die symbolischen Namen der DSECT können zur Adressierung innerhalb der Operandenliste verwendet werden, nachdem das zugeordnete Basisadressregister (hier Register 6) mit der Anfangsadresse der Operandenliste (hier BINDPAR) geladen worden ist.

(11) Wenn der **BIND**-Makro nicht fehlerfrei ausgeführt wurde, wird eine Fehlermeldung über SYSOUT ausgegeben und der Programmlauf von PROGA beendet.

(12) Wie (6), aber nach dem Nachladen von PROGB.

(13) Wie (7), aber nach dem Nachladen von PROGB.

(14) Wenn der **VSVI1**-Makro nicht fehlerfrei ausgeführt wurde, wird eine Fehlermeldung über SYSOUT ausgegeben und der Programmlauf von PROGA beendet.

(15) Meldungen nach SYSOUT informieren darüber, dass die Programmausführung in PROGA fortgesetzt und anschließend Binde- und Ladeinformation mit dem Makro **VSVI1** ausgegeben wurde.

(16) Ausgabefeld für den **VSVI1**-Makro

(17) Der Makroaufruf **BIND** mit MF=D erzeugt eine DSECT, die den Aufbau der Operandenliste des **BIND**-Makros beschreibt. Der Operand PREFIX=X bewirkt, dass alle symbolischen Namen in dieser DSECT (Feldnamen und Equates) mit dem Buchstaben X beginnen.

(18) Der Makroaufruf **VSVI1** mit MF=D erzeugt eine DSECT, die den Aufbau der Operandenliste des **VSVI1**-Makros beschreibt. Der Operand PREFIX=Y bewirkt, dass alle symbolischen Namen in dieser DSECT (Feldnamen und Equates) mit dem Buchstaben Y beginnen.

(19) Die CSECT-Anweisung definiert den Programmabschnitt PROGB mit den Attributen AMODE=ANY und PAGE.

(20) ENTRY-Anweisung für die symbolische Adresse ENTR.

(21) Der Inhalt des Basisregisters von PROGB wird zur Darstellung des Adressierungsmodus und der Ladeadresse ausgegeben.

Ablaufprotokoll

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,proga), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,proga)), -
//        test-support=*aid
%  ASS6011 ASSEMBLY TIME: 786 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 232 MSEC
//compile source=*library-element(macexmp.lib,progb), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,progb)), -
//        test-support=*aid
%  ASS6011 ASSEMBLY TIME: 191 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 93 MSEC
//end
%  ASS6012 END OF ASSEMBH
/add-file-link link-name=plamlib,file-name=macexmp.lib ————————————————  (1)
/load-executable-program library=macexmp.lib,element-or-symbol=proga -    (2)
//        program-mode=*any,test-options=*aid
%  BLS0523 ELEMENT 'PROGA', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'PROGA', VERSION ' ' OF '<date> <time>' LOADED
/%in back;%in error;%r  ——————————————————————————————————————————————  (3)
PROGA: BASE REG.= 80000002 ———————————————————————————————————————————  (4)
STOPPED AT LABEL: BACK , SRC_REF: 53, SOURCE: PROGA , PROC: PROGA
```

```
/%d adr1 %x ──────────────────────────────────────────────────────────── (5)
*** TID: 00100136 *** TSN: 1E17 ********************************************
CURRENT PC: 00000054    CSECT: PROGA  ***************************************
V'00000144' = ADR1     + #'00000000'
00000144 (00000000) D7D9D6C7 C1404040 00000000 00000384    PROGA   .......d
00000154 (00000010) F0200000 D3D6C3C1 D37BC4C5 C6C1E4D3    0...LOCAL#DEFAUL
00000164 (00000020) E3404040 40404040 40404040 00000000    T           ....
00000174 (00000030) FFFFFFFF C5000000 40404040 40404040    ~~~~E...
00000184 (00000040) 40404040 40404040 D1D1D1D1 D1D1D1D1            JJJJJJJJ
00000194 (00000050) D1D1D1D1 D1D1D1D1 D1D1D1D1 D1D1D1D1    JJJJJJJJJJJJJJJJ
           REPEATED LINES:     4
000001E4 (000000A0) D1D1D1D1 D1D1D1D1 D1D1D1D1 D1D1D1D1    JJJJJJJJJJJJJJJJ
000001F4 (000000B0) D1D1D1D1                                JJJJ
/%r
PROGB: BASE REG.= 81000002 ──────────────────────────────────────────────── (6)
VSVI1 PROCESSED
RETURN TO PROGA
STOPPED AT LABEL: ERROR , SRC_REF: 208, SOURCE: PROGA , PROC: PROGA
/%d adr1 %x ──────────────────────────────────────────────────────────── (7)
CURRENT PC: 00000126    CSECT: PROGA  ***************************************
V'00000144' = ADR1     + #'00000000'
00000144 (00000000) D7D9D6C7 C2404040 01000000 0000008A    PROGB   ........
00000154 (00000010) F0680000 D3D6C3C1 D37BC4C5 C6C1E4D3    0...LOCAL#DEFAUL
00000164 (00000020) E3404040 C5D5E3D9 40404040 01000000    T   ENTR    ....
00000174 (00000030) 00000000 F1600000 D3D6C3C1 D37BC4C5    ....1-..LOCAL#DE
00000184 (00000040) C6C1E4D3 E3404040 D7D9D6C7 C1404040    FAULT   PROGA
00000194 (00000050) 00000000 00000384 F0200000 D3D6C3C1    .......d0...LOCA
000001A4 (00000060) D37BC4C5 C6C1E4D3 E3404040 40404040    L#DEFAULT
000001B4 (00000070) 40404040 00000000 FFFFFFFF C5000000        ....~~~~E...
000001C4 (00000080) 40404040 40404040 40404040 40404040
000001D4 (00000090) D1D1D1D1 D1D1D1D1 D1D1D1D1 D1D1D1D1    JJJJJJJJJJJJJJJJ
000001E4 (000000A0) D1D1D1D1 D1D1D1D1 D1D1D1D1 D1D1D1D1    JJJJJJJJJJJJJJJJ
000001F4 (000000B0) D1D1D1D1                                JJJJ
```

(1) Der im BIND-Aufruf des Programmes PROGA verwendete Dateikettungsname wird zugewiesen.

(2) Der DBL wird aufgerufen, um das Programm zu binden und zu laden.

(3) Mit dem AID-Kommando %INSERT werden die Testpunkte BACK und ERROR festgelegt. Das %RESUME-Kommando übergibt die Steuerung an das aufgerufene Programm.

(4) Der Inhalt des Basisregisters von PROGA wird ausgegeben. 31-Bit-Adressierung ist eingestellt (Bit 2^{31} = 1); die Ladeadresse liegt unterhalb der 16MB-Grenze.

(5) In das Feld ADR1 wurde die Binde- und Ladeinformation übertragen. ADR1 wurde mit X'D1' vorbesetzt. Der Aufruf des Makros **VSVI1** erfolgte vor dem Nachladen von PROGB. Die ersten 8 Byte zeigen den Namen PROGA des ersten Programmabschnitts. Es folgt die Ladeadresse X'00000000' und die Länge X'0000037C' (892 Byte). Die nächsten beiden Byte zeigen Typ und Attribut des Programmabschnitts. Typ X'F0' (CSECT) und Attribut X'20' (AMODE=31). Die nachfolgenden Werte X'0000' dienen der Ausrichtung. Anschließend folgt der 16 Byte lange Kontextname (LOCAL#DEFAULT). Die folgenden Felder enthalten den „Leereintrag“ für Name (X'40....40'), Ladeadresse (X'00000000'), Länge (X'FFFFFFFF'), Attribut (X'C5') und Kontextnamen (X'40....40').

(6) Nach dem Laden von PROGB wird der Programmlauf in PROGB fortgesetzt. Der Inhalt des Basisregisters wird ausgegeben. 31-Bit-Adressierungsmodus ist eingestellt. Die Ladeadresse liegt oberhalb 16MB. Nach PROGB wird PROGA fortgesetzt.

(7) Nach dem Nachladen von PROGB stehen jetzt in ADR1 mehrere DBL-Einträge. Zuerst erfolgte der Eintrag für PROGB und den ENTRY in PROGB, anschließend der Eintrag für PROGA. Die Ladeadresse von PROGB ist X'01000000', die Länge X'00000082' (130 Byte). Der Typ ist X'F0' (CSECT) und die Attribute sind X'68' (AMODE=ANY und PAGE). Der ENTRY-Name ist C'ENTR', Typ X'F1' (ENTRY), Attribut X'60' (AMODE=ANY). Danach folgt der 16 Byte lange Kontextname (LOCAL#DEFAULT). Anschließend folgt der Eintrag für PROGA und abschließend der Leereintrag.

VTCSET – Logische Steuerzeichen definieren

Allgemeines

Anwendungsgebiete: Abfragen und Zugriff zu Listen und Tabellen; siehe Seite 158
Verkehr mit Datenstationen; siehe Seite 164
Makrotyp: O-Typ; siehe Seite 28

- Stand der Beschreibung: VTSU V13.3A

Makrobeschreibung

Durch den Makroaufruf **VTCSET** werden symbolische Namen generiert, mit denen logische Steuerzeichen in Line-Mode-Ausgabenachrichten eingefügt bzw. Line-Mode-Eingaben aufgefunden werden können.

Makroaufrufformat und Operandenbeschreibung

VTCSET
prefix

prefix
gibt die Zeichenfolge an, die den symbolischen Namen vorangestellt wird. Sie darf bis zu 5 Zeichen lang sein.

In den nachfolgenden Beschreibungen wird an Stelle der Zeichenfolge, die den symbolischen Namen vorangestellt wird, „&P.“ als Präfix gesetzt.

1. Logische Satzsteuerzeichen

&P.NL
Logisches Zeilenende (New Line)

Wirkung bei der Ausgabe:
– Besondere Anzeigeformen an der Datenstation werden, bis auf den Zeichensatz, auf die Standard-Anzeigeform zurückgesetzt (normal, Standard-Farbe, ungeschützt im Line-Modus, geschützt im Extended-Line-Modus). Wird die Nachricht im Extended-Line-Modus ausgegeben, so ist der Zeilenrest unsichtbar und geschützt.
– Das definierte logische Zeilenendezeichen wird ausgegeben (außer im Extended-Line-Modus).

– Die nächste Zeile wird in den Standardzustand gesetzt: Normal, halbhell, Standard-Farbe, Standard-Zeichensatz, ungeschützt im Line-Modus, geschützt im Extended-Line-Modus. Der Standard-Zeichensatz ist der Zeichenvorrat 0 bei Datensichtstationen vom Typ 9763 und der 1.Zeichenvorrat bei den übrigen Datensichtstationen und bei Druckern.
  Der Standardzustand wird nicht eingestellt, wenn für die Datenstation Betriebsart 2 und HOM=YES festgelegt wurde.
– Falls die Fortsetzung der Ausgabe einen Datenüberlauf an der Datenstation verursachen würde, wird die definierte Überlaufkontrolle ausgeführt (außer im Extended-Line-Modus).
– Der Cursor wird an den Anfang der nächsten Zeile gesetzt.
– Bei feldorientierter Anzeige und strukturierter Ausgabe (Standard) wird ein Feldanfang generiert.

&P.NP
Logisches Seitenende (New Page)

Wirkung bei der Ausgabe:
– Besondere Anzeigeformen auf der Datenstation werden auf die Standard-Anzeigeform zurückgesetzt (normal, Standard-Zeichensatz, Standard-Farbe, ungeschützt im Line-Modus, geschützt im Extended-Line-Modus mit UPDATE=NO, ungeschützt im Extended-Line-Modus mit UPDATE=YES). Das Bildschirmformat wird auf die Größe 24x80 zurückgesetzt. Der Standard-Zeichensatz ist der Zeichenvorrat 0 bei Datensichtstationen vom Typ 9763 und der 1.Zeichenvorrat bei den übrigen Datensichtstationen und bei Druckern.
– Das definierte logische Zeilenendezeichen wird ausgegeben.
– Der Hardcopy-Druck wird ausgelöst (bei HCOPY=YES).
– Die definierte Überlaufkontrollaktion wird ausgeführt.
– Eine neue Seite wird eingerichtet (bei Sichtstationen wird der Bildschirm gelöscht und das Bildschirmformat 24x80 eingestellt, bei Druckern wird ein Papiervorschub ausgelöst). Falls in der gleichen Nachricht schon Seitenvorschübe mittels ASF (Automatic Sheet Feeding) vorgenommen wurden, wird NP durch das zuletzt verwendete ASF ersetzt.
– Der Cursor wird an den Zeilenanfang gesetzt.
– Bei feldorientierter Anzeige und strukturierter Ausgabe (Standard) wird ein Feldanfang generiert.

&P.CL
Logisches Satzende (Current Line)

Wirkung bei der Ausgabe (nur bei Druckern und Fernschreibern):
– Besondere Anzeigeformen an der Datenstation werden auf den Normalzustand zurückgesetzt (Normal, Standard-Zeichensatz, ungeschützt; der Standard-Zeichensatz ist der 1.Zeichenvorrat bei den betroffenen Geräten).
– Das definierte logische Zeilenendezeichen wird ausgegeben.
– Der Cursor wird an den Anfang der aktuellen Zeile gesetzt.

&P.VPAddd
(nur Datensichtstation, für Drucker siehe Seite 1039) Positionierung auf das erste ungeschützte Feld einer Zeile (Vertical Position Absolute) (binäre oder dreistellige dezimale Angabe)

Wirkung bei der Ausgabe:
Absolute Zeilenpositionierung auf das erste ungeschützte Feld der Zeile ddd der Datensichtstation. Bei der Kombination mit HPA wird gleichzeitig auf die absolute Spalte positioniert.

Hinweis

- Ist ddd Null oder größer als 255, wird für VPA das Ersatzzeichen (SUB) eingesetzt und ddd ausgegeben.
- Ist ddd kleiner oder gleich 255, jedoch größer als die maximale Zeilenanzahl, wird statt VPA die Funktion NL ausgeführt.

&P.HPAddd
(nur Datensichtstationen, für Drucker siehe Seite 1039) Positionieren auf Spalte (Horizontal Position Absolute) (dreistellige dezimale Angabe)

HPA wird nur im Extended-Line-Modus bearbeitet und wenn es direkt nach einem gültigen VPA angegeben wird. HPA gibt die absolute Spalte der Zeile an, die durch das vorangegangene VPA festgelegt wurde. Wenn das vorangegangene Steuerzeichen VPA keinen gültigen Wert hat, oder HPA nicht direkt nach VPA angegeben wird, wird HPA ddd ignoriert. Bei 3270-Datenstationen wird auf den Anfang des folgenden ungeschützten Feldes positioniert.

Wirkung bei der Ausgabe:
Absolute Spaltenpositionierung in der Zeile, die durch das vorangegangene VPA festgelegt wurde.

Hinweis

- Ist ddd Null oder größer als 255, wird für HPA das Ersatzzeichen (SUB) eingesetzt und ddd ausgegeben.
- Ist ddd kleiner oder gleich 255, jedoch größer als die maximale Spaltenanzahl, wird HPA ddd ignoriert und nur die Funktion VPA ausgeführt.

2. Logische Anzeigesteuerzeichen

&P.EM1
Hervorgehobene Anzeige 1 (Emphasized Layout 1)

Wirkung bei der Ausgabe:
Die nachfolgenden Textzeichen werden gerätespezifisch hervorgehoben (siehe Tabelle auf Seite 1044) an der Datenstation abgebildet.

Die hervorgehobene Anzeige wird zurückgesetzt durch:
– die logischen Satzsteuerzeichen (NL, NP, HPA, VPA)
– die Anzeigesteuerzeichen (EM2, EM3, EM4, DAR, DIS, NOR)
– die Feldsteuerzeichen (EPA, SPA, NUM, CHS, COL, FLD)

&P.EM2
Hervorgehobene Anzeige 2 (Emphasized Layout 2)

Wirkung bei der Ausgabe:
Die nachfolgenden Textzeichen werden gerätespezifisch hervorgehoben (siehe Tabelle auf Seite 1044) an der Datenstation abgebildet.

Die hervorgehobene Anzeige wird zurückgesetzt durch:
– die logischen Satzsteuerzeichen (NL, NP, HPA, VPA)
– die Anzeigesteuerzeichen (EM1, EM3, EM4, DAR, DIS, NOR)
– die Feldsteuerzeichen (EPA, SPA, NUM, CHS, COL, FLD)

&P.EM3
Hervorgehobene Anzeige 3 (Emphasized Layout 3)

Wirkung bei der Ausgabe:
Die nachfolgenden Textzeichen werden hervorgehoben an der Datenstation abgebildet (siehe Tabelle auf Seite 1044).

Die hervorgehobene Anzeige wird zurückgesetzt durch:
– die logischen Satzsteuerzeichen (NL, NP, HPA, VPA)
– die Anzeigesteuerzeichen (EM1, EM2, EM4, DAR, DIS, NOR)
– die Feldsteuerzeichen (EPA, SPA, NUM, CHS, COL, FLD)

&P.EM4
Hervorgehobene Anzeige 4 (Emphasized Layout 4)

Wirkung bei der Ausgabe:
Die nachfolgenden Textzeichen werden hervorgehoben an der Datenstation abgebildet (siehe Tabelle auf Seite 1044).

Die hervorgehobene Anzeige wird zurückgesetzt durch:
– die logischen Satzsteuerzeichen (NL, NP, HPA, VPA)
– die Anzeigesteuerzeichen (EM1, EM2, EM3, DAR, DIS, NOR)
– die Feldsteuerzeichen (EPA, SPA, NUM, CHS, COL, FLD)

&P.NOR
Normale Anzeige (Normal Layout)

Wirkung bei der Ausgabe:
Die nachfolgenden Textzeichen werden in normaler Darstellung an der Datenstation abgebildet (Rücksetzen der hervorgehobenen Anzeige).

&P.SO
Umschalten des Zeichenvorrates entsprechend der Tabelle Seite 1047 (Shift Out Into Character Set Extension).

Wirkung bei der Ausgabe nur bei bestimmten Datenstationen (siehe Tabelle Seite 1047).

Die nachfolgenden Textzeichen werden in dem gerätespezifischen Zeichenvorrat laut der Tabelle Seite 1047 an der Datenstation abgebildet.

Bedeutung bei der Eingabe:
Die nachfolgenden Textzeichen wurden im datenstationsspezifischen Zeichenvorrat eingegeben.

&P.SI
Zurückschalten des Zeichenvorrates (Shift Into Basic Character Set) entsprechend der Tabelle Seite 1047.

Wirkung bei der Ausgabe:
Die nachfolgenden Textzeichen werden im Grundzeichenvorrat an der Datenstation abgebildet.

Bedeutung bei der Eingabe:
Die nachfolgenden Textzeichen sind aus dem Grundzeichenvorrat.

&P.DAR
Unsichtbare Anzeige (Dark Layout)

Beachten Sie, dass dieser Operand nur noch aus Kompatibilitätsgründen unterstützt wird. Benutzen Sie stattdessen die Operanden &P.EXT DIS und &P.EXT FLD.

Abhängig vom Wert des Betriebsparameters DARPRINTABLE ist DAR entweder ein Feld oder ein logisches Anzeigesteuerzeichen.

Wirkung bei der Ausgabe:
Der Betriebsparameter DARPRINTABLE hat den Wert „N“ (Standard).
Bei feldorientierter Anzeige und strukturierter Ausgabe generiert DAR einen Feldanfang. Die nachfolgenden Textzeichen werden „nicht sichtbar“ dargestellt und können nicht ausgedruckt werden. Das Feld entfällt bei einer Hardcopy.

DAR wird zurückgesetzt durch die logischen Satzsteuerzeichen (NL, NP, HPA, VPA) oder durch die Feldsteuerzeichen (EPA, NUM, CHS, COL und FLD).

Der Betriebsparameter DARPRINTABLE hat den Wert „Y“.
Die nachfolgenden Textzeichen werden „nicht sichtbar“ dargestellt.
Es wird kein Feldanfang generiert.

DAR wird zurückgesetzt durch:

– die logischen Satzsteuerzeichen (NL, NP, HPA, VPA)
– die Anzeigesteuerzeichen (EM1, EM2, EM4, NOR)
– die Feldsteuerzeichen (EPA, SPA, NUM, CHS, COL, FLD)

&P.EXT DIS x
(nur Datensichtstationen)
Setzen von Ausgabeattributen (Set Display Attributes)

Wirkung bei der Ausgabe:
Die nachfolgenden Textzeichen werden gerätespezifisch hervorgehoben an der Datensichtstation abgebildet. Bei den Datensichtstationen 8110, 815x, 816x, 974x, 975x und 3270 wird der inverse Modus ignoriert. Bei der Farbdatensichtstation 9763 werden, abhängig von SIDATA, die Attribute blinkend, unterstrichen, halbhell und die Kombinationen davon in eine Farbauswahl übertragen.

x ist ein hexadezimaler Wert, der über die folgenden Equates auswählt werden kann:

&P.FL	blinkend
&P.UND	unterstrichen/kursiv (siehe EM2)
&P.BLK	nicht sichtbar
&P.RIN	halbhell
&P.INV	invers
&P.RS	die Attribute FL, UND, BLK, RIN und INV werden zurückgesetzt. Beachten Sie, dass RS anders als NOR wirkt. Bei NOR wird automatisch das Attribut „halbhell“ eingestellt.

x hat entweder den Wert von einem dieser Equates oder von der Summe verschiedener Equates.

z.B. `&P.EXT DIS &P.UND+&P.FL text` der nachfolgende Text ist unterstrichen und blinkt

Wird eine Charakteristik nicht durch die Datenstation unterstützt, wird sie ignoriert.

Die hervorgehobene Anzeige wird zurückgesetzt durch:
– die logischen Satzsteuerzeichen (NL, NP, HPA, VPA)
– die Anzeigesteuerzeichen (EM1, EM2, EM3, EM4, DAR, DIS, NOR)
– die Feldsteuerzeichen (EPA, SPA, NUM, CHS, COL, FLD)

3. Logische Feldsteuerzeichen

&P.SPA
Beginn geschützter Bereich (Start Protected Area)

Wirkung bei der Ausgabe (nur bei Datensichtstationen):
Bei feldorientierter Anzeige und strukturierter Ausgabe generiert SPA einen Feldanfang. Die nachfolgenden Textzeichen werden am Bildschirm der Datenstation halbhell und geschützt ausgegeben, d.h., sie können nicht überschrieben und zur DVA zurückübertragen werden.

SPA wird zurückgesetzt durch die logischen Satzsteuerzeichen (NL, NP, HPA, VPA) oder durch die Feldsteuerzeichen (EPA, NUM, CHS, COL und FLD).

Hinweis

Diese Funktion verändert stark die Anzeigeeigenschaft an einigen Datenstationen (Zeilenwechsel bei der DSS 8152, Hardcopyfunktion bei den Datenstationen 816x, 975x und 976x). Sie erfordert daher größte Vorsicht bei der Anwendung.

&P.EPA
Ende geschützter Bereich (End Protected Area)

Wirkung bei der Ausgabe:
Bei feldorientierter Anzeige und strukturierter Ausgabe generiert EPA einen Feldanfang. Die nachfolgenden Textzeichen werden ungeschützt an die Datenstation ausgegeben und hell dargestellt.

EPA wird zurückgesetzt durch die logischen Satzsteuerzeichen (NL, NP, HPA, VPA) oder durch die Feldsteuerzeichen (EPA, NUM, CHS, COL und FLD).

&P.NUM
Numerischer Bereich (Numeric Area)

Wirkung bei der Ausgabe:

Bei einer feldorientierten Anzeige und einer strukturierten Ausgabe generiert NUM einen Feldanfang. Die nachfolgenden Textzeichen werden hell und ungeschützt ausgegeben. In dieses Feld können Sie nur numerische Daten (Zahlen , . * / + -) eingeben.

NUM wird zurückgesetzt durch die logischen Satzsteuerzeichen (NL, NP, HPA, VPA) oder durch die Feldsteuerzeichen (EPA, SPA, CHS, COL und FLD).

&P.CHS dd
Ladbarer Zeichensatz (Loadable Character Set)
CHS wirkt nur bei Datensichtstationen vom Typ 9763.

Wirkung bei der Ausgabe:
Bei einer feldorientierten Anzeige und einer strukturierten Ausgabe generiert CHS einen Feldanfang und es wird ein ladbarer Zeichensatz der Datensichtstation für dieses Feld ausgewählt. dd ist eine zweistellige Dezimalzahl im Bereich 00-07, mit der der gewünschte ladbare Zeichensatz ausgewählt wird. Über **TSTAT** können Sie abfragen, welche Zeichensätze belegt sind.

Beispiel
Geben Sie für dd 00 an, sprechen Sie den ladbaren Zeichensatz 0 an, der im **DCSTA** den symbolischen Namen STACS0T (siehe MONCS) hat.

In das generierte Feld können nur Zeichen aus dem ausgewählten Zeichensatz eingegeben werden.

CHS wird zurückgesetzt durch:
– die logischen Satzsteuerzeichen (NL, NP, HPA, VPA)
– die Anzeigesteuerzeichen (EM1, EM2, EM3, EM4, DAR, NOR, DIS)
– die Feldsteuerzeichen (EPA, SPA, NUM, COL, FLD)

Hinweis
Der Anwender muss dafür sorgen, dass die Zeichensätze der Datensichtstation, die er über das Steuerzeichen CHS logisch ansprechen will, mit den richtigen Zeichenvorräten geladen sind.
In der Systemzeile wird CHS nicht unterstützt.
Die Zeichensätze können physikalisch im physikalischen Modus oder mit EXT TRA im Line-Modus geladen werden. Zeichensätze können mit dem Software-Produkt ICE erstellt werden.

&P.COL dd
Farbenauswahl (Choice of Colors)
COL wirkt nur bei der Farbdatensichtstationen 9763.

Wirkung bei der Ausgabe:
Bei einer feldorientierten Anzeige und einer strukturierten Ausgabe generiert COL einen Feldanfang (auch bei Monochrom-Bildschirmen). Ebenso wird mit COL eine auf der Farbdatensichtstation 9763 verfügbare Farbe für dieses Feld ausgewählt. dd ist eine zweistellige Dezimalzahl im Bereich 00-07, mit der die gewünschte Farbe ausgewählt wird.

00	Standardwert	01	blau	02	rot
03	magenta	04	grün	05	cyan
06	gelb	07	weiß		

Beachten Sie, dass COL in der Systemzeile nicht unterstützt wird.

COL (die Farbauswahl) wird zurückgesetzt durch:
– die logischen Satzsteuerzeichen (NL, NP, HPA, VPA)
– die Anzeigesteuerzeichen (EM1, EM2, EM3, EM4, DAR, NOR, DIS)
– die Feldsteuerzeichen (EPA, SPA, NUM, CHS, FLD)

&P.EXT DIM zz sss
Physikalisches Bildschirmformat (Physical Screen Dimension)
EXT DIM wirkt nur bei Datensichtstationen vom Typ 9763 und nur, wenn es direkt nach NP angegeben wird.

Wirkung bei der Ausgabe:
Mit EXT DIM kann bei Datensichtstationen vom Typ 9763 das Bildschirmformat ausgewählt werden. Das Standardformat ist 24 x 80. Zusätzlich gibt es noch die Formate 32 x 80, 43 x 80 und 27 x 132. Die möglichen Bildschirmformate können mit **TSTAT** abgefragt werden.

Die Anzahl der Zeilen zz wird in zwei Bytes angegeben. Die Anzahl der Spalten sss wird in drei Bytes angegeben.

Beim Einschalten des Geräts und beim Verbindungsaufbau wird automatisch das Standardformat eingestellt. Das Bildschirmformat wird auf das Standardformat zurückgesetzt bei NP, Programmende und wenn der Bildschirm vom System gelöscht wird.

Bei Programmunterbrechung (K2, BKPT) im Line-Modus wird das Bildschirmformat nicht auf das Standardformat zurückgesetzt.

In der Systemzeile wird CHS nicht unterstützt.

&P.EXT FLD x
(nur Datensichtstationen)
Setzen von Feldattributen (Set Field Characteristics)

Wirkung bei der Ausgabe:
Bei einer feldorientierten Anzeige und einer strukturierten Ausgabe wird ein Feldanfang generiert und dem neuen Feld werden bestimmte Feldeigenschaften zugeordnet. Standardmäßig ist dieses neue Feld nicht druckbar. Eine vorangegangene hervorgehobene Anzeige wird für ein geschütztes Feld auf halbhell zurückgesetzt und für alle anderen Felder auf hell gesetzt.

x ist ein hexadezimaler Wert, der über die folgenden Equates auswählt werden kann:

&P.PNS	geschütztes Feld nicht übertragbar
&P.PRS	geschütztes Feld übertragbar (automatische Eingabe)
&P.NUF	numerisches Feld
&P.MOD	vormodifiziertes Feld (nur im Extended-Line-Modus)
&P.MAR	markierbares Feld
&P.PRT	druckbares Feld
&P.INP	ungeschütztes Eingabe-Feld nicht numerisch, nicht markierbar, nicht druckbar
&P.ASK	geschütztes Feld mit automatischem Tabulatorsprung (nur für DSS 3270)

x hat entweder den Wert von einem dieser Equates oder von der Summe verschiedener Equates.

z.B. `&P.EXT.FLD &P.NUF+&P.MAR  Feld` das Feld ist numerisch und markierbar

Beim Kombinieren von Feldeigenschaften bleibt das Ausgabeattribut „halbhell“ erhalten. Andere Ausgabeattribute müssen Sie über das Steuerzeichen DIS anfordern. Dabei müssen die Feldeigenschaften immer vor den Ausgabeattributen gesetzt werden.

FLD wird zurückgesetzt durch die logischen Satzsteuerzeichen (NL, NP, HPA, VPA) oder durch die Feldsteuerzeichen (EPA, SPA, NUM, CHS, COL und FLD).

In der Systemzeile wird FLD nicht unterstützt.

Beachten Sie, dass ein markierbares Feld für die Anwendung nur von Bedeutung ist, wenn der VTSUCB-Parameter READ=MODIFIED benutzt wird.

Werden einem Feld die Feldattribute geschützt, nicht sendbar und markierbar zugeordnet, entsprechen diese Feldattribute dem FHS-Attribut `PROTECTION=DETECTABLE` (geschütztes Auswahlfeld). Wird ein solches Feld im Lese-Modus markiert, erhält der Anwender nur die Feldpositionen VPA und HPA und nicht den Feldinhalt. Durch erneutes Drücken der Taste MAR können Sie das Attribut „markiert“ dieses Feldes zurücknehmen.

4. Logische lokale Steuerzeichen

&P.LOC stz
Erzeugen lokaler Attribute (Set Local Attributes)
LOC wirkt nur bei Datensichtstationen vom Typ 9763.

Wirkung bei der Ein- und Ausgabe:
Das Attribut, das mit dem direkt auf LOC folgenden Steuerzeichen stz angesprochen wird, wird nur lokal eingerichtet. stz kann sein: EM1-EM4, NOR, DAR, DIS, CHS und COL.

Ein lokales Attribut gilt für alle Folgezeichen bis zum nächsten Feldanfang, wenn es vorher nicht explizit zurückgesetzt wird. Das Rücksetzen erfolgt mit LOX (siehe Seite 1036). Lokale Attribute sind zeichengebunden. Mit dem Zeichen verschwindet auch das lokale Attribut, ein eingefügtes Zeichen trägt das lokale Attribut nicht, während die nach rechts wandernden Zeichen ihr lokales Attribut mitnehmen.

Für Datensichtstationen, die nicht vom Typ 9763 sind, und für Drucker gilt folgende Ersatzabbildung:
Die lokalen Anzeigeattribute LOC EM1- LOC EM4, LOC NOR, LOC DIS und LOC DAR werden durch die entsprechenden Anzeigeattribute EM1-EM4, NOR, DIS und DAR ersetzt. Für lokale Zeichensatzattribute (LOC CHS) und Farbattribute (LOC COL) wird keine Ersatzabbildung durchgeführt.

Bei LOC DAR können die nachfolgenden, an der Datensichtstation nicht sichtbaren Textzeichen ausgedruckt werden. Bei der Ersatzabbildung bestimmt der Wert des Betriebsparameters DARPRINTABLE, ob nicht sichtbare Textzeichen ausgedruckt werden können.

Lokale Attribute können nur eingegeben werden, wenn im VTSUCB die Option LOCIN=YES gesetzt wurde. Ansonsten werden sie aus der Eingabenachricht entfernt. Lokale Attribute, die von VTSU nicht logisch unterstützt werden, werden immer aus der Eingabenachricht entfernt.

Hinweis

Die logischen Anzeigesteuerzeichen EM1-EM4 und NOR, die nach LOC gefolgt von EM1-EM4, NOR und DAR stehen, kommen erst zur Wirkung, wenn das lokale Attribut zurückgesetzt wird (durch LOX NOR bzw. LOX LOX). Diese Anzeigesteuerzeichen wirken auch für Zeichen, die überschrieben oder eingefügt werden.

&P.LOX stz
Rücksetzen lokaler Attribute (Reset Local Attributes)

Wirkung bei der Ein- und Ausgabe:
Mit LOX werden lokale Attribute wieder zurückgesetzt.
stz gibt an, welche lokalen Attribute zurückgesetzt werden.
stz kann sein:
- NOR (lokales EM1, EM2, EM3, EM4, DIS, NOR oder DAR wird zurückgesetzt)
- CHS (lokaler ladbarer Zeichenvorrat wird zurückgesetzt)
- COL (Farbauswahl wird zurückgesetzt)
- LOX (alle lokalen Attribute werden zurückgesetzt).

LOX setzt das lokale Attribut auf den zuletzt gültigen nicht-lokalen Wert zurück.

Für Datensichtstationen, die nicht vom Typ 9763 sind, und Drucker gilt folgende Ersatzabbildung:
LOX NOR und LOX LOX setzen auf das letzte Anzeigeattribut zurück, das nicht durch eine Ersatzabbildung zu Stande kam. Für lokale Zeichensatzattribute (LOX CHS) und lokale Farbattribute (LOX COL) wird keine Ersatzabbildung durchgeführt.

Lokale Attribute können nur empfangen werden, wenn im VTSUCB die Option LOCIN=YES gesetzt wurde. Ansonsten werden sie aus der Eingabenachricht entfernt. Lokale Attribute, die von VTSU nicht logisch unterstützt werden, werden immer aus der Eingabenachricht entfernt.

Hinweise

- Bei EXTEND=NO wirkt ein als letztes Zeichen angegebenes logisches Anzeigesteuerzeichen (EM1, EM2, EM3, EM4, DIS, DAR oder NUM) auf das nachfolgende Eingabefeld. Dadurch ist es möglich, eine normale Line-Modus-Eingabe dunkel, numerisch oder hervorgehoben zu steuern. Wenn der Ausgabe keine Eingabe sondern eine weitere Ausgabe folgt, hat das Steuerzeichen keine Wirkung.
- Wenn für ein Feld verschiedene Eigenschaften angefordert werden, geben Sie sie folgender Reihenfoge an: &P.CHS dd &P.COL dd &P.EXT &P.FLD xx &P.EXT &P.DIS xx. In jedem Fall müssen die logischen Anzeigesteuerzeichen den logischen Steuerzeichen für die Feldeigenschaften folgen. Beachten Sie, dass wenn Sie das Steuerzeichen &P.COL benutzen, bei Farbdatensichtstationen die anderen Anzeigeattribute ig-

noriert werden. Physikalische Sequenzen werden weiterhin korrekt generiert. Ebenso werden für Farb- und Monochrom-Bildschirme des Typs 9763 der gleiche logische Puffer benutzt.

– Wenn die zwei logischen Steuerzeichen VPA und HPA kombiniert werden, werden abhängig von der Lage der Steuerzeichen im Puffer, unterschiedliche physikalische Sequenzen generiert.
  – Stehen die Steuerzeichen am Anfang der Nachricht, wird auf die angeforderte Stelle positioniert und ein Feldanfang mit Standard-Attributen generiert.
  – Stehen die Steuerzeichen am Ende der Nachricht, wird auf die angeforderte Stelle positioniert und spezielle Anzeigeformate der Datensichtstation werden auf Standard-Anzeige zurückgesetzt (normal, Standard-Farbe, geschützt). Der Standard-Zeichensatz bleibt erhalten.
  – Stehen die Steuerzeichen innerhalb der Nachricht, wird auf die angeforderte Stelle positioniert und ein Feldanfang mit Standard-Attributen generiert. Spezielle Anzeigeformate der Datensichtstation werden auf die Standard-Anzeige zurückgesetzt (normal, Standard-Farbe, ungeschützt im Line-Modus, geschützt im Extended-Line-Modus). Der Standard-Zeichensatz bleibt erhalten.
– Die Hardcopy-Unterstützung im Line-Modus und im Extended-Line-Modus ist unterschiedlich. Im Extended-Line-Modus wird vom gesamten Bildschirm ein Hardcopy gemacht. Im Line-Modus wird nur vom letzten Feld ein Hardcopy gemacht, das durch ein vorangegangenes Steuerzeichen generiert wurde.
– Felder die mit dem Steuerzeichen EXT FLD erzeugt wurden, müssen als druckbares Feld festgelegt werden.

5. Logische Steuerzeichen zur Druckerunterstützung

Die logischen Steuerzeichen zur Druckerunterstützung sind nur bei EXTEND=NO sinnvoll zu verwenden.

&P.PLD
Vorrücken um halbe Zeile (Partial Line Down)

Wirkung bei der Ausgabe:
Der Drucker wird um eine halbe Zeile vorgerückt, der nachfolgende Text eine halbe Zeile tiefer gedruckt. Rücksetzen durch PLU, am logischen Zeilenende oder am Nachrichtenende.

&P.PLU
Rücksetzen um halbe Zeile (Partial Line Up)

Wirkung bei der Ausgabe:
Der Drucker wird eine halbe Zeile zurückgesetzt, der nachfolgende Text eine halbe Zeile höher gedruckt. Rücksetzen durch PLD, am logischen Zeilenende oder am Nachrichtenende.

&P.VMI d
Zeilenabstand festlegen (Vertical Motion Index)
VMI ist nur am logischen Seitenanfang zulässig.

Wirkung bei der Ausgabe:
Legt den Abstand der Zeilen voneinander fest:
– Für d=1 normaler Abstand (1/6 Zoll).
– Für d=2 enger Abstand (1/8 Zoll).
– Für d=3 Halbzeilenabstand (1/12 Zoll).

Wird durch MLN und am Nachrichtenende zurückgesetzt. Die Anzahl der Zeilen pro Seite wird dem neuen Zeilenabstand angepasst.

&P.HMI d
Zeichenabstand festlegen (Horizontal Motion Index)
HMI ist nur am logischen Zeilenanfang zulässig. Ein über LM eingestellter linker Rand wird zurückgesetzt.

Wirkung bei der Ausgabe:
Legt den Abstand der Zeichen voneinander fest:
– Für d=1 normaler Zeichenabstand (1/10 Zoll).
– Für d=2 Schmalschrift A (Zeichenabstand 1/12 Zoll).
– Für d=3 Schmalschrift B (Zeichenabstand 1/15 - 1/17 Zoll).

Wird durch MLL und am Nachrichtenende zurückgesezt. Die Anzahl der Zeichen pro Zeile wird automatisch angepasst.

&P.NLQ
Schönschrift einschalten (Near Letter Quality Start)

Wirkung bei der Ausgabe:
Wenn ein Drucker die Funktion Near Letter Quality besitzt, wird diese mit NLQ eingeschaltet, ansonsten wird NLQ ignoriert. NLQ wird am Nachrichtenende zurückgesetzt.

&P.NLX
Schönschrift ausschalten (Near Letter Quality Exit)

Wirkung bei der Ausgabe:
Das Ausgabesteuerzeichen NLX schaltet die Funktion Near Letter Quality aus.

&P.LM ddd
Linker Rand (Left Margin)

Wirkung bei der Ausgabe:
Setzt einen linken Rand. Das erste Zeichen aller nachfolgenden Zeilen wird in der durch ddd angegebenen Spalte gedruckt.

LM ist nur zulässig am logischen Zeilenanfang und am Nachrichtenbeginn, wenn dort kein CAP steht. Es wird durch HMI, MLL und am Nachrichtenende zurückgesetzt.

Bei gesetztem linken Rand sind auch mehrere Rückwärtsschritte (BS) hintereinander erlaubt. Damit kann vor den linken Rand positioniert werden.

&P.PTS
Proportionalschrift einschalten (Proportional Type Start)

Wirkung bei der Ausgabe:
Durch eine individuelle Zeichenbreite wird ein schöneres Schriftbild erzeugt.

PTS wirkt bis zum Nachrichtenende oder PTX.

Die Spaltenzählung wird ausgesetzt und erst durch ein auf PTX folgendes NL, NP, ASF, VPA oder HPA wieder eingesetzt.

&P.PTX
Proportionalschrift ausschalten (Proportional Type Exit):

Wirkung bei der Ausgabe:
Schaltet die Proportionalschrift aus.

&P.MLL ddd
Maximale Zeilenlänge (Maximal Line Length):

Wirkung bei der Ausgabe:
Durch ddd wird die maximale Anzahl der Zeichen pro Zeile neu definiert. Gleichzeitig wird der Zeichenabstand auf 1/10 Zoll eingestellt und ein gesetzter linker Rand zurückgesetzt. MLL wirkt nur am logischen Seitenanfang bei Druckern. Die vereinbarte maximale Zeilenlänge bleibt für die gesamte Verbindung erhalten, sofern sie nicht neu festgelegt wird.

&P.MLN ddd
Maximale Zeilenanzahl (Maximal Line Number):

Wirkung bei der Ausgabe:
Durch ddd wird die maximale Anzahl der Zeilen pro Seite neu definiert. Gleichzeitig wird der Zeilenabstand auf 1/6 Zoll eingestellt. MLN wirkt nur am logischen Seitenanfang bei Druckern. Die vereinbarte maximale Zeilenanzahl bleibt für die gesamte Verbindung erhalten, sofern sie nicht neu festgelegt wird.

&P.VPA ddd
Positionieren auf Zeilenanfang (Vertical Position Absolute) (dreistellige dezimale Angabe)

Wirkung bei der Ausgabe:
Der Ausgabetext beginnt in der durch ddd bezeichneten Zeile.

&P.HPA ddd
Positionieren auf Spalte (Horizontal Position Absolute) (dreistellige dezimale Angabe)

Wirkung bei der Ausgabe:
In der aktuellen Zeile werden die nachfolgenden Zeichen ab der durch ddd bezeichneten Spalte ausgegeben.

&P.ASF d
Steuerung des Blatteinzugs bzw. -auswurfs (Automatic Sheet Feeding)
d= Dezimalzahl

Wirkung bei der Ausgabe:
– Für d=0 am Nachrichtenende wird ein Blattauswurf bewirkt.
  Bei Vorsteckeinrichtung: Umschaltung auf Traktor.
– Für d=1,2,3 wird der Blattauswurf mit einem Blatteinzug aus dem durch d bezeichneten Einzugsbehälter kombiniert.
– Für d=9 erfolgt der Blatteinzug von der Vorsteckeinrichtung (nur beim Drucker 9013).

&P.CAP
Keine Positionierung auf den nächsten Zeilenanfang bei Nachrichtenbeginn (Continue Actual Position).

Wirkung bei der Ausgabe:
Beginn der Druckausgabe an der augenblicklichen Position des Druckwagens. Kein Rücksetzen von logischen Steuerzeichen am Nachrichtenende und am Nachrichtenbeginn. Außerdem sind mehrfach PLUs und PLDs erlaubt. Es wird aber jeweils nur ein PLU bzw. PLD am logischen Zeilenende zurückgesetzt.

Um von einem definierten Zustand ausgehen zu können, muss beim erstmaligen Verwenden von CAP innerhalb einer Ausgabesequenz eine Positionierung auf einen Zeilenanfang mittels NL, NP oder VPA vorgenommen werden. CAP ist nur als erstes Zeichen einer Nachricht zulässig. Bei der erstmaligen Verwendung von CAP oder nach Reset-Sequenzen (z.B. HMI-Reset) können Überschreibungen vorkommen, wenn nicht explizit auf einen neuen Zeilenanfang positioniert wird.

6. Logische Steuerzeichen mit besonderen Funktionen

&P.DEL
Löschzeichen (Delete)

Wirkung bei der Ausgabe:
Das Zeichen wird aus dem Ausgabetext entfernt und nicht an die Datenstation weitergeleitet.

&P.BS
Rückwärtsschritt (Backspace; bei DSS nur mit APL-Zusatz und Druckern):

Wirkung bei der Ausgabe:
Das nachfolgende Textzeichen wird über dem vorangegangenen abgebildet (Zusammensetzen eines nicht im Zeichenvorrat enthaltenen Zeichens).

Ein mehrfaches BS ist nur erlaubt, wenn zuvor mit LM ein linker Rand gesetzt wurde.

Bedeutung bei der Eingabe:
Das nachfolgende und das vorangegangene Textzeichen sollen als eine Einheit angesehen werden.

Hinweis

Beim Drucker 9022 wird BS (Rückwärtsschritt) nach SO (Umschalten des Zeichenvorrates) ignoriert.

&P.SUB
Ersatzzeichen (Substitute)

Wirkung bei der Ausgabe:
Dieses logische und sämtliche anderen EBCDIC-Steuerzeichen (Code < X'40'), welche keine logischen Steuerzeichen sind, werden mit dem gültigen Ersatzzeichen (Makro **TCHNG** SUB=OUTIN) an der Datenstation abgebildet.

Wirkung bei der Eingabe:
Das gültige Ersatzzeichen wurde in der Datenstationseingabe erkannt und ersetzt (nur wenn durch Makro **TCHNG** SUB=OUTIN gefordert; siehe Handbuch „TIAM“ [16]).

&P.ESC
Escape

Wirkung bei der Ausgabe:
Setzt das Steuerzeichen ESC ab (EBCDIC-Code X'27'). Dieses Steuerzeichen wird zusammen mit dem folgenden Zeichen unverändert übertragen. Man kann dadurch Gerätefunktionen der Datenstation im Line-Modus nutzen, die logisch nicht unterstützt werden (siehe hierzu Beschreibungen der entsprechenden Datenstationen).

Hinweis

Durch dieses Steuerzeichen wird die Spalten- und Zeilenzählung durch VTSU ausgesetzt (keine Überlaufkontrolle!).
Fortgesetzt wird die Spaltenzählung durch die Steuerzeichen NL, NP oder VPA und die Zeilenzählung (Überlaufkontrolle) durch die Steuerzeichen NP oder VPA.

&P.DC4
wird wie ESC von VTSU behandelt

Wirkung bei der Ausgabe:
Setzt das Steuerzeichen DC4 ab (EBCDIC-Code X'3C'). Dieses Steuerzeichen wird zusammen mit dem folgenden Zeichen an die Datenstation übertragen. Man kann dadurch Gerätefunktionen der Datenstation im Line-Modus nutzen, die logisch nicht unterstützt werden (siehe hierzu Beschreibungen der entsprechenden Datenstationen).

Wirkung auf Spalten- und Zeilenzählung wie im Hinweis unter Steuerzeichen ESC beschrieben.

&P.HT
Horizontaler Tabulatorsprung (Horizontal Tabulation)

Wirkung bei der Ausgabe:
Setzt das Steuerzeichen HT ab (EBCDIC-CODE X'05'), welches unverändert an die Datenstation weitergereicht wird (siehe hierzu Beschreibungen der entsprechenden Datenstationen).

Wirkung auf Spalten- und Zeilenzählung wie im Hinweis unter Steuerzeichen ESC beschrieben.

&P.VT
Vertikaler Tabulatorsprung (Vertical Tabulation)

Wirkung bei der Ausgabe:
Setzt das Steuerzeichen VT ab (EBCDIC-Code X'0B'), welches unverändert an die Datenstation weitergereicht wird (siehe hierzu Beschreibungen der entsprechenden Datenstationen).

Wirkung auf Spalten- und Zeilenzählung wie im Hinweis unter Steuerzeichen ESC beschrieben.

&P.EXT TRA d ll
Transparente Ausgabe (Transparent Output)

Wirkung bei der Ausgabe:
EXT TRA ermöglicht es dem Anwender, Steuerzeichenfolgen ohne Änderung (transparent) an Datensichtstationen und Drucker durchzureichen.
Mit d kann der Anwender einstellen, bei welchen Geräten die transparente Zeichenfolge durchgereicht wird:

d = X'00'
Die Steuerzeichenfolge wird bei allen Gerätetypen durchgereicht.

d = Gerätetyp, der im **TSTAT** geliefert wird:
Die Steuerzeichenfolge wird nur bei dem angegebenen Gerätetyp durchgereicht.

Mit ll gibt der Anwender die Länge der transparenten Steuerzeichenfolge an. ll wird in zwei Bytes entweder dezimal (00-99) oder binär (X'0000'-X'7FFF') angegeben. Die ersten fünf Bytes (EXT TRA d ll) werden dabei nicht mitgezählt. Wenn die Länge größer als der maximale Gerätepuffer ist, wird EXT in SUB umgewandelt.

Hinweis
Wenn Sie das logische Steuerzeichen EXT TRA benutzen, ist es möglich jedes physikalische Steuerzeichen anzugeben.
Wenn Sie über EXT TRA die Positionierung modifizieren wollen, wird die Verarbeitung des Überlaufs nicht mehr garantiert.

&P.EXT RPT dd
Das nachfolgende darstellbare oder NIL-Zeichen wird dd-mal wiederholt (Repeat Symbol)

Wirkung bei der Ausgabe:
EXT RPT ermöglicht es Ihnen ihre Puffer-Größe zu reduzieren, wenn ein Zeichen mehrfach wiederholt wird.
Mit dd legen Sie fest, wie oft das nachfolgende Zeichen wiederholt wird.
dd wird in zwei Bytes angegeben, entweder dezimal (00-99) oder binär (X'0000'-X'7FFF').

Hinweise zu den logischen Steuerzeichen

a) Ist ein Steuerzeichen für ein Ausgabegerät nicht zulässig oder an der verwendeten Stelle nicht anwendbar, wird es zusammen mit der evtl. zugehörigen nachfolgenden Nummer ignoriert oder ersetzt (siehe Beschreibung der einzelnen Steuerzeichen).

b) Sind die auf ein logisches Steuerzeichen, das eine nachfolgende Nummer erwartet, folgenden Bytes als Nummer unzulässig (keine oder unmögliche Nummer), dann wird das betreffende logische Steuerzeichen durch SUB ersetzt und die nachfolgenden Zeichen wie Text behandelt.

c) Ist die dem logischen Steuerzeichen nachfolgende Nummer nur im speziellen Fall unzulässig (z.B. für die augenblickliche Zeilenlänge zu groß), so wird wie unter a) beschrieben verfahren.

d) Die Steuerzeichen ASF, VPA und HPA wirken als logisches Zeilenende und haben auf die bisherigen Steuerzeichen die gleiche Wirkungsweise wie NL (Rücksetzen des 2. Zeichenvorrats usw.).

Wirkung logischer Anzeigesteuerzeichen bei Datenstationen

Daten-station	Wirkung logischer Anzeigesteuerzeichen					
	NOR	**EM1**	**EM2**	**EM3**	**EM4**	**DAR**
8110 TTY [1]	- [2]	-	-	-	-	-
8150	-	-	-	-	-	
8151	ruhig	blinkend	blinkend	blinkend	blinkend	-
8152	gerade	kursiv	kursiv	kursiv	kursiv	-
8160 … 8162 1.BA [3]	halbhell gerade ruhig	halbhell gerade blinkend	halbhell kursiv ruhig	hell gerade ruhig	hell kursiv ruhig	nicht sichtbar
8160 2. BA	hell gerade ruhig	hell gerade blinkend	hell kursiv ruhig	halbhell gerade ruhig	halbhell kursiv ruhig	nicht sichtbar
9748 9749 9750 9751 1.BA	halbhell gerade ruhig	halbhell gerade blinkend	halbhell unter-strichen ruhig	hell gerade ruhig	hell unter-strichen ruhig	nicht sichtbar
9748 9749 9750 9751 2.BA	hell gerade ruhig	hell gerade blinkend	hell unter-strichen ruhig	halbhell gerade ruhig	halbhell unter-strichen ruhig	nicht sichtbar
9752	gelb gerade	gelb blinkend	weiß gerade	grün gerade	rot gerade	nicht sichtbar
9755 9756 9758 1 .BA	halbhell gerade	halbhell gerade blinkend	halbhell unter-strichen	hell gerade	hell unter-strichen	nicht sichtbar
9755 9756 9758 2.BA	hell gerade	hell gerade blinkend	hell unter-strichen	halbhell gerade	halbhell unter-strichen	nicht sichtbar

Daten-station	Wirkung logischer Anzeigesteuerzeichen					
	NOR	**EM1**	**EM2**	**EM3**	**EM4**	**DAR**
9763 u. 9759 1. BA Monochrom Bildschirm	halbhell gerade	halbhell gerade blinkend	halbhell unter-strichen	hell gerade	hell unter-strichen	nicht sichtbar
9763 u. 9759 2. BA Monochrom Bildschirm	hell gerade	hell gerade blinkend	hell unter-strichen	halbhell gerade	halbhell unter-strichen	nicht sichtbar
9763 1.BA Farb-Bildschirm	gelb	cyan	weiß	grün	rot	nicht sichtbar
9763 2.BA Farb-Bildschirm	grün	rot	rot	gelb	weiß	nicht sichtbar
3270 1. BA	halbhell gerade	hell gerade	hell gerade	hell gerade	hell gerade	nicht sichtbar
3270 2. BA	hell gerade	halbhell gerade	halbhell gerade	halbhell gerade	halbhell gerade	nicht sichtbar
3279 1. BA	grün	rot	rot	rot	rot	blau
3279 2. BA	grün	weiß	weiß	weiß	weiß	grün
8121 8122	normal	kursiv	kursiv	kursiv	kursiv	-
9001	normal	unter-strichen	unter-strichen	unter-strichen	unter strichen	-
9002	normal	kursiv	unter-strichen	kursiv	kursiv unter-strichen	-
9003	normal	kursiv	rot	kursiv	rot und kursiv	-
9004	normal	Schatten-schrift	unter-strichen	fett	fett und unter-strichen	-
9013	normal	unter-strichen	unter-strichen	fett	fett und unter-strichen	-
9012	normal	unter-strichen	unter-strichen	fett	fett und unter-strichen	-
9011-18/19	normal	kursiv [4]	unter-strichen	fett	fett und unter-strichen	-

Daten-station	Wirkung logischer Anzeigesteuerzeichen					
	NOR	**EM1**	**EM2**	**EM3**	**EM4**	**DAR**
9001-31 / 8931	normal	kursiv [4]	unter-strichen	fett	fett und unter-strichen	-
9021	normal	kursiv	unter-strichen	fett	fett und unter-strichen	-
9022	normal	Schatten-schrift	unter-strichen	fett	fett und unter-strichen	-

[1] TTY: PT80, T100, T1000

[2] -: Steuerzeichen wird ignoriert

[3] BA: Betriebsart

[4] Nur bei Anschluss an eine DSS 9763 oder an eine BAM-Steuerung; ansonsten unterstrichen oder ohne Wirkung

Wirkung logischer Anzeigesteuerzeichen bei Datenstationen

Daten-station	Wirkung logischer Anzeigesteuerzeichen				
	SO	**SI**	**SPA**	**EPA**	**NUM**
8110 TTY [1]	- [2]	-	-	-	-
8150	-	-	geschützt halbhell	ungeschützt hell	-
8151	-	-	geschützt halbhell	ungeschützt hell	-
8152	APL-Zeichenvorrat	erster Zeichenvorrat	geschützt halbhell	ungeschützt hell	-
8160	-	-	geschützt halbhell	ungeschützt hell	ungeschützt hell [3]
8162	zweiter Zeichenvorrat	erster Zeichenvorrat	geschützt halbhell	ungeschützt hell	ungeschützt hell [3]
9748 9749 9750/9751	-	-	geschützt halbhell	ungeschützt hell	ungeschützt hell [3]
9752	-	-	geschützt gelb	ungeschützt grün	ungeschützt grün [3]
9755 9756 9758	-	-	geschützt halbhell	ungeschützt hell	ungeschützt hell [3]
9763 u.9759 Monochrom Bildschirm	-	-	geschützt halbhell	ungeschützt hell	ungeschützt hell [3]
9763 Farb-Bildschirm	-	-	geschützt gelb	ungeschützt grün	ungeschützt grün [3]
3270	-	-	geschützt halbhell	ungeschützt hell	ungeschützt hell [3]
3279 1. BA [4]	-	-	weiß	rot	rot
3279 2. BA	-	-	weiß	weiß	weiß
812x	-		-	-	-
9001	-	-	-	-	-
9002 [5]	zweiter Zeichenvorrat	erster Zeichenvorrat	-	-	-
9003	zweiter Zeichenvorrat	erster Zeichenvorrat	-	-	-

Daten-station	Wirkung logischer Anzeigesteuerzeichen				
	SO	**SI**	**SPA**	**EPA**	**NUM**
9004	zweiter Zeichenvorrat	erster Zeichenvorrat	-	-	-
9013 [5]	Zeichenvorrats erweiterung [6]	Grundein-stellung des Zeichen-vorrats [7]	-	-	-
9012	Zeichen-vorrats-erweiterung [6]	Grundein-stellung des Zeichen-vorrats [7]	-	-	-
9011-18/19	zweiter Zeichenvorrat	erster Zeichenvorrat	-	-	-
9001-31/8931	-	-	-	-	-
9021	sekundärer font	primärer font	-	-	-
9022	Zeichenvorrats erweiterung [8]	Grundein-stellung des Zeichen-vorrats [9]	-	-	-

[1] TTY: PT80, T100, T1000

[2] -: Steuerzeichen wird ignoriert

[3] BA: Betriebsart

[4] nur numerische Eingaben möglich

[5] abhängig vom Druckertyp (siehe entsprechendes Handbuch zur Druckerbeschreibung)

[6] rechte Hälfte der ISO-8-Bit-Codetabelle; sinnvoll z.B. bei Teletex-Zeichenvorrat

[7] linke Hälfte der ISO-8-Bit-Codetabelle

[8] rechte Hälfte der ISO-8-Bit-Codetabelle; sinnvoll z.B. bei Teletex-Zeichenvorrat

[9] linke Hälfte der ISO-8-Bit-Codetabelle

Anwendbarkeit der logischen Steuerzeichen bei den einzelnen Druckern und Datensichtstationen

Typ	PLD PLU	LM	PTS PTX	VPA	HPA	ASF	MLL	MLN	BS	CAP	ESC DC4	HT	VT
9004	X	X	X	X	X	X	X	X	X	X	X	X	X
9001	X	-	-	-	-	-	X	X	-	X	X	X	X
9003	-	-	-	-	-	-	X	X	X	X	X	X	X
9002	-	-	-	X	X	-	X	X	X	X	X	X	X
9013	X	X	X	X	X	X	X	X	X	X	X	X	X
9012	X	X	X	X	X	X	X	X	X	X	X	X	X
9011 18/19	X	-	X	-	X	X	X	X	X	X	X	X	X
9001-31/ 8931	X	-	X	-	X	-	X	X	X	X	X	X	X
9022	X	X	X	X	X	X	X	X	X	X	X	X	X
9021	X	-	X	X	X	X	X	X	X	X	X	X	X
812x	-	-	-	-	-	-	X	X	X	X	X	-	-
816x 975x 974x	-	-	-	X	*	-	-	-	-	-	X	X	X
976x	-	-	-	X	*	-	-	-	-	-	X	X	X
3270	-	-	-	X	*	-	-	-	-	-	X	X	X
3279	-	-	-	X	*	-	-	-	-	-	X	X	X

Bedeutung:

X Funktion wird ausgelöst
- das logische Steuerzeichen wird unterdrückt
* wenn ein gültiges Steuerzeichen VPA vorangestellt ist, wird das logische Steuerzeichen unterdrückt

Hinweise

– Der Drucker 9022 ignoriert Proportionalschrift, wenn die eingestellte Schriftart Proportionalschrift nicht unterstützt.
– Beim Drucker 9013 ist Proportionalschrift nur sinnvoll, wenn ein geeigneter Zeichenvorrat ausgewählt ist.
– Bei den Druckern 9013 und 9002 wird durch das Steuerzeichen VPAddd eine durch ddd festgelegte Anzahl von Zeilenvorschüben ausgeführt.

– Beim Drucker 9002 wird durch das Steuerzeichen HPAddd entweder eine durch ddd festgelegte Anzahl von Leerzeichen eingefügt, oder die Spalte festgelegt, ab der die nachfolgenden Zeichen ausgegeben werden sollen. Sie können zwischen beiden Möglichkeiten wählen. Das Einfügen der Leerzeichen ist Standard.

Anwendbarkeit der logischen Steuerzeichen für die Ausgabe

Daten-station	CHS	LOC LOX	EXT DIM	EXT TRA	NLQ NLX	EXT DIS	EXT FLD	COL	EXT RPT
9763	X	X	X	X	-	X	X	X	X
975x 9748 9749 816x 3270	-	[1]	-	X	-	X	X	-	X
9001 9002 9003 9004 9013	-	[1]	-	X	-	-	-	-	X
9012	-	[1]	-	X	-	-	-	-	X
9011-18/19	-	[1]	-	X	X	-	-	-	X
9001-31/8931	-	[1]	-	X	X	-	-	-	X
9022	-	[1]	-	X	-	-	-	-	X
9021	-	[1]	-	X	-	-	-	-	X

[1] Es wird eine Ersatzabbildung durchgeführt (siehe Steuerzeichen LOC und LOX)

Bedeutung:

X Funktion wird ausgelöst
- das logische Steuerzeichen wird ignoriert

Hinweis

Bei Datensichtstationen vom Typ 9763 werden bei dem logischen Steuerzeichen EXT DIM jeweils nur die Bildschirmformate unterstützt, die bei **TSTAT** geliefert werden.

Wirkung der Ausgabeattribute

Datenstation	FL	UND	BLK	RIN	INV
8110	- [1]	-	-	-	-
815x	-	-	-	-	-
8160	blinkend	kursiv	nicht sichtbar	halbhell	-
9750	blinkend	unterstrichen/ invers [2]	nicht sichtbar	halbhell	-
9755	blinkend	unterstrichen/ invers [3]	nicht sichtbar	halbhell	-
9758 9756	blinkend	unterstrichen invers [3]	nicht sichtbar	halbhell	invers
9763 9759	blinkend	unterstrichen invers [3]	nicht sichtbar	halbhell	invers
3270	-	-	nicht sichtbar und nicht druckbar	halbhell	-

[1] -: Steuerzeichen wird ignoriert

[2] wird über Schaltbrücke oder ROM gesetzt

[3] auswählbar über SIDATA

Wirkung der Feldattribute

Daten-station	PNS	PRS	NUF	MOD	MAR	PRT	ASK
8110	- [1]	-	-	-	-	-	-
815x	-	-	-	-	-	-	-
8160	geschützt nicht übertrag-bar	geschützt übertrag-bar	numerisch	vor-modifiziert	markier-bar	druckbar	-
9750	geschützt nicht übertrag-bar	geschützt übertrag-bar	numerisch	vor-modifiziert	markier-bar	druckbar	-
9755	geschützt nicht übertrag-bar	geschützt übertrag-bar	numerisch	vor-modifiziert	markier-bar	druckbar	-
9758 9756	geschützt nicht übertrag-bar	geschützt übertrag-bar	numerisch	vor-modifiziert	markier-bar	druckbar	-
9763 9759	geschützt nicht übertrag-	geschützt übertrag-bar	numerisch	vor-modifiziert	markier-bar	druckbar	-
3270	geschützt nicht übertrag-bar	-	numerisch	vor-modifiziert	auswähl-bar [2]	druckbar [3]	geschützt automat. Tabulator-sprung

[1] -: Steuerzeichen wird ignoriert

[2] das erste Zeichen des Feldes ist ein Bestimmungs-Zeichen

[3] bei der DSS 3270 ist ein nicht druckbares Feld automatisch nicht sichtbar

Zeichen- und Zeilenabstand bei den einzelnen Druckern in Zoll

Drucker	HMI1	HMI2	HMI3	VMI1	VMI2	VMI3
9004	1/10	1/12	1/15	1/6	1/8	1/12
9001	1/10	1/12	1/17	1/6	1/8	1/12
9003	1/10	1/12	1/15	1/6	1/8	1/12
9002	1/10	1/10	1/10	1/6	1/6	1/6
9013	1/10	1/12	1/15 nur beim 1. ZV	1/6	1/8	1/12
9012	1/10	1/12	1/12	1/6	1/8	1/12
9011 18/19	1/10	1/12	1/15	1/6	1/8	1/12
9001 31/8931	1/10	1/12	1/15	1/6	1/8	1/12
9022	1/10	1/12	1/15	1/6	1/8	1/12
9021	1/10	1/12	1/15	1/6	1/8	1/12
812x	1/10	1/10	1/10	1/6	1/6	1/6

Hinweise

– Beim Drucker 9011-31/8931 wird nach HMI3 Fettdruck (EM3, EM4) nicht ausgeführt.

– Beim Drucker 9021 wird HMI3 nur dann mit 1/15 Zoll ausgeführt, wenn ein entsprechender Font eingestellt ist, ansonsten wird der nächstmögliche engere Zeichenabstand genommen.

Wirkung des Reset bei Verbindungsaufbau auf logisch unterstützte Funktionen

Drucker	**HMI**	**VMI**	**PTS**	**SO**	**CHS**	**ASF**	**EM1-4**	**NLQ**	**CR**
9001	1/10	S	-	-	ZV1	-	R	-	S
9004	S	S	R	R	-	ASF1	R	-	-
9013	S	S	R	R	S	ASFn, n(S)	R	-	S
9012	M	M	M	R	M	ASFn, n(M)	R	M	M
9011 18/19	M	M	M	R	M	Trakt ASFn, n(M)	EM1: M EM2: R EM3: M	M	M
9001 31/8931	S	S	R	-	ZV1	-	R	S	-
9022	S	S	R	R	R	ASF1	R	-	-
9021	S	S	R	R	R	ASF1	R	-	-
VTSU Standard	**1/10**	**1/6**	**R**	**R** [1]	**ZV1**	**-**	**R**	**R**	**CR**

[1] Linke Hälfte der ISO-8-Bit-Codetabelle bei Druckern mit 8-Bit-Zeichenvorräten, ansonsten Grundzeichenvorrat

Bedeutung:

S Rücksetzen auf Schalterstellung
M Rücksetzen auf Menüeinstellung
R Rücksetzen
- Funktion wird nicht logisch unterstützt
ZV Zeichenvorrat
n(S) n wird durch Schalter bestimmt
n(M) n wird durch Menüauswahl bestimmt

Hinweise

– VTSU-Standard
Für die logische Unterstützung der Funktionen wird davon ausgegangen, dass die VTSU-Standardwerte per Menü oder Schalter eingestellt sind. Abweichungen von der Standardeinstellung liegen in der Verantwortung des Anwenders.

– Automatischer Einzelblatteinzug
Beim Drucker 9011-18/19 bewirkt die Menüeinstellung für automatischen Einzelblatteinzug (ASF) nur eine Auswahl des Traktors oder des Einzelblatteinzugs.

Layout der DSECT

```
           VTCSET BSP
1 *
1 *              VIRTUAL TERMINAL CONTROL CHARACTER SET
1 *
1 *
1 *              LOGICAL RECORD DELIMITERS
1 *
1 BSPNL    EQU   X'15'           LOGICAL LINE END (CONT NEXT LINE)
1 BSPNP    EQU   X'0C'           LOGICAL PAGE END (CONT NEXT PAGE)
1 BSPCL    EQU   X'0D'           LOGICAL LINE END (CONT SAME LINE)
1 BSPVPA   EQU   X'29'           LOG VERTICAL POS ABSOLUT (CONT LINE N)
1 BSPHPA   EQU   X'2A'           LOG HORIZONT POS ABSOLUT (CONT COL N)
1 BSPASF   EQU   X'21'           LOG SHEED FEDDING FROM CASETTE N  D1
1 BSPCAP   EQU   X'20'           CONTINUE ACTUAL POSITION AT MSG BEGIN
1 *
1 *              LOGICAL UNIT DELIMITERS
1 *
1 BSPEM1   EQU   X'1D'           EMPHASIZED LAYOUT 1
1 BSPEM2   EQU   X'1F'           EMPHASIZED LAYOUT 2
1 BSPEM3   EQU   X'13'           EMPHASIZED LAYOUT 3
1 BSPEM4   EQU   X'14'           EMPHASIZED LAYOUT 4
1 BSPNOR   EQU   X'1E'           NORMAL LAYOUT
1 BSPDAR   EQU   X'12'           DARK LAYOUT
1 BSPPLD   EQU   X'2B'           PARTIAL LINE DOWN
1 BSPPLU   EQU   X'2C'           PARTIAL LINE UP
1 *
1 BSPSO    EQU   X'0E'           SHIFT OUT TO 2ND CHARACTER SET
1 BSPSI    EQU   X'0F'           SHIFT IN TO NORMAL CHARACTER SET
1 *
1 BSPSPA   EQU   X'36'           START PROTECTED AREA
1 BSPEPA   EQU   X'08'           END PROTECTED AREA
1 BSPNUM   EQU   X'11'           START NUMERIC (UNPROTECTED) AREA
1 *
1 BSPCHS   EQU   X'06'           CHARACTER SET D1D2
1 BSPCOL   EQU   X'17'           COLOUR CHOICE
1 BSPLOC   EQU   X'09'           LOCAL ATTRIBUTE START S1
1 BSPLOX   EQU   X'0A'           LOCAL ATTRIBUTE EXIT S1
1 *
1 BSPVMI   EQU   X'24'           VERTICAL MOVEMENT INDICATOR  D1
1 BSPHMI   EQU   X'23'           HORIZONTAL MOVEMENT INDICATOR  D1
1 BSPLM    EQU   X'38'           LEFT MARGIN  D1D2D3
1 BSPPTS   EQU   X'1A'           PROPORTIONAL TYPING START
1 BSPPTX   EQU   X'1B'           PROPORTIONAL TYPING END
1 BSPMLL   EQU   X'33'           MAXIMAL LINE LENGTH
1 BSPMLN   EQU   X'35'           MAXIMAL LINE NUMBER (ON PAGE)
1 BSPNLQ   EQU   X'39'           NEAR LETTER QUALITY START
```

```
1 BSPNLX   EQU   X'3B'           NEAR LETTER QUALITY EXIT
1 *
1 *              SPECIAL FUNCTIONS
1 *
1 BSPDEL   EQU   X'07'           DELETE
1 BSPBS    EQU   X'16'           BACKSPACE
1 BSPSUB   EQU   X'3F'           SUBSTITUTE
1 *
1 *              DELIMITER EXTENSION
1 *
1 BSPEXT   EQU   X'3E'           DELIMITER EXTENSION BYTE
1 *
1 *              EXTENDED LOGICAL DELIMITERS
1 *
1 BSPTRA   EQU   C'T'            TRANSPARENT OUTPUT X1L1L2
1 BSPDIM   EQU   C'D'            DIMENSION OF SCREEN D1D2D3D4D5
1 BSPRPT   EQU   C'R'            REPEAT NEXT CHARACTER NN TIMES
1 BSPDIS   EQU   C'I'            SET DISPLAY ATTRIBUTES
1 BSPRS    EQU   X'00'            RESET DISPLAY ATTRIBUTES
1 BSPFL    EQU   X'01'            FLASHING
1 BSPUND   EQU   X'02'            UNDERSCORED
1 BSPBLK   EQU   X'04'            BLANKED
1 BSPRIN   EQU   X'08'            REDUCED INTENSITY
1 BSPINV   EQU   X'10'            INVERSE
1 BSPFLD   EQU   C'F'            SET FIELD CHARACTERISTICS
1 BSPINP   EQU   X'00'            INPUT FIELD
1 BSPPNS   EQU   X'01'            PROTECTED NOT SENDABLE
1 BSPPRS   EQU   X'20'            PROTECTED SENDABLE
1 BSPNUF   EQU   X'02'            NUMERIC
1 BSPMOD   EQU   X'04'            PRE-MODIFIED
1 BSPMAR   EQU   X'08'            MARKABLE
1 BSPPRT   EQU   X'10'            PRINTABLE
1 BSPASK   EQU   X'40'            AUTOMATIC SKIP
1 *
1 *
1 *              PHYSICAL UNIT DELIMITERS
1 *
1 BSPESC   EQU   X'27'           ESCAPE  X
1 BSPDC4   EQU   X'3C'           DC4  X
1 BSPHT    EQU   X'05'           HORIZONTAL TABULATION
1 BSPVT    EQU   X'0B'           VERTICAL TABULATION
1 *
1                *,VTCSET     080    941024   53531028
```

Ein **Beispiel** für den Extended-Line-Mode finden Sie im Anschluss an die Beschreibung des Makros **WRTRD** (Beispiel 5).

VTSUCB – VTSU-Parameter für Ein-/Ausgabe erstellen

Allgemeines

Anwendungsgebiete: Ein-/Ausgabe von Dateien und Sätzen; siehe Seite 159
Verkehr mit Datenstationen; siehe Seite 164
Makrotyp: S-Typ, MF-Format **3**: C-/D-/L-/M-Form; siehe Seite 29

- Stand der Beschreibung: VTSU V13.3A

Makrobeschreibung

Der VTSU-Control-Block (VTSUCB) ist eine Programmschnittstelle, über die der Benutzer VTSU-Parameter für die Ein- und Ausgabe mit den Makros **RDATA**, **WROUT** und **WRTRD** erstellen kann.
Die Kopplung an die Ein- und Ausgabeschnittstelle ist mit folgenden Aufrufen möglich:

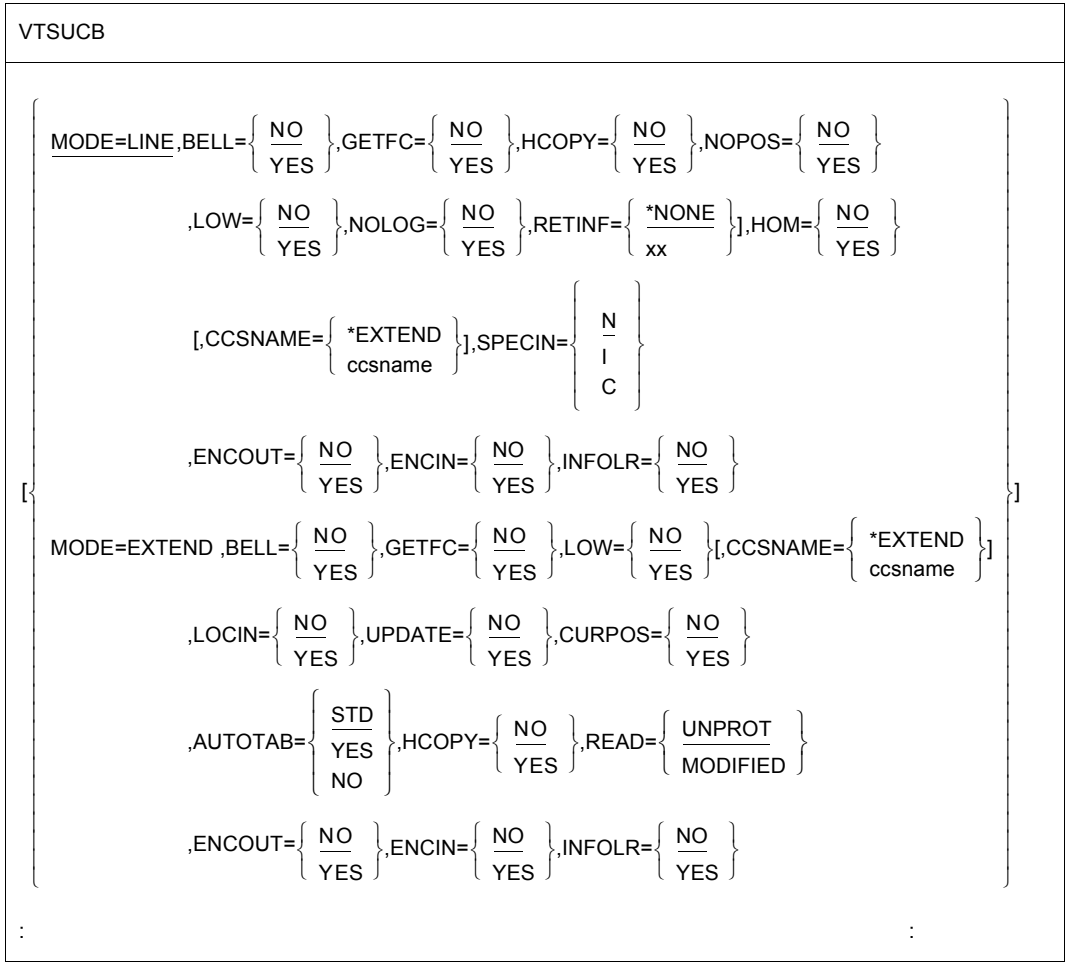
```
    WROUT satz,fehler[,PARMOD=31],VTSUCBA=adrvtsucb
oder WRTRD satz1,,satz2,,[länge],fehler,[,PARMOD=31],VTSUCBA=adrvtsucb
oder RDATA satz,fehler,[länge][,A][,PARMOD=31],VTSUCBA=adrvtsucb
```

Dabei benennt der Operand VTSUCBA eine Adresse (adrvtsucb), ab der die VTSU-Parameter stehen. Diese Angaben ersetzen die Edit-Parameter der jeweiligen Makros. Der Makro **VTSUCB** erstellt einen solchen Datenbereich.

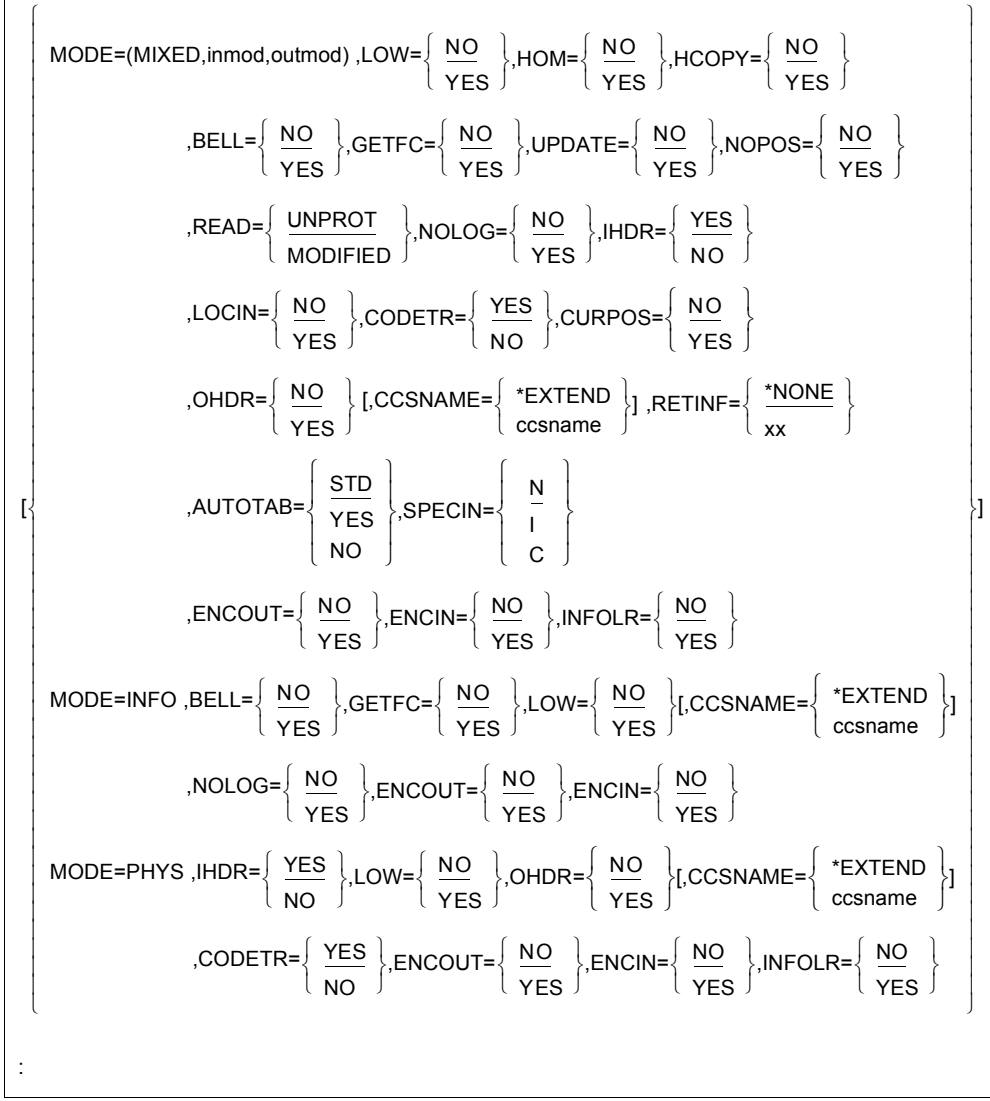
Die **RDATA**/**WROUT**/**WRTRD**-Schnittstellen mit Edit-Parameter können parallel zur **VTSUCB**-Schnittstelle benutzt werde. Neue Funktionen (ab VTSU V9.0B) können jedoch nur über den **VTSUCB** genutzt werden, da die Edit-Parameter nicht mehr erweitert werden.

Erweiterte Zeichensätze werden durch die Operanden CCSNAME und CODETR unterstützt. Diese Operanden werden nur bei MODE=LINE/EXTEND/INFO/FORM/PHYS ausgewertet.

Makroaufrufformat und Operandenbeschreibung



VTSUCB (Fortsetzung)



VTSUCB (Fortsetzung)

```
:                                                                    :
[{ MODE=FORM ,LOW={NO|YES}[,CCSNAME={*EXTEND|ccsname}]
                 ,ENCOUT={NO|YES},ENCIN={NO|YES},INFOLR={NO|YES}
 | MODE=TRANS ,ENCOUT={NO|YES},ENCIN={NO|YES},INFOLR={NO|YES}
 | MODE=CHIP ,ENCOUT={NO|YES},ENCIN={NO|YES}
}]
,MF=L / C / D / M
[,PARAM=adr / (r)]
,PREFIX=Y / p
,MACID=VTC / macid
```

MODE=

LINE
Die aktuelle Datenstation soll als logische Zeilen- bzw. Seiten-Datenstation behandelt werden. Die Nachricht kann durch logische Steuerzeichen strukturiert werden (siehe Makro **VTCSET**).
Für die Ausgabe sind weitere Steuerzeichen unzulässig und werden in ein vom Benutzer definiertes Ersatzzeichen umgewandelt (siehe Kommando MODIFY-TERMINAL-OPTIONS SUBSTITUTE-CHARACTER=).
Ist SYSOUT keine Datenstation, so werden nur die logischen Steuerzeichen NL und NP ausgewertet z.B. bei Ausgaben auf Drucker im Batch-Betrieb.
Bei der Eingabe wird der gerätespezifische Nachrichtenkopf nicht mitgeliefert.

EXTEND
(Nur für Datenstationen 9749, 975x, 9763, 816x und 3270, bei Druckern wird EXTEND intern als Line-Modus verarbeitet)

Die aktuelle Datenstation soll als logische Zeilen- bzw. Seiten-Datenstation behandelt werden. Die Ausgabe des Textes erfolgt standardmäßig geschützt und halbhell. Die Nachricht kann durch logische Steuerzeichen strukturiert werden (siehe Makro **VTCSET**). Die Tasten RU, EFZ, AFZ und LSP sind gesperrt.

Bei TIAM-Anwendungen werden Bereiche, in die der Anwender eingeben kann, mit EPA, DAR oder NUM begonnen und mit SPA beendet. Bei 3270-Datenstationen ist zu beachten, dass die logischen Steuerzeichen Platz auf dem Bildschirm beanspruchen. Mehrere logische Steuerzeichen hintereinander benötigen aber nur einen Platz.
Bei TIAM-Anwendungen wird bei Ein- und Ausgabe NIL als erlaubtes Zeichen behandelt, es wird vom Programm zur Datenstation und umgekehrt geschickt. Bei 3270-Datenstationen ist zu beachten, dass NIL-Zeichen nicht zur DVA übertragen werden.
VTSU-B ergänzt Felder, die bei der Eingabe verkürzt zurückkommen, durch NIL-Zeichen auf ihre ursprüngliche Länge. Dadurch bekommt der Anwender die Felder immer in der Ausgabelänge zurück.
Bei TIAM-Anwendungen wird der Anfang einer Ausgabenachricht an dem dem Cursor folgenden nächsten Zeilenanfang abgebildet. Vor dem 1. Textzeichen wird der Schirm ab Cursor gelöscht, wenn die Nachricht nicht mit VPA beginnt.
Wird bei der Ausgabe das Bildschirmende erreicht, so wird am Schirmanfang fortgesetzt. Diese Fortsetzung ist in jedem Fall bis zum nächsten Feldanfang ungeschützt. Die Bildschirmüberlaufkontrolle ist unwirksam.
Wird in einer Eingabenachricht das Steuerzeichen NL erkannt, so wird die Bearbeitung fortgesetzt und der Returncode X'2C' an der **WRTRD**-Schnittstelle geliefert bzw. der Returncode (Maincode) X'0018' an der **VTSUCB**-Schnittstelle.

INFO
Nachrichten können in einer speziellen Informationszeile (Systemzeile) abgebildet werden, ohne dabei an der Datenstation wichtige Daten zu zerstören.

Die Angabe ist vor allem für die Benutzerprogramme gedacht, die „asynchron“ Nachrichten an Datenstationen senden, ohne die aktuelle Datenstationsanzeige zu kennen.

Die Abbildung erfolgt:
- bei Datenstationen mit Hardware-Anzeigezeile immer geschützt in einer Hardware-Anzeigezeile (z.B. DSS 9749, 9750, 9763)
- bei Datenstationen ohne Hardware-Anzeigezeile geschützt in der 24. Bildschirmzeile. Zuvor muss jedoch eine Ausgabe mit MODE=PHYS oder MODE=FORM erfolgt sein und **TCHNG** INFOLIN=YES gesetzt worden sein.
- in allen anderen Fällen:
  wie eine normale Line-Mode-Nachricht.

Ist die Nachricht länger als eine Bildschirmzeile, wird sie aufgeteilt und Zeile für Zeile ausgegeben.
Das System berücksichtigt dabei die durch das Kommando MODIFY-TERMINAL-OPTIONS OVERFLOW-CONTROL=TIME( ) eingestellte Wartezeit.

Die Systemzeile wird automatisch zurückgesetzt. Die Rücksetzung erfolgt nach der ersten Eingabe, die auf eine Ausgabe in der Systemzeile folgt.

Die Eingabe bei MODE=INFO wird wie eine Line-Mode-Eingabe behandelt.

Hinweis

Wird beim **WRTRD** vor der Systemzeilenausgabe im Line-Mode gearbeitet, wird automatisch ein Eingabefeld eingerichtet.

(MIXED,inmode,outmode)
Das Anwendungsprogramm kombiniert unterschiedliche Moden für Ein- und Ausgabe. Nur MODE=LINE, EXTEND, FORM und PHYS dürfen kombiniert werden.
MODE=CHIP, INFO oder TRANS dürfen nicht kombiniert werden.
Der Standardwert der Operanden LOW, für Ein- und Ausgabe, hängt von dem festgelegten Eingabemodus ab. Bei MODE=PHYS ist LOW=YES, bei MODE=LINE, EXTEND oder FORM ist LOW=NO. Alle anderen Operandenwerten werden entsprechend der geforderten Ein-/Ausgabemodi festgelegt.

PHYS
Die Nachrichten sollen physikalisch, d.h. ohne Aufbereitung durch das System an die Datenstation ausgegeben bzw. von dort eingelesen werden. Damit können spezielle Gerätefunktionen angesprochen werden, für die der LINE- oder FORM-Modus nicht ausreicht.

FORM
Format-Modus. Das Anwenderprogramm arbeitet mit der Software-Komponente FHS bzw. Formatsteuerung, die auch die datenstationsgerechte Aufbereitung der Ausgabenachricht vornimmt.

TRANS
Die Ausgabedaten sollen 'transparent' übertragen werden, d.h. sie bestehen aus beliebigen Binärzeichen (je nach Gerätecode aus 5, 7 oder 8 Bit pro Zeichen), die auf dem Übertragungsweg nicht umgewandelt werden. Ist der Übertragungsweg nicht „potenziell transparent“ generiert, wird die Ausgabe mit dem Returncode X'04' der **WROUT**/**WRTRD**-Schnittstelle zurückgewiesen.

CHIP
Die Ausgabenachricht wird mit dem Geräteprotokoll (810 Protokoll) an das Chipkartenterminal weitergeleitet. Die Nachricht muss im expandierten Modus (siehe Handbücher „Datensichtstationen“) erstellt sein. Ist das Chipkartenterminal nicht ansprechbar, wird die Ausgabe mit dem Returncode X'81' bzw. X'82' (Subcode 2) der **VTSUCB**-Schnittstelle abgewiesen.

Bei der Eingabe wird geprüft, ob die Nachricht vom Chipkartenterminal kommt und das Geräteprotokoll entfernt. Vor die Eingabenachricht wird als erstes Byte der Funktionstastencode gesetzt. Eingabenachrichten, die nicht vom Chipkartenterminal kommen, werden in das Kurztelegramm K14 umgewandelt.

Hinweis

Beim Makro **RDATA** ist der Operand MODE=CHIP nicht zugelassen.

AUTOTAB=
Der automatische Tabulatorsprung von einem ungeschützten Feld zum nächsten ungeschützten Feld wird festgelegt. Im Mixed-Modus wird dieser Parameter nur akzeptiert, wenn Ein- und Ausgabe-Modus den Wert EXTEND haben. Andernfalls wird er ignoriert.

STD
Die Verarbeitung ist vom Betriebsparameter EXPROPOS abhängig.

YES
Sobald Sie am Ende eines ungeschützten Feldes ein Zeichen eintragen, springt der Cursor automatisch von diesem ungeschützten Feld weiter zum nächsten ungeschützten Feld (auch wenn EXPROPOS=Y).

Hinweise
- Bei 3270 Datensichtstationen ist es immer möglich, den Cursor mit den Pfeiltasten auf geschützten Feldern zu bewegen. Ist jedoch AUTOTAB=YES, springt der Cursor automatisch von einem Eingabefeld zum nächsten, sobald am Ende eines Eingabefeldes ein Zeichen eingetragen wird.
- Der Betriebsparameter EXPROPOS wird von der Datensichtstation 3270 ignoriert.

NO
Der Cursor springt nicht automatisch (auch wenn EXPROPOS=N).

BELL=
bestimmt, ob bei der Ausgabe ein akustisches Signal ertönt.

NO
Bei der Ausgabe ertönt kein akustisches Signal.

YES
Bei der Ausgabe ertönt am Ende der Nachricht ein akustisches Signal (nur bei den Datenstationen 9749, 975x, 9763 816x und 3270 mit einem speziellen Gerätezusatz).

CCSNAME=
Der Name des zu verwendenden Zeichensatzes, für diese Nachricht, wird festgelegt. Der Code-Namen der EBCDIC-Variante muss angeben werden. Der Name der entsprechenden ISO-Code-Variante wird automatisch abgelehnt. Der Name darf maximal 8 Byte lang sein. Bei einem Wechsel des verwendeten Zeichensatzes vor der neuen Ausgabe wird automatisch der Bildschirm gelöscht.

ccsname
Zeichensatzname, Name eines beliebigen EBCDIC-Codes. Wird kein Name angegeben, wird automatisch im Standard-Modus gearbeitet. Dies ist entweder ein 7-bit-Modus oder ein mit dem Kommando MODIFY-TERMINAL-OPTIONS aktivierter 8-bit-Modus.

***EXTEND**
Es wird automatisch der erweiterte Anwender-Standard-Code benutzt.

CODETR=
Bei der physikalischen Ausgabe wird festgelegt, ob die Nachricht aus dem spezifizierten Code oder in den spezifizierten Code übersetzt werden soll. Dieser Operand ist nur bei einer Ausgabe auf Druckern sinnvoll, die mit ESCAPE-Sequenzen arbeiten, die nicht entsprechend des EBCDIC-Kerns codiert sind. Diese speziellen ESCAPE-Sequenzen werden von VTSU ignoriert.

YES
VTSU übersetzt die Nachricht aus dem spezifizierten oder in den spezifizierten Code. Die Standard-ESCAPE-Sequenzen werden von VTSU erkannt und übersprungen.

NO
Die Kontrollzeichen SO/SI werden ausgewertet. Eine weitere Code-Umwandlung durch VTSU entfällt.

CURPOS=
Für den Extended-Line-Modus und für den Mixed-Modus wird festgelegt, ob nach der Eingabe die Cursorposition zurückgeliefert werden soll. Im Mixed-Modus wird dieser Parameter nur akzeptiert, wenn Ein- und Ausgabe-Modus den Wert EXTEND haben. Andernfalls wird er ignoriert.

NO
Es wird keine Information über die Cursorposition geliefert. Die Felder YVTCPOSL und YVTCPOSC sind undefiniert.

YES
Die Felder YVTCPOSL und YVTCPOSC werden mit der aktuellen Cursorposition (Zeilen und Spalten) nach der Eingabe zurückgeliefert.

ENCIN=
Gibt an, ob die nächste Eingabe verschlüsselt sein muss oder nicht (gilt nur, wenn die Verbindung von einer Emulation aufgebaut wird, die die Verschlüsselung unterstützt).

YES
Die nächste Eingabe muss verschlüsselt sein.

NO
Die nächste Eingabe darf nicht verschlüsselt sein (dennoch kann eine Verschlüsselung der Nachricht nicht verhindert werden, wenn die Verschlüsselung auf andere Weise angefordert wurde).

ENCOUT=
Gibt an, ob die aktuelle Ausgabe verschlüsselt werden muss oder nicht (gilt nur, wenn die Verbindung von einer Emulation aufgebaut wird, die die Verschlüsselung unterstützt).

YES
Die Ausgabe muss verschlüsselt sein.

NO
Die Ausgabe darf nicht verschlüsselt sein (dennoch kann eine Verschlüsselung der Nachricht nicht verhindert werden, wenn die Verschlüsselung auf andere Weise angefordert wurde).

GETFC=
bestimmt, ob ein Funktionstastencode übergeben wird.

NO
Es soll kein Funktionstastencode übergeben werden.

YES
Der logische Funktionstastencode, der die Taste darstellt, die die Datenübertragung an der Datenstation auslöst, wird als erstes Zeichen der Nachricht übertragen.

HCOPY=
legt fest, ob die Ausgabenachricht nicht nur auf eine Datensichtstation, sondern auch über ein angeschlossenes Hardcopy-Gerät (Drucker) ausgegeben werden soll.

NO
Die Ausgabenachricht wird nur über die Datensichtstation ausgegeben.

YES
Die Ausgabenachricht für eine Datensichtstation wird gleichzeitig über ein dort angeschlossenes Hardcopy-Gerät (Drucker) ausgedruckt.

Hinweise
- Der Hardcopy-Ausdruck erfolgt nur dann, wenn für die Datensichtstation beim Verbindungsaufbau oder durch das Kommando MODIFY-TERMINAL-OPTIONS ein Hardcopy-Gerät zugewiesen wurde. Bei 3270-Datenstationen muss das Hardcopy-Gerät beim Verbindungsaufbau zugewiesen (generiert) sein.
- Wird HCOPY=YES verwendet und kein EXTEND-Mode benutzt und enthält die Nachricht das logische Steuerzeichen SPA, EPA, NUM oder DAR (falls DARPRINTABLE=N ist), so wird nicht die gesamte Nachricht, sondern nur der letzte ungeschützte Teil der Nachricht abgedruckt.
- Wird gleichzeitig MODIFY-TERMINAL-OPTIONS OVERFLOW-CONTROL=NO verwendet, kann es vorkommen, dass nur ein Teil der Ausgabe auf dem Hardcopy-Gerät wiedergegeben wird.

HOM=
(Nur für Datensichtstationen 816x, 9749, 975x ,9763 und 3270).

NO
Die Nachricht soll strukturiert und heterogen ausgegeben werden, d.h. als Ausgabeeinheit wird eine logische Zeile betrachtet.

Wirkung bei Datensichtstationen 816x, 975x, 9763 und 3270 bei Betriebsart 1 (nur für TIAM-Anwendungen):

Einzelne logische Zeilen können getrennt modifiziert und damit gezielt zurückübertragen werden.

YES
Die Nachricht soll unstrukturiert und homogen ausgegeben werden, d.h. die gesamte Nachricht wird als eine Ausgabeeinheit betrachtet. Die Nachrichtenlänge ist durch die Größe des Ausgabepuffers im System beschränkt.

Wirkung bei Datensichtstationen 816x, 975x, 9763 und 3270 bei Betriebsart 1 (nur für TIAM-Anwendungen):
Durch Modifikation eines Zeichens einer Ausgabenachricht kann die gesamte Nachricht wieder zurückübertragen werden, sofern diese nicht durch logische Anzeigesteuerzeichen explizit strukturiert wird.

IHDR=
gibt an, wie mit dem Nachrichtenkopf verfahren wird.

YES
Der gesamte Nachrichtenkopf wird an das Benutzerprogramm übergeben (Standardwert für MODE=PHYS).
Bei 3270-Datenstationen besteht der Nachrichtenkopf aus dem Code der Sendetaste (AID-Byte) und der zwei Byte langen Schreibmarkenposition.

NO
Der Nachrichtenkopf wird nicht an das Benutzerprogramm übergeben.

INFOLR=
Gibt an, ob die Informationszeile zurückgesetzt werden muss.

YES
Die Informationszeile muss zurückgesetzt werden.

NO
Die Informationszeile muss nicht zurückgesetzt werden.

LOCIN=
legt fest, wie mit lokalen Attributen in der Eingabenachricht verfahren wird.
Dieser Operand betrifft nur Datensichtstationen, die lokale Attribute unterstützen (z.B. Datenstation 9763).

NO
Lokale Attribute werden aus der Eingabenachricht entfernt und nicht an den Anwender weitergegeben.

YES
Sind lokale Attribute in der Eingabenachricht, werden sie an den Anwender als logische Steuerzeichen weitergereicht (siehe Makro **VTCSET**).

LOW=
legt fest, ob zwischen Klein- und Großschreibung unterschieden werden soll. Die Voreinstellung des Operanden LOW ist abhängig vom Operanden MODE:

```
MODE=MIXED             die Voreinstellung hängt vom festgelegten
                       Eingabemodus ab
MODE=PHYS              LOW=YES ist Voreinstellung
MODE=LINE/EXTEND/FORM  LOW=NO ist Voreinstellung
```

NO
Alle Kleinbuchstaben werden dem Benutzerprogramm als Großbuchstaben übergeben.

YES
Dem Benutzerprogramm werden auch Kleinbuchstaben übergeben (Standardwert für MODE=PHYS).

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. PREFIX, MACID und PARAM) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

NOLOG=
bestimmt, ob logische Steuerzeichen ausgewertet werden sollen.

NO
Alle logischen Steuerzeichen werden ausgewertet und spezielle physikalische Steuerzeichen werden durchgelassen (siehe Makro **VTCSET** z.B. ESC,DC4). Andere Zeichen < X'40' werden durch SUB ersetzt. Abdruckbare Zeichen werden durchgelassen.

YES
Logische Steuerzeichen werden nicht ausgewertet. Alle Zeichen, die im EBCDIC-Code < X'40' sind, werden durch SUB (Schmierzeichen) ersetzt. Nur abdruckbare Zeichen werden durchgelassen.

NOPOS=
(Nur für Drucker). Für den Line-Modus und für den Mixed-Modus wird der Ausgabeort von Nachrichten festgelegt.
Im Mixed-Modus wird dieser Parameter nur akzeptiert, wenn der Ausgabemodus MODE=LINE ist. Andernfalls wird er ignoriert.

NO
Die Ausgabenachricht beginnt am Anfang der nächsten Zeile.

YES
Die Ausgabenachricht beginnt am Anfang der aktuellen Zeile.

OHDR=
gibt an, wie mit dem benutzerindividuellen Nachrichtenvorspann verfahren wird.

NO
Der Nachrichtenkopf wird dem Ausgabetext nicht vorangestellt.

YES
Die Nachricht enthält einen benutzerindividuellen Nachrichtenkopf, den das System dem Ausgabetext voranstellt. Die Länge des Nachrichtenkopfes +1 muss im ersten Byte der Nachricht binär angegeben werden.

Hinweis
Bei Ausgabe auf die Datensichtstationen 8160, 975x, 9763 und daran lokal angeschlossene Drucker ist zu beachten, dass das System (mit MODE=LINE) oder FHS (mit MODE=FORM) mit Operandenangaben (PAG) arbeitet und keinen Nachrichtenkopf verwendet (PARAM0, PARAM1). Die Unterschiede zwischen diesen beiden Arbeitsweisen sind in den Handbüchern der Datensichtstationen bzw. Drucker beschrieben.

READ=
Für den Extended-Line-Modus und für den Mixed-Modus wird der physikalische Lese-Modus festgelegt.
Im Mixed-Modus wird dieser Parameter nur akzeptiert, wenn Ein- und Ausgabe-Modus den Wert EXTEND haben. Andernfalls wird er ignoriert.

Beachten Sie, dass bei DCAM-Anwendungen im Extended-Line-Modus bei einem Aufruf YSEND gefolgt von einem Aufruf YRECEIVE beide Aufrufe den gleichen physikalischen Lese-Modus (UNPROT oder MODIFIED) haben müssen.

UNPROT
Alle ungeschützten Felder, auch die nicht geänderten, werden an Sie zurückgeliefert. Um die geänderten Werte festzustellen, müssen Sie die empfangenen Daten mit den Ausgabedaten vergleichen.

MODIFIED
Nur die modifizierten Felder werden an Sie zurückgeliefert. Jedem modifizierten Feld wird im Benutzerpuffer seine Position am Bildschirm vorangestellt (siehe logische Steuerzeichen VPA und HPA).

RETINF=
entscheidet, ob von Druckerstationen eine Rückmeldung gefordert wird.

***NONE**
Es wird keine Rückmeldung geliefert.

xx
zwei beliebige abdruckbare Zeichen, die mit der Rückmeldung zurückgeliefert werden sollen. Hochkommas, die zurückgeliefert werden sollen, müssen verdoppelt werden (z.B. RETINF=""). Die Rückmeldung ist 4 Byte lang und wird wie folgt aufgebaut:

```
Byte 0       Identifikation (X'41' positiv/ X'42' negativ)
Byte 1-2     RETINF-Byte
Byte 3       Information über Drucker-Status (druckerabhängig)
```

SPECIN=
fordert eine spezielle Eingabe an. Wird eine spezielle Eingabe gefordert, muss zuvor der Operand SPECIN in der anfordernden Ausgabenachricht angegeben werden.

N
Normale Eingabe von der Datenstation.

I
Die Daten werden vom Ausweisleser gelesen. Die Eingabedaten können aus der Ausweisinformation oder dem Kurztelegramm K14 bestehen. Diese Angabe ist nur bei Datenstationen 9749, 975x, 9763, 816x und 3270 mit definiertem Ausweisleser möglich.
Im Unterschied zu den TRANSDATA-Geräten können bei 3270-Datenstationen jederzeit Daten von einem definierten Ausweisleser eingegeben werden. Wenn Eingaben vom Ausweisleser angefordert werden, wird jede andere Eingabe in K14 umgewandelt.

C
Die Eingabedaten sind vertraulich und bleiben an der Datenstation unsichtbar. Dies erfolgt durch Dunkelsteuerung bzw. Löschen des Bildschirms (wodurch das Bildschirmformat auf 24x80 zurückgesetzt wird), oder durch Überschreiben der Eingabezeile an Schreibstationen.

UPDATE=
Für den Extended-Line-Modus und für den Mixed-Modus können Sie bei Formatausgaben festlegen, ob der ganze Bildschirm neu aufgebaut wird, oder nur die modifizierten Zeilen aktualisiert werden sollen. Eine modifizierte Zeile ist eine Zeile in der entweder ein existierendes Feld aktualisiert oder ein neues Feld erzeugt wird. Im Mixed-Modus wird dieser Parameter nur akzeptiert, wenn Ein- und Ausgabe-Modus den Wert EXTEND haben. Andernfalls wird er ignoriert.

NO
Bei der ersten logischen neuen Seite, wird der ganze Bildschirm neu aufgebaut.

YES
Es werden nur die modifizierten Zeilen aktualisiert.
Wenn Sie ein neues Feld erzeugen, sollten Sie darauf achten, dass das Ende dieses neuen Feldes mit dem Ausgabeattribut 'nicht sichtbar' und dem Feldattribut 'geschützt' versehen ist. Wird ein neues Feld erzeugt, werden bis zum nächsten Feldanfang binäre Nullen ausgegeben. Durch das Ausgabeattribut 'nicht sichtbar' wird die Ausgabe von

binären Nullen unterdrückt und es werden Leerzeichen ausgegeben.
Durch das Feldattribut 'geschützt' wird verhindert, dass das nachfolgende Feld durch das neue Feld überschrieben werden kann.
Beachten Sie, dass beim Aktualisieren des Bildschirms der gleiche Zeichensatz (CCSNAME) verwendet werden muss, wie beim Erstellen des Ursprungs-Bildschirms. Andernfalls wird der Ursprungs-Bildschirm gelöscht und es werden nur die aktualisierten Zeilen ausgegeben.
Weiter müssen Sie beachten, dass durch das Aktualisieren Feldattribute nicht implizit zurückgesetzt werden. Zum Beispiel behält ein Feld, dem das Attribut 'vormodifiziert' zugeordnet wurde auch nach dem Aktualisieren das Attribut 'vormodifiziert'. Attribute müssen somit explizit zurückgesetzt werden.

Rückinformation und Fehleranzeigen

Standard-header:

c	c	b	b	a	a	a	a

Über die Ausführung des Makros VTSUCB wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'00'	X'0000'	Erfolgreiche Bearbeitung.
X'58'	X'00'	X'0008'	Erfolgreiche Bearbeitung, aber RETINF-Byte ist falsch. Operand ignoriert.
X'02'	X'00'	X'0008'	Erfolgreiche Bearbeitung, aber Mode für aktuelle Datenstation unzulässig. Ersatzabbildung
X'00'	X'01'	X'FFFF'	Funktion nicht ausgeführt. UNIT oder FUNCT fehlerhaft.
X'xx' [1)]	X'01'	X'0004'	Funktion nicht ausgeführt. Operandenfehler im VTSUCB.
X'40'	X'01'	X'0004'	Funktion nicht ausgeführt. Länge des benutzerspezifischen Nachrichtenkopfes ungültig.
X'50'	X'01'	X'0004'	Funktion nicht ausgeführt. Länge des Benutzerpuffers für die Eingabe ungültig.
X'60'	X'01'	X'0004'	Funktion nicht ausgeführt. Die angeforderte XHCS-Funktion ist nicht verfügbar. XHCS ist nicht geladen.
X'61'	X'01'	X'0004'	Funktion nicht ausgeführt. Die XHCS-Funktion wird für 7-bit-Datenstationen angefordert.
X'62'	X'01'	X'0004'	Funktion nicht ausgeführt. Die XHCS-Funktion wird nicht unterstützt.
X'80'	X'01'	X'0004'	Funktion nicht ausgeführt. Operand MODE für Kommandotyp ungültig.
X'86'	X'01'	X'0004'	Funktion nicht ausgeführt. Der CCSNAME ist inkompatibel zu den Geräten.
X'1E'	X'01'	X'0004'	Funktion nicht ausgeführt. Ungültiger CCS-Name.
X'00'	X'03'	X'FFFF'	Funktion nicht ausgeführt. VTSUCB-VERSION ist fehlerhaft.
X'xx'	X'20'	X'0004'	Funktion nicht ausgeführt. interner Fehler (zur Diagnose)
X'00'	X'40'	X'000C'	Ausgabenachricht abgeschnitten.
X'00'	X'40'	X'0010'	Eingabenachricht abgeschnitten.
X'00'	X'40'	X'0018'	Extended line mode: Eingabenachricht verkürzt.

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'02'	X'40'	X'0004'	Funktion nicht ausgeführt. Mode für aktuelle Datenstation unzulässig, keine Ersatzabbildung.
X'10'	X'40'	X'0020'	Funktion nicht ausgeführt. Begrenzte Information.
X'81'	X'40'	X'0004'	Funktion nicht ausgeführt. Für die Datenstation ist kein Chipkartenterminal verfügbar.
X'82'	X'40'	X'0004'	Funktion nicht ausgeführt. Chipkartenterminal vorhanden, aber nicht erreichbar.
X'83'	X'40'	X'0004'	Funktion nicht ausgeführt. Datensichtstation lehnt Nachricht für Chipkartenterminal ab.

Weitere Returncodes, deren Bedeutung durch Konventionen makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

1) In Subcode 2 wird der erste falsche Parameter im **VTSUCB** angezeigt.

08: Längenangabe im VTSUCB falsch
10: Operand MODE falsch
11: Operand HCOPY falsch
12: Operand BELL falsch
13: Operand NOLOG falsch
14: Operand READ falsch
15: Operand HOM falsch
16: Operand RETINF falsch
17: Operand LOCIN falsch
18: Operand OHDR falsch
19: Operand CODETR falsch
1A: Operand IHDR falsch
1B: Operand LOW falsch
1C: Operand SPECIN falsch
1D: Operand GETFC falsch
1E: Operand CCSNAME falsch
1F: Operand CURPOS falsch
20: Operand UPDATE falsch
22: Operand AUTOTAB falsch
23: Operand NOPOS falsch
24: Operand ENCOUT falsch
25: Operand ENCIN falsch
26: Operand INFOLR falsch

Das linke Byte des Maincodes wird zurzeit nicht belegt und auf den Wert X'00' gesetzt. Wenn im Standard-Header Fehler sind, die der Selbstidentifikation des Produktes dienen (z.B falsche Version), wird der Wert auf X'FF' gesetzt.

Fehlermeldungen, die ohne Verwendung des VTSUCB an die Ein-/Ausgabeschnittstelle der Zugriffsmethoden geliefert werden, werden in gleicher Form auch bei Verwendung des VTSUCB an die Ein-/Ausgabeschnittstelle geliefert. Zusätzlich dazu wird eine entsprechende Fehlerinformation im VTSUCB geliefert. Fehlerinformationen, die nur den VTSUCB betreffen, werden an den Ein-/Ausgabeschnittstellen durch einen eigenen Returncode (X'24') angezeigt und im VTSUCB-Returncode näher erklärt.

Layout der DSECT

```
           VTSUCB MF=D,PREFIX=A
1 AVTSUCB    DSECT
1          FHDR  MF=(C,AVTC),EQUATES=NO
2          DS    0A
2 AVTCFHE  DS    0XL8            0   GENERAL PARAMETER AREA HEADER
2 *
2 AVTCIFID DS    0A              0   INTERFACE IDENTIFIER
2 AVTCFCTU DS    AL2             0   FUNCTION UNIT NUMBER
2 *                                  BIT 15    HEADER FLAG BIT,
2 *                                  MUST BE RESET UNTIL FURTHER NOTICE
2 *                                  BIT 14-12 UNUSED, MUST BE RESET
2 *                                  BIT 11-0  REAL FUNCTION UNIT NUMBER
2 AVTCFCT  DS    AL1             2   FUNCTION NUMBER
2 AVTCFCTV DS    AL1             3   FUNCTION INTERFACE VERSION NUMBER
2 *
2 AVTCRET  DS    0A              4   GENERAL RETURN CODE
2 AVTCSRET DS    0AL2            4   SUB RETURN CODE
2 AVTCSR2  DS    AL1             4   SUB RETURN CODE 2
2 AVTCSR1  DS    AL1             5   SUB RETURN CODE 1
2 AVTCMRET DS    0AL2            6   MAIN RETURN CODE
2 AVTCMR2  DS    AL1             6   MAIN RETURN CODE 2
2 AVTCMR1  DS    AL1             7   MAIN RETURN CODE 1
2 AVTCFHL  EQU   8               8   GENERAL OPERAND LIST HEADER LENGTH
2 *
1 *
1 *  SUBCODE 1 VALUES
1 *
1 AVTCECPM     EQU   X'01'         ERROR CLASS PARAMETER ERROR
1 AVTCECIN     EQU   X'20'         ERROR CLASS INTERNAL ERROR
1 AVTCECSP     EQU   X'40'         ERROR CLASS SPECIAL ERROR
1 *
1 *  SUBCODE 2 VALUES
1 *
1 AVTCERLN     EQU   X'08'         ERROR IN LENGTH OF VTSUCB
```

```
1 AVTCERMO     EQU   X'10'         ERROR IN MODE PARAMETER
1 AVTCERHC     EQU   X'11'         ERROR IN HARDCOPY PARAMETER
1 AVTCERBE     EQU   X'12'         ERROR IN BELL PARAMETER
1 AVTCERNO     EQU   X'13'         ERROR IN NOLOG PARAMETER
1 AVTCERRD     EQU   X'14'         ERROR IN READ PARAMETER
1 AVTCERHO     EQU   X'15'         ERROR IN HOM PARAMETER
1 AVTCERRE     EQU   X'16'         ERROR IN RETINF PARAMETER
1 AVTCERLO     EQU   X'17'         ERROR IN LOCIN PARAMETER
1 AVTCEROH     EQU   X'18'         ERROR IN OUTPUT HEADER PARAMETER
1 AVTCERCO     EQU   X'19'         ERROR IN CODETR PARAMETER
1 AVTCERIH     EQU   X'1A'         ERROR IN INPUT HEADER PARAMETER
1 AVTCERLW     EQU   X'1B'         ERROR IN LOWER INPUT PARAMETER
1 AVTCERSP     EQU   X'1C'         ERROR IN SPECIAL INPUT PARAMETER
1 AVTCERGE     EQU   X'1D'         ERROR IN GET FUNCTION CODE PARAMETER
1 AVTCERCC     EQU   X'1E'         INVALID CCSNAME
1 AVTCERCP     EQU   X'1F'         ERROR IN CURPOS PARAMETER
1 AVTCERUP     EQU   X'20'         ERROR IN UPDATE PARAMETER
1 AVTCERWA     EQU   X'21'         ERROR IN WARINFO PARAMETER
1 AVTCERAT     EQU   X'22'         ERROR IN AUTOTAB PARAMETER
1 AVTCERNP     EQU   X'23'         ERROR IN NOPOS PARAMETER
1 AVTCEREO     EQU   X'24'         ERROR IN ENCOUT PARAMETER
1 AVTCEREI     EQU   X'25'         ERROR IN ENCIN PARAMETER
1 AVTCERIR     EQU   X'26'         ERROR IN INFOLR PARAMETER
1 *
1 AVTCERXH     EQU   X'60'         XHCS FCT REQUESTED BUT XHCS NOT LOADED
1 AVTCER7B     EQU   X'61'         XHCS FCT REQUESTED FOR 7-BIT TERMINAL
1 AVTCERBS     EQU   X'62'         XHCS FCT SUPPORT ONLY FROM BS2000 V10
1 *
1 AVTCERIN     EQU   X'7A'         NO PLACE ENOUGH TO INSERT SI/SO CHARS
1 AVTCER1L     EQU   X'7B'         NO PLACE ENOUGH TO INSERT PARO1L CHARS
1 *
1 AVTCERM1     EQU   X'80'         MODE NOT VALID FOR COMMAND TYPE
1 AVTCERM2     EQU   X'81'         MODE=CHIP USED BUT NO CKT ANNOUNCED
1 AVTCERC1     EQU   X'82'         CKT NOT AVAILABLE FROM TERMINAL
1 AVTCERC2     EQU   X'83'         OTHER ERROR CODE FROM DSS BY CKT-MSG
1 AVTCERX2     EQU   X'86'         VTSUCB CCSN INCOMPATIBLE WITH DEVICE
1 AVTCERE1     EQU   X'87'         ENCRYPTION FOR OUTPUT NOT SUPPORTED
1 AVTCERE2     EQU   X'88'         ENCRYPTION FOR INPUT  NOT SUPPORTED
1 AVTCERE3     EQU   X'89'         INFO LINE RESET INVALID WITH MODE
1 *
1 AVTCERO1     EQU   X'40'         HEADER LENGTH OF OUTPUT MSG NOT VALID
1 AVTCERI1     EQU   X'50'         USER BUFFER LEN FOR INPUT NOT VALID
1 *
1 *  MAINCODE VALUES FOR SUBCODE 1 = X'40'
1 *
1 AVTCMRPM     EQU   X'04'         WRONG PARAMETER FOR DEVICE
1 AVTCMROT     EQU   X'0C'         OUTPUT TRUNCATION
1 AVTCMRIT     EQU   X'10'         INPUT TRUNCATION
```

```
1 AVTCMRNL     EQU   X'18'         NL IN EXT LINE INPUT MESSAGE
1 *
1 *  MAINCODE VALUES FOR SUBCODE 1 = X'00'
1 *
1 AVTCMRCO     EQU   X'08'         CORRECTED ERROR
1 *
1 *
1 AVTCLEN      DS    H             LENGTH OF VTSUCB
1 *
1 AVTCINM      DS    C             INPUT MODE FOR MODE=MIXED
1 AVTCOUTM     DS    C             OUTPUT MODE FOR MODE=MIXED
1 *
1          DS    XL4           RETURN INFO (NOT YET USED)
1 *
1 AVTCMODE     DS    C             MODE OF MESSAGE
1 AVTCLINE     EQU   C'L'          LINE MODE
1 AVTCEXT      EQU   C'E'          EXTENDED LINE
1 AVTCINFO     EQU   C'I'          INFO LINE MESSAGE
1 AVTCPHYS     EQU   C'P'          PYHSICAL MODE
1 AVTCTRAN     EQU   C'T'          TRANSPARENT MODE
1 AVTCFORM     EQU   C'F'          FORM MODE
1 AVTCCHIP     EQU   C'C'          CHIPCARD MODE (FOR CKT)
1 AVTCMIXD     EQU   C'M'          MIXED MODE
1 *
1 AVTCHC       DS    C             HARCOPY FUNCTION
1 AVTCHCN      EQU   C'N'          NO HARDCOPY
1 AVTCHCY      EQU   C'Y'          LOCAL/CENTRAL HARDCOPY
1 *
1 AVTCBEL      DS    C             BELL FUNCTION
1 AVTCBELN     EQU   C'N'          NO BELL
1 AVTCBELY     EQU   C'Y'          BELL AFTER OUTPUT
1 *
1 AVTCNLG      DS    C             NO LOG CHARS TO INTERPRET FUNCTION
1 AVTCNLGN     EQU   C'N'          LOGICAL CHARACTERS TO INTERPRET
1 AVTCNLGY     EQU   C'Y'          NO LOGICAL CHARACTERS TO INTERPRET
1 *
1 AVTCRBYT     DS    CL2           RETURN INFO BYTES
1 *
1 AVTCRIN      DS    C             RETURN INFORMATION FUNCTION
1 AVTCRINN     EQU   C'N'          NO RETURN INFORMATION
1 AVTCRINY     EQU   C'Y'          RETURN INFORMATION REQUIRED
1 *
1 AVTCLOC      DS    C             INPUT OF LOCAL CHARACTERS
1 AVTCLOCN     EQU   C'N'          NO LOCAL CHARACTERS REQUIRED
1 AVTCLOCY     EQU   C'Y'          LOCAL CHARACTERS REQUIRED
1 *
1 AVTCOHD      DS    C             OUTPUT HEADER FUNCTION
1 AVTCOHDN     EQU   C'N'          NO OUTPUT HEADER IN USER MSG
```

```
1 AVTCOHDY     EQU   C'Y'          OUTPUT HEADER IN USER MESSAGE
1 *
1 AVTCCTR      DS    C             CODE TRANSLATION FUNCTION
1 AVTCCTRN     EQU   C'N'          NO CODE TRANSLATION DONE BY VTSU
1 AVTCCTRY     EQU   C'Y'          CODE TRANSLATION TO/FROM CCS REQ.
1 *
1 AVTCIHD      DS    C             INPUT HEADER FUNCTION
1 AVTCIHDN     EQU   C'N'          NO INPUT HEADER REQUIRED
1 AVTCIHDY     EQU   C'Y'          INPUT HEADER REQUIRED
1 *
1 AVTCLOW      DS    C             LOWER CHARACTERS FUNCTION
1 AVTCLOWN     EQU   C'N'          TRANSLATE LOWER CHARACTERS
1 AVTCLOWY     EQU   C'Y'          RETAIN LOWER CHARACTERS
1 *
1 AVTCSPIN     DS    C             SPECIAL INPUT FUNCTION
1 AVTCNSPI     EQU   C'N'          NO SPECIAL INPUT
1 AVTCIDIN     EQU   C'I'          INPUT FROM ID-CARD READER
1 AVTCCOIN     EQU   C'C'          CONFIDENTIAL INPUT
1 *
1 AVTCFC       DS    C             FUNCTION CODE
1 AVTCFCN      EQU   C'N'          NO FUNCTION CODE REQUIRED
1 AVTCFCY      EQU   C'Y'          FUNCTION CODE REQUIRED
1 *
1 AVTCHOM      DS    C             HOMOGENEOUS OUTPUT
1 AVTCHOMN     EQU   C'N'          NO HOMOGENEOUS OUTPUT REQUIRED
1 AVTCHOMY     EQU   C'Y'          HOMOGENEOUS OUTPUT REQUIRED
1 *
1 AVTCNOP      DS    C             OUTPUT ON SAME LINE
1 AVTCNOPN     EQU   C'N'          OUTPUT STARTS ON NEXT LINE
1 AVTCNOPY     EQU   C'Y'          OUTPUT STARTS ON CURRENT LINE
1 *
1 AVTCCCNA     DS    CL8           CODED CHARACTER SET NAME
1 *
1 AVTCCUR      DS    C             CURSOR POSITION REQUESTED
1 AVTCCURN     EQU   C'N'          CURSOR POSITION NOT RETURNED
1 AVTCCURY     EQU   C'Y'          CURSOR POSITION GIVEN AFTER INPUT
1 *
1 AVTCPOSL     DS    XL1           CURSOR POSITION (LINE)
1 AVTCPOSC     DS    XL1           CURSOR POSITION (COLUMN)
1 *
1 AVTCREAD     DS    C             READ MODE (EXTENDED LINE MODE)
1 AVTCRDUN     EQU   C'U'          READ UNPROTECTED
1 AVTCRDMO     EQU   C'M'          READ MODIFIED
1 *
1 AVTCUPD      DS    C             SCREEN UPDATE IN EXTENDED LINE MODE
1 AVTCUPDN     EQU   C'N'          NO SCREEN UPDATE -> REFRESH
1 AVTCUPDY     EQU   C'Y'          SCREEN UPDATE
1 *
```

```
1 AVTCWAR      DS    C             WAR BYTE REQUESTED
1 AVTCWARN     EQU   C'N'          NO INFO ABOUT WAR BYTE
1 AVTCWARY     EQU   C'Y'          VALUE OF WAR BYTE TO RETURN
1 *
1 AVTCWARI     DS    XL1           RETURNED WAR BYTE VALUE
1 *
1 AVTCAT       DS    C             AUTOMATIC TABULATION
1 AVTCATS      EQU   C'S'          STANDARD AUTOMATIC TABULATION
1 AVTCATN      EQU   C'N'          AUTOMATIC TABULATION NOT REQUESTED
1 AVTCATY      EQU   C'Y'          AUTOMATIC TABULATION REQUESTED
1 *
1 AVTCEO       DS    C             ENCRYPTION FOR OUTPUT
1 AVTCEON      EQU   C'N'          ENCRYPTION FOR OUTPUT NOT REQUESTED
1 AVTCEOY      EQU   C'Y'          ENCRYPTION FOR OUTPUT REQUESTED
1 *
1 AVTCEI       DS    C             ENCRYPTION FOR INPUT
1 AVTCEIN      EQU   C'N'          ENCRYPTION FOR INPUT NOT REQUESTED
1 AVTCEIY      EQU   C'Y'          ENCRYPTION FOR INPUT REQUESTED
1 *
1 AVTCIR       DS    C             INFO LINE RESET
1 AVTCIRN      EQU   C'N'          INFO LINE RESET NOT REQUESTED
1 AVTCIRY      EQU   C'Y'          INFO LINE REQUESTED
1 *
1              DS    XL1           RESERVED
1 *
1 AVTC#        EQU   *-AVTCFHE        LENGTH OF DSECT
1               *,VTSUCB     350    980309
```

WRCPT – Fixpunkt schreiben

Allgemeines

Anwendungsgebiet: Fixpunktschreiben; siehe Seite 166
Makrotyp: S-Typ, MF-Format **1**:
31-Bit-Schnittstelle: Standardform/L-/D-/E-Form; siehe Seite 29

Makrobeschreibung

Der **WRCPT**-Makro schreibt einen Fixpunkt in eine anzugebende Datei (katalogisierte PAM-Datei). Der Fixpunkt umfasst Kennungsinformationen, Programmzustand, auf Programmzustand bezogener Systemzustand und Speicherinhalt (virtueller Speicher). Ein abgebrochenes Programm kann mit dem Kommando RESTART-PROGRAM unter Bezugnahme auf einen Fixpunkt fortgesetzt werden (siehe Handbuch „Kommandos“ [19]).

Zu beachten:
– Der OPEN auf die Datei wird von der Fixpunktroutine ausgeführt.
– Der Operand O steuert, ob die Datei überschrieben wird oder nicht.
– Das Verzweigen zur Fehlerroutine bzw. zur benutzereigenen Restart-Routine muss der Anwender selbst durchführen.

Makroaufrufformat und Operandenbeschreibung

```
WRCPT

{ LINK=linkname   }
{ FILE=pfadname   }

[,IDENT=fixid]
,O=NO / YES
,ENDAID=NO / YES
[,MF=L / (E,..) / D]
```

LINK=
bezeichnet über einen Dateikettungsnamen die Datei, in die der Fixpunkt geschrieben werden soll. Die Datei muss als PAM-Datei katalogisiert und mit diesem Dateikettungsnamen in der TFT eingetragen sein (Kommando ADD-FILE-LINK, siehe „Kommandos“ [19]).

linkname
Dateikettungsname der Datei.

FILE=
bezeichnet die Datei, in die der Fixpunkt geschrieben werden soll.
Die Datei muss bereits als PAM-Datei katalogisiert sein.

pfadname
Pfadname der Datei.

IDENT=
bezeichnet eine Zeichenfolge zur Benennung des Fixpunktes. Die Benennung erfolgt intern, wenn der Operand nicht spezifiziert wird oder die Zeichenfolge mit einem Leerzeichen beginnt.

fixid
Zeichenfolge. Länge = 6 Byte. Das erste Zeichen darf kein Leerzeichen sein.

O=
beschreibt, ob die angegebene Datei vor dem Eintragen des Fixpunktes (logisch) gelöscht werden soll oder nicht.

NO
Die angegebene Datei wird nicht gelöscht. Der Fixpunkt wird an das Ende der angegebenen Datei geschrieben.

YES
Der Inhalt der angegebenen Datei wird vor dem Eintrag des Fixpunktes gelöscht.

ENDAID=
Gibt an, ob vorhandene AID-Verbindungen beendet werden sollen.

NO
Wurde vor dem Fixpunktschreiben mit AID gearbeitet, bleiben alle AID-Maßnahmen gültig. Das Fixpunktschreiben wird mit dem Returncode '68' abgelehnt.

YES
Der Fixpunkt wird in jedem Fall geschrieben. Wurde vor dem Fixpunktschreiben mit AID gearbeitet, wird die Verbindung zu AID beendet. Alle bisher gesetzten Haltepunkte sind nach dem Fixpunktschreiben unwirksam.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Funktionsweise

Beim Aufruf prüft die Fixpunktroutine den Datenbereich auf Gültigkeit. Die Routine wartet auf Beendigung ausstehender Ein-/Ausgaben und bestimmt, wo in der Datei der Fixpunkt zu schreiben ist. Der Fixpunkt umfasst Kennungsinformationen, Programmzustand, dazu bezogenen Systemzustand und virtuelle Speicherinhalte. Dies wird für den logischen Wiederanlauf eines Programms bei einem Fixpunkt benötigt.

Wenn das Schreiben des Fixpunktes erfolgreich abgeschlossen ist, wird eine Nachricht auf SYSOUT gegeben, die dem Benutzer zum Wiederanlauf dient. Wenn ein Fehler während des Erstellens eines Fixpunktes erkannt wird, wird ein Fehlerschlüssel ausgegeben. Der Fehlerschlüssel wird im Standardheader des Datenbereichs hinterlegt. Die erste Ausgabe der Fixpunktverarbeitung ist eine Anzahl von PAM-Blöcken mit Kennungsinformation, notwendigen Kontrollblöcken, Informationen für das Wiedereröffnen der Datei und die Inhalte des virtuellen Speichers. Die Fixpunktverarbeitung gibt eine Meldung nach SYSOUT aus, wenn der Fixpunkt erfolgreich abgeschlossen wurde. Diese Nachricht verknüpft eine spezifizierte Kennung mit einer Halbseiten-Nummer für die folgende Verwendung beim Wiederanlauf.

Der Benutzer kann den Zeitaufwand bei der Fixpunktverarbeitung reduzieren, wenn er die Speicherplatzzuweisung für die Datei kontrolliert. Die geschätzte Anzahl von PAM-Blöcken für einen einzelnen Fixpunkt kann mit folgender Formel errechnet werden:

$P = 2n + 12$

wobei n die Anzahl der virtuellen Seiten ist, die dem Programm zurzeit des Fixpunktes zugewiesen sind. Es folgt daraus, dass der Fixpunkt dann genommen werden sollte, wenn der Bedarf an Klasse-5- und Klasse-6-Speicher für ein Programm gering ist.

Weiterhin sollte die anfängliche Zuweisung des Speicherplatzes für die Datei groß genug sein, um alle angestrebten Fixpunkte unterzubringen und Sekundärzuweisungen zu vermeiden. Wenn das nicht ausführbar ist, ist als sicherzustellen, dass die Sekundärzuweisung gleich oder größer als die Anforderung für einen Fixpunkt ist.

Bei einem Wiederanlauf (RESTART) an der Stelle eines mit **WRCPT** geschriebenen Fixpunktes wird das Programm mit dem den **WRCPT**-Aufruf folgenden Befehl fortgesetzt. Damit der Anwender die Möglichkeit hat zu entscheiden, ob zuvor die Fixpunktroutine oder die Wiederanlaufroutine aktiv war, wird von der Wiederanlaufroutine der sekundäre Returncode im Byte 5 des Standardheaders auf „R“ gesetzt. Der Anwender kann so entscheiden, ob eine eigene Routine zur Behandlung des Wiederanlaufs durchlaufen werden soll.

Hinweise zum Makroaufruf

- Bei Angabe eines **WRCPT**-Makros werden die vorhandenen Kopien (S.IN.tsn. ..) von Prozedur-/ENTER-Dateien zum LOGOFF-Zeitpunkt nicht gelöscht. Diese Dateien müssen für einen problemlosen Wiederanlauf vorhanden sein.
- Ein Fixpunkt kann nicht gesetzt werden:
  - bei Programmen, die Intertaskkommunikation verwenden.
  - in Programmen, die als Shared-Code laufen, von Shared-Code-Programmen aufgerufen werden oder die Shared-Code-Programme aufrufen.
  - in Programmen, die gemeinsamen Speicherbereich (Memory Pool), Serialisierung, ISAM SHARED UPDATE, UPAM SHARED UPDATE benutzen.
  - wenn die Kommandosprache SDF in Memory Pools im Klasse-5-Speicher geladen ist.
  - wenn RFA-Verbindungen geöffnet sind.
  - in Programmen, die Bänder im Multi-Auftragsverwaltungs-Mode MAV bearbeiten.
- Das RESTART-PROGRAM-Kommando muss in der gleichen Rechner- und Gerätekonfiguration und mit demselben System angegeben werden, in der auch der Makro **WRCPT** aufgerufen wurde.
- Bei Dateigenerationsgruppen, die während des **WRCPT**-Zeitpunktes eröffnet waren, darf der Basiswert zwischen dem **WRCPT**- und dem RESTART-Zeitpunkt nicht verändert werden.
- Zum Fixpunktzeitpunkt aktive AUDIT-Funktionen werden ausgeschaltet.
- Der Zustand der Benutzerdaten wird zum Zeitpunkt des Wiederanlaufs **nicht** automatisch wiederhergestellt. Dafür muss der Benutzer vor dem Wiederanlauf in geeigneter Weise selbst sorgen.
- Während der Fixpunktausgabe kann eine Nutzerdatei von HSMS nicht auf eine Hintergrund-Speicherebene verdrängt werden; das MIGRATE-Bit im Katalogeintrag wird auf den Wert INHIBIT gesetzt (siehe Handbuch „DVS Makros“ [7]).
  Wenn beim Wiederanlauf die Nutzerdatei auf eine Hintergrund-Speicherebene verdrängt ist, muss das RESTART-PROGRAM-Kommando mit dem Operanden FILE-CHANGE=ALLOWED angegeben werden. In diesem Fall wird die Coded File Identification (CFID) nicht geprüft.
- Wird der Fixpunkt in einer Prozedur geschrieben, dann wird nach dem Kommando RESTART-PROGRAM nicht nur das Programm weiter durchlaufen, sondern auch die Prozedur.

Hinweise zur Fixpunktdatei

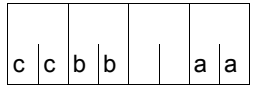
– Liegt die Fixpunktdatei auf gemeinschaftlichem Datenträger, so darf für sie (im FCB des Makroaufrufs) nur die Katalogkennung des eigenen Systems angegeben werden, (siehe auch Handbuch „HIPLEX MSCF“ [26]).

– Die Fixpunktdatei darf nicht auf NK4-Pubsets liegen und muss folgende Attribute haben:
BUF-LEN = STD(1)
BLK-CONTR ≠ DATA

– Die Fixpunktdatei darf nicht auf Net-Storage liegen.

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros WRCPT wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aa=Maincode):

X'cc'	X'aa'	Erläuterung
X'44'	X'00'	Erfolgreiche Fixpunktverarbeitung; Zusatz: Temporäre Dateien waren eröffnet oder Linkname für temporäre Jobvariablen existiert.
	X'04'	Arbeitsbereich konnte nicht zugewiesen werden.
	X'08'	Operandenfehler.
	X'14'	BUF-LEN der Fixpunktdatei ist ≠ STD(1)
X'04' X'08' X'10' X'1C' X'20' X'24' X'28' X'2C' X'30'	X'18'	Fehler beim Eröffnen der Datei; mit den Zusätzen: Kein Arbeitsbereich zur Verfügung. Anzahl der PAM-Request-Blöcke = 0 (für Fixpunktdatei). Remote- bzw. SHARUPD-Zugriffe oder Eventing wird bei Fixpunktverarbeitung nicht unterstützt. Fehler im Zusammenhang mit Jobvariablen. Fehler bei Validierung eines eröffneten FCB. Aus ISAM-Sicht Schreiben eines Fixpunkts nicht möglich. Fehler beim Warten auf die Beendigung einer ausstehende Ein-/Ausgabe für eine SAM-Datei. Fixpunkt kann nicht ausgegeben werden (UPAM) Fixpunkt kann wegen asynchroner Ein-/Ausgaben für eine BTAM-Datei nicht ausgegeben werden.
	X'1C'	Die Fixpunktdatei ist eine Banddatei mit LABEL=NSTD oder LABEL=NO; kein Fixpunkt.
	X'20'	Fehler beim Lesen des Katalogeintrags der Fixpunktdatei.
	X'24'	Memory Pool wird benutzt oder ISAM-Datei wird Shared-Update verarbeitet; kein Fixpunkt.
	X'28'	Teilnahme an Serialisierung; kein Fixpunkt.
	X'2C'	Teilnahme an Ereignissteuerung; kein Fixpunkt.
	X'30'	Contingency-Prozess ist vorhanden; kein Fixpunkt.
	X'34'	DQPAM-Fehler; kein Fixpunkt.
	X'38'	Bandende wurde während der Fixpunktausgabe erreicht; kein Fixpunkt.
	X'40'	Fixpunktausgabe wird im Sicheren System nicht unterstützt; kein Fixpunkt.
	X'48'	Die Fixpunktdatei hat das Format BLK-CONTR=DATA; keine Fixpunktausgabe.
	X'4C'	FASTPAM ist noch aktiv; keine Fixpunktausgabe.

X'cc'	X'aa'	Erläuterung
	X'50'	Die Fixpunktdatei ist keine PAM-Datei.
	X'58'	Fehler beim Zugriff auf die Fixpunktdatei. Der Fehler wird in den Subcodes näher beschrieben. Erläuterung des Fehlercodes (siehe Handbuch „DVS-Einführung“ [8]). Beispiel: Datei existiert nicht zum OPEN-Zeitpunkt (FSTAT-Fehler).
	X'5C'	Falscher Operand beim FILE-Aufruf für die Fixpunktdatei (z.B SHAREUPD=YES).
	X'60'	Die Fixpunktdatei ist bereits geöffnet; keine Fixpunktausgabe.
	X'64'	Makrofehler während der Fixpunktausgabe; keine Fixpunktausgabe.
	X'68'	Keine Fixpunktverarbeitung, wenn mit AID Haltepunkte vereinbart wurden.
	X'6C'	MAREN-Fehler; keine Fixpunktausgabe.
	X'7C'	Data Space wird benutzt.
	X'76'	POSIX ist aktiv.
	X'70'	Interner Fehler bei der PCB-Sicherung; keine Fixpunktausgabe.
	X'74'	Interne Probleme mit SYSFILE-Umgebung; keine Fixpunktausgabe.

Bei erfolgreicher Fixpunktverarbeitung (X'00') wird Subcode1 gelöscht. Die Wiederanlaufroutine setzt Subcode1 auf C'R'.

WRLST – Satz nach SYSLST übertragen

Allgemeines

Anwendungsgebiet: Ein-/Ausgabe von Dateien und Sätzen; siehe Seite 163
Makrotyp: S-Typ, MF-Format **1**:
24-Bit-Schnittstelle: Standardform/E-/L-Form
31-Bit-Schnittstelle: Standardform/E-/L-/C-/D-Form; siehe Seite 29

Die Datei SYSLST ist eine vom Betriebssystem pro Task angelegte temporäre (System-) Datei. Der Inhalt von SYSLST wird bei Taskbeendigung auf den Drucker ausgegeben und die Datei gelöscht. Das Kommando ASSIGN-SYSLST bietet dem Anwender die Möglichkeit, der SYSLST-Datei eine eigene (katalogisierte) Datei, eine S-Variable oder ein Element einer PLAM-Bibliothek zuzuordnen.

Im Gegensatz zur (System-)Datei SYSLST sind die (System-)Dateien SYSLST01 bis SYSLST99 nur wirksam, wenn ihnen katalogisierte Dateien zugeordnet werden (siehe auch Makro **SYSFL**).

Makrobeschreibung

Mit dem Makro **WRLST** kann der Anwender einen Datensatz in die Datei SYSLST oder eine der Dateien SYSLST01 bis SYSLST99 schreiben. Mit jedem Makroaufruf wird ein Datensatz in die spezifizierte Datei übertragen. Der Eintrag erfolgt in die Datei SYSLST, wenn der Operand NUMBER nicht angegeben wird.

Makroaufrufformat und Operandenbeschreibung

```
WRLST

{ satz,fehler[,NUMBER=n][,PARMOD={ 24 }][ } }
{          { 31 }]               }
{ (1)                            }
[,MF=(D,pre) / D / I / (E,..) / L / C / (C,pre)]]
```

satz
symbolische Adresse des Datensatzes, der übertragen werden soll. Der Satz beginnt mit dem Satzlängenfeld, gefolgt von dem Vorschubsteuerzeichen und den zu übertragenden Daten.

Satzaufbau (Beispiel):

```
satz     DC     Y (satzend-satz)
         DS     CL2              reservierte Byte
         DC     X'nm'            Druckvorschubzeichen
daten    DC     C'datensatz'     zu übertragender Datensatz
satzend  EQU    *
```

Hinweis

Das Druckvorschubzeichen gibt den gewünschten Papiervorschub auf dem Drucker an. Die gültigen Druckvorschubzeichen sind:

X'4n' Zeilenvorschub um n vor dem Drucken und 1 Zeile nach dem Drucken. Der Bereich von n ist $(0 - F)_{16}$; das entspricht einem Vorschub von maximal 16 Zeilen. Mit n = 0 wird eine Zeile nach dem Drucken vorgeschoben.

X'0n' Vorschieben um n Zeilen nach dem Drucken. Der Bereich von n ist $(0 - F)_{16}$; das entspricht einem Vorschub von maximal 15 Zeilen (n = 0 entspricht keinem Vorschub).

X'Cn' Springe unmittelbar zu Kanal n der Lochstreifensteuerung des Druckers. Der Bereich von n ist $(1 - B)_{16}$.

X'8n' Drucke und springe zu Kanal n der Lochstreifensteuerung des Druckers. Der Bereich von n ist $(1 - B)_{16}$.

fehler
symbolische Adresse zu der verzweigt wird, wenn einer der folgenden Fehler bei der Makroausführung auftritt (s. Rückinformation):
– nicht behebbarer Fehler
– Operandenfehler
– Satz wird abgeschnitten
– Speicherplatz
Im Fehlerfall enthält Register R14 die Adresse des dem **WRLST**-Aufruf folgenden Befehls. Der Fehlercode wird im Register R15 übergeben.
31-Bit-Schnittstelle: Bei Angabe fehler = 0 (Adresse X'00..0') wird das Programm mit dem den **WRLST**-Makro folgenden Befehl fortgesetzt.

NUMBER=
bezeichnet eine der Dateien SYSLST01 bis SYSLST99.

n
(zweistellige) Zahl aus der Menge (01,02, ..., 99).

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert. Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

(1)
Register R1 enthält die Adresse des Datenbereichs. Die Liste ist auf Wortgrenze auszurichten.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der D-Form des Makroaufrufs kann ein Präfix (pre = 1..3 Buchstaben), wie im Aufrufformat dargestellt, angegeben werden.
Es gibt keine Voreinstellung.

Mit MF=I wird der Standardheader initialisiert.

Bei MF=E wird der Standardheader im Datenbereich nicht automatisch versorgt. Die Adresse des Datenbereichs muss in Register R1 übergeben werden. Die Liste ist auf Wortgrenze auszurichten; Aufbau siehe unten.

Aufbau des Datenbereichs

Adressierungs-Modus	Byte	Inhalt
24-Bit-Modus	0 1 - 3 4 - 7	SYSLSTn (n = Nummer der SYSLST-Datei). Adresse des zu übertragenden Satzes (Operand satz). Adresse, zu der im Fehlerfall verzweigt wird (Operand fehler).
31-Bit-Modus	0 - 7 8 - 11 12 - 15 16	Standardheader. Aufbau: Byte 0-1: X'0024' 2 : X'01' 3 : X'01' 4-7 : Returncode Adresse, zu der im Fehlerfall verzweigt wird (Operand fehler). Adresse des zu übertragenden Satzes (Operand satz). SYSLSTn (Nummer der SYSLST-Datei; Operand NUMBER)

Hinweise zum Makroaufruf

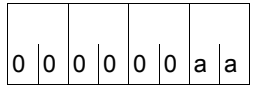
– Bei normaler Beendigung wird mit dem der Makroauflösung folgenden Befehl fortgesetzt. Ein Abschneiden tritt bei Überschreiten der Länge von 137 oder 165 Byte auf (4 Byte Satzlängenfeld + 1 Byte Vorschubsteuerzeichen + 132 bzw. 160 Zeichen) gemäß dem im Makro **SYSFL** angegebenen Druckertyp. Register R14 enthält die Adresse des dem Makroaufruf folgenden Befehls. Register R15 wird nicht verändert, wenn kein Fehler auftritt.

– Wenn SYSLST nicht katalogisiert wurde, geschieht Folgendes: SYSLST wird nach Auftragsbeendigung auf den Drucker ausgegeben. Danach wird die SYSLST-Datei gelöscht.

– Wird SYSLST bzw. SYSLSTn einer katalogisierten Datei zugeordnet, kann diese auch langfristig bearbeitet werden. Der Benutzer ist in diesem Fall selbst für das Ausdrucken bzw. das Löschen der Datei verantwortlich.

Rückinformation und Fehleranzeigen

R15:



Über die Ausführung des Makros WRLST wird im Register R15 ein Returncode übergeben.

X'aa'	Erläuterung
X'04'	Nicht behebbarer Fehler
X'08'	Operandenfehler
X'0C'	Abschneiden des zu schreibenden Satzes. Satzinhalt (ohne Satzlängenfeld) überschreitet die Größe des Ein-/Ausgabepuffers: 2040 Standardfall, 2044 bei Ausgabe in eine katalogisierte Datei.
X'10'	Kein weiterer Speicherplatz für die SYSLSTn zugeordnete Datei. Die letzte PAM-Seite wurde zugewiesen und kann noch beschrieben werden.

31-Bit-Schnittstelle:

Im Standardheader wird kein Returncode übergeben. Bei fehlerhafter Versorgung des Feldes UNIT im Standardheader wird ein Userdump erzeugt.
Die zwei linksbündigen Byte des Registers R15 werden nicht versorgt.

Wird SYSLST einer katalogisierten Datei zugeordnet, die keine sekundäre Speicherplatzanforderung (S-ALLOC=0 im Dateikatalog) erlaubt, und es ist kein weiterer Speicherplatz in dieser Datei verfügbar, wird der Returncode X'10' zurückgegeben.

WROUT – Satz nach SYSOUT übertragen

Allgemeines

Anwendungsgebiete: Ein-/Ausgabe von Dateien und Sätzen; siehe Seite 163
Verkehr mit Datenstationen; siehe Seite 164
Kommunikation; siehe Seite 167

Makrotyp: S-Typ, MF-Format **1**:
24-Bit-Schnittstelle: Standardform/E-/L-Form
31-Bit-Schnittstelle: Standardform/E-/L-/C-/D-Form; siehe Seite 29

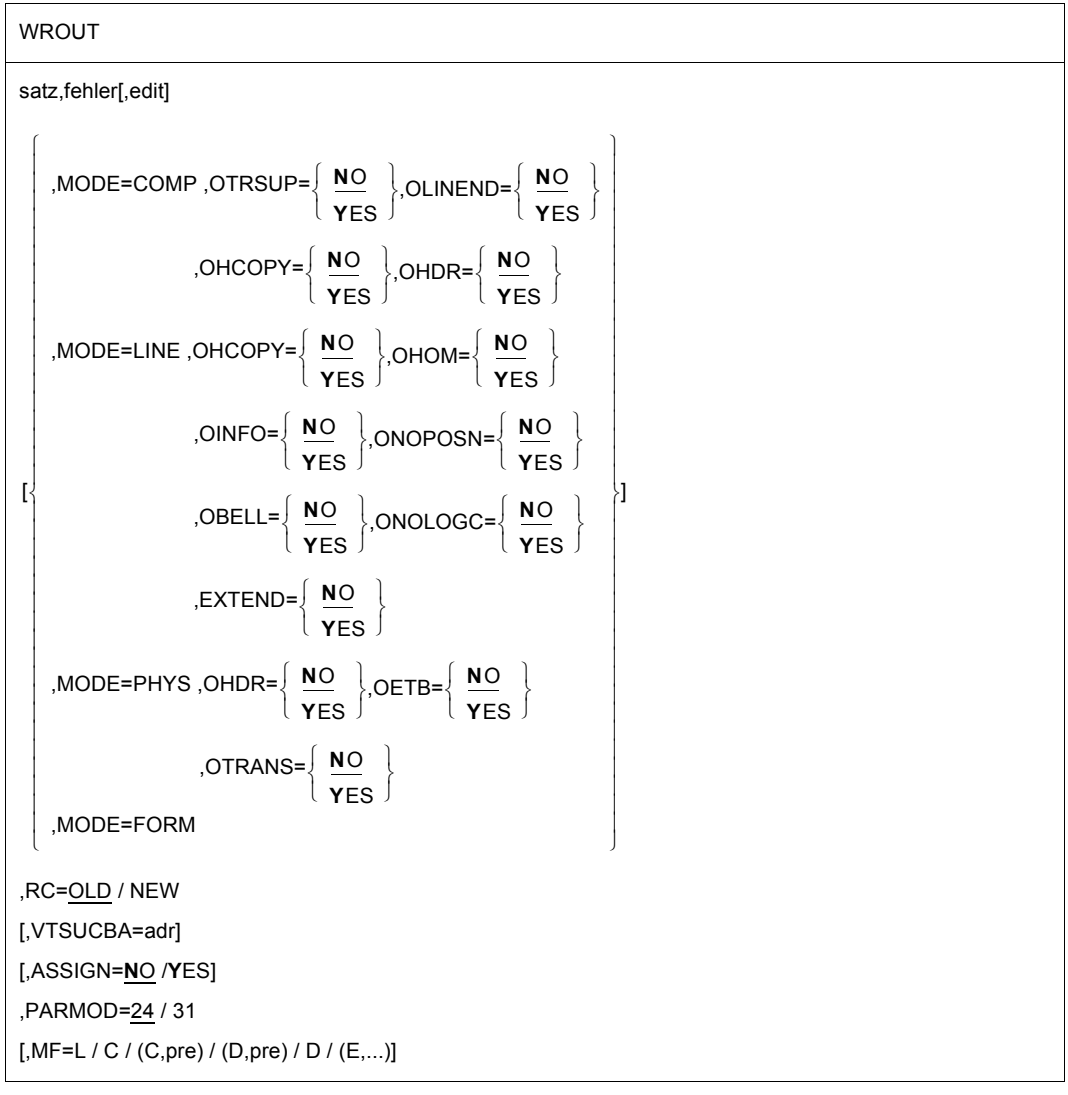
- Stand der Beschreibung: TIAM V13.2A
- Für die 31-Bit-Schnittstelle zu beachten:
  – Bei Verwendung von MF=C/D werden für den Standardheader keine symbolische Namen und Equates erzeugt. Bei dynamischer Versorgung des Datenbereichs sollten die Initialisierungswerte für den Standardheader aus einem mit MF=L erzeugten Datenbereich übernommen werden.
  – Im Standardheader wird kein Returncode übergeben.
- Der Makro **CUPAB** generiert eine DSECT des Datenbereichs des **WROUT** für den Aufruf im 24-Bit-Adressierungsmodus.

Makrobeschreibung

Mit dem Makro **WROUT** kann eine Nachricht in die Datei SYSOUT übertragen werden. SYSOUT kann einem Element einer PLAM-Bibliothek, einer S-Variablen, einer katalogisierten SAM- oder ISAM-Datei, insbesondere der prozessführenden Datenstation zugeordnet sein.

Wird der Operand LOGGING=PARAMETERS (LISTING=YES) im Kommando SET-LOGON-PARAMETERS oder MODIFY-JOB-OPTIONS angegeben, so werden die Ausgaben nach SYSOUT auch nach SYSLST übertragen.

Makroaufrufformat 1 und Operandenbeschreibung



satz
Symbolische Adresse des auszugebenden Datensatzes. Der Satz beginnt mit dem Satzlängenfeld, gefolgt von dem Vorschubsteuerzeichen und der auszugebenden Nachricht.

Satzaufbau und Beispiel:

Byte 0-1: Länge der Nachricht + 4 Byte Satzlängenfeld, die Länge muss > 5 sein.
Byte 2-3: reserviert
Byte 4: Vorschubsteuerzeichen; wird nur bei Ausgabe auf Drucker ausgewertet
Byte 5-n: Nachricht

```
satz     DC    Y (satzend-satz)
         DS    CL2               reservierte Byte
         DC    X'nm'             Druckvorschubzeichen
         DC    nachricht'        zu übertragende Nachricht
satzend  EQU   *
```

Hinweis

Wenn SYSOUT eine katalogisierte Datei ist, so werden Sätze, die größer als 2044 bzw. 2032 Byte sind, abgeschnitten (Returncode X'0C') und nur der erste Teil des Satzes in die Datei geschrieben. Die unterschiedliche maximale Ausgabelänge ist abhängig von der Pamkey-Verwendung (FORMAT=NONKEY/KEY).

fehler
Symbolische Adresse einer Fehlerroutine im Benutzerprogramm, zu der im Fehlerfall verzweigt wird.
Im Fehlerfall enthält Register R14 die Adresse des Befehls, der dem **WROUT**-Aufruf folgt. Der Fehlercode wird im Register R15 übergeben.

31-Bit-Schnittstelle: Bei Angabe fehler = 0 (Adresse X'00..0') wird das Programm mit dem Befehl fortgesetzt, der dem **WROUT**-Aufruf folgt.

edit
Aufbereitungsfunktion (Edit-Options) für eine Nachricht, die zur Datenstation übertragen werden soll. Die Aufbereitung wird übergangen, wenn SYSOUT keine Datenstation ist. Dieser Operand ist nicht erforderlich, wenn Standardfunktionen (alle Edit-Bits=0) verwendet werden, bei einer MODE-Angabe oder bei Nutzung des VTSU-Control-Blocks. Durch Direktangabe (X'xx') kann nur das 1. Edit-Byte für Ausgabe auf die beim **CUPAB**-Makro beschriebene Bedeutung gesetzt werden.

Hinweis

Dieser Operand wird nur noch aus Kompatibilitätsgründen unterstützt. Die Edit-Byte sollten über MODE-Angaben (siehe Operand MODE) oder über den VTSU-Control-Block (siehe Operand VTSUCBA) gesteuert werden.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form und D-Form des Makroaufrufs kann ein Präfix (pre = 1..3 Buchstaben), wie im Aufrufformat dargestellt, angegeben werden. Voreinstellung: pre = CUW

PARMOD=
Steuert die Makroauflösung. Es wird entweder die 24-Bit-Schnittstelle oder die 31-Bit-Schnittstelle generiert. Wenn PARMOD nicht angegeben wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für Assembler (= 24-Bit-Adressierung).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2GB).
Datenlisten beginnen mit dem Standardheader.

Die MODE-Angaben zusammen mit den Edit-Optionen werden nur noch aus Kompatibilitätsgründen unterstützt. Sie werden jetzt im VTSU-Control-Block (VTSUCB, siehe Makro **VTSUCB**) zusammengefasst.

MODE=
Dieser Operand wird nur ausgewertet, wenn SYSDTA der Datensichtstation zugeordnet ist.

COMP
Kompatibler Betriebsmodus. Vom Benutzerprogramm können sämtliche Edit-Optionen über die symbolischen Operanden OTRSUP bis OHDR (siehe unten) verwendet werden. Eventuell direkt im Operanden edit gemachte Angaben werden ignoriert. Steuerzeichen in der Ausgabenachricht sind zulässig, werden aber vom System nicht auf Richtigkeit geprüft. Diese Betriebsweise ist kompatibel zu früheren Versionen des Betriebssystems.
Für die Geräte 8160, 8162, 9749, 975x, 9763, 3270 und X.29-Datenstationen wird dieser Modus wie MODE=LINE behandelt. Die Edit-Option OLINEND wird ignoriert. Die Edit-Optionen OTRSUP und OHDR werden abgewiesen (RC: X'08').

LINE
Die aktuelle Datenstation soll als logische Zeilen- bzw. Seiten-Datenstation behandelt werden. Die Nachricht kann durch logische Steuerzeichen strukturiert werden (siehe Makro **VTCSET**). Ist SYSOUT keine Datenstation, so werden nur die logischen Steuerzeichen NL und NP ausgewertet, z.B bei Ausgaben auf Drucker im Batch-Betrieb. Weitere Steuerzeichen sind unzulässig und werden vom System in ein vom Benutzer definiertes Ersatzzeichen umgewandelt (siehe Kommando MODIFY-TERMINAL-OPTIONS SUBSTITUTE-CHARACTER=).

FORM
Format-Modus. Das Benutzerprogramm arbeitet mit der Software-Komponente FHS bzw. Formatsteuerung, die auch die datenstationsgerechte Aufbereitung der Ausgabenachricht vornimmt.

PHYS
Die Nachricht soll physikalisch, d.h. ohne Aufbereitung durch das System an die Datenstation ausgegeben werden. Damit können spezielle Gerätefunktionen angesprochen werden, für die der LINE- oder FORM-Modus nicht ausreicht. Wird keine der zulässigen Edit-Optionen angegeben, stellt das System der Nachricht einen gerätespezifischen Standard-Nachrichtenkopf voran.

EXTEND=
legt fest, ob die Felder für Ausgabetexte geschützt oder ungeschützt angelegt werden.

NO
Die Bedienerführung erfolgt durch das System. Geschützt gegen Überschreiben ist nur die Eingabeaufforderung durch das System bzw. das Anwenderprogramm.

Ausgaben erfolgen ungeschützt und halbhell.

NIL-Zeichen im Ausgabetext werden in das Ersatzzeichen umgewandelt, bei Eingabe entfernt.

Je nach Betriebsart wird am Anfang der Ausgabe der Bildschirm ab der Schreibmarke gelöscht.

YES
(Nur für Datenstationen 9749, 975x, 9763, 816x und 3270)
Diese Angabe unterstützt die Verwendung von geschützten und ungeschützten Feldern mit den logischen Steuerzeichen EPA, NUM und SPA (siehe Makro **VTCSET**).

Die Ausgabe des Textes erfolgt standardmäßig geschützt und halbhell. Die Nachricht kann durch logische Steuerzeichen strukturiert werden (siehe Makro **VTCSET**). Bei 3270-Datenstationen ist zu beachten, dass die logischen Steuerzeichen Platz auf dem Bildschirm beanspruchen. Mehrere logische Steuerzeichen hintereinander benötigen aber nur einen Platz. Bereiche, in die der Terminalbediener eingeben kann, werden mit EPA oder NUM begonnen und mit SPA beendet.

Bei Ein- und Ausgabe wird NIL als erlaubtes Zeichen behandelt, es wird vom Programm zur Datenstation und umgekehrt geschickt. Bei 3270-Datenstationen ist zu beachten, dass NIL-Zeichen nicht zur DVA übertragen werden. Felder, die bei der Eingabe verkürzt zurückkommen, werden durch NIL-Zeichen auf ihre ursprüngliche Länge ergänzt. Dadurch bekommt der Anwender die Felder immer in der Ausgabelänge zurück.

Der Anfang einer Ausgabenachricht wird an dem der Schreibmarke folgenden nächsten Zeilenanfang abgebildet. Vor dem 1.Textzeichen wird der Schirm ab Schreibmarke gelöscht, wenn die Nachricht nicht mit VPA beginnt.
Wird bei der Ausgabe das Bildschirmende erreicht, so wird am Schirmanfang fortgesetzt. Diese Fortsetzung ist in jedem Fall ungeschützt. Die Bildschirmüberlaufkontrolle ist unwirksam.
Die Tasten RU, EFZ, AFZ und LSP sind gesperrt. Alle anderen Edit-Options außer OBELL, ILCASE, IGETFC werden ignoriert (siehe Programmierhinweise).

OBELL=
bestimmt, ob bei der Ausgabe ein akustisches Signal ertönt.

NO
Bei der Ausgabe ertönt kein akustisches Signal.

YES
Bei der Ausgabe ertönt am Ende der Nachricht ein akustisches Signal (nur bei den Datenstationen 9749, 975x, 9763 816x und bei 3270 mit einem speziellen Gerätezusatz).

OETB=
bestimmt für die Ausgabenachricht das abschließende Steuerzeichen.

NO
Die Ausgabenachricht zur Datenstation wird mit dem Steuerzeichen ETX abgeschlossen.

YES
Die Ausgabenachricht zur Datenstation wird mit dem Steuerzeichen ETB abgeschlossen.

OHCOPY=
legt fest, ob die Ausgabenachricht nicht nur auf eine Datensichtstation, sondern auch über ein angeschlossenes Hardcopy-Gerät (Drucker) ausgegeben werden soll.

NO
Die Ausgabenachricht wird nur über die Datensichtstation ausgegeben.

YES
Die Ausgabenachricht für eine Datensichtstation wird gleichzeitig über ein dort angeschlossenes Hardcopy-Gerät (Drucker) ausgedruckt. Das Hardcopy-Gerät muss generiert werden oder mittels **TCHNG** zugewiesen werden.

Bei 3270-Datenstationen:
Der Inhalt des gesamten Bildschirmes wird über das Hardcopy-Gerät ausgedruckt. Dadurch werden eventuell auch frühere Aus- und Eingaben ausgedruckt. Bei mehreren direkt aufeinander folgenden Ausgaben wird die Hardcopy-Funktion nur bei der letzten Ausgabe ausgelöst. Der Hardcopy-Abdruck erfolgt nur dann, wenn für die Datensichtstation beim Verbindungsaufbau ein Hardcopy-Gerät generiert wurde.

Wurde OINFO=YES oder EXTEND=YES angegeben, erfolgt kein Hardcopy-Abdruck.
Wird OHCOPY=YES verwendet und enthält die Nachricht das logische Steuerzeichen SPA, EPA, CHS oder NUM, so wird nicht die gesamte Nachricht, sondern nur der letzte ungeschützte Teil der Nachricht abgedruckt.
Wird gleichzeitig OVERFLOW-CONTROL=NO (Kommando MODIFY-TERMINAL-OPTIONS) verwendet, kann es vorkommen, dass nur ein Teil der Ausgabe auf dem Hardcopy-Gerät wiedergegeben wird.

OHDR=
gibt an, wie das System mit dem Nachrichtenkopf verfahren soll.

NO
Der Nachrichtenkopf (im US-ASCII-Code) wird dem Ausgabetext nicht vorangestellt.

YES
Die Nachricht enthält einen benutzerindividuellen Nachrichtenkopf, den das System dem Ausgabetext voranstellt. Die Länge des Nachrichtenkopfes plus 1 muss in Byte 5 der Nachricht binär angegeben werden. Der Nachrichtenkopf ist im US-ASCII-Code anzugeben.

Bei 3270-Datenstationen:
Der Nachrichtenkopf wird im EBCDIC-Code angegeben und besteht aus dem CMD-Byte und dem WCC-Byte. Vor diesem Nachrichtenkopf muss ein Byte mit dem Inhalt X'01' gesetzt werden (Länge des TRANSDATA-Nachrichtenkopfes + 1).

Hinweis
Bei Ausgabe auf die Datensichtstationen 8160, 975x, 9763 und daran lokal angeschlossene Drucker ist zu beachten, dass das System (MODE=LINE bzw. COMP) oder FHS mit MODE=FORM keinen Nachrichtenkopf verwendet (PARAMO, PARAM1), sondern mit Operandenangaben (PAG) arbeitet. Die Unterschiede zwischen diesen beiden Arbeitsweisen sind in den Handbüchern der Datensichtstationen bzw. Drucker beschrieben.

OHOM=
gibt an, ob die Nachricht strukturiert oder homogen ausgegeben werden soll.

NO
Die Nachricht soll strukturiert, nicht homogen ausgegeben werden, d.h. als Ausgabeeinheit wird eine logische Zeile betrachtet. Die Nachrichtenlänge ist nicht beschränkt, falls die logischen Zeilen nicht länger als 255 Zeichen sind.

Wirkung bei Betriebsart 1:
Einzelne logische Zeilen können getrennt modifiziert und damit gezielt zurückübertragen werden.

YES
Nur bei Datensichtstationen 816x, 9749, 975x und 9763 anzuwenden.

Die Nachricht soll unstrukturiert, homogen ausgegeben werden, d.h. die gesamte Nachricht wird als eine Ausgabeeinheit betrachtet. Die Nachrichtenlänge ist durch die Größe des Ausgabepuffers im System beschränkt.

Wirkung bei Betriebsart 1:
Durch Modifikation eines Zeichens einer Ausgabenachricht kann die gesamte Nachricht wieder zurückübertragen werden, sofern diese keine logischen Anzeigesteuerzeichen enthält.

Hinweis

Ist SYSOUT einer Datei zugeordnet, wird der Makro **WROUT** nicht ausgeführt und die SYSLST-Protokollierung unterdrückt.

OINFO=

legt fest, ob die Nachricht in einer speziellen Informationszeile ausgegeben werden soll.

NO

Die Nachricht wird nicht in der speziellen Informationszeile ausgegeben.

YES

Die Nachricht kann in einer speziellen Informationszeile abgebildet werden, ohne an der Datenstation wichtige Daten zu zerstören.

Die Angabe ist vor allem für die Benutzerprogramme gedacht, die „asynchron“ Nachrichten an Datenstationen senden, ohne die aktuelle Datenstationsanzeige zu kennen. Die Abbildung erfolgt:

– geschützt in einer Hardware-Anzeigezeile (z.B. DSS 9749, 9750, 9752) oder
– geschützt in der letzten Bildschirmzeile (z.B. DSS 816x, 9751, 9753, 3270), wenn es im Benutzerprogramm festgelegt wird (siehe Makroaufruf **TCHNG**, Operand INFOLIN), nach vorhergegangener Ausgabe mit MODE=FORM oder MODE=PHYS.
– in allen anderen Fällen, wie eine normale Line-Mode-Nachricht.

Ist die Nachricht länger als eine Bildschirmzeile, wird sie aufgeteilt und Zeile für Zeile ausgegeben. Das System berücksichtigt dabei die mit MODIFY-TERMINAL-OPTIONS OVERFLOW-CONTROL=TIME() eingestellte Wartezeit.

Bei OINFO=YES wird die Angabe OHCOPY=YES ignoriert, d.h. es werden weder die Informationszeile noch der Bildschirminhalt abgedruckt.

Erst nach der nächsten Eingabe, auf die eine Ausgabe folgt, wird die Hardware-Anzeigezeile zurückgesetzt.

OLINEND=

gibt an, wie mit den Wagenrücklauf-/Zeilenvorschubzeichen verfahren wird.

NO

Jede Ausgabenachricht beginnt bei einer Datenstation immer mit einer neuen Zeile. Die dafür notwendigen Steuerzeichen werden vom System der Nachricht vorangestellt bzw. bei Erreichen des physikalischen Zeilenendes in die Nachricht eingeschoben, wenn es die betreffende Datenstation erforderlich macht.

YES

Die Ausgabe der Nachricht zur Datenstation erfolgt ohne vom System bereitgestellte Steuerzeichen für Wagenrücklauf/ Zeilenvorschub. Die Steuerung muss vom Benutzerprogramm vorgenommen werden.

ONOLOGC=
gibt an, ob logische Steuerzeichen ausgewertet werden sollen.

NO
Alle logischen Steuerzeichen werden ausgewertet und spezielle physikalische Steuerzeichen werden durchgelassen (siehe Makro **VTCSET** z.B. ESC, DC4). Andere Zeichen < X'40' werden durch SUB ersetzt. Zeichen ≥ X'40' werden durchgelassen.

YES
Logische Steuerzeichen werden nicht ausgewertet. Alle Zeichen, die im EBCDIC-Code < als X'40' sind, werden durch SUB ersetzt.
Nur Zeichen ≥ X'40' werden durchgelassen.

ONOPOSN=
bestimmt, wo die Nachricht beginnen soll.

NO
Die Nachricht beginnt am Anfang der nächsten Zeile.

YES
Die Nachricht beginnt am Anfang der aktuellen Zeile (gilt nur für Schreibstationen).

OTRANS=
gibt an, ob die Ausgabedaten normiert oder transparent übertragen werden sollen.

NO
Die Ausgabedaten sollen normiert übertragen werden, d.h. es findet eine Codeumsetzung statt.

YES
Die Ausgabedaten sollen transparent übertragen werden, d.h. sie bestehen aus beliebigen Binärzeichen (je nach Gerätecode aus 5, 7 oder 8 Bit pro Zeichen), die auf dem Übertragungsweg nicht umgewandelt werden. Ist der Übertragungsweg nicht „potenziell transparent“ generiert, wird die Ausgabe mit RC:X'04' abgewiesen.

OTRSUP=
gibt an, ob die Übersetzung von Gerätecode in EBCDIC unterdrückt werden soll.

NO
Die Übersetzung der Nachricht von EBCDIC in Gerätecode wird nicht unterdrückt, d.h. das Programm liefert die Nachricht in EBCDIC-Code. Im System wird die Nachricht in den Gerätecode übersetzt.

YES
Die Übersetzung der Nachricht wird unterdrückt. Das Programm muss in diesem Fall die Nachricht im Gerätecode liefern.

RC=
legt fest, wo der Returncode abgelegt wird.
Dieser Operand ist nur für eine 31-Bit-Schnittstelle zulässig.

OLD
Der Returncode wird im rechtsbündigen Byte des Registers R15 abgelegt.

NEW
Der Returncode wird sowohl im Register R15 als auch im Standardheader abgelegt. Alle 4 Byte des Registers R15 sind für die Auswertung belegt. Ein 4 Byte-Returncode wird nur zurückgeliefert, wenn SYSDTA von der Datensichtstation liest. In allen anderen Fällen wird nur ein 1 Byte-Returncode zurückgeliefert, unabhängig vom Wert des Returncodes.

VTSUCBA=adr
bestimmt die Adresse eines mit MF=L erzeugten VTSUCB.
Bei Benutzung des Operanden VTSUCBA wird der Operand MODE und die folgenden Edit-Optionen ignoriert (im Datenbereich wird ihr Wert auf X'FF' gesetzt). Das bedeutet, dass alle gewünschten Edit-Optionen im VTSUCB angegeben werden müssen.
Die Angabe des Operanden ist nur für die 31-Bit-Schnittstelle zulässig und wird nur ausgewertet, wenn SYSDTA einer Datensichtstation zugeordnet ist. Standardmäßig erfolgt keine Verwendung des VTSUCB.

ASSIGN=
legt fest, ob Änderungen in der SYSOUT-Zuweisung angezeigt werden sollen.
Das Benutzerprogramm wird von der anfänglichen Standardzuweisung und von jeder folgenden Änderung der SYSOUT-Zuweisung über die Fehlerroutine im Benutzerprogramm verständigt. Bei einer erkannten Änderung der SYSOUT-Zuweisung wird der Satz nicht geschrieben.
Dieser Operandenwert ist nur für die 31-Bit-Schnittstelle zulässig.

NO
Änderungen der SYSOUT-Zuweisung sollen nicht angezeigt werden.

YES
Änderungen der SYSOUT-Zuweisung sollen angezeigt werden.

Hinweis

Die SYSOUT-Zuweisung wird in einem Ausgabefeld der Parameterliste eingetragen. Das Benutzerprogramm kann so auf die geänderten Randbedingungen reagieren, eventuell notwendige Umsetzungen des Ausgabesatzes durchführen und dann das Schreiben mit dem korrigierten Satz erneut ausführen.

Programmierhinweise

für die Verwendung des Operanden MODE=LINE mit EXTEND=YES
(für 3270-Datenstationen siehe Anhang)

Mit EXTEND=YES im LINE-Mode kann mit Formaten gearbeitet werden, ohne dass hierbei eine Formatsteuerung benutzt werden muss.

– Soll der Terminalbediener wie mit Formaten arbeiten, so muss die 1. Ausgabe mit NP beginnen, um den Bildschirm zu löschen und mit dem Text in Position (1.1) beginnen zu können.

– Mit NL wird auf den nächsten Zeilenanfang positioniert und der Rest des Bildschirmes gelöscht, mit VPAn auf den Anfang der Zeile n, wobei der Rest des Bildschirmes erhalten bleibt.

– Positionierung innerhalb einer Zeile kann nur durch Text, Zwischenräume oder NIL-Zeichen erfolgen.

– Der Text nach VPAn, NL und CHS wird geschützt und halbhell dargestellt.

– Ungeschützte Felder werden mit „EPA Text SPA“ ausgegeben.
Numerische Felder werden mit „NUM text SPA“ ausgegeben.

– Mit VPAn am Ende der Nachricht kann die Schreibmarke auf den Anfang des ersten ungeschützten Feldes der Zeile n gesetzt werden, wobei der Bildschirminhalt erhalten bleibt. Beginnt in Zeile n kein ungeschütztes Feld, so wird auf den Anfang des ersten ungeschützten Feldes nach Zeile n positioniert.
Wird am Ende der Nachricht kein VPAn angegeben, so wird die Schreibmarke in das erste ungeschützte Feld auf dem Bildschirm gesetzt.

– Folgeausgabe / Bild-Update
Mit NP wird ein neues Bild erzeugt.
Mit VPAn am Anfang der Ausgabe wird Zeile n des Bildes geändert. Mit VPAn können eine oder mehrere Zeilen übersprungen werden. In der aktuellen Zeile wird ab der Schreibmarke ein dunkler Bereich erzeugt, der bis zum Zeilenende bzw. einem davor liegenden Feld reicht. Anschließend wird in die Zeile n positioniert. Am Ende der Änderung sollte die Schreibmarke wieder mit VPAn positioniert werden, da sonst der Bildschirm ab der Schreibmarke gelöscht würde (s.o.). Bei der Verwendung von NL innerhalb einer Änderung wird ebenfalls der Bildschirm ab Schreibmarke gelöscht. Soll dies verhindert werden, muss eine neue Zeile immer mit VPAn angesprungen werden.

Makroaufrufformat 2 und Operandenbeschreibung

WROUT
(1) [,PARMOD=24 / 31

(1)
Register R1 enthält die Adresse des Datenbereichs. Die Liste auf Wortgrenze ausrichten.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

Aufbau des Datenbereichs

Adressierungs-Modus	Byte	Inhalt
24-Bit-Modus	0	Output Edit Byte 1
	1 - 3	Adresse des auszugebenden Datensatzes (Operand satz).
	4	Output Edit Byte 2
	5 - 7	Adresse, zu der im Fehlerfall verzweigt wird (Operand fehler).
31-Bit-Modus	0 - 7	Standardheader. Aufbau siehe Seite 43 Die Initialisierungswerte sind einem mit MF=L erzeugten Datenbereich zu entnehmen. Bei RC=OLD wird kein Returncode im Standardheader übergeben.
	8 - 11	Adresse, zu der im Fehlerfall verzweigt wird (Operand fehler).
	12 - 15	Adresse des auszugebenden Datensatzes (Operand satz).
	16	Output Edit Byte 1
	17	Output Edit Byte 2
	18	reserviert (X'00')
	19	Flag mit Anzeigen für Verwendung des VTSUCB und für Returncode-Verhalten.
	20-23	Adresse des VTSUCB
	24-25	reserviert (X'0000')
	26	Änderung der SYSOUT-Zuweisung soll mitgeteilt werden (Eingabewert)
	27	Änderung der SYSOUT-Zuweisung (Ausgabewert)

Bei Verwendung der 24-Bit-Schnittstelle sind die Werte für Output Edit Byte 1/2 der beim Makro **CUPAB** angegebenen Tabelle zu entnehmen.

Bei Verwendung der 31-Bit-Schnittstelle sind die Werte einer mit MF=C/D erzeugten Liste zu entnehmen.
Dies gilt auch für die Steuerung, ob eine Änderung der SYSOUT-Zuweisung angezeigt werden soll (Byte 26) und für die Werte der SYSOUT-Zuweisung (Byte 27).

Werte für Änderung der SYSOUT-Zuweisung

Wert	Bedeutung
X'00'	Zuordnung für SYSOUT nicht geändert
X'01'	SYSOUT ist einer Datei zugeordnet
X'02'	SYSOUT ist einer Datenstation zugeordnet
X'03'	SYSOUT ist einer S-Variablen zugeordnet
X'04'	SYSOUT ist einem Element einer PLAM-Bibliothek zugeordnet

Hinweis

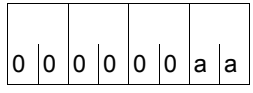
Wenn die Systemdatei SYSOUT eine katalogisierte Datei ist, so werden Sätze, die größer als 2044 bzw. 2032 Byte sind, abgeschnitten (Returncode X'0C') und nur der erste Teil des Satzes in die Datei geschrieben. Die unterschiedliche maximale Ausgabelänge ist abhängig von der Pamkey-Verwendung (FORMAT=NONKEY/KEY).
Bei EAM-Dateien werden Sätze, die größer als 2040 Byte sind, ebenfalls abgeschnitten (Returncode X'0C').

Rückinformation und Fehleranzeigen

Während der Makrobearbeitung enthält Register R1 die Adresse des Datenbereichs.

bei PARMOD=24:

R15:



Über die Ausführung des Makros WROUT wird im Register R15 ein Returncode übergeben.

X'aa'	**Erläuterung**
X'04'	Nicht behebbarer Fehler.
X'08'	Operandenfehler.
X'0C'	Abschneiden des Ausgabesatzes. Satzinhalt (ohne Satzlängenfeld) überschreitet die Größe des Ein-/Ausgabepuffers: bei Datensichtstationen abhängig vom Gerätetyp und der Netzgenerierung, 2044 Byte bei Ausgabe in eine katalogisierte Datei.
X'10'	An der Datenstation wurde „BREAK“ während der Ausführung des Makroaufrufs gegeben.
X'20'	Ungültige Edit Option; wurde vom System korrigiert.
X'38'	Fehler im Zusammenhang mit POSIX.

bei PARMOD=31:

bei **RC=OLD:**

Zusätzlich zu den bei PARMOD=24 beschriebenen Returncodes können die Returncodes

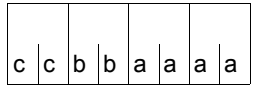
X'aa'	**Erläuterung**
X'00'	Normale Beendigung.
X'24'	Fehler im VTSUCB.

auftreten, sowie die Returncodes, die durch Konvention makroübergreifend festgelegt sind (siehe Tabelle „Standard-Returncodes“ auf Seite 43).

bei **RC=NEW:**

Die Returncodes werden im Standardheader und im Register R15 eingetragen.

Standard-header:



Über die Ausführung des Makros WROUT wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	**X'bb'**	**X'aaaa'**	**Erläuterung**
X'00'	X'00'	X'0000'	Erfolgreiche Bearbeitung der Funktion.
X'00'	X'00'	X'0020'	Erfolgreiche Berabeitung der Funktion, ein aufgetretener Operanden-fehler ist durch TIAM/VTSU korrigiert worden.
X'00'	X'01'	X'0008'	Nicht korrigierter Operandenfehler.
X'07'	X'01'	X'0008'	Nicht korrigierter Operandenfehler: die RESERVED-Felder sind nicht 0.
X'00'	X'20'	X'0004'	Interner Fehler.
X'02'	X'20'	X'0004'	Interner Fehler: BCAM-Nachricht verloren.
X'06'	X'20'	X'0004'	Interner Fehler: negative Transportquittung.
X'00'	X'20'	X'0028'	Interner Fehler: Probleme bei der Speicherzuordnung. SYSOUT ist einer Datei zugeordnet, die bei der Primärzuweisung des Speicherplatzes den Speicher völlig belegt und deren Sekundärzuwei-sung Null ist.
X'00'	X'40'	X'000C'	Ausgabesatz wurde abgeschnitten.
X'00'	X'40'	X'0010'	BREAK während der Ausführung.
X'00'	X'40'	X'0030'	Ein-/Ausgabe abgebrochen.
X'01'	X'80'	X'0004'	Interner BCAM-Engpass.
X'09'	X'80'	X'0038'	Fehler im Zusammenhang mit POSIX: Ein-/Ausgabe-Serialisierungsfehler.
X'0A'	X'40'	X'0038'	Fehler im Zusammenhang mit POSIX: Wenn die LOGON-Task im Systemmodus ist, sind keine Ein-/Ausgaben von mit fork{} erzeugten Prozessen möglich.
		X'24'	VTSU-Fehler. Außer Maincode (rechtes Byte) siehe Fehlerinformation im VTSUCB-Header.

Beispiel 1

```
  WROUT1   START
           PRINT NOGEN
  WROUT1   AMODE 31
  WROUT1   RMODE 24
           BALR  3,0
           USING *,3
           WROUT MESSAGE,ERROR,PARMOD=31
2                *,@DCEO      999    921011   53531004
           TERM
  ERROR    TERM  DUMP=Y
  *
  MESSAGE  DC    Y(ENDMESS-MESSAGE)      Record length
           DS    CL2                     Reserved
           DC    X'01'                   Print feed control character
           DC    'EXAMPLE WROUT 1'       Text
  ENDMESS  EQU   *
           END
```

Ablaufprotokoll:

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,wrout1), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,wrout1))
%  ASS6011 ASSEMBLY TIME: 293 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 79 MSEC
//end
%  ASS6012 END OF ASSEMBH
/start-executable-program library=macexmp.lib,element-or-symbol=wrout1, -
/      prog-mode=*any
%  BLS0523 ELEMENT 'WROUT1', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'WROUT1', VERSION ' ' OF '<date> <time>' LOADED
EXAMPLE WROUT 1
```

Beispiel 2

```
  WROUT2   START
           PRINT NOGEN
           BALR  10,0
           USING *,10
           WROUT MESSAGE,ERROR,PARMOD=31,MODE=LINE,OBELL=Y
2                *,@DCEO      999    921011   53531004
  ERROR    NOP   0
           TERM
  *
  MESSAGE  DC    Y(ENDMESS-MESSAGE)
           DS    3X
           DC    C'Output WROUT 2'
  ENDMESS  EQU   *
           END
```

Beispiel 3

Verwendung des VTSUCB

```
  WROUT3   START
           PRINT NOGEN
           BALR  10,0
           USING *,10
           WROUT  MESSAGE,ERROR,PARMOD=31,VTSUCBA=VTSUPAR
  ERROR    NOP   0
           TERM
  *
  VTSUPAR  VTSUCB MODE=LINE,BELL=YES
1                *,VTSUCB     350    980309
  MESSAGE  DC    Y(ENDMESS-MESSAGE)
           DS    3X
           DC    C'Output WROUT 3'
  ENDMESS  EQU   *
           END
```

Beispiel 4

Verwendung von Format 2

```
  WROUT4   START
           PRINT NOGEN
           BALR  10,0
           USING *,10
           LA    1,PARAM
           WROUT (1),PARMOD=31
  ERROR    NOP   0
           TERM
  *
  PARAM    WROUT MESSAGE,ERROR,MF=L,PARMOD=31,MODE=LINE,OBELL=Y
2                *,@DCEO      999    921011   53531004
  MESSAGE  DC    Y(ENDMESS-MESSAGE)
           DS    3X
           DC    C'Output WROUT 4'
  ENDMESS  EQU   *
           END
```

WRTRD – Kombinierte Ein-/Ausgabe

Allgemeines

Anwendungsgebiete: Ein-/Ausgabe von Dateien und Sätzen; siehe Seite 163
Verkehr mit Datenstationen; siehe Seite 164
Kommunikation; siehe Seite 167

Makrotyp: S-Typ, MF-Format **1**:
24-Bit-Schnittstelle: Standardform/E-/L-Form
31-Bit-Schnittstelle: Standardform/E-/L-/C-/D-Form; siehe Seite 29

- Stand der Beschreibung: TIAM V13.2A
- Für die 31-Bit-Schnittstelle zu beachten:
  - Die C-/D-Form wird aufgerufen mit MF=C/D bzw. MF=(C,p)/(D,p).
    p = Präfix (max. 3 Zeichen); Voreinstellung: p = CUB. Das Präfix verändert nur die Feldnamen (nicht die symbolischen Namen bei den Equates). Ein zu langes Präfix wird auf 3 Zeichen gekürzt.
  - Bei Verwendung von MF=C/D werden für den Standardheader keine symbolischen Namen und Equates erzeugt. Bei dynamischer Versorgung des Datenbereichs sollten die Initialisierungswerte für den Standardheader aus einem mit MF=L erzeugten Datenbereich übernommen werden.
  - Im Standardheader wird kein Returncode übergeben.
- Der Makro **CUPAB** generiert eine DSECT des Datenbereichs des **WRTRD** für die 24-Bit-Schnittstelle.

Makrobeschreibung

WRTRD kann nur im Teilnehmerbetrieb verwendet werden. **WRTRD** sendet eine Nachricht zur Datenstation und liest unmittelbar anschließend eine Nachricht von der Datenstation. Außer der zur Datenstation hin übertragenen Nachricht erscheint dabei kein weiteres Eingabeaufforderungszeichen.
Beim Ablauf des Makros werden im Fall von Format 1 die spezifizierten Operanden in einer Operandentabelle abgespeichert und die Anfangsadresse dieser Tabelle in Register R1 geladen. Im Fall von Format 2 wird die im Anwenderprogramm spezifizierte Tabelle verwendet.

Makroaufrufformat 1 und Operandenbeschreibung

WRTRD

satz1,[edit1],satz2,[edit2],[länge],fehler

[{
,MODE=COMP ,OTRSUP={NO / YES},OLINEND={NO / YES},OHDR={NO / YES},IHDR={NO / YES}
,OHCOPY={NO / YES},ILCASE={NO / YES},IGETBS={NO / YES},ILINEND={NO / YES}
,ITRSUP={NO / YES}
,MODE=LINE ,OHCOPY={NO / YES},OHOM={NO / YES},ONOPOSN={NO / YES},IGETIC={NO / YES}
,OBELL={NO / YES},ONOLOGC={NO / YES},EXTEND={NO / YES},IGETFC={NO / YES}
,ILCASE={NO / YES},IGETBS={NO / YES},ICFD={NO / YES}
,MODE=FORM ,IGETBS={NO / YES},ILCASE={YES / NO}
,MODE=PHYS ,OHDR={NO / YES},OTRANS={NO / YES},OETB={NO / YES},ITRSUP={NO / YES}
,IHDR={YES / NO},ILCASE={NO / YES},IGETBS={NO / YES}
}]

,RC=OLD / NEW

[,VTSUCBA=adr]

[,TIMER=wert]

,PARMOD=24 / 31

[,MF=L / C / (C,pre) / (D,pre) / D / (E,...)]

satz1
symbolische Adresse des auszugebenden Datensatzes. Der Satz beginnt mit dem Satzlängenfeld, gefolgt von einem (beliebigen) Zeichen und der auszugebenden Nachricht.

Satzaufbau und Beispiel:

Byte 0-1: Länge des Satzes + 4 Byte Satzlängenfeld.
Byte 2-3: reserviert
Byte 4: beliebiges Zeichen; wird weder übertragen noch ausgewertet.
Byte 5-n: Datensatz

```
SATZ     DC     Y (SATZEND-SATZ)
         DS     CL2              reservierte Byte
         DC     X'00'
         DC     C'DATENSATZ'     zu übertragender Datensatz
SATZEND  EQU    *
```

edit1
Dieser Operand gibt die Aufbereitung für den zur Datenstation zu übertragenden Satz an. Durch Direktangabe (X'xx') kann hier nur das 1. Edit-Byte für Ausgabe auf die beim **CUPAB**-Makro beschriebene Bedeutung gesetzt werden. Dieser Operand ist nicht erforderlich, wenn Standardfunktionen (alle Edit-Bits = 0) verwendet werden, bei einer MODE-Angabe oder bei Nutzung des VTSU-Control-Blocks.

satz2
symbolische Adresse eines Feldes, in das der von der Datensichtstation eingelesene Datensatz übertragen wird. Der Datensatz wird als Satz variabler Länge eingelesen (die ersten 4 Byte enthalten die Satzlänge).

Satzaufbau und Beispiel:

Byte 0-1: Länge des Satzes + 4 Byte Satzlängenfeld.
Byte 2-3: reserviert
Byte 4-n: Datensatz

```
SATZ2    DS    0CL74
LÄNGE    DS    CL2
RESERV   DS    CL2
DATEN    DS    CL70
```

edit2
gibt die Aufbereitung für den zum Benutzerprogramm zu übertragenden Satz an. Durch Direktangabe (X'xx') kann hier nur das 1. Edit-Byte für Eingabe auf die beim **CUPAB**-Makro beschriebene Bedeutung gesetzt werden. Dieser Operand ist nicht erforderlich, wenn Standardfunktionen (alle Edit-Bits = 0) verwendet werden, bei einer MODE-Angabe oder bei Nutzung des VTSU-Control-Blocks.

Hinweis

Die Operanden edit1 und edit2 werden nur noch aus Kompatibilitätsgründen unterstützt. Die Edit-Bytes sollten über MODE-Angaben oder den VTSU-Control-Block (Operand VTSUCBA) gesteuert werden.

länge
Länge des unter satz2 angegebenen Feldes (einschl. 4 Byte Satzlängenfeld);
5 ≤ länge ≤ 32767.
Bei fehlender Angabe wird das Längenattribut des angegebenen Feldes angenommen.

fehler
symbolische Adresse, zu der im Fehlerfall verzweigt wird. Im Fehlerfall enthält Register R14 die Adresse des dem **WRTRD**-Aufruf folgenden Befehls. Der Fehlercode wird im Register R15 übergeben.
31-Bit-Schnittstelle: Bei Angabe fehler = 0 (Adresse X'00..0') wird das Programm mit dem den **WRTRD**-Makro folgenden Befehl fortgesetzt.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form und D-Form des Makroaufrufs kann ein Präfix (pre = 1..3 Buchstaben), wie im Aufrufformat dargestellt, angegeben werden. Voreinstellung: pre = CUB

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

Die MODE-Angaben zusammen mit den Edit-Optionen werden nur noch aus Kompatibilitätsgründen unterstützt. Sie werden jetzt im VTSU-Control-Block (VTSUCB, siehe Makro **VTSUCB**) zusammengefasst.

MODE=COMP
Kompatibler Betriebsmodus. Vom Benutzerprogramm können sämtliche Edit-Options über die symbolischen Operanden OTRSUP bis OPTAPE (siehe unten) verwendet werden. Eventuell direkt im Operanden edit gemachte Angaben werden ignoriert. Steuerzeichen in der Ausgabenachricht sind zulässig, werden aber vom System nicht auf Richtigkeit geprüft. Diese Betriebsweise ist kompatibel zu früheren Versionen des Betriebssystems.
Für die Geräte 8160, 8162, 9749, 975x, 9763, 3270 und X.29-Datenstationen wird dieser Modus wie MODE=LINE behandelt. Die Edit-Options OLINEND und ILINEND werden ignoriert. Die Edit-Options OTRSUP, OHDR, ITRSUP und IHDR werden abgewiesen (RC: X'08').

LINE
Die aktuelle Datenstation soll als logische Zeilen- bzw. Seiten-Datenstation behandelt werden. Die Nachricht kann durch logische Steuerzeichen strukturiert werden (siehe Makro **VTCSET**).
Für die Ausgabe sind weitere Steuerzeichen unzulässig und werden in ein vom Benutzer definiertes Ersatzzeichen umgewandelt (siehe Kommando MODIFY-TERMINAL-OPTIONS SUBSTITUTE-CHARACTER=). Ist SYSOUT keine Datenstation, so werden nur die logischen Steuerzeichen NL und NP ausgewertet z.B. bei Ausgaben auf Drukker im Batch-Betrieb.
Bei der Eingabe wird der gerätespezifische Nachrichtenkopf nicht mitgeliefert.

FORM
Format-Modus. Das Benutzerprogramm arbeitet mit der Software-Komponente FHS bzw. Formatsteuerung, die auch die datenstationsgerechte Aufbereitung der Ausgabenachricht vornimmt.

PHYS
Die Nachrichten sollen physikalisch, d.h. ohne Aufbereitung durch das System an die Datenstation ausgegeben bzw. von dort eingelesen werden. Damit können spezielle Gerätefunktionen angesprochen werden, für die der LINE- oder FORM-Modus nicht ausreicht. Wird keine der zulässigen Edit-Options angegeben, stellt das System der Ausgabenachricht einen gerätespezifischen Standard-Nachrichtenkopf voran, aus der Eingabenachricht wird der gerätespezifische Nachrichtenkopf nicht entfernt. Kleinbuchstaben werden in Großbuchstaben umgewandelt und evtl. die Backspace-Funktion ausgeführt.

EXTEND=
legt fest, ob die Felder für Ausgabetexte geschützt oder ungeschützt angelegt werden sollen.

NO
Die Bedienerführung erfolgt durch das System. Geschützt gegen Überschreiben ist nur die Eingabeaufforderung durch das System bzw. das Anwenderprogramm.

Die Ausgaben erfolgen ungeschützt und halbhell.

NIL-Zeichen im Ausgabetext werden in das Ersatzzeichen umgewandelt, bei Eingabe entfernt.

Je nach Betriebsart wird am Anfang der Ausgabe der Bildschirm ab der Schreibmarke gelöscht.

YES
(Nur für Datenstationen 9749, 975x, 9763, 816x und 3270)
Diese Angabe unterstützt die Verwendung von geschützten und ungeschützten Feldern mit den logischen Steuerzeichen EPA, DAR, NUM und SPA (siehe Makro **VTCSET**).

Die Ausgabe des Textes erfolgt standardmäßig geschützt und halbhell. Die Nachricht kann durch logische Steuerzeichen strukturiert werden (siehe **VTCSET**). Bei 3270-Datenstationen ist zu beachten, dass die logischen Steuerzeichen Platz auf dem Bildschirm beanspruchen. Mehrere logische Steuerzeichen hintereinander benötigen aber nur einen Platz. Bereiche, in die der Terminalbediener eingeben kann, werden mit EPA, DAR oder NUM begonnen und mit SPA beendet.

Bei Ein- und Ausgabe wird NIL als erlaubtes Zeichen behandelt, es wird vom Programm zur Datenstation und umgekehrt geschickt. Bei 3270-Datenstationen ist zu beachten, dass NIL-Zeichen nicht zur DVA übertragen werden. VTSU-B ergänzt Felder, die bei der Eingabe verkürzt zurückkommen, durch NIL-Zeichen auf ihre ursprüngliche Länge. Dadurch bekommt der Anwender die Felder immer in der Ausgabelänge zurück.

Der Anfang einer Ausgabenachricht wird an dem der Schreibmarke folgenden nächsten Zeilenanfang abgebildet. Vor dem 1.Textzeichen wird der Schirm ab Schreibmarke gelöscht, wenn die Nachricht nicht mit VPA beginnt.
Wird bei der Ausgabe das Bildschirmende erreicht, so wird am Schirmanfang fortgesetzt. Diese Fortsetzung ist in jedem Fall ungeschützt. Die Bildschirmüberlaufkontrolle ist unwirksam.
Die Tasten RU, EFZ, AFZ und LSP sind gesperrt.
Alle anderen Edit-Options außer OBELL, ILCASE, IGETFC werden ignoriert.
Wird in einer Eingabenachricht das Steuerzeichen NL erkannt, so wird die Bearbeitung fortgesetzt und der Returncode X'2C' geliefert.

ICFD=
gibt an, ob vertrauliche Daten geschützt werden sollen.

NO
Es sollen keine Vorkehrungen zum Schutz vertraulicher Daten getroffen werden.

YES
Die Eingabedaten sind vertraulich und sollen an der Datenstation unsichtbar bleiben. Dies erfolgt je nach Datenstation durch Dunkelsteuerung bzw. Löschen des Bildschirms oder durch Überschreiben der Eingabezeile bei Schreibstationen.

IGETBS=
bestimmt, ob „Underline“ (X'6D') in das Benutzerprogramm übertragen wird. Die Angabe des Operanden ist nur für 8103-Datensichtstationen sinnvoll.

NO
„Underline“ wird nicht in das Benutzerprogramm übertragen.
Stattdessen wird vom System die Korrekturfunktion durchgeführt.

YES
Die Zeichen „Underline“ werden dem Benutzerprogramm übergeben. Es findet keine Auswertung durch das System statt.

IGETFC=
bestimmt, ob ein Funktionstastencode übergeben wird.

NO
Es soll kein Funktionstastencode übergeben werden.

YES
Das 1. Byte des Einlesebereichs soll den normierten Funktionstastencode enthalten. Dieser identifiziert die zur Auslösung der Datenübertragung an der Datenstation betätigte Taste. Siehe Tabelle der normierten Funktionstastencodes im Anhang, Seite 1193.

IGETIC=
legt fest, ob die Eingabequelle verändert werden soll.

NO
Die Eingabequelle soll nicht verändert werden.

YES
Die Eingabe soll vom angeschlossenen Ausweisleser erfolgen. Die Eingabedaten können nur aus der Ausweisinformation oder aus dem Kurztelegramm K14 bestehen. Diese Angabe ist nur bei den Datenstationen 9749, 975x, 9763, 816x und 3270 mit einem definierten Ausweisleser möglich (siehe auch Makro **TSTAT** TYPE=TCHAR).

Im Unterschied zu den TRANSDATA-Geräten können bei 3270-Datenstationen jederzeit Daten von einem definierten Ausweisleser eingegeben werden. Wenn Eingaben vom Ausweisleser angefordert werden, wird jede andere Eingabe in K14 umgewandelt.

Hinweis
Der Operand IGETIC wird ignoriert, wenn gleichzeitig der Operand ICFD angegeben wird oder wenn kein Ausweisleser angeschlossen ist. Die Eingabequelle bleibt unverändert.

IHDR=
gibt an, wie mit dem Nachrichtenkopf verfahren wird.

NO
Der Nachrichtenkopf wird nicht an das Benutzerprogramm übergeben.

YES
Der gesamte Nachrichtenkopf wird an das Benutzerprogramm übergeben.
Bei 3270-Datenstationen besteht der Nachrichtenkopf aus dem Anwendungskennzeichen (AID-Byte) und der zwei Byte langen Schreibmarkenposition.

ILCASE=
legt fest, ob zwischen Klein- und Großschreibung unterschieden werden soll.

NO
Alle Kleinbuchstaben werden dem Benutzerprogramm als Großbuchstaben übergeben.

YES
Dem Benutzerprogramm werden auch Kleinbuchstaben übergeben.

ILINEND=
gibt an, wie mit den Wagenrücklauf-/Zeilenvorschubzeichen verfahren wird.

NO
Die Wagenrücklauf-/Zeilenvorschubzeichen werden dem Benutzerprogramm nicht übergeben.

YES
Die Zeichen Wagenrücklauf/Zeilenvorschub werden in das Benutzerprogramm übertragen.

ITRSUP=
gibt an, ob die Übersetzung von Gerätecode in EBCDIC unterdrückt werden soll.

NO
Die Übersetzung von Gerätecode in EBCDIC wird nicht unterdrückt. Das Benutzerprogramm erhält die Nachricht im EBCDIC-Code.

Ausnahme
Der Nachrichtenkopf bei den Datensichtstationen 816x, 9749, 975x und 9763 wird immer im Gerätecode geliefert.

YES
Die Übersetzung von Gerätecode in EBCDIC wird unterdrückt. Das Benutzerprogramm erhält die Nachricht im Gerätecode.

OBELL=
bestimmt, ob bei der Ausgabe ein akustisches Signal ertönt.

NO
Bei der Ausgabe ertönt kein akustisches Signal.

YES
Bei der Ausgabe ertönt am Ende der Nachricht ein akustisches Signal (nur bei Datenstationen 9749, 975x, 9763 816x und bei 3270 mit einem speziellen Gerätezusatz).

OETB=
bestimmt für die Ausgabenachricht das abschließende Steuerzeichen.

NO
Die Ausgabenachricht zur Datenstation wird mit dem Steuerzeichen ETX abgeschlossen.

YES
Die Ausgabenachricht zur Datenstation wird mit dem Steuerzeichen ETB abgeschlossen.

OHCOPY=
legt fest, ob die Ausgabenachricht nicht nur auf eine Datensichtstation, sondern auch über ein angeschlossenes Hardcopy-Gerät (Drucker) ausgegeben werden soll.

NO
Die Ausgabenachricht wird nur über die Datensichtstation ausgegeben.

YES
Die Ausgabenachricht für eine Datensichtstation wird gleichzeitig über ein dort angeschlossenes Hardcopy-Gerät (Drucker) ausgedruckt. Das Hardcopy-Gerät muss generiert werden oder mit dem Kommando MODIFY-TERMINAL-OPTIONS zugewiesen werden.

Bei 3270-Datenstationen:
Der Inhalt des gesamten Bildschirmes wird über das Hardcopy-Gerät ausgedruckt. Dadurch werden eventuell auch frühere Aus- und Eingaben ausgedruckt. Bei mehreren direkt aufeinander folgenden Ausgaben wird die Hardcopy-Funktion nur bei der letzten Ausgabe ausgelöst. Der Hardcopy-Abdruck erfolgt nur dann, wenn für die Datensichtstation beim Verbindungsaufbau ein Hardcopy-Gerät generiert wurde.

Wurde EXTEND=YES oder MODE=EXTEND angegeben, erfolgt kein Hardcopy-Abdruck.

Wird OHCOPY=YES verwendet und enthält die Nachricht das logische Steuerzeichen SPA, EPA, CHS oder NUM, so wird nicht die gesamte Nachricht, sondern nur der letzte ungeschützte Teil der Nachricht abgedruckt.

Wird gleichzeitig OVERFLOW-CONTROL=NO (im Kommando MODIFY-TERMINAL-OPTIONS) verwendet, kann es vorkommen, dass nur ein Teil der Ausgabe auf dem Hardcopy-Gerät wiedergegeben wird.

OHDR=
gibt an, wie das System mit dem Nachrichtenkopf verfahren soll.

NO
Der Nachrichtenkopf (im US-ASCII-Code) wird dem Ausgabetext nicht vorangestellt.

YES
Die Nachricht enthält einen benutzerindividuellen Nachrichtenkopf (Der Nachrichtenkopf ist im US-ASCII-Code anzugeben), den das System dem Ausgabetext voranstellt. Die Länge des Nachrichtenkopfes +1 muss inm ersten Byte der Nachricht binär angegeben werden.

Bei 3270-Datenstationen:
Der Nachrichtenkopf wird im EBCDIC-Code angegeben und besteht aus dem CMD-Byte und dem WCC-Byte. Vor diesem Nachrichtenkopf muss ein Byte mit dem Inhalt X'01' gesetzt werden (Länge des TRANSDATA-Nachrichtenkopfes + 1).

Hinweis
Bei Ausgabe auf die Datensichtstationen 8160, 975x, 9763 und daran lokal angeschlossene Drucker ist zu beachten, dass das System (MODE=LINE bzw. COMP) oder FHS mit MODE=FORM keinen Nachrichtenkopf verwendet (PARAMO, PARAM1), sondern mit Operandenangaben (PAG) arbeitet. Die Unterschiede zwischen diesen beiden Arbeitsweisen finden Sie in den Handbüchern der Datensichtstationen bzw. Drucker.

OHOM=
gibt an, ob die Nachricht strukturiert oder homogen ausgegeben werden soll.

NO
Die Nachricht soll strukturiert, nicht homogen ausgegeben werden, d.h. als Ausgabeeinheit wird eine logische Zeile betrachtet. Die Nachrichtenlänge ist nicht beschränkt, falls die logischen Zeilen nicht länger als 255 255 Zeichen sind.

Wirkung bei Datenstationen 816x, 975x, 9763 und 3270 bei Betriebsart 1:
Einzelne logische Zeilen können getrennt modifiziert und damit gezielt zurückübertragen werden.

YES
Nur bei Datensichtstationen 816x, 9749, 975x, 9763 und 3270 anzuwenden.

Die Nachricht soll unstrukturiert, homogen ausgegeben werden, d.h. die gesamte Nachricht wird als eine Ausgabeeinheit betrachtet. Die Nachrichtenlänge ist durch die Größe des Ausgabepuffers im System beschränkt.

Wirkung bei Datensichtstationen 816x, 975x, 9763 und 3270 bei Betriebsart 1:
Durch Modifikation eines Zeichens einer Ausgabenachricht kann die gesamte Nachricht wieder zurückübertragen werden, sofern diese keine logischen Anzeigesteuerzeichen enthält.

OLINEND=
gibt an, wie mit den Wagenrücklauf-/Zeilenvorschubzeichen verfahren wird.

NO
Jede Ausgabenachricht beginnt bei einer Datenstation immer mit einer neuen Zeile. Die dafür notwendigen Steuerzeichen werden vom System der Nachricht vorangestellt bzw. bei Erreichen des physikalischen Zeilenendes in die Nachricht eingeschoben, wenn es die betreffende Datenstation erforderlich macht.

YES
Die Ausgabe der Nachricht zur Datenstation erfolgt ohne vom System bereitgestellte Steuerzeichen für Wagenrücklauf/Zeilenvorschub. Die Steuerung muss vom Benutzerprogramm vorgenommen werden.

ONOLOGC=
gibt an, ob logische Steuerzeichen ausgewertet werden sollen.

NO
Alle logischen Steuerzeichen werden ausgewertet und spezielle physikalische Steuerzeichen werden durchgelassen (siehe Makro **VTCSET** z.B. ESC, DC4). Andere Zeichen < X'40' werden durch SUB ersetzt. Zeichen ≥ X'40' werden durchgelassen.

YES
Logische Steuerzeichen werden nicht ausgewertet. Alle Zeichen, die im EBCDIC-Code < X'40' sind, werden durch SUB ersetzt.
Nur Zeichen ≥ X'40' werden durchgelassen.

ONOPOSN=
bestimmt, wo die Nachricht beginnen soll.

NO
Die Nachricht beginnt am Anfang der nächsten Zeile.

YES
Die Nachricht beginnt am Anfang der aktuellen Zeile (gilt nur für Schreibstationen).

OTRANS=
gibt an, ob die Ausgabedaten normiert oder transparent übertragen werden sollen.

NO
Die Ausgabedaten sollen normiert übertragen werden, d.h. es findet eine Codeumsetzung statt.

YES
Die Ausgabedaten sollen transparent übertragen werden, d.h. sie bestehen aus beliebigen Binärzeichen (je nach Gerätecode aus 5, 7 oder 8 Bit pro Zeichen), die auf dem Übertragungsweg nicht umgewandelt werden. Ist der Übertragungsweg nicht „potenziell transparent“ generiert, wird die Ausgabe mit dem Returncode X'04' zurückgewiesen.

OTRSUP=
gibt an, ob die Übersetzung von Gerätecode in EBCDIC unterdrückt werden soll.

NO
Die Übersetzung der Nachricht von EBCDIC in Gerätecode wird nicht unterdrückt, d.h. das Programm liefert die Nachricht in EBCDIC-Code. Im System wird die Nachricht in den Gerätecode übersetzt.

YES
Die Übersetzung der Nachricht wird unterdrückt. Das Programm muss in diesem Fall die Nachricht im Gerätecode liefern.

RC=
legt fest, wo der Returncode abgelegt wird.
Dieser Operand ist nur für eine 31-Bit-Schnittstelle zulässig.

OLD
Der Returncode wird im rechtsbündigen Byte des Register R15 abgelegt.

NEW
Der Returncode wird sowohl im Register R15 als auch im Standardheader abgelegt. Alle 4 Byte des Registers R15 sind für die Auswertung belegt. Ein 4 Byte-Returncode wird nur zurückgeliefert, wenn SYSDTA von der Datensichtstation liest. In allen anderen Fällen wird nur ein 1 Byte-Returncode zurückgeliefert, unabhängig vom Wert des Returncodes.

TIMER=
legt eine max. Wartezeit für die Eingabe fest. Falls innerhalb der festgelegten Wartezeit keine Eingabe erfolgt, wird ein Returncode zurückgegeben. Die Angabe dieses Operanden ist nur für die 31-Bit-Schnittstelle zulässig.

wert
Wartezeit zwischen 10 und 3600 Sekunden.
Standardwert ist UNLIMITED, d.h. es wird kein Timer benutzt.

VTSUCBA=
bestimmt die Adresse eines mit MF=L erzeugten VTSUCB.
Bei Benutzung des Operanden VTSUCBA wird der Operand MODE und die folgenden Edit-Optionen ignoriert (im Datenbereich wird ihr Wert auf X'FF' gesetzt), d.h. alle gewünschten Edit-Optionen müssen im VTSUCB angegeben werden.
Die Angabe dieses Operanden ist nur für die 31-Bit-Schnittstelle zulässig.
Standardmäßig erfolgt keine Verwendung des VTSUCB.

adr
symbolische Adresse (Name) des VTSUCB.

Programmierhinweise

für die Verwendung des Operanden MODE=EXTEND bzw. EXTEND=YES
(für 3270-Datenstationen siehe Anhang)

Mit MODE=EXTEND bzw. EXTEND=YES im LINE-Mode kann mit Formaten gearbeitet werden, ohne dass hierbei eine Formatsteuerung benutzt werden muss.

– Soll der Anwender wie mit Formaten arbeiten, so muss die 1. Ausgabe mit NP beginnen, um den Bildschirm zu löschen und mit dem Text in Position (1.1) beginnen zu können.

– Mit NL wird auf den nächsten Zeilenanfang positioniert und der Rest des Bildschirmes gelöscht, mit VPAn auf den Anfang der Zeile n, wobei der Rest des Bildschirmes erhalten bleibt.

– Positionierung innerhalb einer Zeile kann nur durch Text, Zwischenräume oder NIL-Zeichen erfolgen.

– Der Text nach VPAn, NL bzw. CHS wird geschützt und halbhell dargestellt.

– Ungeschützte Felder werden mit „EPA Text SPA“ erzeugt.
Dunkelgesteuerte ungeschützte Felder werden mit „DAR text SPA“ erzeugt.
Numerische Felder werden mit „NUM text SPA“ erzeugt.

– Mit VPAn am Ende der Nachricht kann die Schreibmarke auf den Anfang des ersten ungeschützten Feldes der Zeile n gesetzt werden, wobei der Bildschirminhalt erhalten bleibt. Beginnt in Zeile n kein ungeschütztes Feld, so wird auf den Anfang des ersten ungeschützten Feldes nach Zeile n positioniert.
Wird am Ende der Nachricht kein VPAn angegeben, so wird die Schreibmarke in das erste ungeschützte Feld auf dem Bildschirm gesetzt.

– Folgeausgabe / Bild-Update
Mit NP wird ein neues Bild erzeugt.
Mit VPAn am Anfang der Ausgabe wird Zeile n des Bildes geändert. Mit VPAn können eine oder mehrere Zeilen übersprungen werden. In der aktuellen Zeile wird ab der Schreibmarke ein dunkler Bereich erzeugt, der bis zum Zeilenende bzw. einem davor liegenden Feld reicht. Anschließend wird in die Zeile n positioniert. Am Ende der Änderung sollte die Schreibmarke wieder mit VPAn positioniert werden, da sonst der Bildschirm ab der Schreibmarke gelöscht würde (s.o.). Bei der Verwendung von NL innerhalb einer Änderung wird ebenfalls der Bildschirm ab Schreibmarke gelöscht. Soll dies verhindert werden, muss eine neue Zeile immer mit VPAn angesprungen werden.

Makroaufrufformat 2 und Operandenbeschreibung

WRTRD
(1)

(1)
Register R1 enthält die Adresse des Datenbereichs. Die Liste ist auf Wortgrenze auszurichten.

Aufbau des Datenbereichs

Adressierungs-Modus	Byte	Inhalt
24-Bit-Modus	0	Output Edit Byte 1.
	1 - 3	Adresse des auszugebenden Datensates (Operand satz1).
	4	Input Edit Byte 1.
	5 - 7	Adresse des Feldes, in das der einzulesende Datensatz übertragen wird (Operand satz2).
	8	Output Edit Byte 1.
	9	Input Edit Byte 2.
	10 - 11	max. Länge des einzulesenden Datensatzes (Operand länge).
	12 - 15	Adresse, zu der im Fehlerfall verzweigt wird (Operand fehler).
31-Bit-Modus	0 - 7	Standardheader. Aufbau siehe Seite 43 Die Initialisierungswerte sind einem mit MF=L erzeugten Datenbereich zu entnehmen. Bei RC=OLD wird kein Returncode im Standardheader übergeben.
	8 - 11	Adresse, zu der im Fehlerfall verzweigt wird (Operand fehler).
	12 - 15	Adresse des auszugebenden Datensatzes (Operand satz1).
	16 - 19	Adresse des Feldes, in das der einzulesende Datensatz übertragen wird (Operand satz2).
	20	Output Edit Byte 1.
	21	Output Edit Byte 2.
	22	Input Edit Byte 1.
	23	Input Edit Byte 2.
	24 - 25	max. Länge des einzulesenden Datensatzes (Operand länge).
	26	reserviert (X'00').
	27	Flag mit Anzeigen für die Verwendung des VTSUCB und für Returncode-Verhalten.
	28 - 31	Adresse des VTSUCB
	32 - 33	Werte des Timers
	34 - 35	reserviert (X'00000000')

Bei Verwendung der 24-Bit-Schnittstelle sind die Werte für Input/Output Edit Byte 1/2 der beim Makro **CUPAB** angegebenen Tabelle zu entnehmen.
Bei Verwendung der 31-Bit-Schnittstelle sind die Werte einer mit MF=C/D erzeugten Liste zu entnehmen.
Wird während einer Schreib-/Lese-Operation ein „BREAK“ erkannt, so wird der Befehlszähler auf den Anfang der Makroauflösung zurückgesetzt, sodass nach der Behandlung der Unterbrechung der Makroaufruf wiederholt wird.
Wenn die Länge des zu schreibenden Satzes (minus 4 Byte für das Längenfeld und 1 Byte reserviert) die Größe des Puffers für die Datenstation überschreitet, wird der Satz verkürzt. Es wird mit dem Returncode X'10' in Register R15 zur Fehleradresse des Benutzers verzweigt.

Wenn die Länge des zu lesenden Satzes die angegebene Länge (minus 4 Byte für das Längenfeld) überschreitet, wird der Satz verkürzt und es wird mit dem Returncode X'0C' in Register R15 zur Fehleradresse des Benutzerprogramms verzweigt.

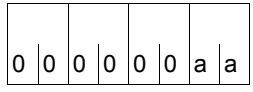
Rückinformation und Fehleranzeigen

Edit Options (Aufbereitungsparameter), die für ein Gerät oder für die gewählte MODE-Art nicht zulässig sind, werden vom System so weit möglich korrigiert (siehe Returncode X'20').

Während der Makrobearbeitung enthält Register R1 die Adresse der Operandenliste.

bei PARMOD=24:

R15:



Über die Ausführung des Makros WRTRD wird im Register R15 ein Returncode übergeben.

X'aa'	Erläuterung
X'00'	Erfolgreiche Bearbeitung der Funktion.
X'04'	Nicht behebbarer Fehler.
X'08'	Operandenfehler.
X'0C'	Satz beim Einlesen abgeschnitten. Die Länge des Satzes überschreitet die angegebene Länge. Die Eingabenachricht wird von einem MSV-Terminal mit einem Header versehen. Hat die Nachricht bereits die Länge des Systempuffers, wird durch Voran Stellen dieses Headers die Nachricht abgeschnitten.
X'10'	Ausgabesatz abgeschnitten. Die Länge des auszugebenden Satzes überschreitet die Größe des Puffers für die Datenstation. Die überschüssigen Zeichen werden nicht ausgegeben.
X'14'	WRTRD wurde in einem Batch-Auftrag aufgerufen.
X'18'	Eingabeende (ETX) erkannt.
X'20'	Ungültiges Edit-Option-Byte, wurde vom System korrigiert.
X'2C'	Die Eingabe beginnt mit Steuerzeichen NL (nur bei Edit-Option EXTEND=YES) Bei 3270-Datenstationen: Eingabelänge verkürzt.
X'38'	Fehler im Zusammenhang mit POSIX.

bei PARMOD=31:

bei **RC=OLD**:

Zusätzlich zu den bei PARMOD=24 beschriebenen Returncodes kann der Returncode

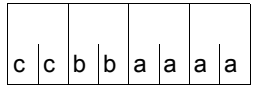
X'aa'	**Erläuterung**
X'24'	Fehler im VTSUCB.

auftreten, sowie die Returncodes, die durch Konvention makroübergreifend festgelegt sind (siehe Tabelle „Standard-Returncodes“ auf Seite 43).

bei RC=NEW:

Die Returncodes werden sowohl im Standardheader als auch im Register R15 eingetragen.

Standard-header:



Über die Ausführung des Makros WRTRD wird im Standardheader folgender Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode):

X'cc'	**X'bb'**	**X'aaaa'**	**Erläuterung**
X'00'	X'00'	X'0000'	Erfolgreiche Bearbeitung der Funktion.
X'00'	X'00'	X'0020'	Erfolgreiche Berabeitung der Funktion, ein aufgetretener Operandenfehler ist durch TIAM/VTSU korrigiert worden.
X'00'	X'01'	X'0008'	Nicht korrigierter Operandenfehler.
X'07'	X'01'	X'0008'	Nicht korrigierter Operandenfehler: die RESERVED-Felder sind nicht 0
X'08'	X'01'	X'0008'	Nicht korrigierter Operandenfehler: Der Wert des Operanden TIMER ist nicht im erlaubten Intervall von 10 bis 3600 Sekunden.
X'00'	X'20'	X'0004'	Interner Fehler.
X'02'	X'20'	X'0004'	Interner Fehler: BCAM-Nachricht verloren.
X'05'	X'20'	X'0004'	Interner Fehler: Eingabenachricht zu lang.
X'06'	X'20'	X'0004'	Interner Fehler: negative Transportquittung.
X'00'	X'40'	X'0004'	Ein-/Ausgabe abgebrochen.
X'00'	X'40'	X'000C'	Eingabesatzlänge > spezifizierte Länge: Eingabesatz wurde abgeschnitten.
X'00'	X'40'	X'0010'	Abschneiden des Ausgabesatzes.

X'cc'	X'bb'	X'aaaa'	Erläuterung
X'00'	X'40'	X'0014'	BREAK im WRTRD. Wird während einer Schreib-/Lese-Operation ein „BREAK“ erkannt und ist RC=NEW, wird dem Anwender der Returncode X'00400014' geliefert. Ist RC=OLD, wird der TU-Befehlszähler auf den Anfang der Makroauflösung zurückgesetzt, sodass nach der Behandlung der Unterbrechung der Makroaufruf wiederholt werden kann.
X'00'	X'40'	X'0018'	Ende der Eingabe.
X'00'	X'40'	X'002C'	NL erkannt. Dieser Returncode kann nur bei Edit-Optionen mit EXTEND=YES auftreten.
X'00'	X'40'	X'0034'	Timer-Ablauf (innerhalb der festgelegten Wartezeit erfolgte keine Eingabe).
X'00'	X'00'	X'0014'	Fehler bei SYSFILE: WRTRD im Batch-Betrieb.
X'01'	X'80'	X'0004'	Interner BCAM-Engpass.
X'09'	X'80'	X'0038'	Fehler im Zusammenhang mit POSIX: Ein-/Ausgabe-Serialisierungsfehler.
X'0A'	X'40'	X'0038'	Fehler im Zusammenhang mit POSIX: Wenn die LOGON-Task im Systemmodus ist, sind keine Ein-/Ausgaben von mit fork{} erzeugten Prozessen möglich.
		X'24'	VTSU-Fehler. Außer Maincode (rechtes Byte) siehe Fehlerinformation im VTSUCB-Header.

Beispiel 1

Das Beispiel realisiert die Ausgabe einer Meldung, die vom Anwender beantwortet werden soll. Die Abfrage wird wiederholt, solange als Antwort „N“ eingegeben wird. Bei einer Antwort ≠ „N“ wird das Programm beendet. Der Aufruf des Makros **WRTRD** erfolgt im Makroaufrufformat 1.

```
  WRTRD1   START
           PRINT NOGEN
  WRTRD1   AMODE 31
  WRTRD1   RMODE 24
           BALR  3,0
           USING *,3
  QUEST    WRTRD QUERY,,INPUT,,5,END,PARMOD=31
2                *,@DCEO      999    921011   53531004
2                *,@DCEI      999    921011   53531002
           CLI   REPLY,'N'
           BE    QUEST
  END      WROUT TEXT,TERM,PARMOD=31
2                *,@DCEO      999    921011   53531004
  TERM     TERM
  **** Definitions ****
  QUERY    DC    Y(ENDQU-QUERY)
           DS    CL2
           DC    X'01'
           DC    'TERMINATE PROGRAM (Y/N) ?'
  ENDQU    EQU   *
  INPUT    DS    0CL5
           DS    CL4
  REPLY    DS    CL1
  TEXT     DC    Y(ENDTEXT-TEXT)
           DS    CL3
           DC    C'*** The WRTRD1 program was terminated. ***'
  ENDTEXT  EQU   *
           END
```

Ablaufprotokoll:

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,wrtrd1), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,wrtrd1))
%  ASS6011 ASSEMBLY TIME: 310 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 82 MSEC
//end
%  ASS6012 END OF ASSEMBH
/start-executable-program library=macexmp.lib,element-or-symbol=wrtrd1, -
/      prog-mode=*any
%  BLS0523 ELEMENT 'WRTRD1', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'WRTRD1', VERSION ' ' OF '<date> <time>' LOADED
TERMINATE PROGRAM (Y/N) ?
n
TERMINATE PROGRAM (Y/N) ?
y
*** The WRTRD1 program was terminated. ***
```

Beispiel 2

Dieses Beispiel realisiert den gleichen Ablauf wie Beispiel 1, der Aufruf des Makros **WRTRD** erfolgt jedoch im Makroaufrufformat 2.

```
  WRTRD2   START
           PRINT NOGEN
           BALR  10,0
           USING *,10
  LOOP     LA    1,PARAM
           WRTRD (1),PARMOD=31
           CLI   REPLY,'N'
           BE    LOOP
  END      WROUT TEXT,TERM,PARMOD=31
2                *,@DCEO      999    921011   53531004
  TERM     TERM
  **** Definitions ****
           DS    OF
  PARAM    WRTRD QUERY,,INPUT,,5,END,MF=L,PARMOD=31
2                *,@DCEO      999    921011   53531004
2                *,@DCEI      999    921011   53531002
  *
  QUERY    DC    Y(ENDQU-QUERY)
           DS    3X
           DC    'TERMINATE PROGRAM (Y/N) ?'
  ENDQU    EQU   *
  INPUT    DS    0CL5
  LENGTH   DS    CL2
  UNUSED   DS    CL2
  REPLY    DS    CL1
  TEXT     DC    Y(ENDTEXT-TEXT)
           DS    CL3
           DC    C'*** The WRTRD2 program was terminated. ***'
  ENDTEXT  EQU   *
           END
```

Beispiel 3

```
  WRTRD3   START
           PRINT NOGEN
           BALR  10,0
           USING *,10
           WRTRD MESSAGE,,INPUT,,40,ERROR,PARMOD=31,MODE=LINE,           C
                 OBELL=Y,ILCASE=Y,ICFD=Y
2                *,@DCEO      999    921011   53531004
2                *,@DCEI      999    921011   53531002
  ERROR    NOP   0
           TERM
  **** Definitions ****
  MESSAGE  DC    Y(ENDMESS-MESSAGE)
           DS    3X
           DC    C'Output WRTRD example 3'
  ENDMESS  EQU   *
  INPUT    DS    0CL14
  LENGTH   DS    CL2
  UNUSED   DS    CL2
  DATA     DS    CL10
           END
```

Beispiel 4

Verwendung des VTSUCB

```
  WRTRD4   START
           PRINT NOGEN
           BALR  10,0
           USING *,10
           WRTRD MESSAGE,,INPUT,,40,ERROR,PARMOD=31,VTSUCBA=VTSUPAR
  ERROR    NOP   0
           TERM
  *
  VTSUPAR  VTSUCB MODE=LINE,BELL=YES,LOW=YES,SPECIN=C
1                *,VTSUCB     350    980309
  MESSAGE  DC    Y(ENDMESS-MESSAGE)
           DS    3X
           DC    C'Output WRTRD example 4'
  ENDMESS  EQU   *
  INPUT    DS    0CL40
  LENGTH   DS    CL2
  UNUSED   DS    CL2
  DATA     DS    CL36
           END
```

Beispiel 5

WRTRD mit Extended-Line-Mode

```
  WRTRD5   START
           PRINT GEN
           BALR  3,0
           USING *,3
  QUEST    WRTRD OUTPUT,,INPUT,,90,END,MODE=LINE,EXTEND=YES,PARMOD=31
1 QUEST    ##SPASS S0001S,S0001D                                    A312
2          CNOP  0,4
2 QUEST    BAS   1,S0001S           ADDRESS AND SKIP PARAMS
1 S0001D     DS 0F                                                  A340
1          FHDR  UNIT=36,FUNCT=19,VERS=2
2          DS    0A
2          DS    0XL8                GENERAL OPERAND LIST HEADER
2          DC    AL2(36)             FUNCTION UNIT NUMBER
2          DC    AL1(19)             FUNCTION NUMBER
2          DC    AL1(2)              FUNCTION INTERFACE VERSION NUMBER
2          DC    X'FFFFFFFF'          Returncode is virgin
1          DC    A(END)                   ERROR RETURN ADDRESS
1          DC    AL4(OUTPUT)              MESSAGE AREA ADDRESS
1          DC    AL4(INPUT)               READ IN AREA DDRESS
1          DS    AL1(0)                   PLACE FOR O.EDIT BYTE 1
1          DS    AL1(0)                   PLACE FOR O.EDIT BYTE 2
1          DS    AL1(0)                   PLACE FOR I.EDIT BYTE 1
1          DS    AL1(0)                   PLACE FOR I.EDIT BYTE 2
1          DC    AL2(90)                  NUMBER OF CHARS. TO BE READ
1          DC    AL1(0)                   RESERVED 2
1          DC    AL1(0)                   FLAG BYTE 1
1          DC    AL4(0)                   VTSUCB ADDRESS
1          DC    AL2(0)                   INPUT TIMER VALUE          009
1          DC    H'0'                     RES_FOR_TIAM               007
1 *
1          @DCEO OTRSUP=,OLINEND=,OMANUAL=,                             C
1                OHCOPY=,OPTAPE=,ONOPOSN=,                              C
1                OHDR=,OETB=,OHOM=,OEXTEND=YES,                         C
1                MODE=LINE,DCEDIT=,OBELL=,OTRANS=,                      C
1                ONOLOGC=,                                              C
1                RDA1=-16,RDA2=-15
2          ORG   *-16
2          DC    AL1(4)
2          ORG   *+16-1
2          ORG   *-15
2          DC    AL1(4)
2          ORG   *+15-1
2                *,@DCEO      999    921011   53531004
1 *
```

```
1          @DCEI DCEDIT=,MODE=LINE,RDA1=-14,RDA2=-13,                    C
1                ITRSUP=,ILINEND=,ICFD=,                                 C
1                IGETBS=,ILCASE=,IHDR=,                                  C
1                IGETFC=,IGETIC=,IEXTEND=YES
2          ORG   *-14
2          DC    AL1(32)
2          ORG   *+14-1
2          ORG   *-13
2          DC    AL1(32)
2          ORG   *+13-1
2                *,@DCEI      999    921011   53531002

1 *
1 S0001S     DS 0Y                                                      A340
1          SVC   39                    SYSFILE SVC
1 *
           CLC   INLNAME(4),='XXXX'
           BNE   QUEST
  END      TERM
1 END      DS    0H                                                     206
1          LA    1,S0006D                                               205
1          B     S0006S                                                 200
1 S0006D     DS 0F                                                      200
1          FHDR  UNIT=6,FUNCT=40,VERS=1                                 207
2          DS    0A
2          DS    0XL8               GENERAL OPERAND LIST HEADER
2          DC    AL2(6)             FUNCTION UNIT NUMBER
2          DC    AL1(40)            FUNCTION NUMBER
2          DC    AL1(1)             FUNCTION INTERFACE VERSION NUMBER
2          DC    X'FFFFFFFF'         Returncode is virgin
1          DC    XL1'01'                                                207
1          DC    XL1'00'
1          DC    XL1'00'
1          DC    XL1'04'
1          DC    CL4'    '
1 S0006S     DS 0Y                                                      200
1          SVC   9
  *
           VTCSET LOG
1 *
1 *              VIRTUAL TERMINAL CONTROL CHARACTER SET
1 *
1 *
1 *              LOGICAL RECORD DELIMITERS
1 *
1 LOGNL    EQU   X'15'           LOGICAL LINE END (CONT NEXT LINE)
1 LOGNP    EQU   X'0C'           LOGICAL PAGE END (CONT NEXT PAGE)
1 LOGCL    EQU   X'0D'           LOGICAL LINE END (CONT SAME LINE)
```

```
1 LOGVPA   EQU   X'29'           LOG VERTICAL POS ABSOLUT (CONT LINE N)
1 LOGHPA   EQU   X'2A'           LOG HORIZONT POS ABSOLUT (CONT COL N)
1 LOGASF   EQU   X'21'           LOG SHEED FEDDING FROM CASETTE N  D1
1 LOGCAP   EQU   X'20'           CONTINUE ACTUAL POSITION AT MSG BEGIN
1 *
1 *              LOGICAL UNIT DELIMITERS
1 *
1 LOGEM1   EQU   X'1D'           EMPHASIZED LAYOUT 1
1 LOGEM2   EQU   X'1F'           EMPHASIZED LAYOUT 2
1 LOGEM3   EQU   X'13'           EMPHASIZED LAYOUT 3
1 LOGEM4   EQU   X'14'           EMPHASIZED LAYOUT 4
1 LOGNOR   EQU   X'1E'           NORMAL LAYOUT
1 LOGDAR   EQU   X'12'           DARK LAYOUT
1 LOGPLD   EQU   X'2B'           PARTIAL LINE DOWN
1 LOGPLU   EQU   X'2C'           PARTIAL LINE UP
1 *
1 LOGSO    EQU   X'0E'           SHIFT OUT TO 2ND CHARACTER SET
1 LOGSI    EQU   X'0F'           SHIFT IN TO NORMAL CHARACTER SET
1 *

1 LOGSPA   EQU   X'36'           START PROTECTED AREA
1 LOGEPA   EQU   X'08'           END PROTECTED AREA
1 LOGNUM   EQU   X'11'           START NUMERIC (UNPROTECTED) AREA
1 *
1 LOGCHS   EQU   X'06'           CHARACTER SET D1D2
1 LOGCOL   EQU   X'17'           COLOUR CHOICE
1 LOGLOC   EQU   X'09'           LOCAL ATTRIBUTE START S1
1 LOGLOX   EQU   X'0A'           LOCAL ATTRIBUTE EXIT S1
1 *
1 LOGVMI   EQU   X'24'           VERTICAL MOVEMENT INDICATOR  D1
1 LOGHMI   EQU   X'23'           HORIZONTAL MOVEMENT INDICATOR  D1
1 LOGLM    EQU   X'38'           LEFT MARGIN  D1D2D3
1 LOGPTS   EQU   X'1A'           PROPORTIONAL TYPING START
1 LOGPTX   EQU   X'1B'           PROPORTIONAL TYPING END
1 LOGMLL   EQU   X'33'           MAXIMAL LINE LENGTH
1 LOGMLN   EQU   X'35'           MAXIMAL LINE NUMBER (ON PAGE)
1 LOGNLQ   EQU   X'39'           NEAR LETTER QUALITY START
1 LOGNLX   EQU   X'3B'           NEAR LETTER QUALITY EXIT
1 *
1 *              SPECIAL FUNCTIONS
1 *
1 LOGDEL   EQU   X'07'           DELETE
1 LOGBS    EQU   X'16'           BACKSPACE
1 LOGSUB   EQU   X'3F'           SUBSTITUTE
1 *
1 *              DELIMITER EXTENSION
1 *
1 LOGEXT   EQU   X'3E'           DELIMITER EXTENSION BYTE
```

```
1 *
1 *              EXTENDED LOGICAL DELIMITERS
1 *
1 LOGTRA   EQU   C'T'            TRANSPARENT OUTPUT X1L1L2
1 LOGDIM   EQU   C'D'            DIMENSION OF SCREEN D1D2D3D4D5
1 LOGRPT   EQU   C'R'            REPEAT NEXT CHARACTER NN TIMES
1 LOGDIS   EQU   C'I'            SET DISPLAY ATTRIBUTES
1 LOGRS    EQU   X'00'            RESET DISPLAY ATTRIBUTES
1 LOGFL    EQU   X'01'            FLASHING
1 LOGUND   EQU   X'02'            UNDERSCORED
1 LOGBLK   EQU   X'04'            BLANKED
1 LOGRIN   EQU   X'08'            REDUCED INTENSITY
1 LOGINV   EQU   X'10'            INVERSE
1 LOGFLD   EQU   C'F'            SET FIELD CHARACTERISTICS
1 LOGINP   EQU   X'00'            INPUT FIELD
1 LOGPNS   EQU   X'01'            PROTECTED NOT SENDABLE
1 LOGPRS   EQU   X'20'            PROTECTED SENDABLE
1 LOGNUF   EQU   X'02'            NUMERIC
1 LOGMOD   EQU   X'04'            PRE-MODIFIED
1 LOGMAR   EQU   X'08'            MARKABLE
1 LOGPRT   EQU   X'10'            PRINTABLE
1 LOGASK   EQU   X'40'            AUTOMATIC SKIP
1 *
1 *
1 *              PHYSICAL UNIT DELIMITERS
1 *
1 LOGESC   EQU   X'27'           ESCAPE  X

1 LOGDC4   EQU   X'3C'           DC4  X
1 LOGHT    EQU   X'05'           HORIZONTAL TABULATION
1 LOGVT    EQU   X'0B'           VERTICAL TABULATION
1 *
1                *,VTCSET     080    941024   53531028
  INPUT    DS    0CL90
           DS    CL4
  INLNAME  DS    CL20
  INFNAME  DS    CL12
  INSTR    DS    CL30
  INZIP    DS    CL4
  INCITY   DS    CL20
  OUTPUT   DS    0H
           DC    Y(ENDOUT-OUTPUT)
           DS    CL2
           DC    X'01'
           DC    AL1(LOGNP)
           DC    AL1(LOGNL)
           DC    AL1(LOGNL)
           DC    AL1(LOGSPA)
```

```
          DC    AL1(LOGEM3)
          DC    C'PLEASE ENTER NAME AND ADDRESS'
          DC    AL1(LOGNL)
          DC    AL1(LOGNL)
          DC    AL1(LOGSPA)
          DC    C' LAST NAME:     '
          DC    AL1(LOGEPA)
NAME      DS    CL20
          DC    AL1(LOGNL)
          DC    AL1(LOGSPA)
          DC    C' FIRST NAME:    '
          DC    AL1(LOGEPA)
FNAME     DS    CL12
          DC    AL1(LOGNL)
          DC    AL1(LOGSPA)
          DC    C' STREET:        '
          DC    AL1(LOGEPA)
STREET    DS    CL30
          DC    AL1(LOGNL)
          DC    AL1(LOGSPA)
          DC    C' ZIP and CITY:  '
          DC    AL1(LOGEPA)
          DC    AL1(LOGNUM)
ZIP       DS    CL4
          DC    AL1(LOGSPA)
          DC    C'  '
          DC    AL1(LOGEPA)
CITY      DS    CL20
          DC    AL1(LOGNL)
          DC    AL1(LOGNL)
          DC    AL1(LOGSPA)
          DC    AL1(LOGEM3)
          DC    C'FOR PROGRAM TERMINATION, ENTER "XXXX" FOR LAST NAME'
ENDOUT    EQU   *
          END
```

Ablaufprotokoll:

```
/start-assembh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '<ver>' OF '<date>' LOADED
%  ASS6010 <ver> OF BS2000 ASSEMBH  READY
//compile source=*library-element(macexmp.lib,wrtrd5), -
//        compiler-action=module-generation(module-format=llm), -
//        module-library=macexmp.lib, -
//        listing=parameters(output=*library-element(macexmp.lib,wrtrd5))
%  ASS6011 ASSEMBLY TIME: 314 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 116 MSEC
//end
%  ASS6012 END OF ASSEMBH
/start-executable-program library=macexmp.lib,element-or-symbol=wrtrd5, -
/      prog-mode=*any
%  BLS0523 ELEMENT 'WRTRD5', VERSION '@' FROM LIBRARY
   ':2OSG:$QM212.MACEXMP.LIB' IN PROCESS
%  BLS0524 LLM 'WRTRD5', VERSION ' ' OF '<date> <time>' LOADED
%PLEASE ACKNOWLEDGE
```

```
PLEASE ENTER NAME AND ADDRESS

 LAST NAME:     .....................
 FIRST NAME:    ...........
 STREET:         ......................
 ZIP and STREET: .... ..............................

FOR PROGRAM TERMINATION, ENTER "XXXX" FOR LAST NAME
```

6 Anhang

Im Anhang sind folgende Abschnitte und Tabellen enthalten:

– nur noch aus Kompatibilitätsgründen unterstützte Makros

  CDUMP
  GETSW
  GETUS
  HSITYPE
  MRSINF
  MRSSTA
  MSG7
  SETSW
  SETUS
  SINF
  TABLE

– eine Tabelle mit allen im Handbuch beschriebenen Makros in alphabetischer Reihenfolge

– eine Tabelle mit allen im Handbuch beschriebenen Makros in der Reihenfolge ihrer SVC-Nummern

– eine Tabelle mit weiteren Makros von BS2000 OSD/BC, die nicht in diesem Handbuch beschrieben werden

– eine Tabelle mit den normierten Funktionstastencodes

6.1 Makros, die nur noch aus Kompatibilität unterstützt werden

CDUMP – User-, System- oder Areadump ausgeben

Allgemeines

Anwendungsgebiet: Testhilfe; siehe Seite 166
Makrotyp: nur für User- und Systemdumps:
S-Typ, MF-Format **1**: Standardform/E-/L-/D-Form;
nur für Areadumps:
S-Typ, MF-Format **2**: Standardform/E-/L-/D-/C-Form; siehe Seite 29

- Seit BS2000/OSD-BC V3.0 steht der neue Makro **CDUMP2** zur Verfügung.

Makrobeschreibung

Der Makro **CDUMP** erstellt (in einer eigenen Dumptask) einen Speicherabzug für die Task, die **CDUMP** aufgerufen hat. Durch Angabe des Operanden SCOPE kann der Anwender bestimmen, ob ein Area-, User- oder Systemdump ausgegeben werden soll.

Makroaufrufformate und Operandenbeschreibung

In der nachfolgenden Formatdarstellung werden zur Erstellung eines System-/User- bzw. Areadumps getrennte Aufrufformate angegeben.

Ausgabe eines Areadumps

CDUMP

SCOPE=AREA,{ NUM=#n | AREAS=((anf1,end1),(anf2,end2),...) }

,PC=STD / adr

[,TITLE=adr]

,MODE=STD / **EXP**ANDED

[,DS@=adr]

,MF=S / E / L / C / D

[,PARAM=adr / (r)]

,ID=CD / pre

Ausgabe eines Systemdumps

CDUMP

SCOPE=SYSTEM

,PC=STD / adr / (r)

,EC=STD / adr / (r)

,IW=STD / PC

[, { CODE={ 'name' | adr | (r) } | INSERT={ adr | (r) } }]

[,TITLE=adr / (r)]

,DIAG=NO / YES

[,ELSN=adr / (r)]

,PARMOD=24 / 31

,MF=S / L / (E,..) / D

,ID=CD / pre

Ausgabe eines Userdumps

CDUMP

[SCOPE=USER]
,PC=STD / adr / (r)
,EC=STD / adr / (r)
,IW=STD / PC
[, { CODE={ 'name' / adr / (r) } / INSERT={ adr / (r) } }]
[,TITLE=adr / (r)]
,DS=STD / NO / YES / listadr
,DIV=STD / NO / YES
,PARMOD=24 / 31
,MF=S / L / (E,..) / D
,ID=CD / pre

In der nachfolgenden Operandenbeschreibung sind die Operanden alphabetisch geordnet.

AREAS=
beschreibt die auszugebenden Bereiche durch Angabe ihrer Anfangs- und Endadressen. Als Adresse kann jeder Ausdruck angegeben werden, der für eine Adresskonstante erlaubt ist (s. Beispiel). Maximal 4 Speicherbereiche können als Liste angegeben werden. AREAS darf nicht in Verbindung mit MF=C/D angegeben werden.

((anf1,end1),....)
anf1 = Adresse des ersten Byte (Anfangsadresse) des auszugebenden Speicherbereichs.
end1 = Adresse des letzten Byte (Endadresse) des auszugebenden Speicherbereichs.

CODE=
bezeichnet eine Zeichenfolge zur Kennzeichnung des Dumps. Die Zeichenfolge wird bei der `IDAON51`-Meldung ausgegeben. Die Angabe ist nur in Verbindung mit SCOPE=USER bzw. SCOPE=SYSTEM möglich.

adr
symbolische Adresse des Feldes mit einer beliebigen Zeichenfolge; Länge = 7 Byte.

(r)
r = Register mit dem Adresswert adr.

'name'
name = beliebige Zeichenfolge. Länge = 7 Zeichen.

DIAG=
steuert, ob eine Meldung an den Operator gesendet wird, die die Adresse des CDUMP-SVCs enthält. Der Operand darf nur in Verbindung mit SCOPE=SYSTEM angegeben werden.

NO
ist Voreinstellung: Eine Meldung wird nicht ausgegeben.

YES
Die Meldung wird ausgegeben.

DIV=
gibt an, ob DIV-Fenster im Userdump enthalten sein sollen. Die Angabe ist nur in Verbindung mit SCOPE=USER möglich und wird nur bei PARMOD=31 ausgewertet. Bei PARMOD≠31 wird unabhängig von der Operandenangabe der Wert DIV=YES verwendet. Siehe auch Abschnitt „Erweiterung durch Datenräume“ auf Seite 61.

STD
ist Voreinstellung: Der im Kommando MODIFY-TEST-OPTIONS eingestellte Wert bestimmt, ob DIV-Fenster im Userdump enthalten sein sollen (YES) oder nicht (NO).

NO
Es sollen keine DIV-Fenster im Userdump enthalten sein.

YES
Es sollen alle DIV-Fenster im Userdump enthalten sein.

DS=
bestimmt, welche Datenräume (DS) in den Userdump aufgenommen werden sollen. Die Angabe ist nur in Verbindung mit SCOPE=USER möglich und wird nur bei PARMOD=31 ausgewertet. Bei PARMOD≠31 wird unabhängig von der Operandenangabe der Wert DS=YES verwendet. Siehe auch Abschnitt „Erweiterung durch Datenräume“ auf Seite 61.

STD
ist Voreinstellung: Der im Kommando MODIFY-TEST-OPTIONS eingestellte Wert bestimmt, ob Datenräume in den Userdump aufgenommen werden sollen (YES) oder nicht (NO).

listadr
Adresse einer Liste von SPIDs (8 Byte pro Eintrag). Die Liste muss mit einem Null-Eintrag (D(0)) abgeschlossen werden.

NO
Es werden keine Datenräume in den Userdump aufgenommen.

YES
Es werden alle, aber max. 100 vom Aufrufer genutzte Datenräume in den Userdump aufgenommen.

DS@=
ist ein Zeiger auf einen Data Space Control Block (DSCB), dessen DSECT bei MF=D generiert wird. Im DSCB können ein Datenraum und die zugehörigen Bereiche definiert werden. Die Angabe ist nur in Verbindung mit SCOPE=AREA möglich.

Datenstruktur eines DSCB:

```
CDDDSCB   DSECT ,      DATA SPACE CTRL BLOCK
CDDDS@    DS    A                Zeiger auf nächsten DSCB
CDDSPID   DS    XL8              SPID des DS
CDDNUM    DS    H                Anzahl der Bereiche im DS
CDDSTRT   DS    A                Beginn des ersten Bereiches
CDDEND    DS    A                Ende des ersten Bereiches
```

Der Anwender kann DSCB-Blöcke verketten, um Bereiche mehrerer Datenräume seiner Task angeben zu können. Standardmäßig sind keine Bereiche aus einem Datenraum angegeben. Siehe auch Abschnitt „Erweiterung durch Datenräume“ auf Seite 61.

adr
symbolische Adresse des DSCB.
adr = NULL entspricht dem Standardwert.

EC=
bestimmt, woher der Ereigniscode (Eventcode, Unterbrechungsgewicht) geholt werden soll. Die Angabe ist nur in Verbindung mit SCOPE=USER/SYSTEM zulässig, bei SCOPE=AREA wird der der Ereigniscode stets aus dem Aufrufer-Stack geholt.

STD
ist Voreinstellung: Der Ereigniscode soll aus dem Aufrufer-Stack geholt werden.

adr
symbolische Adresse des Feldes mit dem Ereigniscode

(r)
r = Register mit dem Ereigniscode (rechtsbündig)

ELSN=
bezeichnet die Adresse eines Feldes mit der Nummer eines Error-Log-Sequence-Blocks, in den der Aufrufer spezielle Daten in die Error-Log-Datei abgelegt hat.
Feldlänge = 4 Byte; Ausrichtung auf Wortgrenze erforderlich. Die Nummer muss als Binärzahl eingegeben werden. Der Operand darf nur in Verbindung mit SCOPE=SYSTEM angegeben werden.

adr
symbolische Adresse (Name) des Feldes

(r)
r = Register mit dem Adresswert adr

INSERT=
legt einen Text fest, der mit der Meldung `IDAON51` ausgegeben wird. In ihm können z.B. nähere Angaben zur Ursache des Speicherabzuges gemacht werden.
Der Anwender muss diesen Text, der bis zu 60 Zeichen lang sein darf, in einem Datenbereich hinterlegen, der folgenden Aufbau hat:

Byte 1: Länge (sedezimal) des auszugebenden Textes (in Byte). Enthält Byte 1 den Wert 0, wird kein INSERT-Text ausgegeben.

Byte 2 bis n (n≤61): Auszugebender Text.
Bei SCOPE=AREA wird der Operand INSERT ignoriert.

adr
symbolische Adresse des Datenbereiches, der Längenangabe und Text enthält

(r)
r= Register mit der Adresse des Datenbereichs, der Längenangabe und Text enthält

IW=
bestimmt, woher das Unterbrechungsgewicht geholt werden soll. Die Angabe ist nur in Verbindung mit SCOPE=USER und nur für 24-Bit-Schnittstelle zulässig. Der Operand wird nur noch aus Kompatibilität unterstützt und sollte in Programmen durch EC=... ersetzt werden.

STD
ist Voreinstellung: Gibt an, dass das Unterbrechungsgewicht aus dem Aufrufer-Stack geholt werden soll.

PC
gibt an, dass das Unterbrechungsgewicht im 1. Byte des bei dem Operanden PC angegebenen Feldes/Registers steht.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei MF=C oder MF=D kann ein Präfix ID (pre = 2 Buchstaben), wie im Aufrufformat dargestellt, angegeben werden.

Für Areadumps sind folgende Angaben Pflicht: Bei MF=C/D/E muss der Operand SCOPE, bei MF=C/D zusätzlich der Operand NUM angegeben werden.
Für Systemdumps ist folgende Angabe Pflicht: Bei MF=C/D/E muss der Operand SCOPE angegeben werden.

MODE=
legt bei einem Areadump den Umfang der auszugebenden Diagnosedaten fest. Der Operand darf nur zusammen mit SCOPE=AREA angegeben werden.

STD
ist Voreinstellung: bewirkt, dass zusätzlich zu den angegebenen Bereichen im Klasse-6- und Klasse-5-Speicher nur der Modul AIDSYSD und die Bereiche mit COMAREA, P1-PCB und TCB ausgegeben werden (siehe Abschnitt Makrobeschreibung, Areadump).

EXP[ANDED]
veranlasst, dass zusätzlich zu den angegebenen Bereichen in Klasse-6- und Klasse-5-Speicher auch der Bereich mit COMAREA sowie alle weiteren Systembereiche wie beim Userdump ausgegeben werden (siehe Abschnitt Makrobeschreibung, Userdump).

NUM=
bezeichnet die Anzahl der auszugebenden Bereiche. NUM darf nur in Verbindung mit MF=L/C/D angegeben werden. Die Anfangs- und Endadressen der auszugebenden Bereiche müssen (dynamisch) im erzeugten Datenbereich eingetragen werden. In Verbindung mit MF=C/D ist NUM Pflichtangabe.

#n
n = Anzahl der auszugebenden Bereiche; $1 \leq n \leq 2048$.
(Das Zeichen # muss der Zahl vorangestellt werden).

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD**.

24
Die Angabe ist nicht in Verbindung mit SCOPE=AREA zulässig. Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen.
(Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

PC=
bezeichnet ein Register oder Feld, das den zu protokollierenden Befehlszähler enthält.

STD
ist Voreinstellung: Der Befehlszähler soll aus dem Aufrufer-Stack geholt werden.

adr
symbolische Adresse (Name) des Feldes (Wortes) mit dem Befehlszähler

(r)
r = Register, das den Befehlszähler enthält. Angabe nur in Verbindung mit SCOPE=USER/SYSTEM möglich.

SCOPE=
gibt an, ob ein User-, System- oder Areadump ausgegeben werden soll.

USER
ist Voreinstellung: Ein Userdump wird ausgegeben.

SYSTEM
Ein Systemdump wird ausgegeben. Angabe nur erlaubt für Anwender mit Leseprivilegierung ≥ 3.

AREA
Ein Areadump wird ausgegeben.

TITLE=
bezeichnet eine zusätzliche Titelzeile für den Speicherabzug.
Länge = 132 Zeichen.

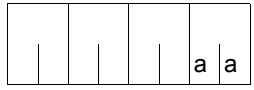
Hinweis
Dieser Operand wird nur noch aus Kompatibilitätsgründen unterstützt, d.h. DAMP bereitet diese Titelzeile nicht auf. Die Angabe des Operanden hat daher keine Wirkung.

adr
symbolische Adresse des Feldes mit der Titelzeile

(r)
r = Register mit dem Adresswert adr. Angabe nur in Verbindung mit SCOPE=USER/SYSTEM möglich.

Rückinformation und Fehleranzeigen

R15 / Standard-header:



Über die Ausführung des Makros CDUMP wird im rechtsbündigen Byte des Registers R15 ein Returncode übergeben. Bei Verwendung der 31-Bit-Schnittstelle wird dieser Returncode auch im Standardheader (Maincode) übergeben.

X'aa'	**Erläuterung**
X'00'	Dump fehlerfrei abgeschlossen
X'04'	Dump unter Verwendung von Standardwerten abgeschlossen
X'08'	Dump wegen /OPTION DUMP-Operand unterdrückt
X'0C'	Dump wegen Systemkonventionen unterdrückt
X'10'	Dump wegen schwerer CDUMP-Operandenfehler unterdrückt
X'14'	Dump wegen unzureichender Testprivilegierung unterdrückt
X'18'	Dump wegen Fehler in DMS-Routinen unterdrückt
X'1C'	Dump wegen Systemfehler unterdrückt
X'20'	Dump unterdrückt, weil CDUMP zuvor unterbrochen wurde
X'24'	Dump wegen eines Fremdtaskdump-Fehlers
X'28'	Dump wegen SHUTDOWN-Bearbeitung unterdrückt.
X'2C'	Dump vom Aufrufer unterdrückt.
X'30'	Alle Bereichsangaben sind ungültig. Keine Dumpausgabe.
X'34'	Falsche Angabe bei NUM=... . Keine Dumpausgabe.
X'38'	Einige Dumpbereiche liegen nicht im Adressraum des Aufrufers oder die Angaben sind inkonsistent. Mindestens ein Speicherbereich wurde aber ausgegeben.
X'3C'	Dump unterdrückt, da DMS-Ready noch nicht erreicht ist.

GETSW – Auftragsschalter abfragen

Allgemeines

Anwendungsgebiet: Benutzer- und Auftragsschalter; siehe Seite 73
Makrotyp: O-Typ; siehe Seite 28

- Seit BS2000/OSD-BC V1.0 steht der neue Makro **SWITCH** zur Verfügung, der die Funktionalitäten der Makros GETSW, GETUS, SETSW und SETUS vereinigt.

Makrobeschreibung

(zur allgemeinen Beschreibung der Auftragsschalter siehe Makro **SETSW**)

Der Makro **GETSW** überträgt die Stellung der Auftragsschalter des laufenden Auftrags in das Register R0.
Die Schalter sind in aufsteigender Reihenfolge den Bits des Registers R0 (von rechts nach links) zugeordnet:

```
Bit 2^0  →  Schalter 0
Bit 2^1  → Schalter 1
   :              :
   :              :
Bit 2^31 → Schalter 31
```

Es gilt: Bit 2^n = 0: Schalter n ausgeschaltet
Bit 2^n = 1: Schalter n eingeschaltet
$0 \leq n \leq 31$

Ein Returncode über die Makroausführung wird nicht übergeben.

Makroaufrufformat

GETSW

GETUS – Benutzerschalter abfragen

Allgemeines

Anwendungsgebiet: Benutzer- und Auftragsschalter; siehe Seite 73
Makrotyp: O-Typ; siehe Seite 28

- Seit BS2000/OSD-BC V1.0 steht der neue Makro **SWITCH** zur Verfügung, der die Funktionalitäten der Makros GETSW, GETUS, SETSW und SETUS vereinigt.

Makrobeschreibung

(zur allgemeinen Beschreibung der Benutzerschalter siehe Makro **SETUS**).

Der Makro **GETUS** überträgt die Stellung der Benutzerschalter der angegebenen Benutzerkennung in das Register R0. Der Anwender kann seine eigene oder auch eine fremde Benutzerkennung angeben.
Die Schalter sind in aufsteigender Reihenfolge den Bits des Registers R0 (von rechts nach links) zugeordnet:

```
Bit 2^0  →  Schalter 0
Bit 2^1  → Schalter 1
   :              :
   :              :
Bit 2^31 → Schalter 31
```

Es gilt: Bit 2^n = 0: Schalter n ausgeschaltet
Bit 2^n = 1: Schalter n eingeschaltet
$0 \leq n \leq 31$

Makroaufrufformat und Operandenbeschreibung

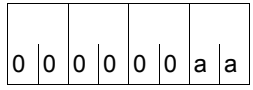
GETUS
[(1)]

(1)
1 = Register R1. Das Register R1 muss den Adresswert des Feldes mit der Benutzerkennung enthalten.
Die Benutzerkennung muss linksbündig in das Feld eingetragen werden (Angabe ohne „$“ und ggf. mit Leerzeichen auf 8 Stellen ergänzen).

Makroaufruf ohne Operand
Die Schalterstellungen für die Benutzerkennung aus dem Kommando SET-LOGON-PARAMETERS werden übergeben.

Rückinformation und Fehleranzeigen

R15:



Über die Ausführung des Makros GETUS wird im rechtsbündigen Byte des Registers R15 ein Returncode übergeben.

X'aa'	**Erläuterung**
X'00'	Normale Ausführung.
X'04'	Operandenfehler.
X'08'	Benutzerkennung existiert nicht.
X'0C'	Benutzerkennung ist nicht mehr gültig.
X'20'	Interner Fehler.

HSITYPE – Informationen über aktuelles HSI ausgeben

Allgemeines

Anwendungsgebiete: Abfragen und Zugriff zu Listen und Tabellen; siehe Seite 158
XS-Programmierung; siehe Seite 168
Makrotyp: R-Typ; siehe Seite 28

- HSITYPE wird durch den Makro **NSIINF** abgelöst.

Makrobeschreibung

Der Makro **HSITYPE** versorgt den Anwender mit Informationen über das aktuelle HSI (Hardware-Software-Interface). Die Information wird in ein Feld ausgegeben, dessen Adresse in einem Register zu übergeben ist. Die ausgegebene Information ist während der Dauer der laufenden BS2000-Session gültig.

Einziger ausgegebener Wert (es wird nur noch XS31-Hardware unterstützt):

XS31: Adressierbarer Speicherbereich > 16MB; alle Adressen werden abhängig vom Adressierungsmodus als 24-Bit- oder 31-Bit-Adressen interpretiert.

Der Makro generiert eine 31-Bit-Schnittstelle.

Makroaufrufformat und Operandenbeschreibung

HSITYPE
(r)

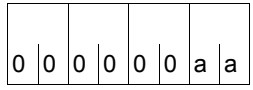
(r)
r = Register mit der Adresse des Feldes, in das die Information übertragen wird.
Feldlänge = 4 Byte; das Feld ist auf Wortgrenze auszurichten.

Registerverwendung

– Register R1 wird bei Makroausführung überschrieben.
– Register R15 enthält den Returncode.

Rückinformation und Fehleranzeigen

R15:



Über die Ausführung des Makros HSITYPE wird im Register R15 ein Returncode übergeben.

X'aa'	**Erläuterung**
X'00'	Funktion ausgeführt.
X'04'	Funktion nicht ausgeführt. Ungültige Adresse angegeben oder Registerfehler.
X'08'	Funktion nicht ausgeführt. Systemfehler (bei internen Aufruf von REQM).
X'18'	Funktion nicht ausgeführt. Ungültiges Makroformat.

MRSINF – MSCF-Informationen abfragen

Allgemeines

Anwendungsgebiet: Mehrrechnersysteme; siehe Seite 168
Makrotyp: S-Typ, MF-Format **2**: Standardform/C-/D-/L-/E-/M-Form; siehe Seite 29

- Seit BS2000/OSD-BC V3.0 steht der neue Makro **MCSINFO** zur Verfügung, der die Makros MRSINF und MRSSTA ersetzt.

Makrobeschreibung

Der Makro **MRSINF** informiert den Anwender über ausgewählte oder alle Rechner in einem MSCF-Kommunikationsnetz. In einem vom Anwender bereitzustellenden Ausgabebereich wird für jeden Rechner ein Satz mit folgenden Informationen hinterlegt:

– BCAM-Name des Rechners
– externe Darstellung der SYSID des Rechners
– BS2000-Version, die an diesem Rechner eingesetzt wird
– Erreichbarkeit des Rechners im MSCF-Netz: Unterschieden wird dabei zwischen einem lokalen Rechner, einem Rechner, zu dem eine Verbindung besteht und einem Rechner, zu dem keine Verbindung besteht.
– Verbundtyp des Rechners: Unterschieden wird dabei zwischen einem für LCS (Loosely Coupled System) und einem für CCS (Closely Coupled System) definierten Rechner.

Makroaufrufformat und Operandenbeschreibung

MRSINF
KEY=HOST / SYSID / ALL [,HOST='bcamname' / adr] [,ESYSID='sysid' / adr] [,AREA=adr / (r)] [,AREAL=länge / adr] ,MF=S / E / L / M / C / D [,PARAM=adr / (r)] ,PREFIX=M / p ,MACID=RSI / macid ,VERSION=2 / 1

KEY=
zeigt an, ob Informationen über das gesamte MSCF-Netz oder einen bestimmten Rechner ausgegeben werden sollen und legt fest, über welchen Operanden der Rechner ausgewählt wird.

HOST
ist Voreinstellung: Es werden Informationen über den Rechner des MSCF-Kommunikationsnetzes angefordert, dessen BCAM-Name im Operanden HOST angegeben ist.

SYSID
Es werden Informationen über den Rechner des MSCF-Kommunikationsnetzes angefordert, dessen SYSID (System-Identifikation) im Operanden ESYSID angegeben ist.

ALL
Es werden Informationen über das gesamte MSCF-Kommunikationsnetz angefordert.

HOST=
legt über einen BCAM-Namen den Rechner fest, über den Informationen ausgegeben werden.

'bcamname'
bcamname = BCAM-Name des Rechners, über den Informationen angefordert werden. Der Name muss in Apostrophe eingeschlossen werden.
Dieser Operandenwert ist bei MF=S oder MF=L anzugeben.

adr
symbolische Adresse eines Feldes, in dem der Anwender den BCAM-Namen des Rechners hinterlegt.
Dieser Operandenwert kann bei MF=M angegeben werden.

ESYSID=
legt über eine SYSID den Rechner fest, über den Informationen ausgegeben werden.

'sysid'
sysid = SYSID des Rechners, über den Informationen angefordert werden. sysid darf ein bis drei Zeichen lang sein und muss in Apostrophe eingeschlossen werden.
Dieser Operandenwert ist bei MF=S oder MF=L anzugeben.

adr
symbolische Adresse eines Feldes, in dem der Anwender die SYSID des Rechners hinterlegt.
Dieser Operandenwert kann bei MF=M angegeben werden.

AREA=
vereinbart die Adresse eines Ausgabebereiches, in dem **MRSINF** die angeforderten Informationen übergibt.

adr
symbolische Adresse eines Feldes, zur Aufnahme der angeforderten MSCF-Informationen. Das Feld ist auf Wortgrenze auszurichten.
Bei KEY=ALL können bis zu 164 Ausgabesätze von jeweils 16 Byte Länge in dieses Feld geschrieben werden. Der erste Ausgabesatz enthält stets die Informationen über den lokalen Rechner.

(r)
r = Register mit dem Adresswert adr.
Ein Register kann nur bei MF=M angegeben werden.

AREAL
legt die Länge des Ausgabebereiches für die angeforderten MSCF-Informationen fest.

Ist der Ausgabebereich zu klein gewählt, so wird die ausgegebene Information abgeschnitten. Der Anwender wird darüber mit einem Returncode informiert.

länge
Länge (in Byte) des Ausgabebereiches, d.h. des Feldes mit der Adresse adr.
Dieser Operandenwert ist bei MF=S oder MF=L anzugeben.

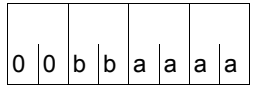
adr
symbolische Adresse eines Feldes (Halbwort), in dem der Anwender (als Binärzahl) den Wert für länge hinterlegt.
Dieser Operandenwert kann bei MF=M angegeben werden.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form, D-Form oder M-Form des Makroaufrufs kann ein Präfix PREFIX und bei der C-Form oder M-Form zusätzlich eine Macid MACID angegeben werden (siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29).

Rückinformation und Fehleranzeigen

Standard-header:



Über die Ausführung des Makros MRSINF wird im Standardheader folgender Returncode übergeben (bb=Subcode1, aaaa=Maincode):

X'bb'	X'aaaa'	Erläuterung
X'00'	X'0000'	Funktion erfolgreich ausgeführt; kein Fehler.
X'01'	X'0001'	Operandenwerte nicht im zulässigen Wertebereich.
X'01'	X'0002'	Der Ausgabebereich ist zu kurz. Informationen wurden abgeschnitten.
X'01'	X'0003'	Der Ausgabebereich ist nicht auf Wortgrenze ausgerichtet.
X'01'	X'0004'	Für den Ausgabebereich wurde eine ungültige Adresse angegeben.
X'20'	X'0020'	Interner Fehler. Nähere Angaben zur Fehlerursache enthält die SERSLOG-Datei.
X'40'	X'0040'	Der angegebene Rechner ist nicht bekannt.
X'40'	X'0041'	Die angegebene SYSID ist nicht bekannt.
X'40'	X'0042'	Die angegebene SYSID ist ungültig.

Weitere Returncodes, deren Bedeutung durch Konvention makroübergreifend festgelegt ist, können der Tabelle „Standard-Returncodes“ auf Seite 43 entnommen werden.

Das aufrufende Programm wird beendet, wenn folgende Fehler auftreten:
– Der Datenbereich ist dem Aufrufer nicht zugewiesen.
– Der Datenbereich ist nicht auf Wortgrenze ausgerichtet.
– Der Datenbereich ist gegen Schreibzugriff geschützt.

MRSSTA – MRS-Zustand ausgeben

Allgemeines

Anwendungsgebiet: Mehrrechnersysteme; siehe Seite 168
Makrotyp: S-Typ, MF-Format **1**: Standardform/L-/E-Form; siehe Seite 29

- Seit BS2000/OSD-BC V3.0 steht der neue Makro **MCSINFO** zur Verfügung, der die Makros MRSINF und MRSSTA ersetzt.

Makrobeschreibung

Dieser Makro steht nur dem Anwender des Mehrrechnersystems zur Verfügung (siehe Handbuch „HIPLEX MSCF“ [26]).

Der **MRSSTA**-Makro gibt aktive und mögliche Verbindungen zwischen dem eigenen (lokalen) Rechner und anderen Rechnern im MSCF-Netzwerk aus.

Makroaufrufformat und Operandenbeschreibung

```
MRSSTA

{ HOST={ 'bcamname' } }
{      { adr1       } }
{      { (r1)       } }
{ AREA={ adr2       } }
{      { (r2)       } }

,MF=S / (E,..) / L
```

HOST=
Angabe eines Rechners, dessen Verbindung zum lokalen Rechner abgefragt werden soll.

Bei Angabe dieses Operanden wird nur in Register R15 ein entsprechender Returncode gesetzt.

bcamname
gibt den BCAM-Namen des Rechners (wie bei der BCAM-Generierung festgelegt) an.

adr1
symbolische Adresse eines BCAM-Namens

(r1)
r1 = Register, das die Adresse des BCAM-Namens enthält

AREA=
Adresse eines Bereichs, in dem alle aktiven und möglichen Verbindungen zum lokalen Rechner abgespeichert werden sollen. Der Bereich muss auf Halbwortgrenze beginnen. Vor jedem Makroaufruf muss in die ersten beiden Byte (Längenfeld) die Länge des Benutzer-Bereichs eingetragen werden. Nach dem Makroaufruf enthält das Längenfeld die Gesamtlänge aller Einträge für die Rechnernamen. Der erste Name ist stets der des lokalen Rechners. Falls auch der Operand HOST angegeben wurde, wird der Operand AREA ignoriert.

Nach der Ausführung des Makros enthält der mit AREA angegebene Bereich folgende Informationen:

Byte 1 - 8:	Rechnername
Byte 9:	Returncode aus Register R15, der bei Angabe des Operanden HOST gesetzt wird (außer X'04A7')
Byte 10:	X'FF' im letzten Eintrag, andernfalls X'00'

adr2
symbolische Adresse des Bereichs, in dem die Verbindungen abgespeichert werden sollen

(r2)
r2 = Register, das die Adresse des Bereichs enthält

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

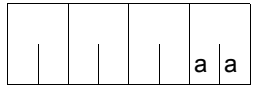
Hinweise

Da die Länge eines Rechnernamen-Eintrages 10 Byte beträgt, ergibt sich die erforderliche Größe des Benutzer-Bereiches zu Anzahl-Rechner * 10 + 2 Byte.

Zu MSCF-Kommunikation siehe auch Handbuch „HIPLEX MSCF“ [26].

Rückinformation und Fehleranzeigen

R15:



Über die Ausführung des Makros MRSSTA wird im rechtsbündigen Byte des Registers R15 ein Returncode übergeben.

Ret.Code	Erläuterung
X'00'	HOST ist lokal.
X'00'	Alle bekannten Rechnernamen sind in den Benutzerbereich gespeichert worden.
X'04'	HOST ist an das lokale MSCF angeschlossen.
X'08'	HOST ist nicht an den lokalen Rechner angeschlossen.
X'0C'	Die MSCF-Kommunikation ist nicht aktiv.
X'10'	Rechner im MSCF-Netz nicht bekannt.
X'14'	Operanden-Fehler.
X'18'	Gegebene Länge zu klein, um alle Namen zu speichern.
X'FF'	MSCF ist nicht in diesem BS2000 enthalten.
X'04A7'	Unzulässige Adresse des MRSSTA-Datenbereichs.

MSG7 – Meldung ausgeben

Allgemeines

Anwendungsgebiet: Meldungswesen; siehe Seite 165
Makrotyp: S-Typ, MF-Format **1**:
(31-Bit-Schnittstelle): Standardform/L-/D-/C-/E-Form;
siehe Seite 29

- Der Makro **MSG7** wird nicht weiterentwickelt; neue Funktionalität wird ausschließlich im Makro **MSG7X** angeboten, der - im Unterschied zum Makro **MSG7** - auch das neue Layout des Datenbereichs verwendet.

Makrobeschreibung

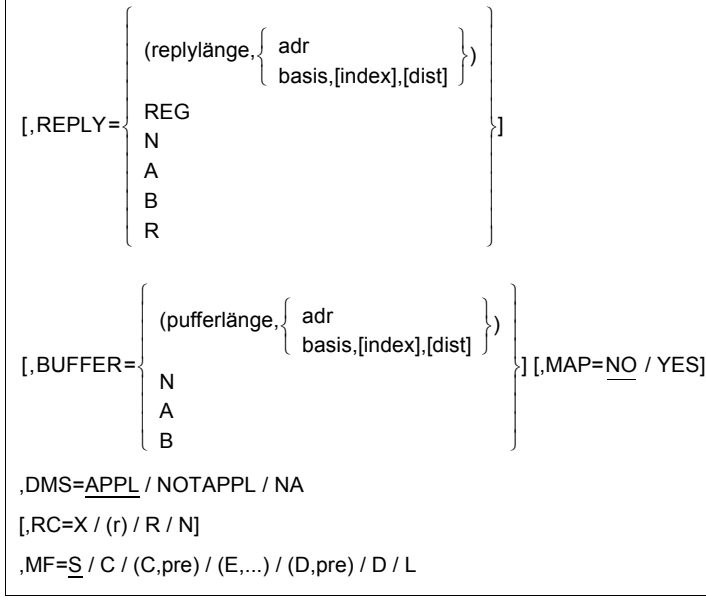
Der **MSG7**-Makro gibt eine Systemmeldung auf SYSOUT, SYSLST, einer Konsole oder in einen Bereich des Benutzerprogramms aus. Der angegebene Meldungsschlüssel muss 7-stellig sein. Die Operandenwerte anzahl2, A, B, C, N, R, T dienen dem Aufbau einer CSECT oder DSECT und können nur in Verbindung mit MF=C/D des Makroaufrufs angegeben werden.

Jede Systemmeldung trägt einen 7-stelligen Meldungsschlüssel. Die ersten 3 Zeichen des Schlüssels bezeichnen die Meldungsklasse; die restlichen 4 Zeichen dienen der laufenden Nummerierung innerhalb einer Klasse. Systemmeldungen können variable Teile „(&nn)“ enthalten, die durch „Inserts“ ersetzt werden können.

Makroaufrufformat und Operandenbeschreibung

```
MSG7

ID=msgid / (r1) / ((r1),(r2)) / (class,(r)) / R / C
[,INSERT={ (insertlänge,{adr | basis,[index],[dist]})
         { ((insertlänge,{adr | basis,[index],[dist]}),...)
         { anzahl1
         { anzahl2
         { (anzahl,B)
         { (anzahl,T)                                      }]
[,LAN='sprache']
,DEST=SYSOUT / SYSLST / CONSOLE / (ausgabeort,...)
[,UCDEST='destcod' / (r) / destadr / N / R / A]
```

MSG7 (Fortsetzung)



ID=
dient der Angabe des Meldungsschlüssels der auszugebenden Systemmeldung.

msgid
7-stelliger Meldungsschlüssel

(r1)
Registerpaar r1 und r1 + 1, das den Meldungsschlüssel enthält:
r1 = Register, das rechtsbündig die Meldungsklasse enthält.
r1 + 1 = Register, das die Meldungsnummer enthält.
r1 muss geradzahlig sein.

((r1),(r2))
Registerpaar, das den Meldungsschlüssel enthält:
r1 = Register, das rechtsbündig die Meldungsklasse enthält.
r2 = Register, das die Meldungsnummer enthält.

(class,(r))
class = 3-stellige Meldungsklasse
r = Register, das die 4-stellige Meldungsnummer enthält.

INSERT=
gibt max. 15 Längen und Adressen von Einfügungen an. Für jede Eintragung wird im Datenbereich der Meldungsverarbeitung ein entsprechender Adressverweis aufgebaut. Werden mehr Einfügungen angegeben, als in der Meldung vorgesehen sind, so werden die weiteren Einfügungen ignoriert.
Eine Einfügung, die nur aus Leerzeichen besteht, wird auf 1 Leerzeichen gekürzt. Leerzeichen am Ende einer Einfügung werden unterdrückt.
Um das Abschneiden der Leerzeichen zu verhindern, muss am Ende der Einfügung das Zeichen X'01' angegeben werden. Findet die Meldungsverarbeitung das Zeichen X'01', werden die bis dahin eingetragenen Leerzeichen nicht unterdrückt.
Enthält ein Meldungstext mehr Einfügungen als im Makroaufruf angegeben, wird für jede übrige Einfügung der Standardwert übernommen. Ist kein Standardwert vorhanden, so wird die Ersatzeinfügung (&nn) eingesetzt.

(insertlänge,...)

`insertlänge` = Länge der Einfügung.
`adr` = symbolische Adresse (Name) des Bereichs
`basis` = Basisregister [1)]
`index` = Indexregister [1)]
`dist` = Distanz [1)]

[1)] Angaben zur Berechnung der Adresse der Einfügung

Einfügungen können übersprungen werden, indem man die ausgelassenen Stellen mit einem Komma belegt, z.B. INSERT=(,(insertlänge2,adr2),,(insertlänge4,adr4)). Eine solche ausgelassene Einfügungen wird durch ihren Standardwert oder durch eine leere Zeichenkette ersetzt.

Bei insertlänge = 0 muss die Einfügung mit einem Satzlängenfeld (4 Byte) beginnen:
Byte 0-1: Länge der Einfügung
Byte 2-3: reserviert.

Hinweise
- Bei indizierter Adressierung Kommas immer mitschreiben.
- Eine Einfügungsliste beginnt mit zwei runden Klammern.
- Summe der Einfügungslängen ≤ 4079 Byte.

anzahl1
Anzahl der INSERT-Verweise, für die Platz im Datenbereich der Meldungsverarbeitung reserviert werden soll.
anzahl1 darf nur zusammen mit MF=L angegeben werden.

anzahl2
Anzahl der Einfügungen, für die Namen in der CSECT/DSECT erzeugt werden sollen.
anzahl2 darf nur in Verbindung mit der C-Form oder D-Form des Makroaufrufs angegeben werden.

(anzahl,B)
Anzahl der Einfügungen, deren Adressen in der Form (basis,index,dist) angegeben werden. Die Angabe wird zum Aufbau der CSECT/DSECT benötigt und kann nur in Verbindung mit der C-Form oder D-Form des Makroaufrufs angegeben werden.

(anzahl,T)
Anzahl der Einfügungen, für die symbolischen Adressen angegeben werden.
Die Angabe wird zum Aufbau der CSECT/DSECT benötigt und kann nur in Verbindung mit der C-Form oder D-Form des Makroaufrufs angegeben werden.

LAN=
bezeichnet die Sprache, in der die Meldungstexte ausgegeben werden sollen.
Dieser Operand wird nicht ausgewertet, wenn DEST=CONSOLE angegeben ist.

'sprache'
sprache = 1 Buchstabe zur Kennzeichnung der Sprache; D = Deutsch, E = Englisch. Weitere Möglichkeiten sind bei der Systemverwaltung zu erfragen. Der Standardwert wird durch den Systemparameter MSGLPRI festgelegt und wird auch dann eingesetzt, wenn eine ungültige Angabe erfolgte.

REPLY=
beschreibt einen Antwortbereich. Der Bereich muss auf Halbwortgrenze ausgerichtet sein und mit einem Satzlängenfeld (4 Byte; Byte 1-2: Länge, Byte 3-4: reserviert) beginnen. Vor der Makroausführung muss in Byte 1-2 die Länge des Antwortbereichs (≤ 4095 Byte) stehen. Bei Ausführung des Makros wird dann die aktuelle Länge der Antwort in Byte 1-2 eingetragen.

Bei Eingabe über den Operanden REPLY werden Kleinbuchstaben in Großbuchstaben umgesetzt. REPLY nur in Verbindung mit DEST=SYSOUT/CONSOLE angeben.

Wird als Antwort ein „?“ eingegeben, das in MIP als Schlüsselwort reserviert ist, zeigt MIP Bedeutung und Maßnahme der betreffenden Meldung an, bevor die Meldung erneut zur Beantwortung ausgegeben wird.

Hinweis
REPLY darf nur in Verbindung mit DEST=SYSOUT/CONSOLE angegeben werden.

(replylänge,...)

`replylänge`	= Länge des Antwortbereichs > 4 Byte
`adr2`	= symbolische Adresse (Name) des Bereichs
`basis`	= Basisregister [1)]
`index`	= Indexregister [1)]
`dist`	= Distanz [1)]

[1)] Angaben zur Berechnung der Adresse des Bereichs

Hinweis
Bei indizierter Adressierung Kommas immer mitschreiben.

REG
Die Antwort steht linksbündig im Register R1. Antwortlänge ≤ 4 Byte.

N
Ein Bereich für die Antwort wird in der CSECT/DSECT nicht generiert.
Die Angabe wird zum Aufbau der CSECT/DSECT benötigt und kann nur in Verbindung mit der C-Form oder D-Form des Makroaufrufs angegeben werden.

A
Die Adresse für den Antwortbereich ist eine symbolische Adresse.
Die Angabe wird zum Aufbau der CSECT/DSECT benötigt und kann nur in Verbindung mit der C-Form oder D-Form des Makroaufrufs angegeben werden.

B
Die Adresse für den Antwortbereich wird in der Form (basis,index,dist) angegeben.
Die Angabe wird zum Aufbau der CSECT/DSECT benötigt und kann nur in Verbindung mit der C-Form oder D-Form des Makroaufrufs angegeben werden.

R
Die Antwort soll in Register R1 übergeben werden.
Die Angabe wird zum Aufbau der CSECT/DSECT benötigt und kann nur in Verbindung mit der C-Form oder D-Form des Makroaufrufs angegeben werden.

DEST=
der Operand beschreibt Ausgabeorte für die konvertierte Systemmeldung. Bei gleichzeitiger Angabe REPLY ist nur DEST=SYSOUT/CONSOLE erlaubt.

SYSOUT
Ausgabe erfolgt nach SYSOUT.

SYSLST
Ausgabe nach SYSLST.

CONSOLE
Ausgabe nach der Konsole.

(ausgabeort,...)
Kombination der genannten Ausgabeorte; Angabe in runden Klammern.

BUFFER=
beschreibt einen Bereich, in den die konvertierte Systemmeldung übertragen werden soll. BUFFER muss auf Halbwortgrenze ausgerichtet sein. In die ersten 2 Byte wird ein Satzlängenfeld eingetragen:

Byte 0-1: Länge des Bereichs
Byte 2-3: reserviert
Byte 4: Output-Steuerzeichen
Byte 5-n: Meldungstext
} WROUT-Format

(pufferlänge,...)

`pufferlänge` = Länge des Bereichs > 16 Byte
`adr` = symbolische Adresse (Name) des Bereichs
`basis` = Basisregister [1)]
`index` = Indexregister [1)]
`dist` = Distanz [1)]

[1)] Angaben zur Berechnung der Adresse des Bereichs

N
Ein Bereich für die konvertierte Systemmeldung wird in der CSECT/DSECT nicht generiert.
Die Angabe wird zum Aufbau der CSECT/DSECT benötigt und kann nur in Verbindung mit der C-Form oder D-Form des Makroaufrufs angegeben werden.

A
Die Adresse für den Bereich ist eine symbolische Adresse.
Die Angabe wird zum Aufbau der CSECT/DSECT benötigt und kann nur in Verbindung mit der C-Form oder D-Form des Makroaufrufs angegeben werden.

B
Die Adresse für den Bereich wird in der Form (basis,index, dist) angegeben.
Die Angabe wird zum Aufbau der CSECT/DSECT benötigt und kann nur in Verbindung mit der C-Form oder D-Form des Makroaufrufs angegeben werden.

MAP=
beschreibt, ob BUFFER eine andere Struktur (Mapping-Format) erhält.

NO
BUFFER wird im **WROUT**-Format eingerichtet.

YES
BUFFER wird im Mapping-Format eingerichtet (Aufbau siehe unten).

UCDEST=
UCON-Ausgabeort (UCON=Universelle CONsole); siehe „Systembetreuung“ [10].
Das Ziel (Ausgabeort) einer Meldung kann wie folgt angegeben werden:
– mnemotechnischer Gerätename für einen bestimmten Bedienungsplatz
– Berechtigungsschlüssel (Routing Code) für Konsole und berechtigte Benutzerprogramme, denen ein bestimmtes Aufgabengebiet zugeordnet ist.
– Berechtigungsname für einen berechtigten Benutzerprogramm.

Hinweise
– UCDEST wird nur ausgeführt, wenn DEST=CONSOLE spezifiziert wurde.
– UCDEST besitzt Vorrang vor dem Operanden RC

'destcod'
folgende Angaben sind für 'destcode' möglich:
– '(mn)'
  wobei mn = 2 Zeichen langer mnemotechnischer Gerätename.
– '< x'
  wobei x = Berechtigungsschlüssel; das Zeichen < muss angegeben werden.
– 'name'
  wobei name = 4 Zeichen langer Name des Benutzerprogramms.

destadr
symbolischer Name eines Bereiches (Wortlänge) mit der Angabe für destcod. Angaben linksbündig; Ausrichtung auf Wortgrenze.

(r)
r = Register mit den Angaben für destcod; linksbündiger Eintrag.

N
Ein Feld für destcod soll in der CSECT/DSECT nicht generiert werden.
Die Angabe wird zum Aufbau der CSECT/DSECT benötigt und kann nur in Verbindung mit der C-Form oder D-Form des Makroaufrufs angegeben werden.

R
Die Angabe für destcod steht in einem Register.
Die Angabe wird zum Aufbau der CSECT/DSECT benötigt und kann nur in Verbindung mit der C-Form oder D-Form des Makroaufrufs angegeben werden.

A
Die Adresse des Feldes mit destcod ist eine symbolische Adresse (destadr).
Die Angabe wird zum Aufbau der CSECT/DSECT benötigt und kann nur in Verbindung mit der C-Form oder D-Form des Makroaufrufs angegeben werden.

DMS=
der Operand spezifiziert den Meldungs-Suchmechanismus.

APPL
Die Meldung wird mit dem DMS gesucht.
(Meldungsdateien + DLAM-Bereich).

NOTAPPL / NA
Die Meldung wird auf Systemebene nur im DLAM-Bereich (ohne DMS) gesucht.
Rückmeldung, wenn die geforderte Meldung nicht gefunden wird:
`msgid,DMS NOT IN MEMORY,Befehlszähler,Modul,inserts.`

RC=
beschreibt den Berechtigungsschlüssel (Routing Code) für das Senden einer Meldung zur Console. Der in der Meldungseinheit stehende Berechtigungsschlüssel wird ignoriert, wenn der Operand RC spezifiziert wird.

x
Berechtigungsschlüssel (1. Zeichen); siehe auch Handbuch „Systembetreuung“ [10].

(r)
r = Register mit der Angabe für x. Eintrag rechtsbündig.

R
Der Berechtigungsschlüssel steht in einem Register.
Die Angabe wird zum Aufbau der CSECT/DSECT benötigt und kann nur in Verbindung mit der C-Form oder D-Form des Makroaufrufs angegeben werden.

N
Ein Feld für den Berechtigungsschlüssel wird in der CSECT/ DSECT nicht generiert.
Die Angabe wird zum Aufbau der CSECT/DSECT benötigt und kann nur in Verbindung mit der C-Form oder D-Form des Makroaufrufs angegeben werden.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Seite 142. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Bei der C-Form und D-Form kann ein Präfix (pre = 1..4 Buchstaben), wie im Aufrufformat dargestellt, angegeben werden.
Voreinstellung: pre = MSGC für die C-Form
pre = MSGD für die D-Form

Die Angabe von weniger als vier Buchstaben für das jeweilige Präfix führt zu der Zeichenfoge: MSGx (x ≙ 1. Buchstabe).

Der Datenbereich enthält zusätzlich eine Beschreibung des Ausgabebereichs, Antwortbereichs und Felder für maximal 15 Einfügungen.

Mapping-Format

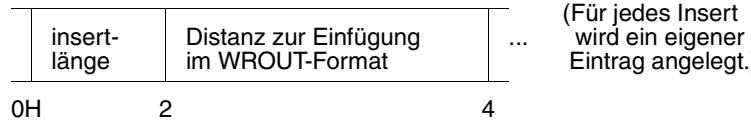
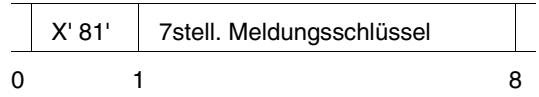
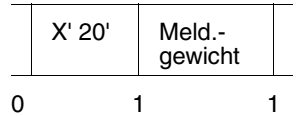
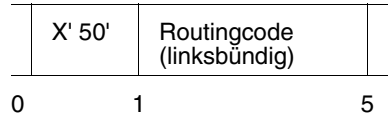
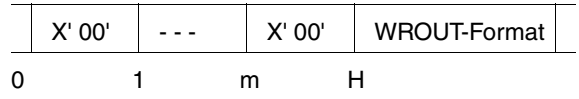
Das Mapping-Format beschreibt die Struktur des Bereichs BUFFER bei MAP=YES. Dabei werden der unter BUFFER beschriebenen Struktur verschiedene Einträge (Mapping-List) vorangestellt.

Struktur des Mapping-Formats:

Byte 0-1:	Länge der Mapping-List
Byte 2-3:	C'MP'
Byte 4-n:	Einträge
Byte n+1 bis p:	weiterer Aufbau wie unter BUFFER beschrieben (WROUT-Format).

Die Einträge beinhalten Informationen über
- die Einfügungen (Inserts)
- den Meldungsschlüssel
- den Berechtigungsschlüssel (Routingcode)
- das Meldungsgewicht
- Füllbytes zum Ausrichten

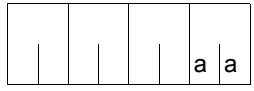
Einträge werden wie folgt abgelegt:

Eintrag über	Aufbau des Eintrags
Insert	 (Für jedes Insert wird ein eigener Eintrag angelegt.)
Meldungs-schlüssel	
Meldungs-gewicht	
Berechtigungs-schlüssel	
Füll-bytes	

Rückinformation und Fehleranzeigen

Der Datenbereich enthält zusätzlich eine Beschreibung des Ausgabebereichs, Antwortbereichs und Felder für maximal 15 Einfügungen.

R15:



Über die Ausführung des Makros MSG7 wird im rechtsbündigen Byte des Registers R15 ein Returncode übergeben.

X' aa'	**Erläuterung**
X'00'	MSG7-Makro ohne Fehler ausgeführt.
X'04'	Abnormale Beendigung der Ein-/Ausgabe.
X'08'	Ungültige Meldungsnummer, Bereichsadresse, mnemotechnischer Gerätename oder Name des berechtigten Benutzerprogramms oder ungültige Bereichslänge.
X'0C'	Benutzer fordert im Batch-Betrieb Antwort an.
X'10'	Es steht kein Speicherplatz zur Verfügung.
X'14'	BREAK während WROUT-Makro.
X'18'	Abgeschnittene Meldung in den Ausgabebereich gebracht.
X'20'	Meldungsausgabe abgebrochen.
X'2C'	DEST=SYSLST wurde zusammen mit REPLY=... spezifiziert.
X'30'	DEST=(...) wurde zusammen mit REPLY=... spezifiziert.
X'41'	Das MIP-Subsystem ist nicht geladen.

SETSW – Auftragsschalter verändern

Allgemeines

Anwendungsgebiet: Benutzer- und Auftragsschalter; siehe Seite 73
Makrotyp: R-Typ; siehe Seite 28

- Seit BS2000/OSD-BC V1.0 steht der neue Makro **SWITCH** zur Verfügung, der die Funktionalitäten der Makros GETSW, GETUS, SETSW und SETUS vereinigt.

Makrobeschreibung

Mit dem Makro **SETSW** kann der Anwender die 32 Auftragsschalter, die seinem Auftrag zugeordnet sind, einschalten, ausschalten oder invertieren.

Makroaufrufformat und Operandenbeschreibung

```
SETSW

[{ ON=(nr,...,nr)     }]
 { OFF=(nr,...,nr)    }
 { INVERT=(nr,...,nr) }
```

ON=(...)
Die angegebenen Schalter werden eingeschaltet.

OFF=(...)
Die angegebenen Schalter werden ausgeschaltet.

INVERT=(...)
Die angegebenen Schalter werden eingeschaltet, wenn sie ausgeschaltet und ausgeschaltet, wenn sie eingeschaltet waren.

nr
Nummer eines Auftragsschalters, der verändert werden soll. Die Schalter sind von 0 bis 31 durchnummeriert.
Es können mehrere Schalter in beliebiger Reihenfolge oder auch als Bereich (z.B. 3-8) angegeben werden.

Makroaufruf ohne Operanden

Statt der Operandenangabe kann die gewünschte Schalterstellung aller Auftragsschalter in Register R0 angegeben werden. Die Schalter sind in aufsteigender Reihenfolge den Bits des Registers R0 (von rechts nach links) zugeordnet:

```
Bit 2^0  →  Schalter 0
Bit 2^1  → Schalter 1
   :              :
   :              :
Bit 2^31 → Schalter 31
```

Es gilt: Bit 2^n = 0: Schalter n ausgeschaltet
Bit 2^n = 1: Schalter n eingeschaltet
$0 \leq n \leq 31$

Rückinformation und Fehleranzeigen

R15:

0	0	0	0	0	0	a	a

Über die Ausführung des Makros SETSW wird im rechtsbündigen Byte des Registers R15 ein Return-code über-geben.

X'aa'	Erläuterung
X'00'	Die gewünschte Schalteränderung wurde ausgeführt.
X'04'	Falsche Operandenangabe. Die Schalteränderung wurde nicht ausgeführt.
X20'	Interner Fehler.

SETUS – Benutzerschalter verändern

Allgemeines

Anwendungsgebiet: Benutzer- und Auftragsschalter; siehe Seite 29
Makrotyp: R-Typ; siehe Seite 28

- Seit BS2000/OSD-BC V1.0 steht der neue Makro **SWITCH** zur Verfügung, der die Funktionalitäten der Makros GETSW, GETUS, SETSW und SETUS vereinigt.

Makrobeschreibung

Mit dem Makro **SETUS** kann der Anwender die Benutzerschalter, die seiner Benutzerkennung zugeordnet sind, ein- und ausschalten oder invertieren.

Makroaufrufformat und Operandenbeschreibung

```
SETUS

[ { ON=(nr,...,nr)       } ]
  { OFF=(nr,...,nr)      }
  { INVERT=(nr,...,nr)   }
```

ON=(...)
Die angegebenen Schalter werden eingeschaltet.

OFF=(...)
Die angegebenen Schalter werden ausgeschaltet.

INVERT=(...)
Die angegebenen Schalter werden eingeschaltet, wenn sie ausgeschaltet und ausgeschaltet, wenn sie eingeschaltet waren.

nr
Nummer eines Benutzerschalters, der verändert werden soll. Die Schalter sind von 0 bis 31 durchnummeriert.
Es können mehrere Schalter in beliebiger Reihenfolge oder auch als Bereich (z.B. 3-8) angegeben werden.

Makroaufruf ohne Operanden

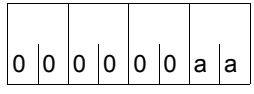
Statt der Operandenangabe kann die gewünschte Schalterstellung aller Benutzerschalter in Register R0 angegeben werden. Die Schalter sind in aufsteigender Reihenfolge den Bits des Registers R0 (von rechts nach links) zugeordnet:

```
Bit 2^0  →  Schalter 0
Bit 2^1  → Schalter 1
   :              :
   :              :
Bit 2^31 → Schalter 31
```

Es gilt: Bit 2^n = 0: Schalter n ausgeschaltet
Bit 2^n = 1: Schalter n eingeschaltet
$0 \leq n \leq 31$

Rückinformation und Fehleranzeigen

R15:



Über die Ausführung des Makros SETUS wird im rechtsbündigen Byte des Registers R15 ein Return-code über-geben.

X'aa'	Erläuterung
X'00'	Die gewünschte Schalteränderung wurde ausgeführt.
X'04'	a) Falsche Operandenangabe. Die Schalteränderung wurde nicht ausgeführt. b) Die Schalterstellungen betreffen einen Auftrag des Anwenders in der WHEN-Warteschlange. Die gewünschte Schalteränderung wurde ausgeführt.
X'20'	Interner Fehler.

SINF – Systeminformationen ausgeben

Allgemeines

Anwendungsgebiet: Abfragen und Zugriff zu Listen und Tabellen; siehe Seite 158
Makrotyp: S-Typ, MF-Format **1**: (Standardform/E-Form/L-Form); siehe Seite 29

- Seit BS2000/OSD-BC V1.0 stehen mit **NSIINF** und **NSIOPT** zwei neue Makros zur Verfügung, die die volle Funktionalität des Makros **SINF** enthalten. Die Funktionalität des **SINF** wurde auf dem Stand BS2000 V10.0 eingefroren. Für Neuentwicklungen sollten die Makros **NSIINF** und **NSIOPT** verwendet werden.

Makrobeschreibung

Der Makro **SINF** informiert über

CPU: Seriennummern und Identifikationen der verfügbaren CPUs
BS2000: Bezeichnung und Version des Betriebssystems, sowie Adressierungsmodus des Betriebssystems
Speicher: Größe des (physikalischen) Hauptspeichers und Anfangsadresse des Betriebssystems im virtuellen Adressraum
HSI: HSI-Basistyp
Server: Server-Typ (Modellreihe)
Systemparameter: Für den nichtprivilegierten Anwender wichtige Systemparameter, wie DEFLUID oder TEMPFILE.

Die ausgewählte Information wird in ein anzugebendes Feld übertragen. Pro Makroaufruf kann nur eine Information abgefragt werden (siehe Liste am Ende der Operandenbeschreibung).

Makroaufrufformat und Operandenbeschreibung

SINF
INFO='info' / adr / (r) ,FIELD=adr / (r) ,LENGTH=länge / (r) [,PARMOD=24 / 31] [,MF=(E,..) / L]

INFO=
bezeichnet die Information, die ausgegeben werden soll.

'info'
info = Kurzbezeichnung der Information. Mögliche Angaben sind der Liste am Ende der Operandenbeschreibung zu entnehmen.

adr
symbolische Adresse eines Feldes, das die Kurzbezeichnung für info enthält. Feldlänge = 8 Byte; Eintrag linksbündig mit nachfolgenden Leerzeichen.

(r)
r = Register mit dem Adresswert adr.

FIELD=
bezeichnet die Adresse des Feldes, in das die Information ausgegeben wird. Feldlänge ≥ Länge der Information (siehe Liste). Die Information wird linksbündig eingetragen.

adr
symbolische Adresse des Feldes.

(r)
r = Register mit dem Adresswert adr.

LENGTH=
beschreibt die Feldlänge des bei FIELD angegebenen Feldes. Die Länge ist der nachfolgenden Liste zu entnehmen. Bei falscher Längenangabe erfolgt kein Eintrag.
Für Systemparameter des Typs C erfolgt die Ausgabe auch in ein zu kleines Ausgabefeld, wenn vom Inhalt des Systemparameters nur Leerzeichen abgeschnitten werden (siehe z.B. Systemparameter DEFLUID).

länge
Länge in Byte, entsprechend der Liste. Angabe als Dezimalzahl.

(r)
r = Register mit der Angabe länge.

MF=
Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB).

Liste der Informationen, die abgefragt werden können

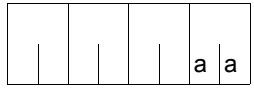
Bezeichnung (info=)	Länge in Byte	Erläuterung der ausgegebenen Werte
CONFNAME	8	Server-Typ (Modellreihe) z.B.: S210-F (im alten Format). Ist der Server-Typ im System nicht eingetragen, wird 7.000␣␣␣ ausgegeben. *Hinweis* Künftig sind die im alten Format ausgegebenen Server-Bezeichnungen möglicherweise nicht mehr eindeutig. Daher wird die Ausgabe im neuen erweiterten Format (INFO=CONFNAMX) empfohlen.
CONFNAMX	21	siehe Makro **NSIINF**, Seite 720.
CPUID	64	Identifikationen der CPUs. Die Ausgabe gliedert sich in 8 Elemente von jeweils 8 Byte Länge: Das i-te Element enthält die Identifikation der i-ten CPU in sedezimaler Darstellung, wie sie von der Hardware übergeben wird. Steht die i-te CPU nicht zur Verfügung, enthält das i-te Element binär Null (X'00...00').
CPUSER	3	Seriennummer der ersten CPU.
	6	Byte 0- 2: Seriennummer der ersten CPU. Byte 3- 5: Seriennummer der zweiten CPU.
	12	Byte 0- 2: Seriennummer der ersten CPU. Byte 3- 5: Seriennummer der zweiten CPU. Byte 6- 8: Seriennummer der dritten CPU. Byte 9-11: Seriennummer der vierten CPU. Ist eine CPU nicht vorhanden, wird in das entsprechende Feld X'000000' eingetragen. Die Angaben haben keine Beziehung zu den CPU-Adressen.
HSIBASE	6	siehe Makro **NSIINF**, Seite 720.
HSILINE	2	siehe Makro **NSIINF**, Seite 720.
HSITYPE	4	Eigenschaften des aktuellen HSI-Typs werden angezeigt. Ausgabe: XS31 Adressierbarer Speicherbereich > 16MB; alle Adressen werden je nach Adressierungsmodus als 24-Bit- oder 31-Bit-Adressen interpretiert.
HSIVM	2	siehe Makro **NSIINF**, Seite 720.
MEMSIZE	4	Größe des für die Software nutzbaren (physikalischen) Hauptspeichers (Angabe in Byte).

Bezeichnung (info=)	Länge in Byte	Erläuterung der ausgegebenen Werte
OSAMODE	2	informiert über den Adressierungsmodus des Betriebssystems. Ausgabe: 31 31-Bit-Adressierungsmodus (Adresslänge = 31Bit).
OSID	12	Byte 0-7: Programmname des Betriebssystems; z.B. 'BS2V190' . Byte 8-11: Version, z.B. 'V190' .
SYSBASE	4	Anfangsadresse des Betriebssystems im virtuellen Adressraum.
<System-parameter>		Wert des Systemparameters

Rückinformation und Fehleranzeigen

Register R1 wird überschrieben.

R15:



Über die Ausführung des Makros SINF wird im rechtsbündigen Byte des Registers R15 ein Returncode übergeben.

X'aa'	Erläuterung
X'00'	Der Makroaufruf war erfolgreich.
X'04'	Ungültige Adresse oder ungültiges Register. Keine Aktion.
X'08'	Interner REQM-Aufruf nicht ausgeführt. Keine Aktion.
X'0C'	Ungültige Angabe im Operand INFO. Keine Aktion.
X'10'	Falsche Längenangabe für das Ausgabefeld. Keine Aktion.
X'14'	Ein unzulässiger Systemparameter wurde bei INFO angegeben. Keine Aktion.
X'18'	Falsches Makroformat angegeben.

TABLE – Ladeinformation übergeben

Allgemeines

Anwendungsgebiet: Binden und Laden; siehe Seite 47
Makrotyp: 24-Bit-Schnittstelle: R-Typ; siehe Seite 28
31-Bit-Schnittstelle: S-Typ;
(Standardform/E-Form/L-Form/C-Form/D-Form);
siehe Seite 29

Für 31-Bit-Schnittstelle zu beachten: Im Standardheader wird kein Returncode übergeben.

- TABLE wird durch die Makros **ETABLE** und **ETABIT** abgelöst.

Makrobeschreibung

Mit dem **TABLE**-Makro übergibt das Benutzerprogramm dem dynamischen Bindelader DBL eine Tabelle mit Einträgen. Diese informieren den DBL über Namen und Adressen von Programmabschnitten (CSECT), Einsprungstellen (ENTRY) und COMMON-Bereichen des Benutzerprogramms, sodass der DBL beim Befriedigen von Externverweisen usw. die ihm übergebenen Werte verwenden kann.

Insbesondere kann ein vom Binder TSOSLNK gebundenes und mit dem Lader ELDE geladenes Benutzerprogramm, das den **LINK**-Makro verwendet, mit dem **TABLE**-Makro den DBL (der durch den **LINK**-Makro aufgerufen wurde) über den Programmaufbau informieren. Dadurch vermeidet man, dass der DBL erneut Moduln lädt, die bereits vom TSOSLNK in das Programm eingebunden wurden.
Während der Makroausführung wird für jeden Tabelleneintrag geprüft, ob der Name eines Ladepunkts schon bekannt ist durch einen vorherigen Ladevorgang oder **TABLE**-Aufruf. Im Ja-Fall wird eine Warnung ausgegeben und die Makroausführung normal fortgesetzt.

Makroaufrufformat 1 und Operandenbeschreibung

```
TABLE
```

$$\left\{\begin{matrix}\text{adr}\\(\text{r})\end{matrix}\right\},\left\{\begin{matrix}\text{länge}\\(\text{r})\end{matrix}\right\}$$

[,PARMOD=24 / 31]

[,MF=L / (E,..) / C / D]

Format 2

PBTABD
[,MF=(D,p) / (C,p)]

adr
symbolische Adresse der Tabelle (Aufbau siehe am Ende der Operandenbeschreibung).

(r)
r = Register mit dem Adresswert adr. Für die 24-Bit-Schnittstelle muss Register R1 verwendet werden.
Bei Verwendung der 31-Bit-Schnittstelle wird Register R1 mit der Adresse des Datenbereichs geladen und kann vom Anwender nicht verwendet werden.

länge
Länge der Tabelle in Byte.

(r)
r = Register mit der Längenangabe. Für die 24-Bit-Schnittstelle muss Register R0 verwendet werden.

PARMOD=
steuert die Makroauflösung. Es wird entweder die 24-Bit- oder die 31-Bit-Schnittstelle generiert.
Wenn PARMOD nicht spezifiziert wird, erfolgt die Makroauflösung entsprechend der Angabe für den Makro **GPARMOD** oder der Voreinstellung für den Assembler (= 24-Bit-Schnittstelle).

24
Die 24-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 24-Bit-Adressen (Adressraum ≤ 16 MB).

31
Die 31-Bit-Schnittstelle wird generiert. Datenlisten und Befehle benutzen 31-Bit-Adressen (Adressraum ≤ 2 GB). Datenlisten beginnen mit dem Standardheader.

MF=
Nur für die 31-Bit-Schnittstelle: Die C-/D-Form wird aufgerufen mit MF=C/D bzw. MF=(C,p)/(D,p).
p = Präfix (max. 3 Zeichen); Voreinstellung: p = I. Das Präfix verändert nur die Feldnamen (nicht die symbolischen Namen bei den Equates). Bei Angabe p = * werden die symbolischen Namen ohne Präfix erzeugt. Bei dynamischer Versorgung des Datenbereichs sollten die Initialisierungswerte für den Standardheader aus einem mit MF=L erzeugten Datenbereich übernommen werden.

Zur allgemeinen Beschreibung des Operanden MF, der dazugehörenden Operandenwerte und der evtl. nachfolgenden Operanden (z.B. für einen Präfix) siehe Abschnitt „S-Typ-Makroaufrufe“ auf Seite 29. Die gültigen MF-Werte sind zu Beginn der Makrobeschreibung bei „Makrotyp“ angegeben und aus dem Aufrufformat ersichtlich.

Der Makro **PBTABD** generiert für die 31-Bit-Schnittstelle eine Beschreibung der Tabelle als DSECT oder Datenabschnitt. Es bedeuten:

MF=
gibt an, ob eine DSECT oder eine Datenliste erzeugt wird.

p = Präfix (max. 3 Zeichen), mit dem alle in der Liste vorkommenden symbolischen Namen beginnen. Bei Angabe p = * werden die Namen ohne Präfix erzeugt.

D
eine DSECT wird erzeugt; Voreinstellung.

C
ein Datenabschnitt wird erzeugt.

Aufbau der Tabelle für den DBL

1. bei Verwendung der 24-Bit-Schnittstelle:

Distanz	Länge	Eintrag/Funktion
0	1	X'02': kennzeichnet CSECT oder ENTRY X'03': kennzeichnet benannten COMMON X'04': kennzeichnet unbenannten COMMON
1	8	Name des ENTRY, COMMON oder CSECT (8 Zeichen)
9	3	errechnete Adresse
12	3	Länge des COMMON (gilt nur für COMMON)

2. bei Verwendung der 31-Bit-Schnittstelle:

Distanz	Länge	Eintrag/Funktion
0	1	X'F0': kennzeichnet CSECT X'F1': kennzeichnet ENTRY X'F3': kennzeichnet COMMON
1	8	Name des ENTRY, CSECT oder COMMON (8 Zeichen)
9	1	Attribute: AMODE = 24 → X'02' AMODE = 31 → X'04' AMODE = ANY → X'06'
10	2	X'00' (wird für Ausrichtung benötigt)
12	4	Ladeadresse (31-Bit-Adresse)
16	4	Länge des COMMON: sonst 4 X'00'

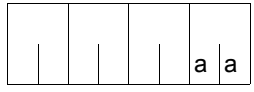
Der Makro **PBTABD** generiert das Layout dieser Tabelle.

Rückinformation und Fehleranzeigen

Über die Ausführung des Makros **TABLE** wird im Register R15 ein Returncode übergeben. Es ist aber zu unterscheiden, ob die 24-Bit- oder die 31-Bit-Schnittstelle aufgerufen wurde (Operand PARMOD oder Makro **GPARMOD**).

bei PARMOD=24:

R15:

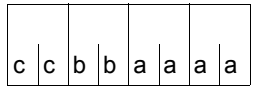


Über die Ausführung des Makros TABLE wird im rechtsbündigen Byte des Registers R15 ein Returncode übergeben.

Ret.Code	Erläuterung
X'00'	Makroaufruf erfolgreich.
X'04'	Makroaufruf nicht erfolgreich; in diesem Fall wird Register R1 gelöscht.

bei PARMOD=31:

R15:



Über die Ausführung des Makros TABLE wird im Register R15 ein gegliederter Returncode übergeben (cc=Subcode2, bb=Subcode1, aaaa=Maincode)
Es bedeuten: cc=00: Funktion normal ausgeführt.
cc=0C: Funktion nicht ausgeführt.

Register R1 wird mit der Adresse des Datenbereichs geladen.

cc	bb	aaaa	Erläuterung
00	00	0000	Normale Ausführung.
0C	00	0004	Datenbereich ist nicht ausgerichtet.
0C	00	0008	Datenbereich nicht zugewiesen oder Adresse in Register R1 liegt außerhalb des Adressraums.
0C	00	0014	Es wurde eine Registernummer > 15 angegeben.
0C	00	0018	Unerlaubte Benutzung eines reservierten Feldes.
0C	00	0100	Fehlerhafte Längenangabe (Operand länge).
0C	00	0104	Falscher Name angegeben (Tabelleneintrag; Distanz 1).
0C	00	0108	Falsche Typangabe (Tabelleneintrag; Distanz 0).
0C	00	010C	Angabe für AMODE nicht korrekt (Tabelleneintrag).

cc	bb	aaaa	Erläuterung
0C	00	0110	ungültige Ladeadresse angegeben (Tabelle Ladeadresse). Die Adresse liegt außerhalb des Klasse-6-Speichers.
0C	00	0200	Systemfehler.
0C	00	0300	Interner Fehler bei Speicheranforderung ($REQM) oder Speicherfreigabe ($RELM) (Systemfehler).
00	01	FFFF	Ungültige Angabe für UNIT/FUNCTION im Standardheader.
00	03	FFFF	Ungültige Angabe für VERSION im Standardheader.

6.2 Makros in alphabetischer Reihenfolge

Makro	Bezeichnung	SVC_{16}	Beschreibung
AINF	Betriebsmittelverbrauch messen	63	Seite 172
ALESRV	Task mit Datenraum verbinden	0D	Seite 198
ALINF	Informationen über Zugriffslisten anfordern	0D	Seite 202
AMODE31	Adressierungsmodus übergeben	-	Seite 205
ARDS	Benutzer-Abrechnungssätze generieren	-	Seite 206
AREC	Benutzer-Abrechnungssatz schreiben	63	Seite 209
ASHARE	Laden von Shared Code in Memory Pools	B7	Seite 214
ASPC	Speicherplatzbelegung erfassen	63	Seite 226
AUDIT	Sprungverfolgungsmodus setzen	5F	Seite 228
BIND	Ladeeinheit binden und laden	B7	Seite 238
BKPT	Programmlauf unterbrechen	5C	Seite 281
CALL	Segmente laden	-	Seite 283
CDUMP	Speicherabzug ohne Programmbeendigung (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 1A 19	Seite 1138
CDUMP2	Speicherabzug ohne Programmbeendigung	1A	Seite 285
CHKEI	Ereigniskennung prüfen	7C	Seite 301
CHKPRV	Privilegien abfragen	31	Seite 304
CHKSI	Serialisierungskennung prüfen	79	Seite 307
CLCOM	Kommunikation beenden	36	Seite 311
CMD [1]	Kommando aufrufen (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 91 58	Seite 313
CONTXT	Zugriff auf Prozessdaten	80	Seite 331
CRYPT	Verschlüsselung von Wörtern	10	Seite 348
CSTAT	Seitenstatus ändern (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 01 4B	Seite 356
CSTMP	Zugriffsart für Memory Pool festlegen (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 01 7A	Seite 360
CTIME	Rechnen mit Zeitstempeln	-	Seite 365
CUPAB	Operandentabelle adressieren	-	Seite 387
DCSTA	Operandentabelle für Datenstationseigenschaften	-	Seite 391

Makro	Bezeichnung	SVC_{16}	Beschreibung
DELFEI	SOLSIG-/POSSIG-Eintrag löschen	BB	Seite 408
DEQAR	Belegung einer Serialisierungskennung aufheben	79	Seite 409
DISCO	Contingency-Definition schließen	7B	Seite 414
DISEI	ereignisgesteuerte Verarbeitung beenden	7C	Seite 417
DISMP	Memory Pool schließen (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 01 7A	Seite 421
DISSI	Zuordnung zu einer Serialisierungskennung lösen	79	Seite 425
DJINF	DSECT/Datenbereiche für Makro JINF erstellen	-	Seite 429
DJSI	DSECT/Datenbereiche für Jobscheduler-Makros erstellen (24-Bit-Schnittstelle)	-	Seite 432
DJSIPL	DSECT/Datenbereiche für Jobscheduler-Makros erstellen (31-Bit-Schnittstelle)	-	Seite 434
DPOFEI	POSSIG-Eintrag erzeugen	BB	Seite 436
DSHARE	Entladen von Shared Code aus Memory Pool	B7	Seite 442
DSOFEI	SOLSIG-Eintrag erzeugen	BB	Seite 445
DSPSRV	Datenraum erzeugen oder zerstören	0D	Seite 450
DTMODE	DSECT/Datenliste für TMODE erstellen	-	Seite 460
ENACO	Contingency-Definition eröffnen	7B	Seite 463
ENAEI	ereignisgesteuerte Verarbeitung eröffnen	7C	Seite 467
ENAMP	Memory Pool eröffnen (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 01 7A	Seite 471
ENASI	Serialisierungskennung zuordnen	79	Seite 481
ENQAR	Belegung einer Serialisierungskennung anfordern	79	Seite 486
ENTER [1]	Batch-Auftrag einleiten (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 91 58	Seite 491
ETABIT	Eintrag für Symboltabelle erzeugen oder ändern	B7	Seite 510
ETABLE	Ladeinformation übergeben	B7	Seite 514
EXIT	STXIT-Contingency-Prozess beenden	80	Seite 522
GCCSN	Anzeige des CCS für Kommando- und Dateneingabe	27	Seite 525
GEPRT	Programmzeit lesen (31-Bit-Schnittstelle) (24-Bit-Schnittstelle, O-Typ) (R-Typ)	 92 18 23	Seite 531

Makro	Bezeichnung	SVC_{16}	Beschreibung
GETPRGV	Programmversion abfragen	B7	Seite 535
GETSW	Auftragsschalter abfragen	48	Seite 1147
GETUS	Benutzerschalter abfragen	41	Seite 1148
GPARMOD	Makroauflösung steuern	-	Seite 538
GTIME	Datum, Uhrzeit und Informationen über Zeitzone anfordern	-	Seite 540
HSITYPE	HSI-Attribute ausgeben	87	Seite 1150
ILEMGT	Verwaltung von ILE	B7	Seite 553
ILEMIT	Listeneintrag für ILE-Liste erzeugen oder ändern	B7	Seite 559
IOSID	Betriebssystemkennung und Version identifizieren	AC	Seite 562
JINF	Jobdaten anfordern (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 BF 51	Seite 565
JMGDJP	DSECT/Datenliste für JMGJPAR erstellen	-	Seite 570
JMGJPAR	Jobparameter ausgeben	-	Seite 571
JOBINFO	Jobdaten anfordern	8C	Seite 573
JSATTCH	Job Scheduler an JMS anschließen (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 BF 51	Seite 577
JSDETCH	Job Scheduler von JMS lösen (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 BF 51	Seite 580
JSEXPCT	JSS-Ereignisse anfordern (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 BF 51	Seite 582
JSINFO	STREAM-PARAMETER abfragen (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 BF 51	Seite 586
JSRUNJB	Job zum Start übergeben (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 BF 51	Seite 588
JSWAKE	Timer Event für Jobscheduler initiieren	BF	Seite 591
LDSLICE	Slice laden	B7	Seite 593
LEVCO	Priorität des Contingency-Prozesses ändern	7B	Seite 598
LGOFF [1]	Auftrag beenden (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 91 58	Seite 601
LKCAN	Lock-Anforderungen löschen	C4	Seite 604

Makro	Bezeichnung	SVC_{16}	Beschreibung
LKCVT	Existierenden Lock konvertieren	C4	Seite 607
LKDEQ	Lock aus der Warteschlange freigeben	C4	Seite 616
LKENQ	Lock generieren und in die Warteschlange einreihen	C4	Seite 621
LKEQU	DLM-eigene Layouts generieren	C4	Seite 632
LKINF	Informationen über Lock-Anforderungen ausgeben	C4	Seite 635
LKLSB	Layout des Lock-Status-Blocks generieren	C4	Seite 640
LPOV	Segment laden	02	Seite 642
MINF	Speicherbelegung ausgeben	01	Seite 645
MRSINF [2]	MRS-Informationen ausgeben	7F	Seite 1152
MRSSTA [2]	MRS-Zustand ausgeben	85	Seite 1156
MSG7	Meldung ausgeben	60	Seite 1159
MSG7X	Meldung ausgeben	26	Seite 652
MSGRC	Returncodes für Meldungs-Makros	-	Seite 670
MSGSHOW	Informationen über Meldungsdateien ausgeben	60	Seite 673
MSGSINIT	globale Bereichszuordnungsliste modifizieren	60	Seite 677
MSGSMOD	globale Bereichszuordnungsliste modifizieren	60	Seite 680
NKDINF	Informationen über (periphere) Konfiguration ausgeben	0E	Seite 685
NKGTYPE	Geräteinformation ausgeben	66	Seite 708
NSIINF	Systeminformationen ausgeben	87	Seite 720
NSIOPT	Systemparameter ausgeben	87	Seite 728
OPCOM	ITC-Teilnahme eröffnen	32	Seite 735
OPSGEN	Steuerung der S-Variablen-Generierung durch MIP	C6	Seite 737
PASS	eine Sekunde warten	4C	Seite 740
PINF	Globale Programminformationen ausgeben	B7	Seite 742
POSSIG	Ereignis signalisieren	7C	Seite 754
RDATA	Satz von SYSDATA lesen (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 27 42	Seite 763
RDUID	Benutzerkennung der aktuellen Task ausgeben	31	Seite 776
RELBF	Empfangswarteschlange löschen	35	Seite 778
RELM	Speicherbereich freigeben (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 01 4A	Seite 779

Makro	Bezeichnung	SVC_{16}	Beschreibung
RELMP	Seiten im Memory Pool freigeben (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 01 7A	Seite 782
REQM	Speicherbereich anfordern (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 01 49	Seite 788
REQMP	Seiten für einen Memory Pool anfordern (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 01 7A	Seite 792
RETCO	Contingency-Prozess beenden	7B	Seite 798
RETRN	Rücksprung mit Register laden	-	Seite 799
REVNT	Ereignis empfangen	34	Seite 801
RPOFEI	POSSIG-Signal senden	BB	Seite 807
RSOFEI	POSSIG-Signal anfordern	BB	Seite 809
SAVE	Register sicherstellen	-	Seite 811
SEGLD	Segmente laden	-	Seite 815
SELPRGV	Programmversion auswählen	B7	Seite 818
SETBF	Pufferlänge für Dialogkommunikation setzen	56	Seite 821
SETIC	Intervallzeitgeber setzen	80	Seite 823
SETSW	Auftragsschalter verändern	47	Seite 1169
SETUS	Benutzerschalter verändern	40	Seite 1171
SEVNT	Ereignis senden	33	Seite 827
SINF	Systeminformation ausgeben	87	Seite 1173
SOLSIG	Signal anfordern	7C	Seite 843
SRMUINF	Eintrag aus Benutzerkatalog ausgeben	B9	Seite 853
STAMCE	MRSCAT-Einträge lesen	21	Seite 867
STXIT	Contingency-Unterbrechungsausgang setzen	80	Seite 904
SUSPEND	Task in Wartezustand versetzen	7B	Seite 916
SWITCH	Benutzer- und Auftragsschalter abfragen und verändern	2A	Seite 918
SYSFL [1]	Systemdateien und TASKLIB zuordnen (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 91 58	Seite 929
SYSTA [1]	Systemdatei- und TASKLIB-Zuweisung ausgeben (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 91 58	Seite 942

Makro	Bezeichnung	SVC_{16}	Beschreibung
TABLE	Ladeinformation übergeben (mit PBTABD) (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 B7 6F	Seite 1177
TCHNG	Datenstationseigenschaften ändern	46	Seite 945
TERM	Programm und Prozedurabschnitt beenden	09	Seite 949
TINF	Taskattribute ausgeben und modifizieren	87	Seite 953
TMODE	Taskeigenschaften abfragen (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 BF 46	Seite 961
TSPRIO	Runprioritäten ausgeben	-	Seite 967
TSTAT	Datenstationseigenschaften abfragen	46	Seite 968
TYPIO	Mitteilung an Konsole	96	Seite 978
UNBIND	Entladen und Entbinden	B7	Seite 982
VMGINF	Informationen über den VM2000-Betrieb ausgeben	67	Seite 993
VPASS	warten	59	Seite 998
VSVI1	Binde- und Ladeinformation ausgeben	B7	Seite 1000
VTCSET	logische Steuerzeichen definieren	-	Seite 1026
VTSUCB	VTSU-Parameter für Ein- und Ausgabe erstellen	-	Seite 1057
WRCPT	Fixpunkt schreiben	05	Seite 1078
WRLST	Satz nach SYSLST übertragen (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 27 45	Seite 1085
WROUT	Satz nach SYSOUT übertragen (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 27 43	Seite 1090
WRTRD	kombinierte Ein-/Ausgabe an der Datenstation (31-Bit-Schnittstelle) (24-Bit-Schnittstelle)	 27 44	Seite 1108

[1] MCLP-Makroaufruf

[2] Makro nur für den Anwender des Software-Produkts HIPLEX MSCF

6.3 Makros in der Reihenfolge ihrer SVC-Nummern

SVC_{16}	SVC_{10}	Makroaufrufe
01	01	CSTMP, DISMP, ENAMP, MINF, RELM, RELMP, REQM, REQMP, CSTAT (alle 31-Bit-Schnittstelle)
02	02	LPOV
05	05	WRCPT
09	09	TERM
0D	13	ALESRV, ALINF, DSPSRV
0E	14	NKDINF
10	16	CRYPT
18	24	GEPRT (O-Typ, 24-Bit-Schnittstelle)
1A	26	CDUMP2 (31-Bit-Schnittstelle)
21	33	STAMCE
23	35	GEPRT (R-Typ)
26	38	MSG7X
27	39	RDATA, WRLST, WROUT, WRTRD (alle 31-Bit-Schnittstelle), GCCSN
2A	42	SWITCH
31	49	CHKPRV, RDUID
32	50	OPCOM
33	51	SEVNT
34	52	REVNT
35	53	RELBF
36	54	CLCOM
40	64	SETUS
41	65	GETUS
42	66	RDATA (24-Bit-Schnittstelle)
43	67	WROUT (24-Bit-Schnittstelle)
44	68	WRTRD (24-Bit-Schnittstelle)
45	69	WRLST (24-Bit-Schnittstelle)
46	70	TMODE (24-Bit-Schnittstelle), TCHNG, TSTAT
47	71	SETSW
48	72	GETSW
49	73	REQM (24-Bit-Schnittstelle)
4A	74	RELM (24-Bit-Schnittstelle)

SVC_{16}	SVC_{10}	**Makroaufrufe**
4C	75	CSTAT (24-Bit-Schnittstelle)
4C	76	PASS
51	81	JINF, JSATTCH, JSDETCH, JSEXPCT, JSINFO, JSRUNJB (alle 24-Bit-Schnittstelle)
56	86	SETBF
58	88	CMD, ENTER, LGOFF, SYSFL, SYSTA (alle 24-Bit-Schnittstelle)
59	89	VPASS
5C	92	BKPT
5F	95	AUDIT
60	96	MSG7, MSGSINIT, MSGSMOD, MSGSHOW
63	99	AINF, AREC, ASPC
66	102	NKGTYPE
67	103	VMGINF
6F	111	TABLE (24-Bit-Schnittstelle)
79	121	ENASI, DISSI, ENQAR, DEQAR, CHKSI
7A	122	ENAMP, DISMP, REQMP, RELMP, CSTMP (alle 24-Bit-Schnittstelle)
7B	123	DISCO, ENACO, LEVCO, RETCO, SUSPEND
7C	124	CHKEI, DISEI, ENAEI, POSSIG, SOLSIG
80	128	CONTXT, EXIT, STXIT, SETIC
87	135	HSITYPE, NSIINF, NSIOPT, SINF, TINF
8C	140	JOBINFO
91	145	CMD, ENTER, LGOFF, SYSFL, SYSTA (alle 31-Bit-Schnittstelle)
92	146	GEPRT (31-Bit-Schnittstelle)
96	150	TYPIO
AC	172	IOSID
B7	183	TABLE (31-Bit-Schnittstelle), ASHARE, BIND, DSHARE, ETABIT, ETABLE, GETPRGV, ILEMGT, ILEMIT, LDSLICE, PINF, SELPRGV, UNBIND, VSVI1
B9	185	SRMUINF
BB	187	DELFEI, DPOFEI, DSOFEI, RPOFEI, RSOFEI
BF	191	JINF, JSATTCH, JSDETCH, JSEXPCT, JSINFO, JSRUNJB, JSWAKE, TMODE (alle 31-Bit-Schnittstelle)
C4	196	LKCAN, LKCVT, LKDEQ, LKENQ, LKEQU, LKINF, LKLSB
C6	198	OPSGEN

6.4 Weitere Makros in BS2000 OSD/BC

In dieser Tabelle sind alle die Makros von BS2000 OSD/BC aufgeführt, die nicht im vorliegenden Handbuch beschrieben sind. Neben einer Kurzbeschreibung der Makrofunktion wird in der Tabelle auf das Handbuch verwiesen, in dem der betreffende Makro ausführlich behandelt wird.

Makro	Kurzbeschreibung	Handbuch
ADDPLNK	Pool-Kettungsnamen definieren	[7]
BINDER	Den BINDER als Unterprogramm aufrufen	[5]
BTAM	Benutzeranforderungen für BTAM abwickeln	[7]
CALENDR	Kalenderdaten erzeugen, ändern und ausgeben	[6]
CATAL	Katalogeintrag bearbeiten	[7]
CHKFAR	Zugriffsrechte auf eine Datei prüfen	[7]
CHNGE	TFT-Eintrag ändern	[7]
CLOSE	Datei schließen	[7]
COMPFIL	Zwei Dateien vergleichen	[7]
COPFILE	Datei kopieren	[7]
CREAIX	Sekundärschlüssel für ISAM-Datei erzeugen	[7]
CREPOOL	ISAM-Pool erzeugen	[7]
DECFILE	Verschlüsselte Datei in unverschlüsselte Datei umwandeln	[7]
DELAIX	Sekundärschlüssel einer ISAM-Datei löschen	[7]
DELPOOL	ISAM-Pool löschen oder freigeben	[7]
DIV	Datei über „Fenster“ im virtuellen Adressraum bearbeiten	[7]
DROPTFT	TFT-Eintrag freigeben	[7]
EAM	Benutzeranforderungen für EAM abwickeln	[7]
ELIM	Satz aus einer ISAM-Datei löschen	[7]
ENCFILE	Unverschlüsselte Datei in verschlüsselte Datei umwandeln	[7]
ERASE	Datei(en) löschen	[7]
EXCALL	Exit-Routinen aus TU-Programmen aus aufrufen	[31]
EXLST	Exit-Adressenliste für Fehlerroutinen anlegen	[7]
EXRTN	Rücksprung aus Fehlerroutinen	[7]
FCB	Dateisteuerblock anlegen	[7]
FCBAD	FCB-Adressen anlegen	[7]
FEOV	Band abschließen und Bandwechsel einleiten	[7]
FILE	Dateimerkmale definieren und Dateiverarbeitung steuern	[7]

Makro	Kurzbeschreibung	Handbuch
FILELST	Variable Operandenbereiche für FILE-Makro erzeugen	[7]
FPAMACC	FASTPAM-Dateizugriffe formuliueren	[7]
FPAMSRV	FASTPAM-Verwaltungsaufrufe formulieren	[7]
FSTAT	Kataloginformationen anfordern	[7]
GET	Satz aus einer Datei lesen	[7]
GETFL	Markierung in einer Datei suchen und Satz lesen	[7]
GETINSP	Installationspfad ausgeben	[32]
GETINSV	Version der Installations-Unit ausgeben	[32]
GETKY	Satz mit dem angegebenen Schlüssel lesen	[7]
GETPROV	Ausgewählte Produktversion ausgeben	[32]
GETR	Nächsten Satz in Richtung Dateianfang lesen	[7]
IDBPL	DSECT für BTAM-Aktionsmakro generieren	[7]
IDFCB	DSECT für FCB generieren	[7]
IDFCBE	DSECT für FCB-Erweiterung generieren	[7]
IDMCB	Versorgung des MFCB (EAM-Steuerblock mit symbolischen Namen)	[7]
IDPPL	UPAM: PAM-Operandenliste	[7]
IMOS...	Makros von IMON-BAS	[32]
IMOK...	Makros von IMON-SIC	[32]
IMPNFIL	Katalogeintrag für Node-Files erstellen (importieren)	[7]
IMPORT	Private Dateien oder Datenträger importieren	[7]
INSRT	Satz in eine Datei einfügen	[7]
ISREQ	Satz-, Bereichs- oder Datenblocksperre aufheben	[7]
LBRET	Aus Benutzer-Kennsatz-Routinen zurückkehren	[7]
LFFSNAP	Dateien von einem Snapset auflisten	[7]
LJFSNAP	Jobvariablen von einem Snapset auflisten	[7]
MAILFIL	Datei per E-Mail an eine Benutzerkennung versenden	[7]
NBMAP	Nachspann einer Nachricht beschreiben	[10]
NBMHE	Format des Nachrichtenheaders beschreiben	[10]
NDWERINF	Status-Byte abfragen	[7]
OPEN	Datei öffnen	[7]
OSTAT	Informationen über geöffnete Dateien ausgeben	[7]
PAM	UPAM-Aktionen ausführen	[7]
PRNT...	Makros zur Ausgabe von Dateien	[23]

Makro	Kurzbeschreibung	Handbuch
PUT	Satz in eine Datei schreiben	[7]
PUTX	Satz in einer Datei ersetzen	[7]
RDTFT	Informationen aus TFT und TST anfordern	[7]
RELTFT	TFT-Eintrag löschen	[7]
RELSE	Datenblock abschließen und Puffer freigeben	[7]
REMPLNK	Pool-Kettungsnamen löschen	[7]
RETRY	Makroaufruf wiederholen	[7]
RFFSNAP	Dateien von einem Snapset restaurieren	[7]
RJFSNAP	Jobvariablen von einem Snapset restaurieren	[7]
SELPROV	Produktversion auswählen	[32]
SETINSP	Installationspfad eintragen oder modifizieren	[32]
SETL	Internen Satzzeiger einer Datei positionieren	[7]
SHOPLNK	Informationen über ISAM-Pool-Kettungsnamen ausgeben	[7]
SHOPOOL	Informationen über ISAM-Pool ausgeben	[7]
SHOWAIX	Informationen über Sekundärschlüssel ausgeben	[7]
SPSINF	Informationen über SPOOL-Parameterdatei ausgeben	[23]
STORE	Einen Satz in die durch den Schlüssel definierte Stelle der Datei übertragen	[7]
VERIF	Datei wiederherstellen	[7]

6.5 Tabelle der normierten Funktionstastencodes

Die erste Spalte (CODE) ist unterteilt in
a) den normalen Funktionstastencode und
b) den Funktionstastencode, wenn der Bildschirm vor der Eingabe gelöscht wurde.

Weiter bedeuten:
1) Bei Verwendung der Zugriffsmethode TIAM reserviert für ESCAPE-/BREAK-Funktion.
2) Standardwerte siehe VTSU-Betriebsparameter COMPKEYS in „VTSU“ [30].

Code a)	b)	**Bedeutung**		**8110**	**8151**	**8152**	**816x**	**974x 9750 9752 9755**	**9756 9758 9759 9762 9763**	**3270** 2)
00	10	Datenübertragung		DÜ	DÜZ, DÜM	DÜZ,DÜM,	DÜ, DÜ1,	DÜ, DÜ1,	SEND	ENTER
01		(normal)	1		FT1	DÜB	DÜ2	DÜ2	K1	PA1
02			2		FT2 1)	F1	K1	K1	K2 1)	PA2 1)
03			3		DVA	F2 1)	K2 1)	K2 1)	K3	PA3,PF6
04			4			F3	K3	K3	K4	PF7
05			5			F1+FZ	K4	K4	K5	PF8
06			6			F2+FZ 1)	K5	K5	K6	PF9
07			7			F3+FZ	K6	K6	K7	PF10
08		Kurz-	8				K7	K7	K8	PF11
09		telegramm	9				K8	K8	K9	PF12
0A			10				K9	K9	K10	PF13
0B			11				K10	K10	K11	PF14
0C			12				K11	K11	K12	PF15
0D			13				K12	K12	K13	PF16
0E			14				K13 K14	K13 K14	K14	PF17
10		Speicher löschen								CLEAR

Code a)	Code b)	Bedeutung		8110	8151	8152	816x	974x 9750 9752 9755	9756 9758 9759 9762 9763	3270 [2]
20	30	Daten-übertragung (markiert)	0			DÜZ, DÜM, DÜB+FZ		DÜZ, DÜM, DÜB+FZ		
21	31		1				F1	F1	F1	PF1
22	32		2				F2	F2	F2	PF2
23	33		3				F3	F3	F3	PF3
24	34		4				F4	F4	F4	PF4
25	35		5				F5	F5	F5	PF5
26	36		6					F6	F6	PF18
27	37		7					F7	F7	PF19
28	38		8					F8	F8	PF20
29	39		9					F9	F9	PF21
2A	3A		10					F10	F10	PF22
2B	3B		11					F11	F11	PF23
2C	3C		12					F12	F12	PF24
2D	3D		13					F13	F13	
2E	3E		14					F14	F14	
2F	3F		15					F15	F15	
30		Datenübertragung (Positionsdaten)				PU				
40	50	Daten-übertragung (speziell)	0				Bypass-Eingabe			
41	51		1				pos. Rückmeldung			
42	52		2				neg. Rückmeldung			
43	53		3							
45	55	magnetischer Ausweisleser						X	X	X
46	56	Daten-übertragung (markiert)	16						F16	
47	57		17						F17	
48	58		18						F18	
49	59		19						F19	
4A	5A		20						F20	
4B	5B		21						F21	
4C	5C		22						F22	
4D	5D		23						F23	
4E	5E		24						F24	
60		Chipkartentermin							X	
80		Ausweis steckt						X	X	X

Abkürzungen

AFZ	Ausfügen Zeile (Tastatur Datensichtstation)
AID	Advanced Interactive Debugger (Dialogtesthilfe)
AL	Access List (Zugriffsliste für Datenräume)
ALE	Access List Entry (Eintrag in der AL)
ALET	Access List Entry Token (Zeiger auf Datenräume)
AR	Access Register (Zugriffsregister für Datenräume)
ASCII	American Standard Code of Information Interchange
BBS	Bandbetriebssystem (obsolet)
BTAM	Basic Tape Access Method
CCS	Coded Character Set (Codiertabelle)
CJC	Conditional Job Control
CLT	Communication Link Table
CLTF	Common Log Task Facility (Fehlerquelle bei Makroaufrufen)
CPU	Central Processing Unit
CSECT	Control Section
DBL	Dynamic Binder Loader (dynamischer Bindelader)
DCAM	Data Communication Access Method
DCM	Data Communication Methods (Kommunikationszugriffssystem)
DGG	Dateigenerationsgruppe (engl: FGG)
DIV	Data In Virtual (Zugriffsmethode für Datenräume)
DLAM	Dynamic Loadable Access Method
DLM	Distributed Lock Manager
DMS	Data Management System (Datenverwaltungssystem; deutsch: DVS)
DRV	Dual Recording by Volume
DSECT	Dummy Section
DSSM	Dynamic SubSystem Management (Subsystemverwaltung)
DVS	Datenverwaltungssystem (englisch: DMS)

EAM	Evanescent Access Method
EBCDIC	Extended Binary Coded Decimal Interchange Code
EFZ	Einfügen Zeile (Tastatur Datensichtstation)
ELDE	Statischer Lader in BS2000
EOF	End of File
EOT	End of Tape/Text
ES	Ereignisschalter
ESA	Enterprise System Architecture (Erweiterung des Adressraums durch Datenräume)
ESD	External Symbol Dictionary
ETX	End of Text
FCB	File Control Block (Dateisteuerblock)
FGG	File Generation Group (Dateigenerationsgruppe; deutsch: DGG)
FIFO	First In First Out (Reihenfolgeprinzip)
FT	File Transfer (Dateiübertragung)
GByte	Gigabyte; 1 GByte = 1024 MByte = 1.048.576 KByte = 1.073.741.824 Byte
GR	General Register (Mehrzweckregister)
HDR1	Header(-label) 1 (Vorsatzetikett)
HIPLEX MSCF	HIPLEX Multi System Control Facility (Software-Produkt)
II	Informationsindikator
IOP	Input / Output Processor (Ein-/Ausgabeprozessor)
ISAM	Indexed Sequential Access Method (Dateistruktur/Zugriffsmethode)
ISD	Internal Symbol Dictionary (Internadressbuch)
ITC	Inter Task Communication
ITN	Internal Task Number (interne Tasknummer)
JCB	Job Control Block
JSS	Job Scheduling Supports
JTBP	Job To Be Processed Block
JTBX	Job To Be Processed Extension
JV	Jobvariable
KByte	Kilobyte; 1 KByte = 1024 Byte
LIFO	Last In First Out (Reihenfolgeprinzip)
LMS	Library Maintenance System (Bibliotheksverwaltungssystem)

LSP	Löschen (Bildschirm-)Speicher (Tastatur Datensichtstation)
LZE	Logisches Zeilenende (Tastatur Datensichtstation)
MByte	Megabyte; 1 MByte = 1024 KByte = 1.048.576 Byte
MCLP	Macro Command Language Processor (Kommandosprachübersetzer)
MOPS	Million Operations per Second
MP	Memory Pool
MPVS	Multiple Public Volume Set
MRS	Mehrrechnersystem
MRSCAT	MRS-Katalog
MZR	Mehrzweckregister
NDM	Nucleus Device Management
NTL	No Time Limit
PAM	Primary Access Method (Dateistruktur/Zugriffsmethode)
PC	Program Counter (Befehlszähler)
PCB	Process Control Block (Systemstack)
PCR	Program Control Register
PT	Programmtabelle
PVS	Public Volume Set
RC	Return Code (Rückinformation bei Makrobearbeitung)
RFA	Remote File Access (Fern-Dateizugriff)
RS	Rücksprungschalter
RU	Roll-up (Tastatur Datensichtstation)
SAM	Sequential Access Method (Dateistruktur/Zugriffsmethode)
SI	Sekundärer Indikator
SPID	SPace IDentification (Kennzeichnung für Datenräume)
SPL	Software Programming Language (Programmiersprache)
SPOOL	Simultaneous Peripheral Operation OnLine
SVC	Supervisor Call (Assembler-Befehl)
TCB	Task Control Block (Task-Steuerblock)
TCS	Tele Communication System (obsolet)
TFT	Task File Table (Task-Dateitabelle)
TIAM	Terminal Interactive Access Method (Zugriffsmethode im Teilnehmerbetrieb)

TOD	Time Of Day (Uhrzeit)
TODR	Time Of Day clock Register
TODX	Extended Time Of Day clock Register
TOS	Tape Operating System (obsolet)
TPR	Task Privileged
TSN	Task Sequence Number (Auftragsnummer)
TSOS	Time Sharing Operating System
TU	Task Unprivileged
UCON	Universal Console
UPAM	User-PAM
UTM	Universal Transaction Monitor
VFB	Vertical Format Buffer (Vorschubinformationspuffer)
VM	Virtual Memory
VSN	Volume Serial Number (Archivnummer)
VTOC	Volume Table of Contents
VTSU-B	Virtual Terminal Support-Basic
XCS	Cross Coupled System

Literatur

Die Handbücher finden Sie im Internet unter *http://manuals.ts.fujitsu.com*. Handbücher, die mit einer Bestellnummer angezeigt werden, können Sie auch in gedruckter Form bestellen.

[1] **Assemblerbefehle** (BS2000)
Sprachbeschreibung

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

[3] **AID** (BS2000)
Advanced Interactive Debugger
Testen von ASSEMBH-Programmen
Benutzerhandbuch

[4] **BLSSERV** (BS2000)
Bindelader-Starter
Benutzerhandbuch

[5] **BINDER** (BS2000)
Benutzerhandbuch

[6] **BS2000 OSD/BC**
CALENDAR
Benutzerhandbuch

[7] **BS2000 OSD/BC**
DVS-Makros
Benutzerhandbuch

[8] **BS2000 OSD/BC**
Einführung in das DVS
Benutzerhandbuch

[9] **BS2000 OSD/BC**
Diagnosehandbuch
Benutzerhandbuch

[10] **BS2000 OSD/BC**
Einführung in die Systembetreuung
Benutzerhandbuch

[11] **BS2000 OSD/BC**
Systeminstallation
Benutzerhandbuch

[12] **DSSM/SSCM**
Verwaltung von Subsystemen in BS2000
Benutzerhandbuch

[13] **BS2000 OSD/BC**
Abrechnungssätze
Benutzerhandbuch

[14] **SECOS** (BS2000)
Security Control System
Benutzerhandbuch

[15] **DCAM** (BS2000)
Makroaufrufe
Benutzerhandbuch

[16] **TIAM** (BS2000)
Benutzerhandbuch

[17] **VM2000** (BS2000)
Virtuelles Maschinensystem
Benutzerhandbuch

[18] **BS2000 OSD/BC**
System Managed Storage
Benutzerhandbuch

[19] **BS2000 OSD/BC**
Kommandos
Benutzerhandbuch

[20] **SDF-A** (BS2000)
System Dialog Facility - Administration
Benutzerhandbuch

[21] **SDF-P** (BS2000)
Programmieren in der Kommandosprache
Benutzerhandbuch

[22] **JV** (BS2000)
Jobvariablen
Benutzerhandbuch

[23] **BS2000 OSD/BC**
Spool & Print - Makros und Exits
Benutzerhandbuch

[24] **RSO** (BS2000)
Remote SPOOL Output
Benutzerhandbuch

[25] **DRV** (BS2000)
Dual Recording by Volume
Benutzerhandbuch

[26] **HIPLEX MSCF** (BS2000)
BS2000-Rechner im Verbund
Benutzerhandbuch

[27] **BS2000 OSD/BC**
Dienstprogramme
Benutzerhandbuch

[28] **MAREN** (BS2000)
Bandverwaltung in BS2000
Benutzerhandbuch

[29] **LMS** (BS2000)
SDF-Format
Benutzerhandbuch

[30] **VTSU**
Virtual Terminal Support
Benutzerhandbuch

[31] **BS2000 OSD/BC**
System Exits
Benutzerhandbuch

[32] **IMON** (BS2000)
Installationsmonitor
Benutzerhandbuch

[33] **BS2000**
Benutzerkommandos (ISP-Format)
Benutzerhandbuch

Nur aus Kompatibilitätsgründen.

Stichwörter

B

E