1 Einleitung

Dieses Kapitel beschreibt kurz das Produkt DRIVE/WINDOWS, die Zielgruppe des Handbuchs sowie das Konzept der Handbuchreihe. Außerdem enthält es eine Liste mit Änderungen gegenüber der Vorgängerversion sowie eine Aufstellung der in den DRIVE/WINDOWS-Handbüchern verwendeten Darstellungsmittel.

1.1 Kurzbeschreibung des Produkts

DRIVE/WINDOWS ist eine Programmiersprache der 4. Generation (4GL) zur Entwicklung kommerzieller Client-Server-Anwendungen. Sie ist die 4GL für Zugriff auf Dateien oder die BS2000 Datenbanksysteme SESAM/SQL V1, SESAM/SQL V2 und UDS . DRIVE-Anwendungen können nach verschiedenen Client-Server-Architekturen auf heterogene Rechner verteilt werden, denn DRIVE/WINDOWS ist auf den Plattformen BS2000, SINIX und MS-Windows verfügbar und unterstützt die Verbindung von Client und Server optimal.

Die einheitliche Sprache mit mächtigen und leicht erlernbaren Anweisungen ermöglicht die Erstellung komplexer Anwendungen für Datenbankzugriff, Report, Oberfläche, Kommunikation und Verarbeitung. DRIVE/WINDOWS versorgt automatisch systemnahe Schnittstellen zu den jeweiligen Komponenten und nimmt dem Programmierer diese Arbeit ab.

Zum Test seiner DRIVE-Anwendung stellt DRIVE/WINDOWS dem Programmierer einen integrierten Debugger zur Verfügung.

DRIVE-Anwendungen können unabhängig vom Einsatz eines Transaktionsmonitors erstellt und getestet werden. Sie sind ohne Änderungen mit oder ohne Transaktionsmonitor ablauffähig.

Zur Steigerung der Performance können die erstellten DRIVE-Anwendungen mit dem Compiler DRIVE/WINDOWS-COMP übersetzt werden.

DRIVE-Server-Anwendungen unter SINIX ermöglichen Ihnen den Dateizugriff und Datenbankzugriff auf INFORMIX und können problemlos als Komponenten einer verteilten Anwendung mit DRIVE-Anwendungen unter BS2000 oder MS-Windows eingebunden werden. Auch hier gibt es, wie im BS2000, eine integrierte Reportfunktion und für performanten Ablauf der erstellten Server-Applikation einen Compiler.

Über die Erstellung von Anwendungen hinaus stellt DRIVE/WINDOWS unter MS-Windows komfortable Werkzeuge zur Verfügung, die voll in den Entwicklungsprozeß integriert sind. case/4/0 unterstützt den Entwurf einer Anwendung. Darauf aufsetzend ermöglicht DRIVE/DESIGNER den nahtlosen Übergang in die Codierphase und die konsistente Generierung und Montage von DRIVE-Quellprogrammen aus den Entwurfsergebnissen. Die Software-Produktionsumgebung von DRIVE/WINDOWS bietet dann für die Programmerstellung, den Test und und die Anwendungssteuerung eine komfortable grafische und menügestützte Oberfläche.

1.2 Zielgruppe

Das Handbuch wendet sich an Programmierer, die DRIVE-Anwendungen oder Teile von Client-Server-Anwendungen mit DRIVE/WINDOWS auf BS2000-Rechnern entwickeln. Dafür benötigt der Programmierer allgemeine Kenntnisse über das Betriebssystem BS2000.

Abhängig vom konkreten Einsatzfall sind weitere Kenntnisse erforderlich über:

– das Datenbanksystem UDS

– das Datenbanksystem SESAM

– den Transaktionsmonitor UTM

– das Format Handling System FHS zur Erstellung von Bildschirmen

– das Betriebssystem des Clients (MS-Windows oder SINIX)

1.3 Konzept der Handbuchreihe

Die Beschreibung von DRIVE/WINDOWS umfaßt hauptsächlich drei Handbücher:

– Das Handbuch "Programmiersystem" [1] gibt einen allgemeinen Überblick über das System DRIVE/WINDOWS und erläutert die Funktionen, die der Vorbereitung und Sicherung, dem Test und Ausführen von DRIVE-Programmen dienen. Es enthält auch die für den Systemverwalter notwendigen Informationen zur Einsatzvorbereitung von DRIVE/WINDOWS und über die Konfiguration von Client-Server-Anwendungen.

– Das Handbuch "Programmiersprache" [2] beschreibt die Regeln für ein DRIVE-Programm. Es wird auf die Programmierlogik, die Erstellung von Bildschirm- und Listenformaten, die Gestaltung von Berichten mit dem Report-Generator und Client-Server-Anwendungen eingegangen.

– Das Handbuch "Lexikon der DRIVE-Anweisungen" [3] enthält, alphabetisch sortiert, alle DRIVE-Anweisungen mit Syntax und der Beschreibung des gesamten Funktionsumfangs. Die SQL-Anweisungen sind in separaten Handüchern beschrieben (s.u.).

  Die Anweisungen sind aufgeteilt in drei Teile: in die DRIVE-Anweisungen, in die Report-Anweisungen für die Erstellung von Listen, Formularen und Berichten sowie in komplexe Unterstrukturen von Anweisungen, die als "Metavariablen" beschrieben sind. Außerdem enthält das Lexikon eine Einführung in die Syntax der DRIVE-Anweisungen sowie eine Aufstellung aller Meldungen und Schlüsselwörter von DRIVE/WINDOWS.

Darüberhinaus gibt es noch Lexika mit DRIVE-SQL-Anweisungen für die verschiedenen Datenbanksysteme:

– "Lexikon der DRIVE-SQL-Anweisungen für SESAM V1" [4]

– "Lexikon der DRIVE-SQL-Anweisungen für SESAM V2" [5]

– "Lexikon der DRIVE-SQL-Anweisungen für UDS" [6]

DRIVE/WINDOWS bietet zusätzlich die volle Funktionalität von DRIVE V5.1 im sogenannten Old-Style- und im Mischbetrieb.
Die Old-Style-Funktionen finden Sie in den Handbüchern DRIVE V5.1 Benutzerhandbuch [14] und Lexikon [15]. Wie Sie Old-Style-Programme in neue DRIVE-Anwendungen integrieren, ist im Handbuch "Programmiersprache" [2] beschrieben, wie Sie DRIVE/WINDOWS für den Old-Style- und Mischbetrieb generieren im Handbuch "Programmiersystem" [1].

1.4 Readme-Datei

Funktionelle Änderungen und Nachträge der aktuellen Produktversion zu diesem Handbuch entnehmen Sie bitte ggf. der produktspezifischen Readme-Datei. Sie finden die Readme-Datei auf Ihrem BS2000-Rechner unter dem Dateinamen SYSRME.*produkt.version.sprache*. Die Benutzerkennung, unter der sich die Readme-Datei befindet, erfragen Sie bitte bei Ihrer zuständigen Systembetreuung. Die Readme-Datei können Sie mit dem Kommando /SHOW-FILE oder mit einem Editor ansehen oder auf einem Standarddrucker mit folgendem Kommando ausdrucken:

```
/PRINT-DOCUMENT dateiname, LINE-SPACING=*BY-EBCDIC-CONTROL
```

bei SPOOL -Versionen kleiner 3.0A:

```
/PRINT-FILE FILE-NAME=dateiname,LAYOUT-CONTROL=
    PARAMETERS(CONTROL-CHARACTERS=EBCDIC)
```

1.5 Änderungen gegenüber der Ausgabe vom Dezember 1993 (DRIVE/WINDOWS V1.1)

Komponenten

– DRIVE/WINDOWS unterstützt SESAM V2. Damit wird der Sprachstandard SQL2 erfüllt.

– SAM- und ISAM-Dateien lassen sich mit DRIVE-Programmen bearbeiten.

– Verteilte DRIVE-Anwendungen ermöglichen den Anschluß zu grafischen Bedienoberflächen auf den Clients (MS-Windows und SINIX).

– Old-Style-Programme lassen sich in neue DRIVE-Anwendungen integrieren durch den Programmaufruf mit CALL. Parameter können an das rufende Programm zurückgegeben werden.

– Im Old-Style ist neben dem Zugriff auf SESAM V1 nun auch der Zugriff auf SESAM V2, LEASY und DMS möglich.

– DRIVE-Programme mit SQL-Anweisungen für SESAM V2 können nur übersetzt werden, wenn keine Transaktion offen ist.

– Die Anschlüsse zu den Produkten TOM-REF und QUERY werden nicht mehr unterstützt.

Datentypen

– DRIVE/WINDOWS unterstützt die Datentypen TIME(3) und TIMESTAMP(3).

– Der Datentyp VARCHAR belegt ein Byte mehr als bisher. Redefinitionen auf VARCHAR-Variablen und Manipulationen des Längenfelds sind nicht mehr möglich.

Funktionen

– Die Anweisung HELP entfällt. Dafür wird das DRIVE-Lexikon als Datei (Softbook) ausgeliefert.

– Für Reports können Hintergrundmuster sowohl für Seiten als auch für einzelne Zeilen festgelegt werden.

– Die Aufrufreihenfolge bei CALL und DO hat sich geändert: Während DRIVE/WINDOWS bisher zunächst nach einem Zwischencode und dann nach einer Source suchte, wird nun immer das Element mit dem neuesten Datum aufgerufen.

– Mit der Anweisung COMPILE werden keine Fehlerlisten erzeugt. Fehlermeldungen werden in den Source-Teil der Übersetzungsliste eingestreut.

- Mit der Anweisung PARAMETER DYNAMIC LIBRARY wird nicht mehr automatisch eine PLAM-Bibliothek angelegt, sondern kann nur eine bereits vorhandene PLAM-Bibliothek zugewiesen werden.
- Benutzereigene Datentypen können als Übergabeparameter an gerufene Programme übergeben werden. Die Anweisung DECLARE TYPE darf vor der PROCEDURE-Anweisung stehen.
- Die Anweisungen COMMIT WORK und ROLLBACK WORK sind ohne die Anweisung OPTION DBSYSTEM= und mit der Anweisung OPTION DBSYSTEM=OFF erlaubt. Ist zum Zeitpunkt ihrer Ausführung keine Datenbanktransaktion offen, so beziehen sie sich nur auf UTM-Transaktionen.
- Mit der Übersetzungsoption OPTION SCREENCHECK legen Sie fest, ob DRIVE/WINDOWS CHECK-Klauseln in den Adressierungshilfen auswerten soll.
- Die DRIVE-Systemvariable &SQL_STATE enthält den SQLSTATE von SESAM V2.
- Die Anweisung PERMIT hat bei SESAM V2 keine Wirkung. Sie setzt nur den SQLSTATE.
- Mit CURRENT TIMESTAMP wird der aktuelle Zeitstempel ausgegeben.
- Sekundenbruchteile (FRACTION) können sowohl als Einheit für Zeitspannen als auch in Zeitangaben angegeben werden.
- Datumzeitausdrücke lassen sich zu einem Ergebnis mit dem Datentyp TIMESTAMP(3) verknüpfen.
- Die Funktion LENGTH liefert die letzte Position in einer Zeichenkette, die kein Leerzeichen ist.
- Mit den Funktionen UPPERSTRING und LOWERSTRING werden Zeichen gemäß der länderspezifischen Einstellung umgesetzt, mit TRSTRING gemäß der benutzereigenen Definition.
- Bei der Berechnung von Zeitspannen mit Uhrzeiten (TIME) rechnet DRIVE/WINDOWS nicht über die Tagesgrenzen, sondern es werden negative Sunden ausgewiesen.
- NULL-Werte werden am Bildschirm, wenn nichts anderes vereinbart ist, durch das Sonderzeichen @ dargestellt.
- Der Aufruf von PASCAL-Programmen wird nicht unterstützt.

Schlüsselwörter

– DRIVE/WINDOWS verwendet neue Schlüsselwörter. Eine Aufstellung aller Schlüsselwörter befindet sich im Anhang des DRIVE-Lexikons [3].

Kompatibilität

– DRIVE-Programme, deren Code-Elemente mit Vorgängerversionen von DRIVE/WINDOWS erstellt wurden, müssen erneut übersetzt werden, damit sie ablauffähig sind.

– Für DRIVE-Programme, die auf SESAM-Datenbanken zugreifen, sind die Migrationshinweise im Handbuch "Lexikon der DRIVE-SQL-Anweisungen für SESAM V2" [5] zu beachten.

1.6 Darstellungsmittel

Die in den DRIVE/WINDOWS-Handbüchern verwendeten Zeichen und Schriftarten haben folgende Bedeutung:

`Schreibmaschinenschrift`

wird für feststehende Namen (z. B. Kommandos auf Betriebssytemebene, Dateinamen) und Fehlermeldungen im Fließtext verwendet. Außerdem wird sie in den Beispielen benutzt.

Kursive Schrift

kennzeichnet als Zwischenüberschrift Beispiele und im Fließtext frei wählbare Namen sowie Metavariablen.

i Dieses Zeichen weist Sie auf eine sehr wichtige Information hin, die Sie unbedingt beachten müssen.

Die verwendete Metasprache wird im DRIVE-Lexikon [3] beschrieben.

Bei Literaturverweisen, z.B. auf die oben erwähnten Handbücher, wird der Kurztitel zusammen mit einer Zahl in eckigen Klammern angegeben. In jedem Handbuch befindet sich im Anhang eine Literaturliste, die nach diesen Zahlen aufsteigend angelegt ist.

2 Aufbau und Syntax der Anweisungen

2.1 Aufbau

DRIVE-Anweisungen bestehen aus folgenden Elementen:

– Schlüsselwörtern
– Namen
– Literalen
– Metavariablen
– Trennzeichen
– Kommentaren

Beispiel

```
CYCLE cursorname INTO variable WHILE charname='literal' /*Schleife*/
```

Schlüsselwörter:	CYCLE, INTO, WHILE
Namen:	cursorname, charname
Literale:	literal
Metavariable:	variable
Trennzeichen:	Leerzeichen, Gleichheitszeichen (=)
Kommentar:	Schleife

Schlüsselwörter

Schlüsselwörter sind Wörter, die so angegeben werden müssen, wie sie im Handbuch dargestellt sind. Eine Aufstellung aller in DRIVE/WINDOWS verwendeten Schlüsselwörter befindet sich zusammen mit deren Abkürzungsmöglichkeiten im Anhang.

Namen

Namen kennzeichnen variable Werte, die bei der Eingabe vom Anwender durch aktuelle Werte ersetzt werden müssen.

Namen dürfen Buchstaben, Ziffern und Sonderzeichen enthalten, wenn keine weitergehenden Einschränkungen beschrieben sind.

Namen, die Buchstaben, Ziffern und Unterstriche (_) enthalten, müssen nicht besonders gekennzeichnet werden. Namen, die darüber hinaus weitere Sonderzeichen enthalten, müssen in Anführungszeichen (") gesetzt werden.

Literale

Literale sind Konstanten, die in der angegebenen Form vom Sprachübersetzer übernommen werden.

Numerische Literale werden direkt angegeben und hexadezimale Literale mit X'*literal*' angegeben.
Alphanumerische Literale müssen in Hochkommas gesetzt werden.
Für Datumzeit-Literale müssen Sie angeben, ob das Literal ein Datum, eine Uhrzeit oder einen Zeitstempel enthält. Für Intervall-Literale müssen Sie eine Einheit für die Zeitspanne angeben.

Hochkommas (') in Literalen müssen durch Hochkommas entwertet werden.

Beispiel

Das Literal "So geht's:" wird folgendermaßen geschrieben:

```
'So geht''s:'
```

Metavariablen

Metavariablen sind komplexe Anweisungsteile, die zur Erleichterung der Übersichtlichkeit aus einer Anweisung ausgegliedert werden. Sie werden in einem eigenen Kapitel beschrieben (siehe Kapitel „DRIVE-Metavariablen“ auf Seite 279).

Trennzeichen

Trennzeichen müssen zwischen Schlüsselwörtern, Namen, Literalen und Metavariablen angegeben werden, um sie eindeutig identifizieren zu können. Als Trennzeichen dienen:

– das Leerzeichen
– das Komma (,)
– der Verknüpfungsoperator ||
– alle Vergleichsoperatoren = < > <= >= <>
– alle Rechenoperatoren + – * / % **

Außerhalb von Zeichenfolgen, die in Hochkomma (') oder Anführungszeichen (") eingeschlossen sind, wirkt auch ein Kommentar oder ein Zeilenende wie ein Trennzeichen.

Kommentare

Ein Kommentar wird eingeleitet durch die Zeichenfolge /* und abgeschlossen durch die Zeichenfolge */. Zwischen diesen Kommentarzeichen kann beliebiger Text stehen, der auch über mehrere Zeilen gehen kann.

Die Zeichenfolgen /* und */ kennzeichnen keine Kommentare, wenn sie in Anführungszeichen (") oder Hochkommas (') stehen.

2.2 Syntax

Für die formale Darstellung der Anweisungen und Metavariablen werden folgende Metazeichen verwendet.

Formale Darstellung	**Erläuterung**	**Beispiel**
GROSSBUCH-STABEN	Großbuchstaben bezeichnen ein Schlüsselwort, das Sie in dieser Form eingeben müssen.	`COLUMNS`
Fettdruck	Die fettgedruckten Buchstaben bezeichnen die Abkürzung eines Schlüsselworts.	**`PERM`**`ANENT`
Kleinbuchsta-ben	Kleinbuchstaben bezeichnen eine Variable, für die Sie einen aktuellen Wert einsetzen müssen.	`LIBRARY=bibliothek`

( )	Runde Klammern sind Bestandteil der Anweisung.
	Runde Klammern müssen angegeben werden, sobald eine Angabe in runden Klammern steht.
		`bibliothek(elemname)`
		oder
		`CONCAT (charausdruck1,`
		`  charausdruck2)`
		oder
		`ATTRIBUTE (attribute, ...)`

{ }	Geschweifte Klammern fassen Einheiten zusammen.
	Geklammert wird von innen nach außen.
	Geschweifte Klammern dürfen nicht angegeben werden.
		`STATUS={ OFF | ADD |`
		`  REMOVE }`
		oder
		`USING { [ RETURN ] [ level ]`
		`  varname datendef }, ...`

[ ]	Eckige Klammern schließen Angaben ein, die weggelassen werden können.
	Geklammert wird von innen nach außen.
	Eckige Klammern dürfen nicht angegeben werden.
		`[ set transaction ]`
		oder
		`[ COBOL | C ] TAC tacname`

Formale Darstellung	**Erläuterung**	**Beispiel**
< >	Spitze Klammern sind Bestandteil der Anweisung. Spitze Klammern müssen angegeben werden, sobald eine Angabe in spitzen Klammern steht.	`aggregat=< { wert | NULL }, ... >`
|	Der senkrechte Strich trennt alternative Operandenwerte. Eine der Alternativen in geschweiften Klammern muß angegeben werden.	`LETTERS={ CAPITAL | BOTH |` `UNCHANGED }`

...	Punkte zeigen an, daß die unmittelbar davor stehende Variable mehrmals wiederholt werden kann.
	Steht vor den Punkten eine mit Klammern zusammengefaßte Einheit, muß sie insgesamt angegeben werden.
	Steht ein Komma oder ein Semikolon vor den Punkten, muß es bei Wiederholungen angegeben werden, um mehrere Angaben der gleichen Strukturstufe zu trennen.
		`AT zeile ...`
		`USING { [ RETURN ] [ level ]`
		`varname datendef }, ...`
		`(attribute, ...)`

3 DRIVE-Anweisungen

ACQUIRE
Speicherbereich anfordern

Diese Anweisung ist gültig

– in der UTM-Startprozedur als DRIVE-Startparameter

ACQUIRE fordert dimensionierbare Speicherbereiche (= Cache-Speicher) für den UTM-Betrieb an. Diese Cache-Speicher dienen als Zwischenspeicher für interne benutzerspezifische DRIVE-Informationen und bewirken eine Performancesteigerung.

```
ACQUIRE MEMORY mlength USER n
```

---

mlength

Größe eines Speicherbereichs innerhalb des Cache-Speichers in Kbyte.

Wertebereich: 0 < *mlength* < 2147483647.
mlength muß eine ganze Zahl sein.

Die Größe des Cache-Speichers in Byte ergibt sich aus:
n * *mlength* * 1024

Die so ermittelte Größe wird noch aufgerundet und zwar auf ein ganzzahliges Vielfaches von:

– 64 Kbyte, wenn DRIVE/WINDOWS im unteren Adreßraum (< 16 Mbyte) geladen ist
– 1 Mbyte, wenn DRIVE/WINDOWS im oberen Adreßraum (> 16 Mbyte) geladen ist

Soll im Mischbetrieb mit DRIVE 5.1 auf denselben Cache-Speicher zugegriffen werden, müssen die Gesamtgrößen übereinstimmen.

n Anzahl der DRIVE-UTM-Anwender, deren interne benutzerspezifische DRIVE-Informationen bei einem UTM-Dialogschrittwechsel gleichzeitig im COMMON MEMORY POOL (Klasse-6-Speicher) zwischengespeichert werden sollen.
Wertebereich: $0 < n < 32767$.

Beispiel

15 DRIVE-UTM-Anwendern wird jeweils 100 Kbyte Speicherplatz zur Zwischenspeicherung benutzerspezifischer Daten zur Verfügung gestellt.

```
ACQUIRE MEMORY 100 USER 15
```

ADD BOX
Dialog-Box ausgeben

Diese Anweisung ist gültig

– im UTM-Betrieb, aber nicht im UTM-Asynchronbetrieb und bei VTV nicht in Auftragnehmer-Umgebung
– im Programm-Modus

ADD BOX gibt eine Dialog-Box aus, die Sie zuvor mit IFG erstellt haben (siehe IFG [28]).

Bereits ausgegebene Bildschirmformate (Teilformate und Dialog-Boxen) bleiben erhalten, werden aber von der Dialog-Box überlagert und sind gesperrt, d.h. es sind keine Benutzereingaben in diese Bildschirmformate möglich.

Die zuletzt ausgegebene Dialog-Box ist die aktuelle Dialog-Box. Benutzereingaben sind nur in dieser aktuellen Dialog-Box möglich.

Die Anweisung ADD BOX darf nur angegeben werden, wenn bereits ein FHS-DE-Teilformat mit der Anweisung DISPLAY screenformat ausgegeben wurde, sonst bricht DRIVE/ WINDOWS das Programm ab.

```
ADD BOX dialogbox
        [ POSITION ( zeile1 , spalte1 ) ] [ TO feld1 ]
        [ CURSOR { POSITION ( zeile2 , spalte2 ) | TO feld2 }
        [ MESSAGE schluessel [ POSITION ( zeile3 , spalte3 ) | TO feld3 ]
```

---

dialogbox — Name des FHS-DE-Formats (max. 7 Zeichen).

Das Format muß mit IFG mit der Eigenschaft "Anzeige in einer Box" erstellt worden sein.

Das Format muß im Deklarationsteil des Programms mit der Anweisung DECLARE SCREEN definiert sein.

POSITION — legt die Position der Dialog-Box fest.

Die Position wird über den Start- oder Bezugspunkt festgelegt. Der Startpunkt ist das erste Zeichen (links oben) der Dialog-Box. Der Bezugspunkt ist das erste Zeichen von *feld1*, falls *feld1* angegeben ist, oder die linke obere Ecke der darunterliegenden Dialog-Box / des darunterliegenden FHS-DE-Teilformats.

	Wenn Sie POSITION nicht oder mit (0,0) angeben, versucht DRIVE/WINDOWS die Dialog-Box mit der Standardverschiebung (+2,+2) zum Bezugspunkt zu positionieren. Ist dies nicht möglich, wird die Dialog-Box so verschoben, daß sie auf den Bildschirm paßt. Wenn POSITION angegeben ist, aber der Platz an der vorgegebenen Position für die Dialog-Box nicht ausreicht, bricht UTM den Vorgang mit PEND ER ab.
zeile1	Zeilenabstand zwischen Bezugspunkt und Startpunkt der Dialog-Box. *zeile1* muß eine ganze Zahl sein.
spalte1	Spaltenabstand zwischen Bezugspunkt und Startpunkt der Dialog-Box. *spalte1* muß eine ganze Zahl sein.
feld1	Feld im zuletzt ausgegebenen FHS-DE-Format (Teilformat und Dialog-Box). *feld1* muß eine einfache Komponente der zugehörigen SCREEN-Variablen sein. Wenn *variable* nicht die Komponente der SCREEN-Variablen zum zuletzt ausgegebenen Bildschirmformat ist, bricht UTM den Vorgang mit PEND ER ab. Damit die Komponenten von SCREEN-Variablen eindeutig zugeordnet werden können, müssen sich die ersten 8 Zeichen der Feldnamen im zuletzt ausgegebenen Bildschirmformat unterscheiden.
CURSOR	Die Schreibmarke wird an eine bestimmte Stelle auf den Bildschirm gesetzt. CURSOR dürfen Sie nur angeben, wenn mit IFG für *dialogbox* das Globalattribut "Schreibmarkenposition" gesetzt wurde (siehe IFG [28]).
POSITION	legt die absolute Position (Zeile / Spalte) der Schreibmarke fest.
zeile2	Zeile (1 ≤ *zeile2* ≤ Anzahl der Bildschirmzeilen). *zeile2* muß eine ganze Zahl sein.
spalte2	Spalte (1 ≤ *spalte2* ≤ Anzahl der Bildschirmspalten). *spalte2* muß eine ganze Zahl sein.
TO	Die Schreibmarke wird auf das erste Zeichen des Felds *feld2* gesetzt. Bei Listen wird die Schreibmarke auf die erste Spalte und die erste Zeile des Listenbereichs gesetzt.
feld2	Feld in der Dialog-Box, die ausgegeben werden soll. *feld2* muß eine Komponente der SCREEN-Variablen zu *dialogbox* sein.

Damit die Komponenten von SCREEN-Variablen eindeutig zugeordnet werden können, müssen sich die ersten 8 Zeichen der Feldnamen in der aktuellen Dialog-Box unterscheiden.

MESSAGE

Die FHS-DE-Meldung mit Meldungsschlüssel *schluessel*, die Sie mit IFG erstellt haben, wird ausgegeben. Abhängig von der Festlegung mit IFG erfolgt die Ausgabe entweder in einer Meldungsbox oder im Meldebereich der Dialog-Box.

MESSAGE dürfen Sie nur angeben, wenn mit IFG für *dialogbox* das Globalattribut "Meldungskennzeichen" gesetzt wurde (siehe IFG [28]).

schluessel

Meldungsschlüssel der FHS-DE-Meldung. *schluessel* kann als Variable (siehe Metavariable *variable*) oder als alphanumerisches Literal (siehe *charliteral* bei Metavariable *literal*) angegeben werden.

schluessel muß in der Form AAAAnnn angegeben werden. A steht für einen Buchstaben (A-Z), n für eine Ziffer (0-9). Für AAAA darf nicht IDHS stehen.

POSITION

legt die absolute Position der Meldungsbox fest. Die Meldungsbox wird mit einer zusätzlichen Verschiebung (+2,+2) zu *zeile3*, *spalte3* positioniert.

POSITION wird nur ausgewertet, wenn mit IFG festgelegt wurde, daß die Meldung in eine Meldungsbox ausgegeben werden soll.

Wenn eine Meldung in einer Meldungsbox ausgegeben werden soll, und Sie weder POSITION noch TO angeben, wird die Meldungsbox in die Mitte des Bildschirms plaziert.

Wenn eine mit MESSAGE POSITION positionierte Meldungsbox eine Schreibmarke überdeckt, die mit CURSOR POSITION gesetzt ist, wird MESSAGE POSITION ignoriert.

zeile3

Zeile (1 ≤ *zeile3* ≤ Bildschirmzeilen). *zeile3* muß eine ganze Zahl sein.

spalte3

Spalte (1 ≤ *spalte3* ≤ Bildschirmspalten). *spalte3* muß eine ganze Zahl sein.

TO

legt fest, daß die Meldungsbox mit der Standardverschiebung (+2,+2) zu *feld3* positioniert werden soll.

TO wird nur ausgewertet, wenn mit IFG festgelegt wurde, daß die Meldung in eine Meldungsbox ausgegeben werden soll.

Wenn eine Meldung in einer Meldungsbox ausgegeben werden soll, und Sie weder TO noch POSITION angeben, wird die Meldungsbox in die Mitte des Bildschirms plaziert.

feld3 Feld in der Dialog-Box, die ausgegeben werden soll. *feld3* muß eine Komponente der SCREEN-Variablen zu *dialogbox* sein.

Damit die Komponenten von SCREEN-Variablen eindeutig zugeordnet werden können, müssen sich die ersten 8 Zeichen der Feldnamen in der aktuellen Dialog-Box unterscheiden.

AT
Testpunkt und Aktion vereinbaren

Diese Anweisung ist gültig

– im TIAM-Betrieb
– im Debug-Modus

AT legt im Debug-Modus Testpunkte und Aktionen für ein Programm fest.

Den vom Anwender definierten Testpunkten ordnet DRIVE/WINDOWS immer eine Anweisung des DRIVE-Programms zu.

Die AT-Anweisung bezieht sich entweder auf das aktuelle im Debug-Modus laufende Programm oder bei der Angabe von *bibliothek* oder *elemname* auf ein Unterprogramm aus der DRIVE-Bibliothek.

Sind keine Aktionen angegeben, wird das Programm am Testpunkt angehalten (= implizite Aktion [STOP]). Der Testpunkt wird zum Haltepunkt. An Haltepunkten kann eine der folgenden Debug-Anweisungen eingegeben werden:

– AT
– BREAK
– BREAK DEBUG
– CONTINUE
– DISPLAY FORM
– DISPLAY LIST
– REMOVE
– SET
– TRACE

Mit der Anweisung CONTINUE wird der Debug-Lauf fortgesetzt.

Für denselben Testpunkt können mit mehreren AT-Anweisungen mehrere Aktionen festgelegt werden. Bei Erreichen eines Testpunkts werden die festgelegten Aktionen in folgender Reihenfolge ausgeführt: Zunächst werden alle Debug-Anweisungen DISPLAY und SET in der Reihenfolge ihrer Eingabe ausgeführt. Danach wird entweder die Debug-Anweisung CONTINUE ausgeführt oder die Debug-Anweisung TRACE ausgeführt oder das Programm angehalten (=implizites [STOP]), je nachdem welche Anweisung zuletzt vereinbart wurde. Die Debug-Anweisung COUNT wird erst ausgeführt, wenn die zum Testpunkt gehörende Programmanweisung fehlerfrei abgeschlossen wurde.

Die Aktionen CONTINUE, TRACE und [STOP] überschreiben sich gegenseitig.

Solange eine Aktion für einen Testpunkt hinterlegt ist, wird diese Aktion jedesmal durchgeführt, wenn die Anweisung ausgeführt wird, der der Testpunkt zugeordnet ist.

```
AT { [ bibliothek(elemname) | elemname ] { zeile ... | zeile1 – zeile2 | ALL } |
     * }
   [ { CONTINUE | COUNT | DISPLAY FORM | DISPLAY LIST | SET | TRACE } ... ]
```

bibliothek	Name der DRIVE-Bibliothek (max. 54 Zeichen), aus der das Programm eingelesen wird. *bibliothek* kann auch der Dateikettungsname der DRIVE-Bibliothek (entsprechend den BS2000-Konventionen) sein. DRIVE/WINDOWS interpretiert *bibliothek* zuerst als Dateikettungsnamen, dann als Bibliotheksnamen. Wird die DRIVE-Bibliothek mit der Anweisung PARAMETER DYNAMIC LIBRARY voreingestellt, so kann die Angabe von *bibliothek* entfallen.
elemname	Name des Elements (max. 31 Zeichen), das das Programm enthält. DRIVE/WINDOWS sucht nach dem zuletzt bearbeiteten Element, unabhängig davon, ob dieses eine Source enthält (S-Element) oder einen Zwischencode (X-Element). Wird keine Angabe für *bibliothek* gemacht, wird die bei PARAMETER DYNAMIC LIBRARY angegebene Bibliothek eingesetzt. Wird keine Angabe für *elemname* gemacht, wird der Testpunkt in dem Programm gesetzt, das im Debug-Modus läuft.
*	Die Angabe * bezeichnet den Testpunkt, der zuletzt eingegeben wurde.
zeile	Die Angabe von *zeile* bezieht sich auf eine Zeilennummer in der Übersetzungsliste. In der Zeile muß eine ausführbare Anweisung beginnen, für die ein Testpunkt festgelegt wird. Ausnahme: Für die PROCEDURE-Anweisung des mit DEBUG gerufenen Programms kann kein Testpunkt festgelegt werden. Es können mehrere Zeilennummern angegeben werden. Beginnen in Zeile *zeile* mehrere Anweisungen, so wird für jede Anweisung ein Testpunkt vereinbart.

zeile1-zeile2	Die Angabe bezieht sich auf Zeilennummern in der Übersetzungsliste. Für jede Programmanweisung, die in einer Zeile dieses Bereichs beginnt, wird ein Testpunkt vereinbart. *zeile1* muß kleiner als *zeile2* sein.
ALL	Für alle ausführbaren Programmanweisungen wird ein Testpunkt vereinbart.
CONTINUE	Nachdem der Testpunkt erreicht und dort hinterlegte Aktionen ausgeführt sind, wird das im Debug-Modus laufende Programm fortgesetzt.
COUNT	Für jede Programmanweisung, die zu einem Testpunkt gehört, wird ein Durchlaufzähler vereinbart. Der Durchlaufzähler wird mit 0 initialisiert und nach jeder fehlerfreien Ausführung der Anweisung um 1 erhöht. Durchlaufzähler werden am Ende der Debug-Sitzung in die Ergebnisliste ausgegeben. Mit der Anweisung REMOVE werden Durchlaufzähler gelöscht.
DISPLAY FORM	siehe Anweisung DISPLAY FORM. Die Kennzeichnung von Variablen mit RETURN wird ignoriert.
DISPLAY LIST	siehe Anweisung DISPLAY LIST.
SET	siehe Anweisung SET.
TRACE	siehe Anweisung TRACE.

AT-Anweisungen beziehen sich auf das jeweilige aktuelle Programm oder Unterprogramm. Sie wirken sich nicht aus auf Folgeprogramme, die mit DO aufgerufen werden.

Beispiel

Das Programm "test" wird im Debug-Modus getestet.

Alle ausführbaren Programmanweisungen werden gezählt. Die Anweisungen in den Zeilen 15, 17 und 20 bis 55 der Übersetzungsliste des Hauptprogramms "test" sind Testpunkte. Nacheinander wird die Variable &var1 auf 1 gesetzt, auf einen Drucker und auf den Bildschirm ausgegeben. Danach wird der Programmablauf fortgesetzt.

An der Anweisung in Zeile 33 der Übersetzungsliste des Unterprogramms "test2", das sich in der voreingestellten Bibliothek befindet, wird die Variable &subvar1 auf 2 gesetzt und auf einen Drucker ausgegeben.

Ab der Anweisung in der Zeile 99 der Übersetzungsliste des Hauptprogramms "test" wird die Ablaufverfolgung des Programms eingeschaltet und der Debug-Lauf fortgesetzt.

```
DEBUG test;          /* DRIVE haelt vor der ersten ausfuehrbaren    */
                     /* Anweisung des Verarbeitungsteils an          */
AT ALL COUNT;
AT 15 SET &var1 = 1
AT 17 DISPLAY LIST &var1
AT 20 - 55 DISPLAY FORM &var1
AT * CONTINUE
AT test2 33 SET &subvar1 = 2
AT test2 33 DISPLAY LIST &subvar1
AT 99 TRACE
CONTINUE             /* Der Debug-Lauf wird jetzt erst fortgesetzt   */
...
```

BREAK
Bildschirm löschen oder logischen Programmteil abbrechen

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Dialog-, Programm- und Debug-Modus mit unterschiedlicher Funktion (s.u.)

Abhängig vom Modus hat BREAK drei Funktionen:

– Im Dialog-Modus wird die laufende Funktion abgebrochen und der Bildschirm gelöscht. Wird BREAK innerhalb eines geschachtelten Programm-Ablaufs angegeben, werden alle Programme abgebrochen, und es wird in den Dialog-Modus geschaltet.

– Im Debug-Modus wird nach BREAK zum Endhaltepunkt (hinter die END PROCEDURE-Anweisung) des getesteten Programms verzweigt.

  Nach BREAK DEBUG wird die Ergebnisliste ausgegeben, falls ein Durchlaufzähler vereinbart wurde (Anweisung AT ... COUNT). Danach wird der Debug-Modus verlassen und in den Dialog-Modus geschaltet.

– Im Programm-Modus bricht BREAK logische Programmteile ab (eine Schleife, eine Verzweigung, ein Programm, eine Programmhierarchie, ein internes Unterprogramm).

Weitere Möglichkeiten ein Programm abzubrechen sind:

– Belegen einer K-Taste mit der Funktion BREAK (Vorbelegung von DRIVE/WINDOWS: K1-Taste)

– Eingeben des BS2000-Kommandos `SEND-MESSAGE TO=PROGRAM,MESSAGE=BREAK`. (Diese Möglichkeit besteht nur im TIAM-Betrieb.)

```
BREAK [ CYCLE | DEBUG | PROCEDURE | SUBPROCEDURE ]
```

CYCLE

Eine Schleife wird abgebrochen und das Programm mit der Anweisung fortgesetzt, die dem zugehörenden END CYCLE folgt.

Wird BREAK CYCLE innerhalb einer Cursorschleife (CYCLE *cursorname* ... bis END CYCLE) angegeben, wird implizit ein CLOSE *cursorname* durchgeführt.

BREAK CYCLE muß innerhalb einer CYCLE-Klammer (CYCLE bis END CYCLE) stehen. Ein BREAK CYCLE innerhalb eines internen Unterprogramms kann keine Schleife im rufenden Programm abbrechen.

DEBUG

Die Anweisung BREAK DEBUG ist nur im Debug-Modus erlaubt. Sie bewirkt die Ausgabe der Ergebnisliste, den Abbruch des Debug-Modus sowie den Übergang in den Dialog-Modus.

PROCEDURE

Ein Programm wird abgebrochen.

Wurde das Programm mit CALL aufgerufen, wird in das rufende Programm zurückgesprungen und mit der Anweisung fortgesetzt, die dem CALL-Aufruf folgt.

Wurde das Programm mit DO aufgerufen, geht DRIVE/WINDOWS in den Dialog-Modus über. Alle benötigten Betriebsmittel werden freigegeben und alle offenen Transaktionen zurückgesetzt. Ein Wiederanlauf des Programms ist nicht möglich.

Wird BREAK PROCEDURE innerhalb eines geschachtelten Programmablaufs angegeben, wird nur die aktuelle Programmstufe abgebrochen. BREAK PROCEDURE innerhalb eines internen Unterprogramms wirkt wie ein END PROCEDURE. Es gelten dieselben Regeln. (Siehe Anweisung END).

Bei BREAK PROCEDURE eines mit DO gerufenen Programms schließt DRIVE/WINDOWS alle mit der Anweisung OPEN FILE geöffneten Dateien. Bei BREAK PROCEDURE eines mit CALL gerufenen Programms bleiben mit der Anweisung OPEN FILE geöffnete Dateien weiterhin geöffnet.

BREAK PROCEDURE darf nicht innerhalb eines DISPATCH-Blocks stehen.

SUBPROCEDURE

Ein internes Unterprogramm wird abgebrochen und das rufende Programm mit der Anweisung fortgesetzt, die dem CALL-Aufruf des rufenden Programms folgt. Das rufende Programm kann ebenfalls ein internes Unterprogramm sein.

BREAK SUBPROCEDURE muß statisch innerhalb einer SUBPROCEDURE-Klammer (SUBPROCEDURE bis END SUBPROCEDURE) stehen.

Regeln bei Datenbankzugriff

Für BREAK DEBUG gilt:

– Ist noch eine Transaktion offen, setzt DRIVE/WINDOWS sie zurück und gibt die Meldung DRI0101 aus.

– Sind temporäre, im Programm-Modus definierte SQL-Objekte (Programm-Cursor oder temporäre Views) vorhanden, werden sie von DRIVE/WINDOWS gelöscht.
Wenn ein SQL-Objekt nicht gelöscht werden kann, gibt DRIVE/WINDOWS die Meldung DRI0150 aus.

– Sind beim Zugriff auf SESAM V2.x dynamische, temporäre Views vorhanden, löscht DRIVE/WINDOWS sie und gibt die Meldung DRI0488 aus.

– Beim Zugriff auf SESAM V2.x setzt DRIVE/WINDOWS eine SET SESSION-, eine SET CATALOG- und eine SET SCHEMA-Anweisung ab, deren jeweiliger Operand durch die letzte vorausgegangene PARAMETER DYNAMIC AUTHORIZATION-, PARAMETER DYNAMIC CATALOG- und PARAMETER DYNAMIC SCHEMA-Anweisung bestimmt ist.

CALL
Unterprogramm aufrufen

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

CALL ruft aus einem Programm sowohl interne als auch externe Unterprogramme auf.

Das rufende Programm wird unterbrochen und das Unterprogramm durchlaufen. Anschließend wird in das rufende Programm zurückgesprungen und mit der Anweisung fortgefahren, die unmittelbar dem CALL-Aufruf folgt.

Unter einem internen Unterprogramm versteht man eine benannte DRIVE-Anweisungsfolge, die innerhalb eines DRIVE-Programms beliebig oft aufgerufen werden kann. Innerhalb eines internen Unterprogramms können andere interne Unterprogramme nur aufgerufen werden, wenn sie vorher definiert wurden.

Unter einem externen Unterprogramm versteht man ein eigenständiges Programm, das von anderen Programmen beliebig oft aufgerufen werden kann. Externe Unterprogramme können in DRIVE/WINDOWS (New-Style oder Old-Style) geschrieben sein oder auch in einer anderen Programmiersprache (z.B. COBOL oder C). Es können auch UTM-Teilprogramme sein, die mit dem Transaktionscode aufgerufen werden.

Unabhängig von der Programmiersprache können externe Unterprogramme lokal oder in einem entfernten System vorhanden sein. Abhängig von der mit PARAMETER DISTRIBUTION angegebenen Verteilungsinformation sucht DRIVE/WINDOWS nach einem Programm im lokalen oder entfernten System, wenn OPTION DISTRIBUTION=ON gesetzt ist. CALL-Anweisungen, die externe Unterprogramme in entfernten Systemen aufrufen, werden Remote-CALL-Anweisungen genannt.

DRIVE-Programme in einem entfernten System dürfen folgende Anweisungen nicht enthalten:

– CALL ... TAC (wenn das UTM-Teilprogramm im lokalen System ausgeführt werden soll)
– COMMIT WORK WITH DISPLAY
– COMMIT WORK WITH SEND MESSAGE
– DISPLAY
– DO
– FILL
– ROLLBACK WORK WITH RESET
– SEND MESSAGE
– STOP WITH DISPLAY
– STOP WITH charausdruck

Externe DRIVE-Unterprogramme können sowohl als Zwischencode als auch als Source vorliegen. DRIVE/WINDOWS sucht unter dem angegebenen Namen nach dem zuletzt bearbeiteten Programm, unabhängig davon, ob dieses als Source oder als Zwischencode vorliegt. Wird ein Zwischencode gefunden, entfällt die Syntax- und Semantiküberprüfung (siehe Anweisung DO).

Bei "CALL ..." ohne Angabe einer Bibliothek sucht DRIVE/WINDOWS zum Übersetzungszeitpunkt der Source nach einem internen Unterprogramm mit dem angegebenen Namen im aktuellen Programm (CALL *subprogname*).
Falls kein internes Unterprogramm mit dem angegebenen Namen existiert, sucht DRIVE/WINDOWS zum Ablaufzeitpunkt ein Element in der Bibliothek, die mit der Anweisung PARAMETER DYNAMIC LIBRARY festgelegt ist (CALL *elemname*). Falls kein Element dieses Namens existiert, wird das Programm abgebrochen.

In einer EXECUTE-Anweisung wird bei der Anweisung "CALL ..." ohne Angabe einer Bibliothek ein externes DRIVE-Unterprogramm aufgerufen.

Rekursive Programmaufrufe sind nicht erlaubt und führen zum Programmabbruch.

```
CALL { subprogname |
       bibliothek(elemname) |
       elemname |
       [ COBOL | C ] { MODULE modulname | TAC tacname }

     [ USING { [ RETURN ]  { ausdruck | NULL }
               [ INIT ausdruck1 [ NOCHECK ] ] [ INDICATOR ] }, ... ] }
```

---

subprogname — Name eines internen Unterprogramms (max. 31 Zeichen). Das Unterprogramm muß im aktuellen, aufrufenden Programm mit SUBPROCEDURE *subprogname* definiert sein.

Die Angabe einer USING-Klausel ist bei CALL *subprogname* nicht erlaubt.

bibliothek — Name der DRIVE-Bibliothek (max. 54 Zeichen), aus der das Unterprogramm eingelesen wird.

bibliothek kann auch der Dateikettungsname der DRIVE-Bibliothek (entsprechend den BS2000-Konventionen) sein.

DRIVE/WINDOWS interpretiert *bibliothek* zuerst als Dateikettungsnamen, dann als Bibliotheksnamen.

	Wird die DRIVE-Bibliothek für das (lokale oder entfernte) System mit der Anweisung PARAMETER DYNAMIC LIBRARY voreingestellt, so kann die Angabe von *bibliothek* entfallen.
elemname	Name des Elements (max. 31 Zeichen), das das Unterprogramm enthält. DRIVE/WINDOWS sucht nach dem zuletzt bearbeiteten Element, unabhängig davon, ob dieses eine Source enthält (S-Element) einen Zwischencode (X-Element) oder einen Objektcode (R-Element). *elemname* kann ein Old-Style-Programm sein, das mit DRIVE V5.1 erstellt wurde. Was Sie beim Aufruf eines Old-Style-Programms beachten müssen, ist in der DRIVE-Programmiersprache [2] beschrieben. Nur wenn die Übersetzungsoption OPTION DISTRIBUTION=ON definiert wurde, wird überprüft, ob mit der Anweisung PARAMETER DISTRIBUTION eine Verteilungsinformation festgelegt wurde. Gemäß der Verteilungsinformation wird ein lokales oder ein entferntes Programm aufgerufen. Ist mit der Anweisung PARAMETER DISTRIBUTION keine Verteilungsinformation definiert, wird das Programm lokal gesucht. Wird keine Angabe für *bibliothek* gemacht, wird die für das (lokale oder entfernte) System mit PARAMETER DYNAMIC LIBRARY voreingestellte Bibliothek eingesetzt.
COBOL	Vorbelegung Sprachspezifische Steuerung des Programmaufrufs für ein in der Programmiersprache COBOL geschriebenes Unterprogramm oder UTM-Teilprogramm.
C	Sprachspezifische Steuerung des Programmaufrufs für ein in der Programmiersprache C geschriebenes Unterprogramm oder UTM-Teilprogramm.
modulname	Name des Unterprogramms (max. 8 Zeichen), das aufgerufen wird (gilt nur für externe Unterprogramme in anderen Programmiersprachen). Das externe Unterprogramm muß in der Bibliothek mit dem Dateikettungsnamen USEROML vorhanden sein. Diese Bibliothek wird automatisch verwendet, wenn mit der Anweisung PARAMETER DYNAMIC LIBRARY keine andere Bibliothek angegegeben wurde.

Das externe Unterprogramm im entfernten System darf keine Datenbankanweisungen (SESAM/UDS) enthalten. Die Parameter werden über einen Parameterübergabebereich ausgetauscht (siehe DRIVE-Programmiersprache [2]).

tacname

Name des Transaktionscodes (max. 8 Zeichen) eines anwendereigenen UTM-Teilprogramms. Das UTM-Teilprogramm kann Teil der lokalen oder einer entfernten UTM-Anwendung sein. Die UTM-Anwendung kann unter SINIX oder unter BS2000 laufen.

tacname darf nicht das Präfix `dri`, `drt`, `drc` oder `sql` haben. Diese Namen sind von DRIVE/WINDOWS reserviert.

Beim Aufruf eines anwendereigenen UTM-Teilprogramms muß die Programmiersprache, in der es geschrieben wurde, angegeben werden.

Nur wenn die Übersetzungsoption OPTION DISTRIBUTION=ON definiert wurde, wird überprüft, ob mit der Anweisung PARAMETER DISTRIBUTION eine Verteilungsinformation festgelegt wurde. Gemäß der Verteilungsinformation wird ein entferntes UTM-Teilprogramm aufgerufen. Ist mit der Anweisung PARAMETER DISTRIBUTION keine Verteilungsinformation definiert, wird das UTM-Teilprogramm lokal gesucht.

Im TIAM-Betrieb wird CALL TAC ignoriert.

USING

Mit USING werden Parameter des rufenden Programms an das aufgerufene Programm übergeben. Die Parameter müssen zuweisungsverträglich sein (siehe DRIVE-Programmiersprache [2]). Im Debug-Modus wird andernfalls beim Aufruf eines externen DRIVE-Programms (CALL *[(bibliothek)]elemname)* ein Parameter-Prompting durchgeführt (siehe DRIVE-Programmiersprache [2]).

Bei CALL MODULE ... USING oder CALL TAC ... USING darf die Gesamtlänge der übergebenen Datenwerte 31 Kbyte nicht überschreiten. Beim Aufruf von externen DRIVE-Unterprogrammen im lokalen System gibt es keine Einschränkung.

Ist das externe Unterprogramm ein DRIVE-Programm, muß es mit PROCEDURE...USING... definiert worden sein, sonst wird das Programm abgebrochen.

Ist das externe Unterprogramm ein UTM-Teilprogramm, werden die übergebenen Parameter im Nachrichtenbereich hinterlegt. Die UTM-Anweisung MGET fordert die übergebenen Daten von UTM für das Teilprogramm an (siehe DRIVE-Programmiersprache [2]).

Für externe Unterprogramme in anderen Programmiersprachen gelten die in der DRIVE-Programmiersprache [2] beschriebenen Regeln.

USING ist beim Aufruf eines internen Unterprogrogramms (CALL *subprogname*) nicht erlaubt.

RETURN

Mit RETURN werden die Parameter gekennzeichnet, die wieder an das rufende Programm zurückgegeben werden sollen. Bei externen Unterprogrammen in DRIVE müssen sie auch im gerufenen Programm mit PROCEDURE ... RETURN ... gekennzeichnet sein. Wird bei PROCEDURE kein RETURN angegeben, wird das Programm abgebrochen.

Bei externen Unterprogrammen in anderen Programmiersprachen werden in die RETURN-Parameter die Datenwerte übernommen, die dieses Unterprogramm im Parameterübergabebereich hinterlegt hat.

Ist das externe Unterprogramm ein UTM-Teilprogramm, werden in die RETURN-Parameter die Datenwerte übernommen, die dieses Unterprogramm im Nachrichtenbereich hinterlegt hat (siehe DRIVE-Programmiersprache [2]).

Eine Variable, die mit RETURN gekennzeichnet ist, darf nur einmal in der USING-Klausel mit RETURN angegeben werden.

ausdruck

Angabe der zu übergebenden Parameter (Sendefelder).

An externe Unterprogramme in DRIVE/WINDOWS können Variablen (auch strukturierte Variablen), Vektoren, Matrizen, Aggregate, Literale und Rechenausdrücke übergeben werden.

An externe Unterprogramme in DRIVE/WINDOWS im entfernten System können Variablen (auch strukturierte Variablen), Vektoren, Matrizen, Literale und Rechenausdrücke übergeben werden.

An externe Unterprogramme in anderen Programmiersprachen (CALL MODULE) und an Transaktionscodes (CALL TAC) können Variablen (auch strukturierte Variablen), Vektoren, oder Matrizen übergeben werden. *ausdruck* muß eine Variable sein (siehe Metavariable *variable*).

Hat bei CALL MODULE oder bei CALL TAC *ausdruck* den NULL-Wert, muß die INDICATOR-Klausel angegeben sein. Ansonsten wird das Programm abgebrochen.

Wird RETURN oder INIT angegeben, muß *ausdruck* eine Variable sein.

NULL	Der NULL-Wert wird an das Unterprogramm übergeben. NULL ist bei CALL MODULE und CALL TAC nicht erlaubt.
INIT	Mit INIT wird *ausdruck* ein Wert zugewiesen. Ist *ausdruck* ein Vektor oder eine Matrix, erhalten alle Komponenten den Wert *literal*.
ausdruck1	*ausdruck1* darf nur Literale, NULL oder Funktionen, deren Argumente Literale sind (nicht CURRENT DATE/TIME/TIMESTAMP) enthalten. D.h. *ausdruck1* muß zum Übersetzungszeitpunkt berechenbar sein.
NOCHECK	legt fest, daß *ausdruck1* nicht auf eine gegebenenfalls vorhandene CHECK-Klausel von *ausdruck* überprüft wird.
INDICATOR	Mit INDICATOR wird eine Indikatorvariable angelegt. Der Wert der Indikatorvariablen gibt an, ob der Übergabeparameter den Wert NULL oder einen definierten Wert enthält. Die Angabe INDICATOR ist nur bei CALL MODULE und bei CALL TAC erlaubt.

Beispiel

Das externe DRIVE-Unterprogramm "mitkorr" in der voreingestellten Bibliothek wird aufgerufen. Dabei wird der Parameter &vgl1 dem Unterprogramm übergeben.

```
CALL mitkorr USING &vgl1;
```

Beziehungen zu anderen Anweisungen

- CALL-Anweisungen, die ein UTM-Teilprogramm aufrufen (CALL TAC), sind nicht erlaubt in Programmen, die mit ENTER gestartet werden.
- CALL-Anweisungen, die ein Old-Style-Programm aufrufen, sind nicht erlaubt in Programmen, die mit ENTER gestartet werden.
- Remote-CALL-Anweisungen werden nicht sequentiell, sondern gleichzeitig ausgeführt, wenn sie innerhalb eines DISPATCH-Blocks stehen (siehe Anweisung DISPATCH).
- Remote-CALL-Anweisungen sind nicht erlaubt in Programmen, die mit ENTER gestartet werden.
- Wurde das Unterprogramm mit der Compiler-Option OPTION OBJECT=ON übersetzt, darf beim Programmaufruf *bibliothek* nicht angegeben werden.
- Wurde ein Programm mit der Übersetzungsoption OPTION DISTRIBUTION=ON übersetzt, sucht DRIVE/WINDOWS entsprechend der Verteilungsinformation nach einem Programm im lokalen oder entfernten System.

Regeln bei Datenbankzugriff

Für Unterprogramme, die im lokalen System aufgerufen werden, gilt:

– Wenn dem rufenden Programm und dem gerufenen Unterprogramm unterschiedliche Datenbanksysteme zugeordnet sind (DBSYSTEM ≠ OFF), wird die Anweisung CALL abgebrochen. Die unterschiedliche Zuordnung von Datenbanksystemen ist nur möglich, wenn mit CALL Zwischencode oder Objektcode aufgerufen wird, der der in einer früheren DRIVE-Sitzung mit einem anderen Datenbanksystem erzeugt wurde.

– Wenn dem rufenden Programm und dem gerufenen Unterprogramm jeweils das Datenbanksystem SESAM V2.x zugeordnet ist (DBSYSTEM = SESAMSQL), wird die Anweisung CALL nur dann ausgeführt, wenn entweder für das rufende Programm keine Transaktion offen ist oder wenn für das gerufene Unterprogramm Zwischencode oder Objektcode erzeugt wurde. Dabei ist zu beachten, daß DRIVE/WINDOWS stets das zuletzt bearbeitete Element verwendet, unabhängig davon, ob dieses eine Source enthält (S-Element), einen Zwischencode (X-Element) oder einen Objektcode (R-Element).

– Wenn dem rufenden Programm ein Datenbanksystem zugeordnet ist (DBSYSTEM ≠ OFF) und dem gerufenen Unterprogramm nicht (DBSYSTEM = OFF), so greift das gerufene Unterprogramm auf dasselbe Datenbanksystem zu wie das rufende Programm.

– Wenn dem gerufenen Unterprogramm ein BS2000-Datenbanksystem zugeordnet ist (DBSYSTEM = UDS / SESAM / SESAMSQL), wird die Anweisung CALL nur dann ausgeführt, wenn dieses Datenbanksystem der geladenen Variante entspricht.

– Wenn das gerufene Unterprogramm ein Old-Style-Programm ist, wird die Anweisung CALL abgebrochen, wenn dem rufenden Programm das Datenbanksystem UDS zugeordnet ist (DBSYSTEM = UDS).

– Wenn das gerufene Unterprogramm ein Old-Style-Programm ist, wird die Anweisung CALL nur dann ausgeführt, wenn für das rufende Programm keine (New-Style-)Transaktion offen ist.

CASE
Bedingte Verzweigung programmieren

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

CASE kennzeichnet den Beginn eines CASE-Blocks, dessen Ende mit END CASE festgelegt wird. Die erste Anweisung nach CASE muß eine OF-Anweisung sein. Die Anweisungsfolge nach einer OF-Anweisung bis zur nächsten OF-Anweisung oder wenn keine weitere OF-Anweisung vorhanden ist, bis zu END CASE, wird als OF-Zweig bezeichnet.

Innerhalb eines CASE-Blocks werden bedingte Verzweigungen festgelegt. Dabei werden Werte mit Mustern verglichen, die in OF-Zweigen definiert sind. Ergibt der Vergleich den Wahrheitswert TRUE, verzweigt DRIVE/WINDOWS zu den nachfolgenden Anweisungen im OF-Zweig.

Anweisungen mit CASE dürfen beliebig oft geschachtelt werden, d.h. in den OF-Zweigen darf wieder CASE ... END CASE vorkommen. CASE, IF und CYCLE dürfen geschachtelt werden, sich aber nicht überlappen.

In einem Programm muß nach der Anweisung CASE [ALL] [ausdruck1] ein Semikolon stehen.

```
CASE [ ALL ] [ ausdruck1 ]
     { OF { ausdruck2, ... | bedingung | REST } [ programming ... ] } ...
```

ALL — Alle OF-Zweige werden ausgeführt, für die der entsprechende Vergleich den Wert TRUE liefert.

ausdruck1 — *ausdruck1* wird mit *ausdruck2* zu einer Bedingung *ausdruck1=ausdruck2* ergänzt und ausgewertet. Ist der Wahrheitswert TRUE wird der OF-Zweig durchlaufen.

Wenn *ausdruck1* fehlt, muß die OF-Anweisung eine *bedingung* enthalten.

Der Wert von *ausdruck1* wird berechnet. Er bleibt bis zur Anweisung END CASE konstant.

Als Vergleichsoperator dient das Gleichheitszeichen.

OF

Es wird entweder eine *bedingung* ausgewertet oder *ausdruck2* mit *ausdruck1* zu einer *bedingung* mit Gleichheitszeichen ergänzt und ausgewertet.

Wenn Sie ALL nicht angegeben haben und der Vergleich den Wert TRUE ergibt, wird der OF-Zweig ausgeführt und dann nach END CASE verzweigt.

Wenn Sie ALL angegeben haben und der Vergleich den Wert TRUE ergibt, wird der OF-Zweig ausgeführt und dann zum nächsten OF-Zweig (falls vorhanden) verzweigt.

Wenn kein Vergleich den Wert TRUE ergibt, führt DRIVE/WINDOWS die Anweisungen nach REST aus oder beendet, falls REST nicht angegeben wurde, die Verzweigung mit END CASE.

ausdruck2

Wurden mit OF mehrere Ausdrücke angegeben, dann werden die Ergebnisse der Einzelvergleiche zwischen *ausdruck1* und *ausdruck2* mit einem logischen OR verknüpft und dann erst der Wahrheitswert (TRUE oder FALSE) überprüft.

bedingung

Ergibt der Vergleich von *bedingung* den Wert TRUE, dann wird der OF-Zweig ausgeführt.

bedingung darf kein *satzelement* enthalten (siehe Metavariable *satzelement* in den SQL-Lexika [4-6]).

REST

Mit OF REST werden die Resultate vorheriger OF-Zweige ausgewertet. OF REST wird nur erreicht, wenn alle vorherigen Bedingungen der OF-Anweisungen das Ergebnis FALSE hatten. Die Anweisung darf nur einmal vorkommen und muß dann der letzte OF-Zweig sein.

programming

siehe Metavariable *programming*

Beispiel

Die Variable &kreuz ist als Vektor mit dem Wiederholungsfaktor 3 definiert. In einem CASE-Block wird geprüft, ob &kreuz(1), &kreuz(2) oder &kreuz(3) einen Wert ungleich ' ' hat (d.h. ob eine Eingabe gemacht wurde).
Abhängig davon, für welche Komponente eine Eingabe gemacht wurde, wird das Unterprogramm "ende", "anzeigen" oder "mahnung" aufgerufen.
Wenn keine Eingabe gemacht wurde, wird die Meldung " Ihre Eingabe bitte (DUE) " ausgegeben .

```
...
DECLARE VARIABLE &kreuz(3) CHAR (1);
...
CASE;
    OF &kreuz(1) <> ' ' CALL ende;
    OF &kreuz(2) <> ' ' CALL anzeigen  USING &database, &projekt, &verzug;
    OF &kreuz(3) <> ' ' CALL mahnung   USING &database, &projekt;
    OF REST SEND MESSAGE 'Ihre Eingabe bitte  (DUE)' WAIT;
END CASE;
...
```

Definition von Fehlerausgängen

Tritt bei Vergleichen oder Berechnungen ein Fehler auf, wird nach END CASE verzweigt und gegebenfalls gemäß WHENEVER ausgewertet. Ablauffehler von DRIVE-Anweisungen innerhalb der OF-Zweige werden so behandelt, wie bei den einzelnen Anweisungen beschrieben.

CLEAR
Variable oder DRIVE-Format zurücksetzen

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

CLEAR setzt Variablen auf den Anfangswert zurück oder löscht noch nicht ausgegebene DRIVE-Formate (DECLARE FORM) bis auf Seitenkopf und -fuß.

Insbesondere können Sie mit CLEAR den Inhalt von Ein- und Ausgabefeldern zurücksetzen. Es ist möglich, sowohl einzelne Ein- oder Ausgabefelder als auch Gruppen von Ein- oder Ausgabefeldern zurückzusetzen. Dasselbe gilt für eine SREEN-Variable (CLEAR *screenvariable*).

Variablen werden auf den bei DECLARE VARIABLE vereinbarten INIT-Wert gesetzt.

Haben Sie bei der Deklaration der Variablen keinen INIT-Wert vereinbart, wird die Variable bei Ausführung der Anweisung CLEAR auf den datentypspezifischen Anfangswert gesetzt (siehe DRIVE-Programmiersprache [2]). Variablen mit einem Zeit-Datentyp, die TEMPORARY deklariert sind, erhalten in diesem Fall das aktuelle Datum (CURRENT DATE), die aktuelle Uhrzeit (CURRENT TIME) oder den aktuellen Zeitstempel (CURRENT TIMESTAMP).

Bei DRIVE-Formaten gilt: Alle FILL-Anweisungen für das angegebene Format, die noch nicht mit DISPLAY ausgegeben worden sind, werden zurückgesetzt. Das bedeutet, daß die Anweisungsfolge

```
DECLARE FORM name ...
...
FILL name ...
FILL name ...
CLEAR name
DISPLAY name ...
```

zur Ausgabe eines Formats führt, das nur den bei DECLARE FORM vereinbarten Seitenkopf und -fuß (TTITLE und BTITLE) enthält, ansonsten aber leer ist.

CLEAR hat keine Wirkung auf ein implizites DISPLAY infolge eines Bildschirmüberlaufs bei einer FILL-Anweisung.

```
CLEAR { variable | formatname }, ...
```

---

variable	Name der Variablen, die zurückgesetzt werden soll. Der Name muß eine in der aktuellen Programmeinheit gültige Variable bezeichnen (siehe DRIVE-Programmiersprache [2]).
formatname	Name des mit DECLARE FORM vereinbarten DRIVE-Formats, das zurückgesetzt werden soll.

CLOSE FILE
Datei schließen

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

CLOSE FILE schließt eine geöffnete Datei.

```
CLOSE FILE datei
```

---

datei	Logischer Name einer Datei (max. 31 Zeichen). Mit diesem Namen muß die Datei im Programm mit der Anweisung DECLARE FILE definiert sein.

Besonderheiten im UTM-Betrieb

– ISAM-Dateien mit dem Merkmal SHARED-UPDATE=YES werden erst beim Beenden der UTM-Anwendung geschlossen.
– Mit der Eröffnungsart "INPUT" geöffnete Dateien werden erst beim Beenden der UTM-Anwendung geschlossen.
– Dateien werden bei einer Bildschirmausgabe (z.B. DISPLAY ..., SEND MESSAGE, = Dialogschrittende) geschlossen und bei Bedarf im nächsten Dialogschritt wieder geöffnet. DRIVE/WINDOWS verwaltet in diesem Fall die aktuelle Dateiposition.

COMPILE
Programm übersetzen

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Dialog-Modus

COMPILE überprüft eine Source auf Syntax- und Semantikfehler. Mit der COMPILE-Anweisung können Optionen angegeben werden, die den Übersetzungslauf steuern (z.B. ob der bei einer fehlerfreien Übersetzung erzeugte Zwischencode in der DRIVE-Bibliothek abgespeichert wird). Die bei COMPILE angegebenen Optionen überschreiben diejenigen in der Source.

Beim Übersetzen der Source wird eine Übersetzungsliste erzeugt. Die Ausgabe dieser Liste wird durch die Übersetzungsoption OPTION LISTING=LIST/LIBRARY/BOTH gesteuert. Die Liste besteht aus

– einem Kopfteil (Elementname, Bibliotheksname, Datum und Uhrzeit der Überprüfung)
– der Sourceliste. Sie enthält alle Anweisungen der Source und die Auflösung aller vorhandenen COPY-Elemente, USE-Elemente und EUA-Formate.
– einer Übersicht über die Übersetzungsoptionen, die Anzahl der Fehler und die Größe der Objekte.

Treten bei der Übersetzung Fehler auf, enthält die Sourceliste zusätzlich noch Fehlermeldungen.

Ein Programm darf nur übersetzt werden, wenn keine Transaktion offen ist.

Stellt COMPILE bei einer Source, die sich in der EDT-Arbeitsdatei 0 befindet, Fehler fest, werden mit der folgenden EDT-Anweisung die Fehlermeldungen in die in der EDT-Arbeitsdatei 0 stehende Source eingefügt (siehe Anweisung EDT). Die Übersetzungsliste steht in der EDT-Arbeitsdatei 9.

Wird COMPILE ohne Operanden angegeben, wird die in der EDT-Arbeitsdatei 0 stehende Source überprüft. In diesem Fall werden in einer Source die OPTION-Angaben LISTING=LIBRARY und CODE=ON nicht ausgeführt und kommentarlos übergangen.

```
COMPILE [ bibliothek1(elemname1) | elemname1 ]

        [ INTO { bibliothek2(*) | bibliothek2(elemname2) | elemname2 } ]

        [ OPTION ]
```

bibliothek1

Name der DRIVE-Bibliothek (max. 54 Zeichen), aus der die zu übersetzende Source eingelesen wird.

bibliothek1 kann auch der Dateikettungsname der DRIVE-Bibliothek (entsprechend den BS2000-Konventionen) sein.

DRIVE/WINDOWS interpretiert *bibliothek1* zuerst als Dateikettungsnamen, dann als Bibliotheksnamen.

Wird die DRIVE-Bibliothek mit der Anweisung PARAMETER DYNAMIC LIBRARY voreingestellt, so kann die Angabe von *bibliothek1* entfallen.

elemname1

Name eines Elements vom Typ S (max. 31 Zeichen), das eingelesen, analysiert und übersetzt wird.

Wird keine *bibliothek1* angegeben, wird die mit PARAMETER DYNAMIC LIBRARY voreingestellte Bibliothek eingesetzt.

Ist *elemname1* in der angegebenen DRIVE-Bibliothek nicht vorhanden, wird eine Fehlermeldung ausgegeben.

Wenn *elemname1* nicht angegeben ist, wird die in der EDT-Arbeitsdatei 0 stehende Source analysiert und übersetzt.

INTO

Mit INTO werden die Elemente und DRIVE-Bibliotheken festgelegt, in die der Zwischencode, der Objektcode oder die Übersetzungsliste geschrieben wird, falls die Übersetzungsoption OPTION CODE=ON, OBJECT=ON oder LISTING=LIBRARY angegeben wurde.

Bei der Option CODE=ON wird ein Element vom Typ X erzeugt, das den erzeugten Zwischencode enthält.

Bei der Option OBJECT=ON wird ein Element vom Typ R erzeugt, das den erzeugten Objektcode enthält.

Bei der Option LISTING=LIBRARY oder =BOTH wird ein Element vom Typ P erzeugt, das die Übersetzungsliste enthält.

	Wird INTO nicht angegeben, gilt: Elemente vom Typ X oder P werden in der mit *bibliothek1* bezeichneten Bibliothek oder in der aktuellen DRIVE-Bibliothek überschrieben. Elemente vom Typ R werden in der Bibliothek *bibliothek1* unter einem Namen abgelegt, der aus den ersten 4 und den letzten 3 Zeichen von *elemname1* (4-3-Regel) gebildet wird. Dabei werden ungültige Zeichen durch das Sonderzeichen # ersetzt.
bibliothek2	Name der DRIVE-Bibliothek (max. 54 Zeichen), in die der erzeugte Zwischencode, Objektcode oder die Übersetzungsliste geschrieben wird. *bibliothek2* kann auch der Dateikettungsname der DRIVE-Bibliothek (entsprechend den BS2000-Konventionen) sein. Wird keine Angabe für *bibliothek2* gemacht, wird *bibliothek1* eingesetzt.
*	Das Element, das den Zwischencode, den Objektcode oder die Übersetzungsliste enthält, erhält den Namen *elemname1*.
elemname2	bezeichnet das Element, das den Zwischencode, den Objektcode oder die Übersetzungsliste enthält. Der Name für Elemente vom Typ X oder P darf max. 31 Zeichen lang sein, für Elemente vom Typ R max. 7 Zeichen. Ist der Name für ein R-Element länger als 7 Zeichen, wird er nach der 4-3-Regel verkürzt. Ungültige Zeichen werden durch # ersetzt. Fehlt eine Angabe für *bibliothek2*, wird *elemname2* in der Bibliothek *bibliothek1*gesichert. Wird weder *bibliothek1* noch *bibliothek2* angegeben, wird die bei PARAMETER DYNAMIC LIBRARY angegebene Bibliothekeingesetzt.
OPTION	OPTION-Anweisung (siehe Anweisung OPTION). Sie geben die gewünschten Optionen an.

Beispiele

Die Source mit dem Namen "prog1" wird aus der aktuellen DRIVE-Bibliothek eingelesen und übersetzt. Der Zwischencode wird unter dem gleichen Namen in der aktuellen DRIVE-Bibliothek gespeichert.

```
COMPILE prog1 OPTION CODE=ON
```

Die Source mit dem Namen "prog1" wird aus der aktuellen DRIVE-Bibliothek eingelesen und übersetzt. Der Zwischencode wird unter dem Namen "prog2" in der aktuellen DRIVE-Bibliothek gespeichert.

```
COMPILE prog1 INTO prog2 OPTION CODE=ON
```

Die Source mit dem Namen "prog1" wird aus der DRIVE-Bibliothek "bib1" eingelesen und übersetzt. Der Zwischencode wird mit dem Namen "prog2" in der DRIVE-Bibliothek "bib2" gespeichert.

Die Angabe "bib1" kann nicht entfallen, weil die DRIVE-Bibliothek mit der Anweisung PARAMETER DYNAMIC LIBRARY nicht festgelegt wurde.

```
COMPILE bib1(prog1) INTO bib2(prog2) OPTION CODE=ON
```

Die Source mit dem Namen "prog1" wird aus der aktuellen DRIVE-Bibliothek eingelesen und übersetzt. Die Übersetzungsliste wird mit dem Namen "list" in der DRIVE-Bibliothek "bib2" gespeichert.

```
COMPILE prog1 INTO bib2(list) OPTION LISTING=LIBRARY
```

Die Source mit dem Namen "prog1" wird aus der aktuellen DRIVE-Bibliothek eingelesen und übersetzt. Die Übersetzungsliste wird mit dem Namen "list" in der DRIVE-Bibliothek "bib2" gespeichert. Zugleich wird die Übersetzungsliste im TIAM-Betrieb nach SYSLIST ausgegeben oder im UTM-Betrieb in die zentrale Druckdatei ausgegeben.

```
COMPILE prog1 INTO bib2(list) OPTION LISTING=BOTH
```

CONTINUE
Schleifendurchlauf oder Debug-Lauf fortsetzen

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm- und Debug-Modus mit unterschiedlicher Funktion (s.u.)

Abhängig vom Modus hat CONTINUE unterschiedliche Funktion:

– Im Programm-Modus kann mit CONTINUE CYCLE in einer durch CYCLE ... END CYCLE geklammerten Folge von Anweisungen vorzeitig zum END CYCLE gesprungen werden.
– Im Debug-Modus wird nach CONTINUE der Programmablauf (im Debug-Modus) fortgesetzt.

Die Anweisung CONTINUE ohne Angabe des Operanden CYCLE ist nur im Debug-Modus erlaubt.

```
CONTINUE [ CYCLE ]
```

CYCLE

CONTINUE CYCLE ist nur im Programm-Modus erlaubt und muß innerhalb einer CYCLE-Klammer (CYCLE bis END CYCLE) stehen.

In Abhängigkeit von der Schleifenbedingung wird die Schleife erneut durchlaufen oder abgebrochen.

COPY
COPY-Element einfügen

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Dialog- und Programm-Modus mit unterschiedlicher Funktion (s.u.)

COPY fügt in ein Programm COPY-Elemente ein. Sie bestehen aus einer Folge von Anweisungen oder Teilen von Anweisungen und müssen in einer DRIVE-Bibliothek gespeichert sein (siehe DRIVE-Programmiersystem [1]). Teilanweisungen können als COPY-Elemente zu einer kompletten Anweisung zusammengefügt werden. COPY-Elemente können sowohl im Deklarationsteil als auch im Verarbeitungsteil stehen.

COPY-Elemente dürfen nicht geschachtelt werden, d.h. innerhalb von COPY-Elementen sind COPY-Anweisungen nicht erlaubt.

Im Programm-Modus wird das COPY-Element wird bei der Übersetzung in die Source kopiert. Es gelten die dort üblichen Einschränkungen (z.B. Programmstruktur).

Im Dialog-Modus wird nur die erste Anweisung aus dem angegebenen COPY-Element an der Datensichtstation ausgegeben. Die Anweisung kann geändert und dann abgeschickt werden.

```
COPY { bibliothek(elemname) | elemname }
```

---

bibliothek

Name der DRIVE-Bibliothek (max. 54 Zeichen), aus der das COPY-Element kopiert wird.

bibliothek kann auch der Dateikettungsname der DRIVE-Bibliothek (entsprechend den BS2000-Konventionen) sein.

DRIVE/WINDOWS interpretiert *bibliothek* zuerst als Dateikettungsnamen, dann als Bibliotheksnamen.

Wird die DRIVE-Bibliothek mit der Anweisung PARAMETER DYNAMIC LIBRARY voreingestellt, so kann die Angabe von *bibliothek* entfallen.

elemname

Name des COPY-Elements (max. 31 Zeichen). Das COPY-Element vom Typ S wird aus der angegebenen DRIVE-Bibliothek kopiert.

elemname muß angegeben werden, sonst wird eine Fehlermeldung ausgegeben.

Wird keine Angabe für *bibliothek* gemacht, wird die bei PARAMETER DYNAMIC LIBRARY angegebene Bibliothek eingesetzt.

Ist das COPY-Element in der DRIVE-Bibliothek nicht vorhanden, wird eine Fehlermeldung ausgegeben.

Beispiel 1

Das COPY-Element "MITVAR" aus der Bibliothek "DRI.LIB" wird in ein Programm eingefügt.

```
COPY "DRI.LIB"(MITVAR)
```

Beispiel 2

Ein *select-ausdruck* vervollständigt als COPY-Element "select1" eine Cursor-Deklaration.

```
DECLARE c1 CURSOR FOR COPY select1;;
```

– das erste Semikolon beendet die COPY-Anweisung
– das zweite Semikolon beendet die DECLARE-Anweisung

Hat das COPY-Element "select1" folgenden Inhalt:

```
SELECT * FROM tab1 WHERE ...
```

dann lautet das Ergebnis:

```
DECLARE c1 CURSOR FOR SELECT * FROM tab1 WHERE ...;
```

CYCLE
Schleife programmieren

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

CYCLE kennzeichnet den Beginn eines CYCLE-Blocks, dessen Ende mit END CYCLE festgelegt wird. Innerhalb eines CYCLE-Blocks wird eine Schleife beliebig oft oder bis ein Endekriterium erreicht ist durchlaufen.

Eine Schleife mit Zugriff auf einen Cursor (CYCLE cursorname) wird so lange durchlaufen, bis das Ende der Cursortabelle erreicht ist. Mit dieser Anweisung lassen sich sehr einfach alle Sätze einer Cursortabelle verarbeiten.

Wird CYCLE ohne Operand angegeben, wird mit CYCLE eine Endlosschleife definiert, die nur durch BREAK CYCLE oder einen Programmabbruch abgebrochen werden kann.

Schleifen dürfen geschachtelt werden. Die Schachtelungstiefe ist beliebig und nur abhängig vom Speicherplatz, den DRIVE/WINDOWS zur Bearbeitung benötigt.

Schleifen, Bedingungen (IF) und Verzweigungen (CASE) dürfen ebenfalls geschachtelt werden, sich aber nicht überlappen.

In einem Programm muß nach der Anweisung CYCLE ein Semikolon stehen.

```
CYCLE [ cursorname INTO variable, ... |

        WHILE bedingung |

        FOR variable1=numausdruck1 [ BY numausdruck2 ] TO numausdruck3 ]
```

---

cursorname — Name eines Cursors. Der Cursor muß bereits deklariert und geschlossen sein.
CYCLE *cursorname* ist eine reine Ausgabefunktion. Wenn kein Satz gefunden wird, ist eine sinnvolle Steuerung der Schleife nicht mehr möglich. Die Schleife wird beendet und der Cursor geschlossen.

INTO	Mit INTO wird zu Beginn jedes Schleifendurchlaufs ein Satz aus der Cursortabelle *cursorname* in *variable* übertragen. Solange Einträge in der Cursortabelle vorhanden sind, wird die Schleife durchlaufen. Der Cursor wird beim ersten Schleifendurchlauf geöffnet. Anschließend wird er auf den nächsten Satz in der Cursortabelle positioniert und die Variable mit Werten dieser Satzelemente versorgt. Die Schleife wird beendet, falls der letzte Satz von *cursorname* erreicht ist. Der Cursor wird bei "TABLE END" oder nach BREAK CYCLE geschlossen. Die CYCLE ... INTO Anweisung führt implizit alle SQL-Aufrufe durch, die in einer 3GL-Umgebung explizit formuliert werden müssen (OPEN, FETCH und CLOSE). Wurde bei WHENEVER ein Fehlerausgang ≠ CONTINUE für "TABLE END" festgelegt, wird dieser nicht ausgeführt, wenn bei CYCLE *cursorname* INTO *variable*, ... das Ereignis "TABLE END" eintritt.
variable	Name einer Variablen.
WHILE	Ist WHILE angegeben, so wird die Schleife immer durchlaufen, wenn *bedingung* einen wahren Wert liefert.
bedingung	Bedingung(en) an Datenwerte. Zu Beginn jedes Schleifendurchlaufs wird überprüft, ob *bedingung* noch erfüllt ist. Solange *bedingung* den Wert TRUE liefert, wird die Schleife durchlaufen. *bedingung* darf keinen *ausdruck* oder * enthalten.
FOR	Zu Beginn der Schleifenverarbeitung wird der Variablen *variable1* der Wert von *numausdruck1* zugewiesen. Vor dem ersten Durchlauf wird geprüft, ob der neue Wert von *variable1* kleiner oder gleich *numausdruck3* ist (hat *numausdruck2* einen negativen Wert, wird auf größer oder gleich geprüft). Wenn das der Fall ist, dann wird die Anweisungsfolge zwischen CYCLE FOR und END CYCLE abgearbeitet. Andernfalls wird die Schleife über END CYCLE beendet. Vor jedem weiteren Durchlauf wird der Wert von *numausdruck2* zur *variable1* addiert und der Vergleich mit *numausdruck3* erneut durchgeführt. Treten bei der Berechnung von Werten oder beim Vergleich von Werten Ablauffehler auf, wird die Schleife beendet und zu END CYCLE verzweigt, wo der Fehler gegebenenfalls mit WHENEVER behandelt werden kann.

numausdruck1 und das Ergebnis der Inkrementierung vor jedem Schleifendurchlauf müssen zuweisungsverträglich zu *variable1* sein.

variable1

variable1 übernimmt eine Kontrollfunktion in der FOR-Schleife. *variable1* darf nicht strukturiert, indiziert, redefiniert oder redefinierend sein. Als Datentypen sind nur NUMERIC, DECIMAL, INTEGER, SMALLINT, EXTENDED DECIMAL oder XDEC erlaubt.

Innerhalb der Schleife darf der Variablen *variable1* kein Wert über eine explizite DRIVE-Anweisung (z.B. SET oder RETURN bei einer Ein-/Ausgabe) zugewiesen werden. *variable1* kann nicht erneut innerhalb der aktuellen Schleife als Laufvariable verwendet werden. Wenn die FOR-Schleife entweder durch Erreichen von *numausdruck3* oder BREAK CYCLE beendet wird, hat *variable1* den letzten aktuellen Datenwert. Dieser Wert kann ungleich *numausdruck3* sein.

numausdruck1

numausdruck1 legt den Startwert der FOR-Schleife fest. Ist der Wert von *numausdruck1* zu Beginn der FOR-Schleife der NULL-Wert, dann wird die Schleife über END CYCLE beendet.

BY numausdruck2

numausdruck2 legt die Schrittweite in der FOR-Schleife fest. *numausdruck2* darf nicht den Wert 0 haben.
Hat *numausdruck2* den NULL-Wert, wird die FOR-Schleife mit einer Fehlermeldung abgebrochen.
Zu Beginn des Schleifendurchlaufs bestimmt das aktuelle Vorzeichen von *numausdruck2* die Laufrichtung der FOR-Schleife für den gesamten Schleifendurchlauf (siehe oben, FOR). Der Wert von *numausdruck2* ist in dieser Zeit konstant.

Fehlt die Angabe BY *numausdruck2*, wird *numausdruck2* = 1 gesetzt, d.h. es wird bei jedem Schleifendurchlauf der Zähler um eins erhöht.

TO numausdruck3

numausdruck3 legt den Endewert in der FOR-Schleife fest. Ist der Wert von *numausdruck3* zu Beginn der FOR-Schleife der NULL-Wert, dann wird die Schleife über END CYCLE beendet.
Der Wert von *numausdruck3* bleibt für den gesamten Schleifendurchlauf konstant.

Innerhalb einer Schleife mit Cursorverarbeitung ist COMMIT WORK nur erlaubt, wenn der Cursor mit STORE gespeichert und mit RESTORE wiederhergestellt wurde.

Definition von Fehlerausgängen

Die Definition eines Fehlerausgangs über WHENEVER ist die einzige Möglichkeit, einen Programmabbruch wegen Semantikfehler bei der Auswertung von Schleifen zu vermeiden. Die Definition muß im Deklarationsteil stehen. Das Programm wird entsprechend des bei WHENEVER definierten Fehlerausgangs nach dem zugehörenden END CYCLE fortgesetzt (siehe Anweisung WHENEVER).

Beispiel 1

Alle Sätze der Cursortabelle "cr" werden solange in die Variable &var übertragen, bis das Ende der Cursortabelle erreicht ist (&DML_STATE='TABLE END'):

```
CYCLE cr INTO &var.*;
/* Verarbeitung */
...
END CYCLE;    /* CYCLE Cursor-Ende */
```

Die Systemvariable &ERROR_STATE='OK' wird gesetzt.

Die angeführten Anweisungen enthalten implizit folgende Anweisungsfolge:

```
OPEN cr;
CYCLE;
FETCH cr INTO &var.*;
IF &DML_STATE='TABLE END'
   THEN
   BREAK CYCLE;
END IF;
/* Verarbeitung */
...
END CYCLE;
CLOSE cr;
```

DRIVE/WINDOWS schließt den Cursor (CLOSE cr) und der Status dieses CLOSE wird in die Systemvariable &ERROR_STATE eingetragen.

Anmerkungen zum Beispiel:

- Es darf kein Transaktionsende in der Schleife sein.
- Wird ein Cursor ohne Treffer eröffnet, dann wird der Verarbeitungsteil nicht durchlaufen.

Beispiel 2

Bis das Ende der Cursortabelle "druckcursor" erreicht ist werden alle Sätze in die Variable &drucksatz.* übertragen und die Ausgabefelder des Listenformats "mitarbeiter" gefüllt.

```
CYCLE druckcursor INTO &drucksatz.*;
   FILL mitarbeiter TABLE NAMES &drucksatz.*;
END CYCLE;
```

Beispiel 3

In einer Schleife wird allen Feldern des Vektors &sprachen(5) der NULL-Wert zugewiesen.

```
SET &index = 1;
CYCLE WHILE &index <= 5;
   SET &sprachen(&index) = NULL;
   SET &index = &index + 1;
END CYCLE;
```

Beispiel 4

Beispiel 4 verhält sich wie Beispiel 3.

```
CYCLE FOR &index=1 TO 5;
   SET &sprachen(&index) = NULL;
END CYCLE;
```

Beispiel 5

In einer Schleife wird den letzten vier Feldern des Vektors &monat(12) der NULL-Wert zugewiesen.

```
CYCLE FOR &index=12 BY -1 TO 9;
   SET &monat(&index) = NULL;
END CYCLE;
```

Beispiel 6

Die folgende Schleife wird nicht durchlaufen, weil mit 12 > 10 bereits vor dem ersten Schleifendurchlauf die Endebedingung *numausdruck3* > *&index* erfüllt ist.

```
CYCLE FOR &index=10 BY -1 TO 12;
   SET &monat(&index) = NULL;
END CYCLE;
```

Beispiel 7

Das Bildschirmformat "format1" wird solange ausgegeben und die Variable &artikel.* wird solange als Satz in die Tabelle "v_artikel" eingefügt, bis die Systemvariable &KFKEY den Wert 'K3' hat (d.h. bis die K3-Taste gedrückt wird).

```
PARAMETER KFKEY = 'K3';
...
CYCLE;
   DISPLAY format1;
   IF &KFKEY = 'K3'
      THEN BREAK CYCLE;
      ELSE INSERT INTO v_artikel VALUES (&artikel.*);
      ...
   END IF;
END CYCLE;
```

DEBUG
Programm starten und in den Debug-Modus wechseln

Diese Anweisung ist gültig

– im TIAM-Betrieb
– im Dialog-Modus

DEBUG schaltet in den Debug-Modus um, startet ein DRIVE-Programm (mit Unterprogramm) oder ein externes DRIVE-Unterprogramm unter der Kontrolle des Debuggers und führt es unter der Kontrolle des Debuggers aus.

Voraussetzung dafür ist, daß das Programm übersetzt und eine Übersetzungsliste erzeugt wurde.
Falls zu einer Source kein Zwischencode vorliegt, bewirkt DEBUG eine implizite Übersetzung. Eine Übersetzungsliste wird allerdings nicht implizit erzeugt, sondern muß zum Beispiel mit der Anweisung OPTION LISTING = LIBRARY erzeugt werden.

Der Ablauf eines Programms wird im Debug-Modus nicht verändert. Es ist jedoch möglich, den Programmablauf zu unterbrechen und bestimmte Aktionen (s.u.) durchführen zu lassen. Damit können im Debug-Modus Programmfehler erkannt und lokalisiert und Fehlerursachen herausgefunden werden.

Nachdem der Debug-Modus gestartet ist, wird automatisch der Anfangshaltepunkt gesetzt. Dieser liegt hinter der PROCEDURE-Anweisung des mit DEBUG gestarteten Programms.

Am Anfangshaltepunkt können Sie Debug-Anweisungen eingeben, sobald die Eingabeaufforderung (*) erscheint.

Folgende Debug-Anweisungen können an Haltepunkten eingegeben werden:

– AT
– BREAK
– BREAK DEBUG
– CONTINUE
– DISPLAY FORM
– DISPLAY LIST
– REMOVE
– SET
– TRACE

Falls das gerufene Programm Übergabeparameter (USING ...) erwartet, kann in der Anweisung DEBUG die USING-Klausel entfallen. DRIVE/WINDOWS erfragt in diesem Fall die notwendigen Parameter über ein Parameter-Prompting. Mit der Anweisung SET können Sie die Parameter versorgen. DRIVE/WINDOWS hält das gerufene Unterprogramm hinter der PROCEDURE-Anweisung an. Sie können Debug-Anweisungen eingeben.

Für den Mischbetrieb gilt bezüglich des Debug-Modus folgendes:
Wird im New-Style ein Old-Style-Programm mit CALL aufgerufen, wird das Old-Style-Programm ausgeführt. Anschließend wird in den New-Style in den Debug-Modus gewechselt.
Wird im New-Style ein Old-Style-Programm mit DO aufgerufen, so wird der Debug-Modus im New-Style beendet und das Old-Style-Programm ausgeführt. Anschließend wird wieder in den New-Style gewechselt.
Wird im Old-Style ein New-Style-Programm mit DO aufgerufen, wird in den New-Style gewechselt, das Programm wird aber nicht im Debug-Modus ausgeführt.

Mit der SET-Anweisung können im Debug-Modus einer Variablen keine Feldattribute und einer SCREEN-Variablen keine Globalattribute zugewiesen werden.

Mit der Anweisung BREAK DEBUG wird der Debug-Modus verlassen und in den Dialog-Modus geschaltet.

```
DEBUG [ bibliothek(elemname) | elemname ]

      [ USING { ausdruck | NULL }, ... ]
```

---

bibliothek
: Name der DRIVE-Bibliothek (max. 54 Zeichen), aus der das Programm analysiert und unter Kontrolle des Debuggers ausgeführt wird.

  bibliothek kann auch der Dateikettungsname der DRIVE-Bibliothek (entsprechend den BS2000-Konventionen) sein.

  DRIVE/WINDOWS interpretiert *bibliothek* zuerst als Dateikettungsnamen, dann als Bibliotheksnamen.

  Wird die DRIVE-Bibliothek mit der Anweisung PARAMETER DYNAMIC LIBRARY voreingestellt, so kann die Angabe von *bibliothek* entfallen.

elemname
: Name des Elements (max. 31 Zeichen), das das Programm enthält.

  DRIVE/WINDOWS sucht nach dem zuletzt bearbeiteten Element, unabhängig davon, ob dieses eine Source enthält (S-Element) oder einen Zwischencode (X-Element).

  Wird keine Angabe für *bibliothek* gemacht, wird die bei PARAMETER DYNAMIC LIBRARY angegebene Bibliothek eingesetzt.

  Wenn *elemname* nicht angegeben wird, verwendet DRIVE/WINDOWS die in der EDT-Arbeitsdatei 0 stehende Source.

USING	Mit USING werden Parameter an das zu startende Programm übergeben. Das zu startende Programm muß mit PROCEDURE ... USING definiert worden sein. Die Parameter dürfen zum Übersetzungszeitpunkt keine Variablen enthalten. Sind die Parameter nicht zuweisungsverträglich oder wird bei der Anweisung DEBUG die USING-Angabe weggelassen, so gibt DRIVE/WINDOWS folgende Meldung aus: `DRI0561 BITTE USING-PARAMETER VERSORGEN`. Der Anwender kann nun mit der SET-Anweisung alle Parameter versorgen oder mit der BREAK DEBUG-Anweisung den Debug-Modus verlassen. Im Debug-Modus wird dieses Parameter-Prompting auch für DRIVE-Unterprogramme durchgeführt, die mit DO oder CALL aufgerufen werden.
ausdruck	bezeichnet die Parameter, die an das zu startende Programm übergeben werden (Sendefelder). Als Übergabeparameter erlaubt sind Literale, Aggregate, deren Komponenten Literale sind, und Rechenausdrücke, die keine Variablen enthalten. Es können auch Werte von Variablen übergeben werden. *ausdruck* muß zum Übersetzungszeitpunkt berechenbar sein.
NULL	Übergabe des NULL-Werts an das zu startende Programm

Verhalten im Fehlerfall

Tritt im Debug-Modus bei einer Anweisung ein Fehler auf, so wird zunächst überprüft, ob für diesen Fehler eine WHENEVER-Behandlung vereinbart ist.

Wurde keine WHENEVER-Behandlung vereinbart, wird das Programm vor der fehlerhaften Anweisung angehalten.

Wurde eine WHENEVER-Behandlung vereinbart, geschieht abhängig von der WHENEVER-Aktion folgendes: Wenn die vereinbarte WHENEVER-Aktion CALL oder CONTINUE ist, wird das Programm weiter ausgeführt. Wenn die WHENEVER-Aktion BREAK ist, wird das Programm vor der fehlerhaften Anweisung angehalten.

Im Debug-Modus wird zum Endhaltepunkt verzweigt, wenn wegen einer fehlenden COMMIT WORK-Anweisung eine Transaktion nicht zurückgesetzt werden kann.

Tritt im Debug-Modus bei einer Debug-Anweisung ein Fehler auf, so gibt DRIVE/WINDOWS zunächst eine Fehlermeldung aus und dann die Meldung: `DRI0553 BITTE DEBUG-ANWEISUNG EINGEBEN`. Das Programm steht an dem Haltepunkt, an dem die fehlerhafte

Debug-Anweisung eingegeben wurde. Debug-Anweisungen, die der fehlerhaften Debug-Anweisung folgen, werden nicht ausgeführt. Eine Korrektur der Fehlerursache wird erst wirksam, wenn der Testpunkt das nächste Mal erreicht wird.

Beziehungen zu anderen Anweisungen

– Wird bei DEBUG weder *bibliothek* noch ein *elemname* angegeben, werden in einer Source die OPTION-Angaben LISTING=LIBRARY und CODE=ON nicht ausgeführt und kommentarlos übergangen.

– Findet bei DEBUG eine Analysephase statt, wird die Übersetzungsoption CODE=ON, die in der Source angegeben ist, ignoriert.

  Andere Optionen werden ausgeführt, z.B. wird bei LISTING=LIBRARY die Übersetzungsliste in ein Bibliothekselement geschrieben.

i Die Ablaufverfolgung kann nicht eingeschaltet werden (TRACE), wenn das Programm in der EDT-Arbeitsdatei 0 steht.

Regeln bei Datenbankzugriff

– Die Anweisung DEBUG wird nur dann ausgeführt, wenn für den Dialog-Modus keine (New-Style-)Transaktion offen ist.

– Wenn dem Dialog-Modus und dem gerufenen Programm unterschiedliche Datenbanksysteme zugeordnet sind (DBSYSTEM ≠ OFF), wird die Anweisung DEBUG abgebrochen. Die unterschiedliche Zuordnung von Datenbanksystemen ist nur möglich, wenn mit DEBUG Zwischencode oder Objektcode aufgerufen wird, der in einer früheren DRIVE-Sitzung mit einem anderen Datenbanksystem erzeugt wurde.

– Wenn dem Dialog-Modus und dem gerufenen Programm jeweils das Datenbanksystem SESAM V2.x zugeordnet ist (DBSYSTEM = SESAMSQL), wird die Anweisung DEBUG nur dann ausgeführt, wenn für den Dialog-Modus keine Transaktion offen ist.

– Wenn dem Dialog-Modus ein Datenbanksystem zugeordnet ist (DBSYSTEM ≠ OFF) und dem gerufenen Programm nicht (DBSYSTEM = OFF), so greift das gerufene Programm auf dasselbe Datenbanksystem zu wie der Dialog-Modus.

– Wenn dem gerufenen Programm ein BS2000-Datenbanksystem zugeordnet ist (DBSYSTEM = UDS / SESAM / SESAMSQL), wird die Anweisung DEBUG nur dann ausgeführt, wenn dieses Datenbanksystem der geladenen Variante entspricht.

– Wenn das gerufene Programm ein Old-Style-Programm ist, wird die Anweisung DEBUG abgebrochen, wenn dem Dialog-Modus das Datenbanksystem UDS zugeordnet ist (DBSYSTEM = UDS).

Beispiel

Das Programm "test" wird im Debug-Modus getestet.

Alle ausführbaren Programmanweisungen werden gezählt. Die Anweisungen in den Zeilen 15, 17 und 20 bis 55 der Übersetzungsliste des Hauptprogramms "test" sind Testpunkte. Nacheinander wird die Variable &var1 auf 1 gesetzt, auf einen Drucker und auf den Bildschirm ausgegeben. Danach wird der Programmablauf fortgesetzt.

An der Anweisung in Zeile 33 der Übersetzungsliste des Unterprogramms "test2", das sich in der voreingestellten Bibliothek befindet, wird die Variable &subvar1 auf 2 gesetzt und auf einen Drucker ausgegeben.

Ab der Anweisung in der Zeile 99 der Übersetzungsliste des Hauptprogramms "test" wird die Ablaufverfolgung des Programms eingeschaltet und der Debug-Lauf fortgesetzt.

```
DEBUG test;          /* DRIVE haelt vor der ersten ausfuehrbaren    */
                     /* Anweisung des Verarbeitungsteils an          */
AT ALL COUNT;
AT 15 SET &var1 = 1
AT 17 DISPLAY LIST &var1
AT 20 - 55 DISPLAY FORM &var1
AT * CONTINUE
AT test2 33 SET &subvar1 = 2
AT test2 33 DISPLAY LIST &subvar1
AT 99 TRACE
CONTINUE             /* Der Debug-Lauf wird jetzt erst fortgesetzt   */
...
```

DECLARE CONSTANT
Konstante definieren

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

DECLARE CONSTANT definiert Konstanten.

Die definierte Konstante kann innerhalb eines DRIVE-Programms syntaktisch für eine Variable stehen, aber nicht für ein Literal.

DECLARE CONSTANT muß im Deklarationsteil des Programms vor allen ausführbaren Anweisungen stehen.

```
DECLARE CONSTANT { varname
                   { literal |
                     MSGSTRING ( numausdruck1 [ [, numausdruck2 ], name ] ) } }, ...
```

---

varname	Name einer Konstanten *varname* muß mit dem "&"-Zeichen beginnen und darf insgesamt max. 32 Zeichen lang sein.
literal	Literal, dessen Wert *varname* zugewiesen bekommt.
MSGSTRING	Der Wert des nachfolgenden Ausdrucks wird zum Übersetzungszeitpunkt erzeugt und im Zwischencode gespeichert. Aus der Meldungsdatei (= aktuelle MIP-Datei) wird die Meldung entnommen, die durch die Parameter festgelegt ist. Wenn drei Parameter (*numausdruck1, numausdruck2, name*) angegeben werden, wird *numausdruck2* ignoriert. Wenn zwei Parameter angegeben werden, muß der zweite Parameter *name* sein (*numausdruck1, name*). Wenn keine eindeutige Meldung gefunden werden kann, gibt DRIVE/WINDOWS zurück: `MELDUNG NICHT GEFUNDEN`.
numausdruck1	Zweiter Teil des Meldungsschlüssels (= Meldungsnummer). *numausdruck1* muß ein numerischer Ausdruck ohne Nachkommastellen sein. Hat *numausdruck1* den Wert NULL, so liefert der gesamte Ausdruck den NULL-Wert.

numausdruck2	*numausdruck2* wird ignoriert, falls ein Wert angegeben wurde. *numausdruck2* wird nur aus Kompatibilitätsgründen unterstützt.
name	erster Teil des Meldungsschlüssels (= Identifikationsschlüssel für die Systemkomponente, die die Meldung erzeugt). *name* darf maximal 3 Zeichen lang sein.

DECLARE FILE
Datei definieren

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

DECLARE FILE definiert Dateien, d.h. Sie machen einem Programm den logischen Namen einer Datei bekannt. Dieser logische Namen wird einer Datei beim Öffnen mit der Anweisung OPEN FILE zugewiesen.

Für die Darstellung von NULL-Werten wird mit dieser Anweisung gleichzeitig ein Ersatzzeichen festgelegt. Wenn für die NULL-Wertdarstellung kein Zeichen definiert ist, gibt DRIVE/WINDOWS beim Versuch, einen NULL-Wert in eine Datei zu schreiben, einen Fehler aus.

DECLARE FILE muß im Deklarationsteil des Programms vor allen ausführbaren Anweisungen stehen.

```
DECLARE FILE { datei [ NULL zeichen ] }, ...
```

---

datei	Logischer Name einer Datei (max. 31 Zeichen).
NULL	Für die NULL-Wertdarstellung wird ein Zeichen festgelegt. Anstelle des NULL-Werts wird dieses Zeichen so oft in die Datei geschrieben oder in der Datei erwartet, wie es der Feldlänge des Datenfeldes entspricht, das den NULL-Wert enthält.
zeichen	Angabe des NULL-Wertzeichens (ein Zeichen). Geben Sie für *zeichen* nicht an: – bei alphanumerischen Datenfeldern: das Leerzeichen (%) – bei numerischen Datenfeldern: die Zeichen: + - , oder . Dies kann zu ungewollten Reaktionen führen. Das Zeichen muß als alphanumerisches Literal (*charliteral*, siehe Metavariable *literal*) oder als sedezimales Zeichen (*sedecliteral*, siehe Metavariable *literal*) angegeben werden.

DECLARE FORM
DRIVE-Bildschirmformat definieren

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

DECLARE FORM definiert ein DRIVE-Bildschirmformat zur Ein- und Ausgabe von Daten.

DECLARE FORM legt einen Speicherbereich für ein Bildschirmformat an und definiert das Layout dieses Bildschirms. Der Speicherbereich wird mit FILL-Anweisungen aufgefüllt. Die Ausgabeformatierung wird durch DISPLAY *formatname* abgeschlossen. Gleichzeitig wird eine Ausgabe angestoßen.

DECLARE FORM muß im Deklarationsteil des Programms vor allen ausführbaren Anweisungen stehen.

Befindet sich im Seitenkopf oder -fuß ein Eingabe- oder Ausgabefeld, das nicht in den dafür vorgesehenen Platz paßt, erfolgt zum Ausführungszeitpunkt Programmabbruch.

```
DECLARE FORM formatname

            [ PERMANENT | TEMPORARY ]
            [ NULL nullwert ]
            [ { COLUMNS n | LINES n }, ... ]

            [ TTITLE [ format ] { [ RETURN ] ausdruck [ INIT ausdruck1 [ NOCHECK ] ]
                                    [ ATTRIBUTE ( attribute, ... ) ] [ mask ] |
                                  NEWLINE n |
                                  TABULATOR n |
                                  BLANK n }, ... ]

            [ BTITLE [ format ] { [ RETURN ] ausdruck [ INIT ausdruck1 [ NOCHECK ] ]
                                    [ ATTRIBUTE ( attribute, ... ) ] [ mask ] |
                                  NEWLINE n |
                                  TABULATOR n |
                                  BLANK n }, ... ]
```

---

formatname	Name des DRIVE-Formats (max. 31 Zeichen).
PERMANENT	Der Inhalt eines mit PERMANENT definierten Formats bleibt über Unterprogrammende hinaus erhalten und ist bei einem erneuten Aufruf desselben Unterprogramms wieder vorhanden.
TEMPORARY	Vorbelegung Der Inhalt eines mit TEMPORARY definierten Formats bleibt bei Unterprogrammende nicht erhalten.
NULL	Für die NULL-Wertdarstellung wird ein Zeichen festgelegt. Eine bei PARAMETER DYNAMIC vereinbarte NULL-Wertdarstellung wird hiermit überschrieben.
nullwert	Angabe des NULL-Wertzeichens (max. 1 Zeichen). Die NULL-Wertdarstellung wird für den alphanumerischen Datentyp (CHARACTER, VARCHAR) oder für die numerischen Datentypen (NUMERIC, DECIMAL, INTEGER, SMALLINT, REAL und FLOAT) festgelegt (siehe Metavariable *nullwert*). Die alphanumerische NULL-Wertdarstellung ist auch gültig für die Zeit-Datentypen. Die numerische NULL-Wertdarstellung ist auch gültig für den Datentyp INTERVAL.
COLUMNS n	Festlegen der Anzahl der Spalten pro Bildschirm. Die Angabe COLUMNS darf innerhalb von DECLARE FORM nur einmal angegeben werden. Für n gilt: $0 < n \leq$ Anzahl der Bildschirmspalten Vorbelegung: Spaltenzahl des jeweiligen Bildschirms
LINES n	Festlegen der Anzahl der Zeilen pro Bildschirm. Die Angabe LINES darf innerhalb von DECLARE FORM nur einmal angegeben werden. Für n gilt: $0 < n \leq$ Anzahl der Bildschirmzeilen - 1 Vorbelegung: Zeilenzahl des jeweiligen Bildschirms - 1 Die letzte Zeile ist als Meldungszeile reserviert. Für LINES gilt: Zeilenzahl (TTITLE) + Zeilenzahl (BTITLE) < LINES.
TTITLE	Definieren eines Seitenkopfes, der nach jedem Bildschirmwechsel ausgegeben wird.

	Die Gesamtanzahl der Zeilen für TTITLE, BTITLE und FILL darf maximal sein: Anzahl der Bildschirmzeilen - 1. Die letzte Zeile ist als Meldungszeile reserviert. Es muß stets mindestens eine FILL-Zeile vorhanden sein.
	Bei einem Bildschirmüberlauf wird der TTITLE als Seitenkopf ausgegeben.
BTITLE	Definieren eines Seitenfußes, der nach jedem Bildschirmwechsel ausgegeben wird.
	Für die Anzahl der Zeilen, die für einen Seitenfuß definiert werden können, gelten die Angaben unter TTITLE.
	Bei einem Bildschirmüberlauf wird der BTITLE als Seitenfuß ausgegeben.
format	*format* legt das Bildschirmformat fest (siehe Metavariable *format*).
	Wird für *format* TABLE angegeben, darf NEWLINE nicht angegeben werden.
	Wird für *format* LINE angegeben, erfolgt auf die Angaben TABULATOR und BLANK ein Zeilenvorschub mit Leerzeile.
RETURN	Mit RETURN wird festgelegt, daß eine Variable zum Eingabefeld wird, das vorbelegt werden kann. Eine Variable, die mit RETURN gekennzeichnet ist, kann pro Format nur einmal als Eingabefeld verwendet werden.
ausdruck	Definieren von Ausgabe- und/oder Eingabefeldern für das Bildschirmformat. *ausdruck* kann sein: Eine oder mehrere Variablen (auch Systemvariablen) und/oder ein oder mehrere Literale.
	Wenn RETURN oder INIT angegeben wird, muß *ausdruck* eine Variable sein, die nicht mit ".*" qualifiziert werden darf.
INIT	*ausdruck* wird ein Anfangswert zugewiesen. Die INIT-Klausel ist nur zulässig, wenn *ausdruck* eine Variable ist. Ist *ausdruck* ein Vektor oder eine Matrix, erhalten alle Komponenten den entsprechenden Anfangswert *literal* oder NULL.
INIT ausdruck1	*ausdruck1* darf nur *literal*, NULL oder Funktion sein, deren Argumente Literale (aber nicht CURRENT DATE/TIME/TIMESTAMP) sind. *ausdruck1* muß zum Übersetzungszeitpunkt berechenbar sein.
NOCHECK	Eine bei der Deklaration der Variablen *ausdruck* angegebene CHECK-Klausel (siehe Metavariable *check*) wird für die Anfangswertzuweisung nicht ausgewertet.

	NOCHECK ist nicht erlaubt bei redefinierten Variablen oder einer Variablen, die eine andere redefiniert.
ATTRIBUTE	Bildschirmformaten werden Feldeigenschaften zugewiesen.
attribute	Feldeigenschaft (siehe Metavariable *attribute*).
	Als Farbeigenschaften können nur angegeben werden:
	GREEN, RED, WHITE und YELLOW.
	Vorbelegung für Ausgabefelder: VISIBLE, PROTECTED, NOUNDERLINE, NORMALINTENSITY.
	Vorbelegung für Eingabefelder: VISIBLE, UNPROTECTED, NOUNDERLINE, HIGHINTENSITY.
	INVISIBLE darf nur für Eingabefelder (mit RETURN) angegeben werden.
mask	Definieren der Darstellungsmöglichkeiten für maskierte Ein- und Ausgaben (= Ausgabeaufbereitung). Die Angabe *mask* ist nur erlaubt, wenn *ausdruck* eine einfache Variable oder eine einfache Komponente ist.
NEWLINE n	Festlegen der Position der Bildschirmfelder. Mit NEWLINE wird ein Vorschub von *n* Zeilen durchgeführt. Mit NEWLINE 1 wird die aktuelle Zeile abgeschlossen. Falls Daten folgen, werden sie in die nächste Zeile geschrieben. Auch wenn die aktuelle Zeile schon ganz gefüllt ist, also schon soviele Zeichen enthält, wie in der COLUMNS-Angabe des entsprechenden DECLARE vereinbart wurden, bewirkt NEWLINE 1 nicht die Ausgabe einer Leerzeile, sondern nur eine Positionierung auf die nächste Zeile.
	Hat *n* den Wert "0", erfolgt ein bedingter Vorschub, d.h. wenn die aktuelle Zeile leer ist, wird NEWLINE 0 ignoriert, wenn die aktuelle Zeile nicht leer ist, wirkt NEWLINE 0 wie NEWLINE 1.
TABULATOR n	Festlegen der Position der Bildschirmfelder. Die Ausgabe wird ab Spalte *n* fortgeführt. Ist der Wert kleiner als die aktuelle Spaltenposition, so erfolgt ein Zeilen- oder Seitenvorschub. Der entstehende Zwischenbereich wird mit Leerzeichen gefüllt.
	Eine Angabe von TABULATOR ohne einen nachfolgenden Wert *n* hat entweder keine Wirkung oder nur einen Zeilenvorschub zur Folge. Es gilt: 1 ≤ TABULATOR ≤ COLUMNS

BLANK n

Festlegen der Position der Bildschirmfelder. BLANK bewirkt, daß *n* Leerzeichen eingefügt werden. Dabei kann ein Zeilen- oder Seitenvorschub erfolgen.

n ist eine ganze Zahl.

Beispiel

Der Seitenkopf des Bildschirmformats "mitarbeiterausgabe" besteht aus fünf Zeilen, der Seitenfuß aus zwei Zeilen.
Der Seitenkopf hat folgenden Aufbau (In der ersten Zeile stehen das aktuelle Datum und die aktuelle Zeit):

```
1995-12-20                                          13:30:31

                          Mitarbeiter
                          ___________
```

Der Seitenfuß besteht aus einer Leerzeile und einer Zeile mit 80 Gleichheitszeichen (=).

```
DECLARE VARIABLE &datum DATE;
DECLARE VARIABLE &zeit TIME;
...
DECLARE FORM mitarbeiterausgabe
   TTITLE &datum,' '(50),&zeit,NL 2,
          ' '(30),'Mitarbeiter',NL 1,
          ' '(29),'-'(13),NL 2
   BTITLE NL 2,'='(80);
```

DECLARE LIST
Listenformat definieren

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

DECLARE LIST definiert ein Listenformat.

DECLARE LIST legt einen Speicherbereich für ein Listenformat an und definiert das Layout dieser Druckerliste. Der Speicherbereich wird mit FILL-Anweisungen aufgefüllt. Die Ausgabeformatierung wird durch DISPLAY *listname* abgeschlossen.

In der Systemvariablen &PAGES ist die Anzahl der bislang gedruckten Seiten hinterlegt. Nach jedem Seitenvorschub wird &PAGES hochgezählt. Eine Initialisierung der Seitenzählung muß durch explizite Versorgung eines Initialisierungswertes erfolgen. Bei Programmbeginn ist diese Versorgung nicht notwendig.

Bei Einsatz unter UTM muß die zentrale Druckdatei generiert sein. Für die Generierung gibt es drei Möglichkeiten:

– die Datei `DRI.LIST.FILE` ist bereits vorhanden,
– die Datei wird über den Dateikettungsnamen `DRILIST` zugewiesen,
– die Datei `DRI.LIST.FILE` wird bei der ersten DISPLAY-Anweisung von DRIVE/WINDOWS generiert.

Die tatsächliche Druckausgabe auf den Drucker erfolgt beim nächsten STOP (siehe Anweisung STOP) oder LIST (siehe Anweisung LIST).

DECLARE LIST muß im Deklarationsteil des Programms vor allen ausführbaren Anweisungen stehen.

```
DECLARE LIST listname

            [ PERMANENT | TEMPORARY ]
            [ NULL nullwert ]
            [ { COLUMNS n | LINES n }, ... ]

            [ TTITLE [ format ] { ausdruck [ mask ] |
                                  NEWLINE n |
                                  TABULATOR n |
                                  BLANK n }, ... ]

            [ BTITLE [ format ] { ausdruck [ mask ] |
                                  NEWLINE n |
                                  TABULATOR n |
                                  BLANK n }, ... ]
```

listname	Name des Listenformats (max. 31 Zeichen).
PERMANENT	Der Inhalt einer mit PERMANENT definierten Liste bleibt über Unterprogrammende hinaus erhalten und ist bei einem erneuten Aufruf desselben Unterprogramms wieder vorhanden.
TEMPORARY	Vorbelegung Der Inhalt einer mit TEMPORARY definierten Liste bleibt bei Unterprogrammende nicht erhalten.
NULL	Für die NULL-Wertdarstellung wird ein Zeichen festgelegt. Eine bei PARAMETER DYNAMIC vereinbarte NULL-Wertdarstellung wird hiermit überschrieben.
nullwert	Angabe des NULL-Wertzeichens (max. 1 Zeichen). Die NULL-Wertdarstellung wird für den alphanumerischen Datentyp (CHARACTER) oder numerischen Datentypen (NUMERIC, DECIMAL, INTEGER, SMALLINT, REAL und FLOAT) festgelegt. Die alphanumerische NULL-Wertdarstellung ist auch gültig für die Zeit-Datentypen. Die numerische NULL-Wertdarstellung ist auch gültig für den Datentyp INTERVAL.
COLUMNS n	Festlegen der Anzahl der Spalten pro Listenzeile. Die Angabe COLUMNS darf innerhalb von DECLARE LIST nur einmal angegeben werden.

	Für *n* gilt: $0 < n \leq 255$
	Vorbelegung: 132
LINES n	Festlegen der Anzahl der Zeilen pro Listenseite. Die Angabe LINES darf innerhalb von DECLARE LIST nur einmal angegeben werden.
	Für *n* gilt: $0 < n \leq 999$
	Vorbelegung: 60
	Für LINES gilt: Zeilenzahl(TTITLE) + Zeilenzahl(BTITLE) <LINES
TTITLE	Definieren eines Listenkopfes, der nach jedem Seitenwechsel ausgegeben wird.
BTITLE	Definieren eines Listenfußes, der nach jedem Seitenwechsel ausgegeben wird.
format	*format* legt das Listenformat fest (siehe Metavariable *format*).
	Wird für *format* TABLE angegeben, darf NEWLINE nicht angegeben werden.
	Wird für *format* LINE angegeben, erfolgt auf die Angaben TABULATOR und BLANK ein Zeilenvorschub mit Leerzeile.
ausdruck	Definieren von Ausgabefeldern für das Listenformat.
mask	Definieren der Darstellungsmöglichkeiten für maskierte Ausgaben (= Ausgabeaufbereitung).
	Die Angabe *mask* ist nur erlaubt, wenn *ausdruck* eine einfache Variable oder eine einfache Komponente ist.
NEWLINE n	Festlegen der Position der Ausgabefelder innerhalb des Listenformats. Mit NEWLINE wird ein Vorschub von *n* Zeilen durchgeführt. Mit NEWLINE 1 wird die aktuelle Zeile abgeschlossen. Falls Daten folgen, werden sie in die nächste Zeile geschrieben. Auch wenn die aktuelle Zeile schon ganz gefüllt ist, also schon soviele Zeichen enthält, wie in der COLUMNS-Angabe des entsprechenden DECLARE vereinbart wurden, bewirkt NEWLINE 1 nicht die Ausgabe einer Leerzeile, sondern nur eine Positionierung auf die nächste Zeile.
	Hat *n* den Wert "0", erfolgt ein bedingter Vorschub, d.h. wenn die aktuelle Zeile leer ist, wird NEWLINE 0 ignoriert, wenn die aktuelle Zeile nicht leer ist, wirkt NEWLINE 0 wie NEWLINE 1.
TABULATOR n	Festlegen der Position der Ausgabefelder innerhalb des Listenformats.

Die Ausgabe wird ab Spalte *n* fortgeführt. Ist der Wert kleiner als die aktuelle Spaltenposition, so erfolgt ein Zeilen- oder Seitenvorschub. Der entstehende Zwischenbereich wird mit Leerzeichen gefüllt.

Eine Angabe von TABULATOR ohne einen nachfolgenden Wert *n* hat entweder keine Wirkung oder nur einen Zeilenvorschub zur Folge. Es gilt: 1 ≤ TABULATOR ≤ COLUMNS

BLANK n — Festlegen der Position der Ausgabefelder innerhalb des Listenformats.
BLANK bewirkt, daß *n* Leerzeichen eingefügt werden. Dabei kann ein Zeilen- oder Seitenvorschub erfolgen.

n ist eine ganze Zahl.

Beispiel

Der Seitenkopf des Listenformats "mitarbeiterliste" besteht aus sechs Zeilen, der Seitenfuß aus zwei Zeilen.
Der Seitenkopf hat folgenden Aufbau (In der ersten Zeile stehen das aktuelle Datum, die aktuelle Zeit und die aktuelle Seitenzahl):

```
1995-12-20  14:30:22                                Seite:          1

                                Mitarbeiter
                                ___________
```

Der Seitenfuß besteht aus einer Leerzeile und einer Zeile mit 80 Gleichheitszeichen (=).

```
DECLARE VARIABLE &datum DATE;
DECLARE VARIABLE &zeit TIME;
...
DECLARE LIST mitarbeiterliste
   TTITLE &datum,'  ',&zeit,TAB 60,'Seite:',&PAGES,NL 3,
          TAB 31,'Mitarbeiter',NL 1,
          TAB 30,'-'(13),NL 2
   BTITLE NL 2,'='(80);
```

DECLARE SCREEN
FHS-Format definieren

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

DECLARE SCREEN definiert ein FHS-Format, das Sie zuvor mit IFG erstellt haben (siehe IFG [28]). DECLARE SCREEN muß im Deklarationsteil des Programms vor allen ausführbaren Anweisungen stehen.

Bei der Übersetzung von DECLARE SCREEN erzeugt DRIVE/WINDOWS aus der Adressierungshilfe des Bildschirmformats eine strukturierte Variable, die SCREEN-Variable. Über diese SCREEN-Variable tauscht DRIVE/WINDOWS Daten mit FHS aus.

Die Adressierungshilfen für DRIVE-Programme müssen für die angesprochenen Formate in der Formatbibliothek gespeichert sein.

```
DECLARE SCREEN screenformat [ variable ]
                            [ PERMANENT | TEMPORARY ]
                            [ ERRORATTRIBUTE ( attribute, ... ) ]
```

---

screenformat	FHS-Formatname (max. 7 Zeichen).
variable	Name der mit DECLARE SCREEN erzeugten Variablen. In diese Variable wird das FHS-Format (Adressierungshilfe) kopiert. Diese Variable wird bei DISPLAY wieder an FHS übergeben. Sie wird auch als SCREEN-Variable bezeichnet. Wird *variable* nicht angegeben, erhält die SCREEN-Variable den FHS-Formatnamen mit vorangestelltem "&". *variable* ist maximal 32 Zeichen lang (inclusive "&").
PERMANENT	Der Inhalt einer mit PERMANENT definierten Variablen bleibt über Unterprogrammende hinaus erhalten und ist bei einem erneuten Aufruf desselben Unterprogramms wieder vorhanden.
TEMPORARY	Vorbelegung Der Inhalt einer ohne PERMANENT definierten Variablen wird bei Unterprogrammende vergessen.

ERRORATTRIBUTE	Datenfeldern werden Feldeigenschaften für den Fehlerfall zugewiesen, die beim automatischen Fehlerdialog ausgewertet werden (siehe Anweisung DISPLAY *screenformat*, SCREENERROR REPEAT). Wird keine Angabe gemacht, gilt die Angabe bei PARAMETER DYNAMIC.
attribute	Feldeigenschaft (siehe Metavariable *attribute*)

Betriebsmittel bereitstellen

DRIVE/WINDOWS sucht die mit DECLARE SCREEN definierten Formate in der Formatbibliothek mit dem Dateikettungsnamen FORMOML. Ist im TIAM-Betrieb kein solcher Dateikettungsname bekannt, sucht DRIVE/WINDOWS die Formate in der Formatbibliothek `($userid.)DRI.LIB`. Mit der Anweisung PARAMETER STATIC FORMLIB kann jedoch vor der ersten Verarbeitungsanweisung eine andere Formatbibliothek zugewiesen werden.

Besonderheiten beim UTM-Betrieb

Sollen im UTM-Betrieb FHS-Formate eingesetzt werden, müssen Sie in der UTM-Startprozedur die Formatbibliothek angeben:

`SET-FILE-LINK LINK-NAME=FORMOML,FILE-NAME=`*`formatbibliothek`* oder
`PARAMETER STATIC FORMLIB=`*`formatbibliothek`*

Außerdem müssen Sie in der UTM-Startprozedur folgende FHS-Startparameter angeben:

`.FHS DE=NO` falls keine FHS-DE-Formate eingesetzt werden oder
`.FHS DE=YES` falls FHS-DE-Formate eingesetzt werden

und

`.FHS MAPLIB=`*`formatbibliothek`*

Beispiel

Das FHS-Teilformat "maske" wird definiert. Die zugehörige SCREEN-Variable erhält den Namen &bildvar.

```
DECLARE SCREEN maske &bildvar
```

DECLARE TYPE
Datentyp definieren

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

DECLARE TYPE definiert benutzereigene Datentypen.

DECLARE TYPE kann in einem Programm vor der Anweisung PROCEDURE oder im Deklarationsteil des Programms vor allen ausführbaren Anweisungen stehen.
Bei Programmen mit OPTION-Anweisungen muß DECLARE TYPE nach den OPTION-Anweisungen stehen.

```
DECLARE TYPE { [ level ] usertyp datendef }, ...
```

---

level	Angabe einer ein- oder zweistelligen Stufennummer. Sie gibt die hierarchische Struktur einer Datentypdeklaration an. Die Stufennummer muß nicht angegeben werden, solange noch keine Unterteilung des zu definierenden Datentyps vorliegt. Die Stufennummer wird nur für Datengruppen und Wiederholungsgruppen verwendet.
usertyp	Name des benutzerdefinierten Datentyps (max. 31 Zeichen)
datendef	Da die Syntax von *datendef* äußerst komplex ist, werden an dieser Stelle nur die Klauseln angegeben, die in *datendef* enthalten sind, aber aus der oben angegebenen Syntax nicht ersichtlich sind. Bezüglich der einzelnen Datentypen, die unter *datendef* angegeben werden können, ist folgendes zu beachten: – Grunddatentyp (für Zeitspannen) Beim Datentyp INTERVAL muß für *datumzeiteinheit datumzeitfeld1* gleich *datumzeitfeld2* sein. – Strukturtyp (strukturierte Variable) *strukturtyp* darf keine LIKE-Klausel enthalten.

– Basistyp (redefinierte Variable)

basistyp darf keine REDEFINES-Klausel enthalten.

In der CHECK-Klausel darf die Prüfbedingung *bedingung* keine Variablen enthalten. Die Prüfbedingung *bedingung* darf das Schlüsselwort VALUE enthalten anstelle der Variablen, in deren Definition *usertyp* verwendet wird.

Beispiele

Die Variable &wohnung hat den benutzereigenen Datebtyp "adresse".

```
DECLARE TYPE     1 adresse,
                 2 strasse  CHARACTER (30),
                 2 plz      NUM       (5),
                 2 ort      CHARACTER (30);
...
DECLARE VARIABLE &wohnung   adresse;
```

Der benutzereigene Datentyp "db_typ" wird in einem Programm deklariert und in der USING-Leiste angegeben.
DECLARE TYPE steht nach der Anweisung OPTION und vor der Anweisung PROCEDURE.

```
OPTION LISTING=LIST;
DECLARE TYPE 1 db_typ,
               2 db_system   CHARACTER (3),
               2 funktion,   CHARACTER (10),
               2 daten,
                 3 spalte    CHARACTER  (20),
                 3 art_name  CHARACTER  (15);
PROCEDURE h_prog USING &db_parameter db_typ;
...
```

DECLARE VARIABLE
Variable definieren

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

DECLARE VARIABLE definiert Variablen. Für die Variablen können Anfangswerte, Prüfbedingungen und Eigenschaften für die Darstellung von Datenwerten festgelegt werden. Außerdem haben Sie die Möglichkeit Variablen zu redefinieren.

DECLARE VARIABLE muß im Deklarationsteil des Programms vor allen ausführbaren Anweisungen stehen.

```
DECLARE VARIABLE { [ level ] varname
                 [ PERMANENT | TEMPORARY ]
                 { datendef | LIKE { CURSOR cursorname | TABLE tabelle } } }, ...
```

---

level — Angabe einer ein- oder zweistelligen Stufennummer.

Sie gibt die hierarchische Struktur einer Variablendeklaration an. Die Stufennummer muß nicht angegeben werden, solange noch keine Unterteilung der zu definierenden Variablen vorliegt. Die Stufennummer wird nur für Datengruppen und Wiederholungsgruppen verwendet.

varname — Name der Variablen.

varname muß mit dem "&"-Zeichen beginnen und darf insgesamt max. 32 Zeichen lang sein.

Es können einfache Variablen, Vektoren, Matrizen, Datengruppen und Wiederholungsgruppen definiert werden.
Der Wertebereich einer Variablen darf ausschließlich seines Indikatorwertebereiches nicht größer als 32 Kbyte sein.

PERMANENT

Festlegen der Lebensdauer der Variablen, falls das Programm, in dem die Variable definiert wird, mit CALL aufgerufen wird:

– Variablen werden nur beim ersten CALL-Aufruf initialisiert. Bei folgenden CALL-Aufrufen bleiben diese Werte erhalten.
– Variablenwerte bleiben über Unterprogrammende hinaus erhalten.

TEMPORARY

Vorbelegung

– Variablen werden bei jedem CALL-Aufruf neu initialisiert.
– Variablenwerte werden bei Unterprogrammende vergessen.

datendef

Datentyp für die zu definierende Variable. Es gibt folgende Datentypen für DRIVE-Variablen:

– alphanumerischer Datentyp
– numerischer Datentyp
– Zeit-Datentyp
– Datentyp INTERVAL
– benutzerdefinierter Datentyp
– strukturierter Datentyp

Zu den strukturierten Datentypen gehören Vektor, Matrix, Datengruppe und Wiederholungsgruppe. Datengruppe und Wiederholungsgruppe werden in den DRIVE-Handbüchern auch unter dem Begriff "Gruppe" zusammengefaßt.

Gruppen bestehen aus mehreren Gruppenkomponenten, Vektoren und Matrizen bestehen aus einer einzigen einfachen Komponente und einem Wiederholungsfaktor, einfache Variablen bestehen nur aus einer einfachen Komponente.

Die Syntax von *datendef* ist äußerst komplex. An dieser Stelle werden nur die Klauseln angegeben, die in *datendef* enthalten sind, aber aus der oben angegebenen Syntax nicht ersichtlich sind.

INIT-Klausel

Mit INIT wird eine Variable mit einem Literal oder dem NULL-Wert initialisiert. Grundsätzlich belegt DRIVE/WINDOWS jede Variable mit einem Anfangswert, auch wenn keine INIT-Klausel angegeben wurde (siehe unten: Vorbelegung der Variablen mit einem Datentyp).

LIKE-Klausel

Mit LIKE wird die Struktur einer Variablen komponentenweise in eine andere Variable (= *variable*) kopiert.

CHECK-Klausel

Mit CHECK wird eine Bedingung vereinbart und während des Programmlaufes überprüft.

MASK-Klausel

Mit MASK werden die Eigenschaften für die Ausgabeaufbereitung der Datentypen festgelegt.

REDEFINES-Klausel

Mit REDEFINES werden für einen Speicherbereich einer Variablen mehrere Beschreibungen angegeben.

LIKE

Mit LIKE wird die Struktur eines Cursors oder einer Tabelle komponentenweise in eine Variable kopiert. Sie muß eine Datengruppe oder eine Wiederholungsgruppe sein. Die Stufennummern werden beim Kopieren entsprechend angepaßt.

Die Angaben zu den Komponenten sind von demselben Datentyp wie die Satzelemente (Spalten) der angegebenen Tabelle oder des Cursors. Wenn Unterschiede zwischen dem Datentyp des Datenbanksystems und DRIVE/WINDOWS bestehen, wird der Datentyp des Datenbanksystems eindeutig und kompatibel in einen DRIVE-Datentyp konvertiert (siehe Programmiersprache [2]).

Die Komponenten erhalten dieselben Namen wie die Satzelemente der Tabelle oder des Cursors. Bei Mehrdeutigkeiten oder bei leeren Zeichenketten (z.B. bei einem Ausdruck innerhalb von select-liste, siehe SQL-Lexika [4-6]) wird der Name "FILLER" benutzt.

cursorname

Name des Cursors, der deklariert sein muß (siehe SQL-Lexika [4-6], Anweisung DECLARE... CURSOR...).

tabelle

Name einer Basistabelle, eines persistenten oder eines temporären Views, die oder der deklariert sein muß (siehe SQL-Lexika [4-6]).

Vorbelegung der Variablen mit einem Datentyp

Wenn keine INIT-Klausel für eine Variable angegeben ist, wird die Variable typgerecht vorbelegt:

Datentyp	**Vorbelegung**
CHARACTER	Leerzeichen (%)
CHARACTER VARYING, VARCHAR	Leere Zeichenkette
DECIMAL, INTEGER, NUMERIC, SMALLINT, REAL, FLOAT, DOUBLE PRECISION, XDEC, EXTENDED DECIMAL	0 (nicht NULL-Wert!)
DATE	Datum des Übersetzungszeitpunktes, falls PERMANENT angegeben ist, sonst Datum des Ausführungszeitpunktes
TIME, TIME(3),	Uhrzeit des Übersetzungszeitpunktes, falls PERMANENT angegegen ist, sonst Uhrzeit des Ausführungszeitpunktes
TIMESTAMP(3)	Zeitstempel des Übersetzungszeitpunktes, falls PERMANENT angegegen ist, sonst Zeitstempel des Ausführungszeitpunktes
INTERVAL	0 (nicht NULL-Wert!)

Beispiel 1

Definition von einfachen Variablen:
Die alphanumerische Variable "name" ist 20 Zeichen, die numerische (gepackte) Variable "nummer" 10 Zeichen lang; &nummer hat 2 Nachkommastellen.

```
DECLARE VARIABLE &name CHAR (20);
DECLARE VARIABLE &nummer NUM (10,2);
```

Beispiel 2

Definition eines Vektors (= eindimensionale Variable):
Im Vektor &fremdsprache(3) ist das alphanumerische Feld "fremdsprache" mit der Länge 10 dreimal ausgeprägt, z.B. für die Fremdsprachen Englisch, Französisch und Spanisch.

```
DECLARE VARIABLE &fremdsprache(3) CHAR (10);
```

Beispiel 3

Definition einer Matrix (= zweidimensionale Variable):
In der Matrix "&monatsumsatz / filiale"(12,5) ist das numerische Feld "monatsumsatz / filiale" 60-mal ausgeprägt. Das Feld "monatsumsatz / filiale" hat die Länge von 12 Zeichen, davon 2 Nachkommastellen. Es enthält den Wert, den eine von 5 Filialen in einem Monat umsetzt.

```
DECLARE VARIABLE "&monatsumsatz / filiale"(12,5) NUM(12,2);
```

Wegen der Sonderzeichen (Leerzeichen und Schrägstrich) im Variablennamen muß dieser in Anführungszeichen (") stehen. Der Umsatz für den Monat August der Filiale 3 steht z.B. in der Variablen "&monatsumsatz / filiale"(08,3).

Beispiel 4

Definition einer Datengruppe:

```
DECLARE VARIABLE 1 &mitarbeiter,
                   2 personal_nr        CHAR (06),
                   2 nachname           CHAR (20),
                   2 vorname            CHAR (20),
                   2 adresse,
                     3 land             CHAR (03),
                     3 strasse          CHAR (26),
                     3 plz              CHAR (10),
                     3 ort              CHAR (20),
                   2 gehalt             NUM  (7,2),
                   2 abt_leiter         CHAR (06),
                   2 abt_mit_           INT,
                   2 proj_mit_          INT;
```

Beispiel 5

Definition einer Wiederholungsgruppe:
Der Wiederholungsfaktor soll 5 sein.

```
DECLARE VARIABLE 1 &w(5),
                   2 i   INT,
                   2 c   CHAR (3);
```

Beispiel 6

In der Datengruppe "saetze" wird für die Komponenten "aabt_mit_nr", "egehalt", "abt_mit_nr" und "gehalt_" über Datentyp und Länge hinaus folgendes vereinbart:

Als Anfangswert wird der Variablen &aabt_mit_nr der Wert 0000 zugewiesen. Werden ihr während der Programmausführung nicht-numerische Werte zugewiesen, wird die Meldung "Geben Sie eine Zahl ein" ausgegeben.

Werden der Variablen &egehalt während der Programmausführung Werte zugewiesen, die kleiner 20000 sind, wird die Meldung "Das Gehalt ist zu niedrig" ausgegeben.

Die Variable &abt_mit_nr redefiniert die Variable &aabt_mit_nr.

Werte der Variable &gehalt_ werden ohne führende Nullen, mit mindestens einer Vorkommastelle, dem Dezimalzeichen, zwei Nachkommastellen und der Zeichenfolge "%DM" dargestellt.

```
DECLARE VARIABLE 1 &saetze,
                   2 aabt_mit_nr CHAR (4) INIT 'OOOO'
                     CHECK &aabt_mit_nr IS NOT NUMERIC
                     MESSAGE 'Geben Sie eine Zahl ein',
                   2 aufnahmesatz,
                     3 enachname CHAR (20),
                     3 evorname CHAR (20),
                     3 egehalt NUM (7,2)
                       CHECK &egehalt < 20000
                       MESSAGE 'Das Gehalt ist zu niedrig',
                   2 drucksatz,
                     3 abt_mit_nr CHAR (4) REDEFINES &aabt_mit_nr,
                     3 nachname_ CHAR (20),
                     3 vorname_ CHAR (20),
                     3 gehalt_ NUM (7,2) MASK 'ZZZZ9P99'' DM''';
```

DRIVE-Systemvariablen

DRIVE/WINDOWS stellt Systemvariablen zur Verfügung, die in DRIVE-Programmen genutzt werden können. Eine Liste der Systemvariablen finden Sie in der DRIVE-Programmiersprache [2].

Der Gültigkeitsbereich der Systemvariablen ist auf das jeweilige Programm begrenzt. Über Systemvariablen können keine Informationen an andere, externe DRIVE-Programme gegeben werden, ausgenommen mit Hilfe der USING-Klausel.

DELETE FILE RECORD
Satz in ISAM-Datei löschen

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

DELETE FILE RECORD löscht in einer geöffneten ISAM-Datei den Datensatz mit dem angegebenen ISAM-Schlüssel.

```
DELETE FILE RECORD datei KEY charausdruck
```

---

datei	Logischer Name einer Datei, in der ein Datensatz gelöscht wird. Mit diesem Namen muß die Datei im Programm mit der Anweisung DECLARE FILE definiert sein.
charausdruck	ISAM-Schlüssel des Datensatzes, der gelöscht wird.

DISPATCH
Unterprogramme im verteilten System parallel aufrufen

Diese Anweisung ist gültig

– im UTM-Betrieb, im TIAM-Betrieb ohne Funktionalität (s.u.)
– im Programm-Modus

DISPATCH kennzeichnet den Beginn eines DISPATCH-Blocks, dessen Ende mit END DISPATCH festgelegt wird. Innerhalb eines DISPATCH-Blocks werden alle CALL-Anweisungen, die Unterprogramme in einem entfernten System aufrufen (Remote-CALL-Anweisungen), gleichzeitig bei END DISPATCH ausgeführt. Alle anderen zulässigen Anweisungen innerhalb eines DISPATCH-Blocks werden sequentiell ausgeführt.

Das rufende Programm wartet bei END DISPATCH bis alle entfernt gerufenen Unterprogramme ausgeführt sind und fährt dann mit der nächsten Anweisung fort.

DISPATCH-Blöcke dürfen folgende Anweisungen nicht enthalten:

– BREAK PROCEDURE
– COMMIT WORK
– DISPATCH
– DO
– STOP

Gleiche RETURN-Parameter in den Remote-CALL-Anweisungen eines DISPATCH-Blocks sind nicht erlaubt.

Tritt ein Fehler bei der Ausführung einer Remote-CALL-Anweisung auf, wird das rufende Programm abgebrochen. Die Anweisung WHENEVER wird nicht unterstützt.

Wenn in einem DISPATCH-Block Variablenwerte vor einer fehlerhaften Remote-CALL-Anweisung verändert wurden, werden die Variablenwerte nicht zurückgesetzt.

Im TIAM-Betrieb werden die Anweisungen DISPATCH und END DISPATCH ignoriert und die CALL-Anweisungen sequentiell ausgeführt.

DISPATCH-Blöcke dürfen nicht mit den Anweisungen BREAK CYCLE, BREAK SUBPROCEDURE oder CONTINUE CYCLE verlassen werden.

DISPATCH-Blöcke, Schleifen (CYCLE), Bedingungen (IF) und Vezweigungen dürfen sich nicht überlappen.

DISPATCH

Beispiel

```
DISPATCH;
  CALL local_proc1          USING RETURN &a;
  CALL remote_proc1         USING RETURN &a;
  SET &b = &a;
  CALL TAC remote_tac       USING RETURN &b;
END DISPATCH;
```

(Zum Beispiel siehe DRIVE-Programmiersprache [2], Abschnitt Parallele Verteilte Verarbeitung anstoßen mit DISPATCH)

Regeln bei Verteilter Transaktionsverarbeitung

Remote-CALL-Anweisungen in einem DISPATCH-Block müssen in der Übersetzungseinheit dieses DISPATCH-Blocks stehen. D.h.: Wird innerhalb eines DISPATCH-Blocks ein externes Unterprogramm mit CALL lokal aufgerufen, so dürfen in diesem externen Unterprogramm keine Remote-CALL-Anweisungen stehen.

DISPLAY FORM
Kompakt-Bildschirmformat definieren und ausgeben

Diese Anweisung ist gültig

- im TIAM-Betrieb
- im UTM-Betrieb, aber nicht im UTM-Asynchronbetrieb und bei VTV nicht in Auftragnehmer-Umgebung
- im Programm- und Debug-Modus

DISPLAY FORM definiert im Ablaufteil eines Programms ein Kompakt-Bildschirmformat ad hoc, füllt es mit Inhalt und gibt das Format auf den Bildschirm aus.

Die Anweisung DISPLAY FORM erlaubt Ein- und Ausgaben in das Bildschirmformat. Sie enthält implizit die DECLARE FORM-, die FILL- und die DISPLAY-Anweisungen. Statt drei Anweisungen genügt eine. Das Kompakt-Bildschirmformat ist eine Sonderform des DRIVE-Formats.

Nach Ausgabe des Kompakt-Bildschirmformats kann der Formatinhalt nicht nochmals ausgegeben werden, da das Format nicht über einen Namen ansprechbar ist.

```
DISPLAY FORM [ format ] { [ RETURN ] ausdruck [ INIT ausdruck1 [ NOCHECK ] ]
                          [ ATTRIBUTE ( attribute, ... ) ] [ mask ] |
                        NEWLINE n |
                        TABULATOR n |
                        BLANK n }, ...

            [ { COLUMNS n | LINES n }, ... ]

            [ TTITLE [ format ] { [ RETURN ] ausdruck [ INIT ausdruck1 [ NOCHECK ] ]
                                  [ ATTRIBUTE ( attribute, ... ) ] [ mask ] |
                                NEWLINE n |
                                TABULATOR n |
                                BLANK n }, ... ]

            [ BTITLE [ format ] { [ RETURN ] ausdruck [ INIT ausdruck1 [ NOCHECK ] ]
                                  [ ATTRIBUTE ( attribute, ... ) ] [ mask ] |
                                NEWLINE n |
                                TABULATOR n |
                                BLANK n }, ... ]
```

---

format	*format* legt das Bildschirmformat fest (siehe Metavariable *format*).
	Wird für *format* TABLE angegeben, darf NEWLINE nicht angegeben werden.
	Wird für *format* LINE angegeben, erfolgt auf die Angaben TABULATOR und BLANK ein Zeilenvorschub mit Leerzeile.
RETURN	Mit RETURN wird festgelegt, daß eine Variable zum Eingabefeld wird, das vorbelegt werden kann. Eine Variable, die mit RETURN gekennzeichnet ist, kann pro Format nur einmal als Eingabefeld verwendet werden.
	Paßt eine Eingabevariable (mit RETURN gekennzeichnet) nicht auf eine Bildschirmseite, so erfolgt zum Ausführungszeitpunkt Programmabbruch.
	Paßt eine Ausgabevariable (ohne RETURN gekennzeichnet) nicht auf eine Bildschirmseite, so wird sie zur Ausgabe abgeschnitten. Die letzten drei Zeichen der Ausgabevariable werden mit ">>>" gekennzeichnet.
	Befindet sich im TTITLE oder BTITLE ein Eingabe- oder Ausgabefeld, das nicht in den dafür vorgesehenen Platz paßt, erfolgt zum Ausführungszeitpunkt Prozedurabbruch.
	RETURN wird im Debug-Modus ignoriert.
ausdruck	Definieren von Ausgabe- und/oder Eingabefeldern für das Bildschirmformat (siehe Metavariable *ausdruck*). *ausdruck* darf nicht länger als 31 Kbyte sein.
INIT	*ausdruck* wird ein Anfangswert zugewiesen. Die INIT-Klausel ist nur zulässig, wenn *ausdruck* eine Variable ist. Ist *ausdruck* ein Vektor oder eine Matrix, erhalten alle Komponenten den entsprechenden Anfangswert *literal* oder NULL.
ausdruck1	*ausdruck1* darf nur *literal*, NULL oder Funktion sein, deren Argumente Literale (aber nicht CURRENT DATE/TIME/TIMESTAMP) sind. *ausdruck1* muß zum Übersetzungszeitpunkt berechenbar sein.
NOCHECK	Eine bei der Deklaration der Variablen *ausdruck* angegebene CHECK-Klausel (siehe Metavariable *check*) wird für die Anfangswertzuweisung nicht ausgewertet.
	NOCHECK ist nicht erlaubt bei redefinierten Variablen oder einer Variablen, die eine andere redefiniert.
ATTRIBUTE	Bildschirmformaten werden Feldeigenschaften zugewiesen.
	ATTRIBUTE wird im Debug-Modus ignoriert.

attribute	Feldeigenschaft (siehe Metavariable *attribute*). Als Farbeigenschaften können nur angegeben werden: GREEN, RED, WHITE und YELLOW. Vorbelegung für Ausgabefelder: VISIBLE, PROTECTED, NOUNDERLINE, NORMALINTENSITY. Vorbelegung für Eingabefelder: VISIBLE, UNPROTECTED, NOUNDERLINE, HIGHINTENSITY. INVISIBLE darf nur für Eingabefelder (mit RETURN) angegeben werden.
mask	Definieren der Darstellungsmöglichkeiten für maskierte Ein- und Ausgaben (= Ausgabeaufbereitung). Die Angabe *mask* ist nur erlaubt, wenn *ausdruck* eine einfache Variable oder eine einfache Komponente ist.
NEWLINE n	Festlegen der Position der Bildschirmfelder. Mit NEWLINE wird ein Vorschub von *n* Zeilen durchgeführt. Mit NEWLINE 1 wird die aktuelle Zeile abgeschlossen. Falls Daten folgen, werden sie in die nächste Zeile geschrieben. Auch wenn die aktuelle Zeile schon ganz gefüllt ist, also schon soviele Zeichen enthält, wie in der COLUMNS-Angabe des entsprechenden DECLARE vereinbart wurden, bewirkt NEWLINE 1 nicht die Ausgabe einer Leerzeile, sondern nur eine Positionierung auf die nächste Zeile. Hat *n* den Wert "0", erfolgt ein bedingter Vorschub, d.h. wenn die aktuelle Zeile leer ist, wird NEWLINE 0 ignoriert, wenn die aktuelle Zeile nicht leer ist, wirkt NEWLINE 0 wie NEWLINE 1.
TABULATOR n	Festlegen der Position der Bildschirmfelder. Die Ausgabe wird ab Spalte *n* fortgeführt. Ist der Wert kleiner als die aktuelle Spaltenposition, so erfolgt ein Zeilen- oder Seitenvorschub. Der entstehende Zwischenbereich wird mit Leerzeichen gefüllt. Eine Angabe von TABULATOR ohne einen nachfolgenden Wert *n* hat entweder keine Wirkung oder nur einen Zeilenvorschub zur Folge. Es gilt: 1 ≤ TABULATOR ≤ COLUMNS
BLANK n	Festlegen der Position der Bildschirmfelder. BLANK bewirkt, daß *n* Leerzeichen eingefügt werden. Dabei kann ein Zeilen- oder Seitenvorschub erfolgen. *n* ist eine ganze Zahl.

COLUMNS n	Festlegen der Anzahl der Spalten pro Bildschirm. Die Angabe COLUMNS darf innerhalb von DISPLAY FORM nur einmal angegeben werden. Für n gilt: $0 < n \leq$ Anzahl der Bildschirmspalten Vorbelegung: Spaltenzahl des aktuellen Bildschirms
LINES n	Festlegen der Anzahl der Zeilen pro Bildschirm. Die Angabe LINES darf innerhalb von DISPLAY FORM nur einmal angegeben werden. Für n gilt: $0 < n \leq$ Anzahl der Bildschirmzeilen - 1 Vorbelegung: Zeilenanzahl des aktuellen Bildschirms - 1 Die letzte Zeile ist als Meldungszeile reserviert. Für LINES gilt: Zeilenzahl (TTITLE) + Zeilenzahl (BTITLE) < LINES.
TTITLE	Definieren eines Seitenkopfes, der nach jedem Bildschirmwechsel ausgegeben wird. Die Gesamtanzahl der Zeilen für TTITLE, BTITLE und FILL darf maximal sein: Anzahl der Bildschirmzeilen - 1. Die letzte Zeile ist als Meldungszeile reserviert. Es muß stets mindestens eine FILL-Zeile vorhanden sein. Bei einem Bildschirmüberlauf wird der TTITLE als Seitenkopf ausgegeben.
BTITLE	Definieren eines Seitenfußes, der nach jedem Bildschirmwechsel ausgegeben wird. Für die Anzahl der Zeilen, die für einen Seitenfuß definiert werden können, gelten die Angaben unter TTITLE. Bei einem Bildschirmüberlauf wird der BTITLE als Seitenfuß ausgegeben.

Die Maskierung von CHAR-Ausdrücken in der NUM-Funktion hat nur die Wirkung, daß der CHAR-Ausdruck bei der Eingabe hinsichtlich der Maske auf Zulässigkeit geprüft wird. D.h. die Maske wirkt nicht bei der Ausgabe.

Beziehungen zu anderen Anweisungen

– Die Bildschirmausgabe wird durch Versorgung der strukturierten Variablen festgelegt (siehe Anweisung DECLARE SCREEN). Diese Variablen können z.B. über SET versorgt werden (siehe Anweisung SET).

Beispiel

Bis auf die 15. Zeile ist das Kompakt-Bildschirmformat leer. In der 15. Zeile steht 5 Zeichen eingerückt: "Geben Sie die laufende Nummer ein:"

```
DECLARE VARIABLE &nummer NUM(3);
...
DISPLAY FORM NL 15, TAB 5,'Geben Sie die laufende Nummer ein: ',RETURN &nummer;
```

DISPLAY formatname
DRIVE-Format ausgeben

Diese Anweisung ist gültig

– im TIAM-Betrieb
– im UTM-Betrieb, aber nicht im UTM-Asynchronbetrieb und bei VTV nicht in Auftragnehmer-Umgebung
– im Programm-Modus

DISPLAY *formatname* schließt eine DRIVE-Formataufbereitung ab und gibt das Format auf dem Bildschirm aus. Nach der Ausgabe wird der Speicherbereich des DRIVE-Bildschirmformats nicht gelöscht. So kann ein Format mehrmals hintereinander ausgegeben werden, wenn keine neue FILL-Anweisung für diesen Bereich gegeben wurde. Wird ein FILL für diesen Bereich angegeben, wird der Formatspeicher gelöscht und ein neuer Inhalt aufgebaut.

```
DISPLAY formatname
```

---

formatname	Name des DRIVE-Formats (max. 31 Zeichen). Das angegebene Format muß im Deklarationsteil des Programms mit DECLARE FORM definiert sein. *formatname* darf nur einmal angegeben werden.

DISPLAY LIST
Kompakt-Listenformat definieren und ausgeben

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm- und Debug-Modus

DISPLAY LIST definiert im Ablaufteil eines Programms ein Kompakt-Listenformat ad hoc, füllt es mit Inhalt und gibt es auf einem Drucker aus.

Die Anweisung DISPLAY LIST enthält implizit die DECLARE LIST-, die FILL- und die DISPLAY-Anweisungen. Statt drei Anweisungen genügt eine. Das Kompakt-Listenformat ist eine Sonderform des DRIVE-Listenformats.

Nach Ausgabe des Kompakt-Listenformats kann der Formatinhalt nicht nochmals ausgegeben werden, da das Format nicht über einen Namen ansprechbar ist.

Bei Einsatz unter UTM muß die zentrale Druckdatei generiert sein. Für die Generierung gibt es drei Möglichkeiten:

– die Datei `DRI.LIST.FILE` ist bereits vorhanden,
– die Datei wird über den Dateikettungsnamen `DRILIST` zugewiesen,
– die Datei `DRI.LIST.FILE` wird bei der 1. DISPLAY-Anweisung von DRIVE/WINDOWS generiert.

Die tatsächliche Druckausgabe auf einen Drucker erfolgt beim nächsten STOP (siehe Anweisung STOP) oder LIST (siehe Anweisung LIST).

```
DISPLAY LIST [ format ] { ausdruck [ mask ] |
                         NEWLINE n |
                         TABULATOR n |
                         BLANK n }, ...

            [ { COLUMNS n | LINES n }, ... ]

            [ TTITLE [ format ] { ausdruck [ mask ] |
                                  NEWLINE n |
                                  TABULATOR n |
                                  BLANK n }, ... ]

            [ BTITLE [ format ] { ausdruck [ mask ] |
                                  NEWLINE n |
                                  TABULATOR n |
                                  BLANK n }, ... ]
```

format — *format* legt das Listenformat fest (siehe Metavariable *format*).

Wird für *format* TABLE angegeben, darf NEWLINE nicht angegeben werden.

Wird für *format* LINE angegeben, erfolgt auf die Angaben TABULATOR und BLANK ein Zeilenvorschub mit Leerzeile.

ausdruck — Definieren von Ausgabefeldern für das Listenformat.

mask — Definieren der Darstellungsmöglichkeiten für maskierte Ausgaben (= Ausgabeaufbereitung).
Die Angabe *mask* ist nur erlaubt, wenn *ausdruck* eine einfache Variable oder eine einfache Komponente ist.

NEWLINE n — Festlegen der Position der Ausgabefelder innerhalb des Listenformats.

Mit NEWLINE wird ein Vorschub von *n* Zeilen durchgeführt. Mit NEWLINE 1 wird die aktuelle Zeile abgeschlossen. Falls Daten folgen, werden sie in die nächste Zeile geschrieben.

Auch wenn die aktuelle Zeile schon ganz gefüllt ist, also schon soviele Zeichen enthält, wie in der COLUMNS-Angabe des entsprechenden DECLARE vereinbart wurden, bewirkt NEWLINE 1 nicht die Ausgabe einer Leerzeile, sondern nur eine Positionierung auf die nächste Zeile.

	Hat *n* den Wert "0", erfolgt ein bedingter Vorschub, d.h. wenn die aktuelle Zeile leer ist, wird NEWLINE 0 ignoriert, wenn die aktuelle Zeile nicht leer ist, wirkt NEWLINE 0 wie NEWLINE 1.
TABULATOR n	Festlegen der Position der Ausgabefelder innerhalb des Listenformats. Die Ausgabe wird ab Spalte *n* fortgeführt. Ist der Wert kleiner, als die aktuelle Spaltenposition, so erfolgt ein Zeilen- oder Seitenvorschub. Der entstehende Zwischenbereich wird mit Leerzeichen gefüllt. Eine Angabe von TABULATOR ohne einen nachfolgenden Wert *n* hat entweder keine Wirkung oder nur einen Zeilenvorschub zur Folge. Es gilt: 1 ≤ TABULATOR ≤ COLUMNS
BLANK n	Festlegen der Position der Ausgabefelder innerhalb des Listenformats. BLANK bewirkt, daß *n* Leerzeichen eingefügt werden. Dabei kann ein Zeilen- oder Seitenvorschub erfolgen. *n* ist eine ganze Zahl.
COLUMNS n	Festlegen der Anzahl der Spalten pro Listenzeile. Die Angabe COLUMNS darf innerhalb von DISPLAY LIST nur einmal angegeben werden. Für *n* gilt: 0 < *n*≤ 255 Vorbelegung: 132
LINES n	Festlegen der Anzahl der Zeilen pro Listenseite. Die Angabe LINES darf innerhalb von DISPLAY LIST nur einmal angegeben werden. Für *n* gilt: 0 < *n* ≤ 999 Vorbelegung: 60 Für LINES gilt: Zeilenzahl(TTITLE) + Zeilenzahl(BTITLE) < LINES
TTITLE	Definieren eines Listenkopfes, der nach jedem Seitenwechsel ausgegeben wird.
BTITLE	Definieren eines Listenfußes, der nach jedem Seitenwechsel ausgegeben wird.

Beispiel

Das Kompakt-Listenformat hat folgenden Aufbau: In der zweiten Zeile stehen das aktuelle Datum, die aktuelle Zeit und die aktuelle Seitenzahl. In der elften Zeile stehen die Dateninhalte der Variablen &nachname, &vorname, &gehalt.

```
1995-12-20  15:03:42                                        Seite:                1


                              Mitarbeiter
                             -------------





    Winterberg               Hannelore                3500.00 DM
```

```
DISPLAY LIST
   NL 1,&datum,'  ',&zeit,TAB 60,'Seite: ',&PAGES,NL 3,
   TAB 31,'Mitarbeiter',NL 1,
   TAB 30,'-'(13),NL 5,
   TAB 5,&nachname,TAB 30,&vorname,TAB 55,&gehalt,' DM';
```

DISPLAY listname
Listenformat ausgeben

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

DISPLAY *listname* schließt eine DRIVE-Listenaufbereitung ab und gibt das Format auf dem Drucker aus.

Nach der Ausgabe wird der Speicherbereich des DRIVE-Listenformats nicht gelöscht. So kann ein Format mehrmals hintereinander ausgegeben werden, wenn keine neue FILL-Anweisung für diesen Bereich gegeben wurde. Wird ein FILL für diesen Bereich angegeben, wird der Formatspeicher gelöscht und ein neues Format aufgebaut.

Bei Einsatz unter UTM muß die zentrale Druckdatei generiert sein. Für die Generierung gibt es drei Möglichkeiten:

– die Datei `DRI.LIST.FILE` ist bereits vorhanden
– die Datei wird über den Dateikettungsnamen `DRILIST` zugewiesen
– die Datei `DRI.LIST.FILE` wird bei der ersten DISPLAY-Anweisung von DRIVE/WINDOWS generiert

Die tatsächliche Druckausgabe auf einen Drucker erfolgt dann beim nächsten STOP (siehe Anweisung STOP) oder LIST (siehe Anweisung LIST).

```
DISPLAY listname
```

---

listname	Name des DRIVE-Listenformats (max. 31 Zeichen). Das angegebene Listenformat muß im Deklarationsteil des Programms mit DECLARE LIST definiert sein.

DISPLAY screenformat
FHS-Format ausgeben

Diese Anweisung ist gültig

- im TIAM-Betrieb nur bei FHS-Formaten ohne Dialogerweiterung (DE)
- im UTM-Betrieb, aber nicht im UTM-Asynchronbetrieb und bei VTV nicht in Auftragnehmer-Umgebung
- im Programm-Modus

DISPLAY *screenformat* gibt ein FHS-Format aus, das Sie zuvor mit IFG erstellt haben (siehe IFG [28]). Sie können mehrere Teilformate gleichzeitig ausgeben. Bei der gleichzeitigen Ausgabe von mehreren Teilformaten dürfen Sie nicht FHS-Formate für die Dialogerweiterung (FHS-DE) mit einfachen FHS-Formaten mischen.

Die Ausgabe eines Bildschirmformats bedeutet: die SCREEN-Variable wird, zusammen mit eventuell vorhandenen Eingabewerten, an FHS übergeben und dann am Bildschirm ausgegeben.

Die SCREEN-Variable ist die strukturierte Variable, die mit DECLARE SCREEN erzeugt wird zur Aufnahme der Adressierungshilfe eines FHS-Formats.

Nach der Ausgabe wird der Inhalt der SCREEN-Variablen nicht gelöscht. Formate dürfen sich am Bildschirm nicht überlappen.

```
DISPLAY screenformat, ...
        [ SCREENERROR { REPEAT | CONTINUE } ]
        [ CURSOR { POSITION ( zeile1, spalte1 ) | TO feld1 }
        [ MESSAGE schluessel [ POSITION ( zeile2, spalte2 ) | TO feld2 ]
```

---

screenformat

FHS-Teilformatname (max. 7 Zeichen).

Das Format muß im Deklarationsteil des Programms mit der Anweisung DECLARE SCREEN definiert sein.

Wenn Sie mehrere FHS-Teilformate gleichzeitig ausgeben lassen, dürfen Sie entweder nur Formate mit Dialogerweiterung (DE) oder nur Formate ohne Dialogerweiterung ausgeben lassen.

Beim automatischen Fehlerdialog darf die letzte Bildschirmzeile nicht verwendet werden. Sie wird von DRIVE/WINDOWS zur Ausgabe von Meldungen verwendet.

Beim benutzergesteuerten Fehlerdialog kann auch die letzte Bildschirmzeile verwendet werden, z.B. zur Ausgabe anwendereigener Fehlermeldungen.

SCREENERROR

Mit SCREENERROR wird das Verhalten von DRIVE/WINDOWS festgelegt, wenn fehlerhafte Feldeingaben vorliegen.

REPEAT

Vorbelegung

Die Teilformate werden solange wiederholt ausgegeben, bis die Eingabe fehlerfrei ist oder ein gewollter Abbruch durch BREAK erfolgt (= automatischer Fehlerdialog). Als fehlerhafte Eingabewerte gelten alle Felder, bei denen die CHECK-Klausel nicht zutrifft (siehe Anweisung DECLARE VARIABLE).

Die fehlerhaften Eingabewerte werden entsprechend dem Operanden ERRORATTRIBUTE, der mit der Anweisung DECLARE SCREEN oder PARAMETER DYNAMIC festgelegt wurde, gekennzeichnet (siehe Anweisungen DECLARE SCREEN und PARAMETER DYNAMIC).

Eine Datenübertragung vom FHS-FORMANT-Format in die SCREEN-Variable erfolgt für jedes Teilformat erst nach korrekter Eingabe aller Bildschirmfelder.

CONTINUE

Das Programm wird nach DISPLAY fortgesetzt (= benutzergesteuerter Fehlerdialog). Bei fehlerhafter Eingabe wird das Programm abgebrochen, wenn keine entsprechende WHENEVER-Anweisung vorliegt, und die Systemvariable &ERROR_STATE erhält folgende Werte:

&ERROR='FORMAT_ERROR'
&FORMAT_NAME=Name des ersten fehlerhaften Teilformats
&VAR_NAME=Name des ersten fehlerhaften Eingabefeldes

Wird das Programm abgebrochen, weil eine unzulässige K/F-Taste betätigt wurde, erhält die Systemvariable &ERROR_STATE folgende Werte:

&ERROR='FORMAT_ERROR'
&FORMAT_NAME= "%K/F_ERROR"
&VAR_NAME=INIT-Wert von DECLARE VARIABLE

Fehlerhafte Eingabewerte können über die Anweisung SET ERRORATTRIBUTE gekennzeichnet werden und die fehlerhaften Formate erneut ausgegeben werden. Als fehlerhafte Eingabewerte gelten alle Felder, bei denen die CHECK-Klausel nicht zutrifft.

	Alle fehlerfreien Felder werden in die strukturierte SCREEN-Variable übertragen. Die restlichen Komponenten bleiben erhalten. CONTINUE wird nur im Zusammenhang mit Verletzungen der CHECK-Klauseln wirksam.
CURSOR	Die Schreibmarke wird an eine bestimmte Stelle auf den Bildschirm gesetzt. CURSOR dürfen Sie nur angeben, wenn das auszugebende Teilformat ein FHS-DE-Format ist und wenn mit IFG für *sreenformat* das Globalattribut "Schreibmarkenposition" gesetzt wurde (siehe IFG [28]).
POSITION	legt die absolute Position (Zeile / Spalte) der Schreibmarke fest.
zeile1	Zeile (1 ≤ *zeile2* ≤ Anzahl der Bildschirmzeilen). *zeile2* muß eine ganze Zahl sein.
spalte1	Spalte (1 ≤ *spalte2* ≤ Anzahl der Bildschirmspalten). *spalte2* muß eine ganze Zahl sein.
TO	Die Schreibmarke wird auf das erste Zeichen des Felds *feld1* gesetzt. Bei Listen wird die Schreibmarke auf die erste Spalte und die erste Zeile des Listenbereichs gesetzt.
feld1	Feld im Teilformat, das ausgegeben werden soll. *feld1* muß eine Komponente der SCREEN-Variablen zu *screenformat* sein. Damit die Komponenten von SCREEN-Variablen eindeutig zugeordnet werden können, müssen sich die ersten 8 Zeichen der Feldnamen aller mit DISPLAY screenformat ausgegebenen Bildschirmformate unterscheiden.
MESSAGE	Die FHS-DE-Meldung mit Meldungsschlüssel *schluessel*, die Sie mit IFG erstellt haben, wird ausgegeben. Abhängig von der Festlegung mit IFG erfolgt die Ausgabe entweder in einer Meldungsbox oder im Meldebereich des Teilformats *screenformat*. MESSAGE dürfen Sie nur angeben, wenn das auszugebende Teilformat ein FHS-DE-Format ist und wenn mit IFG für *screenformat* das Globalattribut "Meldungskennzeichen" gesetzt wurde (siehe IFG [28]).
schluessel	Meldungsschlüssel der FHS-DE-Meldung. *schluessel* kann als Variable (siehe Metavariable *variable*) oder als alphanumerisches Literal (siehe *charliteral* bei Metavariable *literal*) angegeben werden.

	schluessel muß in der Form AAAAnnn angegeben werden. A steht für einen Buchstaben (A-Z), n für eine Ziffer (0-9). Für AAAA darf nicht IDHS stehen.
POSITION	legt die absolute Position der Meldungsbox fest. Die Meldungsbox wird mit einer zusätzlichen Verschiebung (+2,+2) zu *zeile3*, *spalte3* positioniert. POSITION wird nur ausgewertet, wenn mit IFG festgelegt wurde, daß die Meldung in eine Meldungsbox ausgegeben werden soll. Wenn eine Meldung in einer Meldungsbox ausgegeben werden soll, und Sie weder POSITION noch TO angeben, wird die Meldungsbox in die Mitte des Bildschirms plaziert. Wenn eine mit MESSAGE POSITION positionierte Meldungsbox eine Schreibmarke überdeckt, die mit CURSOR POSITION gesetzt ist, wird MESSAGE POSITION ignoriert.
zeile2	Zeile (1 ≤ *zeile3* ≤ Anzahl der Bildschirmzeilen). *zeile3* muß eine ganze Zahl sein.
spalte2	Spalte (1 ≤ *spalte3* ≤ Anzahl der Bildschirmspalten). *spalte3* muß eine ganze Zahl sein.
TO	legt fest, daß die Meldungsbox mit der Standardverschiebung (+2,+2) zu *feld2* positioniert werden soll. TO wird nur ausgewertet, wenn mit IFG festgelegt wurde, daß die Meldung in eine Meldungsbox ausgegeben werden soll. Wenn eine Meldung in einer Meldungsbox ausgegeben werden soll, und Sie weder TO noch POSITION angeben, wird die Meldungsbox in die Mitte des Bildschirms plaziert.
feld2	Feld im Teilformat, das ausgegeben werden soll. *feld2* muß eine Komponente der SCREEN-Variablen zu *screenformat* sein. Damit die Komponenten von SCREEN-Variablen eindeutig zugeordnet werden können, müssen sich die ersten 8 Zeichen der Feldnamen aller mit DISPLAY screenformat ausgegebenen Bildschirmformate unterscheiden.

Beziehungen zu anderen Anweisungen

Die Bildschirmausgabe wird durch Versorgung der strukturierten Variablen festgelegt (siehe Anweisung DECLARE SCREEN). Diese Variablen können z.B. über SET versorgt werden (siehe Anweisung SET).

Beispiel

Das IFG/FHS-Teilformat "maske" wird ausgegeben. Bei fehlerhaften Eingaben wird die Ausgabe des Formats wiederholt.

```
DISPLAY maske SCREENERROR REPEAT
```

DO
Dialog-Programm starten

Diese Anweisung ist gültig

- im TIAM-Betrieb
- im UTM-Betrieb, aber nicht im UTM-Asynchronbetrieb und bei VTV nicht in Auftragnehmer-Umgebung
- im Dialog-Modus
- im Programm-Modus nur in einem Dialog-Programm

Abhängig vom Modus hat DO zwei Funktionen:

- Wird DO im Dialog-Modus angegeben, wird implizit erst ein COMPILE durchgeführt und anschließend das Programm gestartet.

  Wurde für dieses Programm bereits Zwischencode erzeugt und in der aktuellen DRIVE-Bibliothek gespeichert, greift DO auf diesen direkt zu und startet nur noch das Programm. Die Syntax- und Semantiküberprüfung entfällt in diesem Fall.

  DRIVE/WINDOWS sucht unter dem angegebenen Namen nach dem aktuellesten Programm, gleichgültig, ob es als Source oder als Zwischencode vorliegt.

- Wird DO in einem Dialog-Programm angegeben, wird dieses Programm abgebrochen und das Folgeprogramm aufgerufen. Dieses DO hat dieselbe Wirkung wie ein END PROCEDURE mit einem nachfolgenden DO im Dialog-Modus. DO darf nicht angegeben werden, solange eine Transaktion offen oder eine Bildschirmausgabe noch nicht abgeschlossen ist.

Wird das Runtime-System eingesetzt, so kann ein Programm nur dann mit der Anweisung DO aufgerufen werden, wenn es als Zwischencode vorliegt.

```
DO [ bibliothek(elemname) | elemname ]

   [ USING { ausdruck | NULL }, ... ]
```

---

bibliothek — Name der DRIVE-Bibliothek (max. 54 Zeichen), aus der das DRIVE-Programm eingelesen wird.

bibliothek kann auch der Dateikettungsname der DRIVE-Bibliothek (entsprechend den BS2000-Konventionen) sein.

DRIVE/WINDOWS interpretiert *bibliothek* zuerst als Dateikettungsnamen, dann als Bibliotheksnamen.

Wird die DRIVE-Bibliothek mit der Anweisung PARAMETER DYNAMIC LIBRARY voreingestellt, so kann die Angabe von *bibliothek* entfallen.

elemname — Name des Elements (max. 31 Zeichen), das das DRIVE-Programm als Source oder als Zwischencode enthält.

DRIVE/WINDOWS sucht nach dem zuletzt bearbeiteten Element, unabhängig davon, ob dieses eine Source enthält (S-Element) oder einen Zwischencode (X-Element).

elemname kann ein Old-Style-Programm sein, das mit DRIVE V5.1 erstellt wurde. Was Sie beim Aufruf eines Old-Style-Programms beachten müssen, ist in der DRIVE-Programmiersprache [2] beschrieben.

Wird keine Angabe für *bibliothek* gemacht, wird die bei PARAMETER DYNAMIC LIBRARY angegebene Bibliothek eingesetzt.

Ist *elemname* in der angegebenen Bibliothek nicht vorhanden, wird eine Fehlermeldung ausgegeben.

Im UTM-Betrieb und im Programm-Modus muß *elemname* angegeben werden.

Wenn *elemname* nicht angegeben wird, verwendet DRIVE/WINDOWS die in der EDT-Arbeitsdatei 0 stehende Source (nur im Dialog-Modus).

USING — Mit USING werden Parameter an das zu startende Dialog- oder Folgeprogramm übergeben. Dazu müssen die Programme mit PROCEDURE... USING... definiert werden und die Parameter zuweisungsverträglich sein (siehe DRIVE-Programmiersprache [2]). Im Debug-Modus wird andernfalls ein Parameter-Prompting durchgeführt. (siehe DRIVE-Programmiersprache [2]).

ausdruck — Angabe der zu übergebenden Parameter (Sendefelder).

Im Dialog-Modus sind als Übergabeparameter erlaubt Literale, Aggregate, deren Komponenten Literale sind, und Rechenausdrücke, die keine Variablen enthalten.
Innerhalb eines Dialog-Programms können auch Werte von Variablen übergeben werden.

NULL — Übergabe des NULL-Werts an das zu startende Dialog- oder Folgeprogramm.

Anzeigen von Syntax- und Ablauffehlern

Wird ein Programm mit DO gestartet, wird bei Ablauffehlern eine Fehlerliste erzeugt und im TIAM-Betrieb nach SYSLST, im UTM-Betrieb in eine zentrale Druckdatei geschrieben.

Bei der Ausführung eines in der EDT-Arbeitsdatei 0 stehenden Elements wird im Fehlerfalle keine Fehlerliste erzeugt, sondern mit der folgenden EDT-Anweisung werden die Fehlermeldungen in das in der EDT-Arbeitsdatei 0 stehende Element eingefügt (siehe Anweisung EDT).

Eine Fehlerliste wird erzeugt, obwohl das rufende Hauptprogramm in der EDT-Arbeitsdatei 0 steht, wenn über CALL externe Unterprogramme gestartet werden und in diesen die Fehler aufgetreten sind.

Beziehungen zu anderen Anweisungen

– Innerhalb transaktionsgesicherter, als UTM-Asynchronvorgänge gestarteter Programme sind DO-Anweisungen nicht wirksam. Als Ersatz sind in diesem Fall jedoch ENTER-Anweisungen erlaubt.

– Wird bei DO weder *bibliothek* noch ein *elemname* angegeben, werden in einer Source die OPTION-Angaben LISTING=LIBRARY und CODE=ON nicht ausgeführt und kommentarlos übergangen.

– Findet bei DO eine Analysephase statt, wird die Übersetzungsoption CODE=ON, die in der Source angegeben ist, ignoriert.

  Andere Optionen werden ausgeführt, z.B. wird bei LISTING=LIBRARY die Übersetzungsliste in ein Bibliothekselement geschrieben.

– Wurde das Programm mit der Compiler-Option OPTION OBJECT=ON übersetzt, darf beim Programmaufruf *bibliothek* nicht angegeben werden.

Regeln bei Datenbankzugriff

– Die Anweisung DO nur dann ausgeführt, wenn für den Dialog-Modus keine (New-Style-)Transaktion offen ist.

– Wenn dem Dialog-Modus und dem gerufenen Unterprogramm unterschiedliche Datenbanksysteme zugeordnet sind (DBSYSTEM ≠ OFF), wird die Anweisung DO abgebrochen. Die unterschiedliche Zuordnung von Datenbanksystemen ist nur möglich, wenn mit DO Zwischencode oder Objektcode aufgerufen wird, der in einer früheren DRIVE-Sitzung mit einem anderen Datenbanksystem erzeugt wurde.

– Wenn dem Dialog-Modus ein Datenbanksystem zugeordnet ist (DBSYSTEM ≠ OFF) und dem gerufenen Unterprogramm nicht (DBSYSTEM = OFF), so greift das gerufene Unterprogramm auf dasselbe Datenbanksystem zu wie der Dialog-Modus.

– Wenn dem gerufenen Unterprogramm ein BS2000-Datenbanksystem zugeordnet ist (DBSYSTEM = UDS / SESAM / SESAMSQL), wird die Anweisung DO nur dann ausgeführt, wenn dieses Datenbanksystem der geladenen Variante entspricht.

– Wenn das gerufene Unterprogramm ein Old-Style-Programm ist, wird die Anweisung DO abgebrochen, wenn dem Dialog-Modus das Datenbanksystem UDS zugeordnet ist (DBSYSTEM = UDS).

Wenn eine Dialog-Programm die Anweisung DO enthält, beachten Sie bitte auch die Regeln für END PROCEDURE in Programmen, die mit DO aufgerufen werden (siehe Abschnitt „Regeln bei Datenbankzugriff“ auf Seite 111)

Regeln bei Verteilter Transaktionsverarbeitung

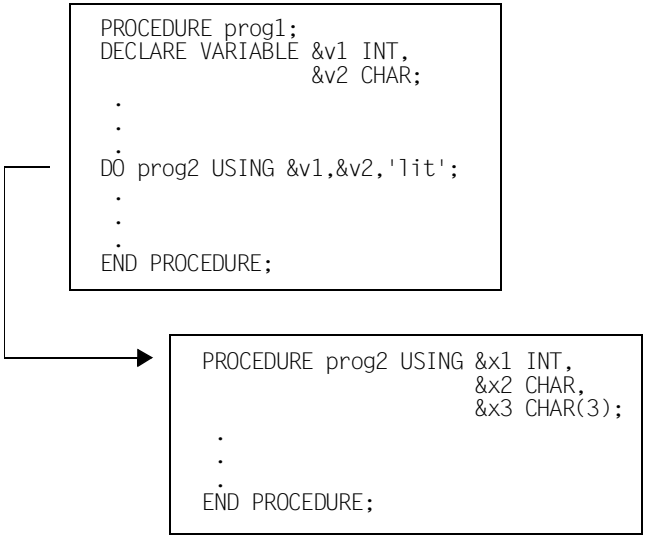
– DO in einem Dialog-Programm ist nur in Auftraggeber-Umgebung erlaubt.

– DO darf nicht in DISPATCH-Blöcken stehen.

– DO darf erst ausgeführt werden, wenn alle Auftragnehmer-Vorgänge beendet sind.

Beispiel

Das Programm "prog1" übergibt die Parameter &v1, &v2 und das Literal "lit" an Programm "prog2". Für jeden Parameter muß im Programm "prog2" ein zuweisungsverträglicher Parameter (&x1, &x2, &x3) definiert sein.



&x1 enthält den Wert von &v1.
&x2 enthält den Wert von &v2.
&x3 enthält den Wert "lit".

EDT
Editor aufrufen

Diese Anweisung ist gültig

– im TIAM-Betrieb
– im Dialog-Modus

EDT ruft den BS2000-Standard-Editor EDT auf.

Wird EDT mit Operanden angegeben, wird eine Source in die EDT-Arbeitsdatei 0 im F-Modus einlesen. Diese Source darf maximal 999 999 Zeilen mit je 256 Zeichen umfassen.

Wird EDT ohne Operanden angegeben, wird in die EDT-Arbeitsdatei 0 verzweigt. Diese Datei ist leer, wenn in der aktuellen DRIVE-Sitzung noch keine Datei in den EDT eingelesen wurde. Wurde bereits eine Datei eingelesen, steht diese in der EDT-Arbeitsdatei 0.

```
EDT [ bibliothek(elemname) | elemname ]

    [ SOURCE | LIST | COPYSOURCE | USERLABEL ]
```

---

bibliothek

Name der DRIVE-Bibliothek (max. 54 Zeichen), aus der das Element *elemname* eingelesen wird.
bibliothek kann auch der Dateikettungsname der DRIVE-Bibliothek (entsprechend den BS2000-Konventionen) sein.
DRIVE/WINDOWS interpretiert *bibliothek* zuerst als Dateikettungsnamen, dann als Bibliotheksnamen.

Wird die DRIVE-Bibliothek mit der Anweisung PARAMETER DYNAMIC LIBRARY voreingestellt, so kann die Angabe von *bibliothek* entfallen.

elemname

Name des einzulesenden Elements (max. 31 Zeichen).

Wird keine *bibliothek* angegeben, wird die bei PARAMETER DYNAMIC LIBRARY angegebene Bibliothek eingesetzt.

Wenn *elemname* nicht angegeben wird, verwendet DRIVE/WINDOWS die in der EDT-Arbeitsdatei 0 stehende Source (nur im Dialog-Modus).

SOURCE	Vorbelegung Das einzulesende Element ist eine Source (S-Element) in der DRIVE-Bibliothek.
LIST	Das einzulesende Element ist eine Übersetzungsliste (P-Element) in der DRIVE-Bibliothek. Die Angabe LIST beim Einlesen der Übersetzungsliste in den Editor ist unbedingt nötig, da sonst nach einer Source mit dem Namen der Übersetzungsliste gesucht würde. Die Übersetzungsliste sollte nicht verändert werden, da sonst die Protokollierung von Programmanweisungen nicht erfolgen kann oder Fehlermeldungshinweise, die sich auf Zeilennummern in Übersetzungslisten beziehen, nicht stimmen.
COPYSOURCE	Das einzulesende Element ist ein COPY-Element (S-Element) in der DRIVE-Bibliothek. Die Angabe COPYSOURCE beim Einlesen des COPY-Elements in den Editor ist unbedingt nötig, da sonst nach einer Source mit dem Namen des COPY-Elements gesucht würde.
USERLABEL	Das einzulesende Element ist ein Benutzerkennsatz (S-Element) in der DRIVE-Bibliothek. Die Angabe USERLABEL beim Einlesen des Benutzerkennsatzes in den Editor ist unbedingt nötig, da sonst nach einer Source mit dem Namen des Benutzerkennsatzes gesucht würde.

BS2000-Standard-Editor EDT

Regeln

Das Zeilenende ist Trenner, d.h. Schlüsselwörter, Namen und Operatoren dürfen nicht über eine Zeile hinausgehen.

Ausnahme:

alphanumerische, hexadezimale Literale, Kommentare und alle Zeichenfolgen, die in Anführungszeichen (") eingeschlossen sind.

Einschränkungen bei der Verwendung von EDT-Funktionen

– Folgende EDT-Anweisungen sind nicht erlaubt:

  @EDIT
  @EXEC
  @LOAD
  @RUN
  @SYSTEM

– Die EDT-Marken 1-4 sind frei verwendbar. Dagegen dürfen die EDT-Marken 5-9 nicht gesetzt werden. Sie sind für DRIVE-internen Gebrauch reserviert.

– Die EDT-Arbeitsdatei 0 sowie die EDT-Arbeitsdateien 1-8 sind frei verwendbar. Die EDT-Arbeitsdatei 9 ist für DRIVE/WINDOWS reserviert. Dorthin wird nach DO und COMPILE ohne die Angabe von *bibliothek(elemname)* die Übersetzungsliste geschrieben.

Prompting zum Schutz vor ungewolltem Überschreiben

Inhalte der EDT-Arbeitsdatei 0, die geändert und noch nicht mit SAVE gespeichert wurden, schützt DRIVE/WINDOWS vor ungewolltem Überschreiben. Soll z.B. mit EDT ein neues Element in die EDT-Arbeitsdatei 0 eingelesen werden, gibt DRIVE/WINDOWS die Meldung `DRI0046 <elemname> UEBERSCHREIBEN? ANTWORT: (Y=JA, N=NEIN)` aus.

Bei Antwort "Y" wird das Element der EDT-Arbeitsdatei 0 mit dem neuen Element überschrieben.

Bei Antwort "N" bleibt das alte Element erhalten und es wird nicht in den EDT verzweigt. Die Source kann mit der Anweisung SAVE gesichert oder mit der EDT-Anweisung (ohne Angaben von Operanden) angesehen werden.

Fehler in DRIVE-Programmen anzeigen (max. 4000 Fehler)

– DO und COMPILE auf in der EDT-Arbeitsdatei 0 stehende Programme:

  Findet DRIVE/WINDOWS bei der Analyse Syntaxfehler, werden bei der folgenden EDT-Anweisung die Fehlermeldungen in das in der EDT-Arbeitsdatei 0 stehende Programm eingefügt.
  Zusätzlich steht in der EDT-Arbeitsdatei 9 die vollständige Übersetzungsliste.
  Die EDT-Arbeitsdatei 9 sollte der Anwender nicht verwenden, da der Inhalt von 9 kommentarlos überschrieben wird.
  Die Zeilen sollten maximal 230 Zeichen lang sein. Längere Zeilen können in der Übersetzungsliste nicht komplett angezeigt werden.

– Wird nach der Analyse eines fehlerhaften DRIVE-Programms in den EDT verzweigt, positioniert DRIVE/WINDOWS die EDT-Arbeitsdatei 0 auf die Zeile mit dem ersten Fehler. Außerdem markiert DRIVE/WINDOWS alle fehlerhaften Programmzeilen mit der EDT-Marke 5. Mit der EDT-Anweisung +(5) und Drücken der Taste F3 kann auf die jeweils nächste fehlerhafte Programmzeile positioniert werden.

– Hinter jede fehlerhafte Programmzeile fügt DRIVE/WINDOWS eine Zeile ein, in der die Fehlerpositionen jeweils durch einen Stern (*) gekennzeichnet werden. DRIVE/WINDOWS markiert diese Zeile mit der EDT-Marke 13.
Hinter jede Zeile mit Fehlerpositionen fügt DRIVE/WINDOWS weitere Zeilen ein. Diese Zeilen enthalten Fehlerhinweise zu den jeweiligen Programmfehlern. DRIVE/WINDOWS markiert diese Zeilen mit der EDT-Marke 13.

– Es werden maximal 48 Fehlermeldungen pro Programmzeile angezeigt.

– Wird mit der EDT-Anweisung HALT oder RETURN in den DRIVE-Dialog-Modus zurückgeschaltet, so werden jene Zeilen mit der Marke 13 gelöscht, die nicht geändert wurden. (D.h. alle Zeilen mit Fehlerpositionen und -hinweisen, die nicht mit gleichem oder anderem Inhalt überschrieben wurden, werden gelöscht.) Alle EDT-Marken werden zurückgesetzt.

  Wird dagegen mit der Taste K1 zurückgeschaltet, bleiben die Zeilen mit Fehlerpositionen und -hinweisen erhalten. Die EDT-Marken 5 und 13 bleiben gesetzt. Bei einem erneuten EDT-Aufruf werden die Fehler wieder angezeigt.

  Wird die EDT-Anweisung DELETE MARK angegeben, werden auch die von DRIVE/WINDOWS gesetzten Marken gelöscht.

– Verhalten bei einem Fehler in einem COPY-Element.

  Die Fehlermeldung wird in die Source hinter dem entsprechenden COPY-Element eingefügt. Anstelle der Sternzeile erscheint eine Meldung. Sie besagt, daß der Fehler in einem COPY-Element aufgetreten ist und in welcher Zeile des COPY-Elements dieser Fehler zu finden ist.

  In der Übersetzungsliste in der EDT-Arbeitsdatei 9 steht das expandierte COPY-Element und die genaue Fehlerzuordnung.

Beziehungen zu anderen DRIVE-Anweisungen

Ein DRIVE-Programm in der EDT-Arbeitsdatei 0 kann mit SAVE abgespeichert werden. Mit SAVE wird der aktuelle Inhalt der EDT-Arbeitsdatei 0 in der aktuellen DRIVE-Bibliothek gespeichert.

END
Ende des logischen Programmteils kennzeichnen

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

END kennzeichnet das Ende von logischen Programmteilen. Unter Programmteilen versteht man: Verzweigungen (CASE), Schleifen (CYCLE), Bedingungen (IF), parallele remote Verarbeitung (DISPATCH), Reporterstellung (REPORT),interne Unterprogramme (SUBPROCEDURE) und komplette Programme (PROCEDURE).

```
END { CASE | CYCLE | DISPATCH | IF | PROCEDURE | REPORT | SUBPROCEDURE }
```

CASE	Ende einer Verzweigung.
CYCLE	Ende einer Schleife.
DISPATCH	Ende eines DISPATCH-Blocks. Alle CALL-Anweisungen im DISPATCH-Block, die Unterprogramme in einem entfernten System aufrufen (Remote-CALL-Anweisungen), werden ausgeführt. Das rufende Programm wird solange unterbrochen bis alle remote Aufträge bearbeitet sind.
IF	Ende einer Bedingung.
PROCEDURE	Ende eines Programms. END PROCEDURE muß die letzte Anweisung des Programms sein und darf nur einmal in einem Programm vorkommen. Das Programm wird ordnungsgemäß abgeschlossen. DRIVE/WINDOWS gibt die Meldung `DRI0088 'programmname' ORDNUNGSGEMAESS BEENDET` aus. Wurde das Programm mit CALL aufgerufen, wird in das rufende Programm zurückgesprungen und mit der Anweisung fortgesetzt, die dem CALL-Aufruf folgt. Die RETURN-Parameterwerte werden an das rufende Programm zurückgegeben. Bei END PROCEDURE eines mit CALL gerufenen Programms bleiben offene Transaktionen weiterhin offen und mit der Anweisung OPEN FILE geöffnete Dateien weiterhin geöffnet.

	Wurde das Programm mit DO aufgerufen, geht DRIVE/WINDOWS in den Dialog-Modus über. Alle benötigten Betriebsmittel werden freigegeben. Ist eine Transaktion offen, wird das Programm mit einer Fehlermeldung abgebrochen und die Transaktion wird zurückgesetzt. Mit der Anweisung OPEN FILE geöffnete Dateien werden geschlossen. Ein Wiederanlauf des Programms ist nicht möglich.
REPORT	Ende einer Reporterstellung.
SUBPROCEDURE	Ende eines internen Unterprogramms. END SUBPROCEDURE ist immer die letzte Anweisung eines Unterprogramms. Das rufende Programm wird unmittelbar hinter dem zum internen Unterprogramm gehörenden CALL-Aufruf fortgesetzt. Innerhalb eines internen Unterprogramms sind offene Schleifen (CYCLE ohne END CYCLE), offene Bedingungen (IF ohne END IF), offene DISPATCH-Blöcke (DISPATCH ohne END DISPATCH) und offene Verzweigungen (CASE ohne END CASE) nicht erlaubt.

Bei der Fehleranalyse kann ein fehlerhaftes END zu etlichen Folgefehlern führen, weil auch nachfolgende, möglicherweise korrekte END-Anweisungen als fehlerhaft oder fehlend gemeldet werden (z.B. wenn eine IF-Struktur innerhalb einer Schleife nicht durch END IF vor dem END CYCLE abgeschlossen wurde).

Beziehungen zu anderen DRIVE-Anweisungen (gilt nur für Hauptprogramme)

– Ausgabebereiche für Formate, die mit FILL FORM/LIST aufgebaut werden, müssen vor Erreichen von END PROCEDURE mit DISPLAY FORM/LIST abgeschlossen sein.

Beziehungen zu anderen DRIVE-Anweisungen (gilt nur für Unterprogramme, die mit CALL aufgerufen werden)

– In Abhängigkeit davon, ob Ausgabebereiche für Formate als PERMANENT oder TEMPORARY deklariert wurden, müssen sie vor Erreichen von END PROCEDURE mit DISPLAY FORM/LIST abgeschlossen sein.

Regeln bei Verteilter Transaktionsverarbeitung

– Für Programme, die mit DO in Auftraggeber-Umgebung gestartet wurden, gilt: Bei END PROCEDURE müssen alle innerhalb des Dialog-Programms gestarteten Auftragnehmer-Vorgänge beendet sein.

– Bei END PROCEDURE eines DRIVE-Programms in Auftragnehmer-Umgebung, das vom Auftraggeber-Vorgang direkt aufgerufen wurde, wird zum Auftraggeber-Vorgang zurückgesprungen. RETURN-Parameter werden an den Auftraggeber-Vorgang zurückgegeben.

Regeln bei Datenbankzugriff

Für END PROCEDURE in Unterprogrammen, die mit CALL im lokalen System aufgerufen werden, gilt:

– Wenn dem gerufenen Unterprogramm ein Datenbanksystem zugeordnet ist (DBSYSTEM ≠ OFF) und dem rufenden Programm nicht (DBSYSTEM = OFF), so greift nach Ablauf des gerufenen Unterprogramms das rufende Programm auf dasselbe Datenbanksystem zu wie das gerufene Unterprogramm.

Für END PROCEDURE in Programmen, die mit DO oder DEBUG aufgerufen werden gilt:

– Ist noch eine Transaktion offen, setzt DRIVE/WINDOWS sie zurück und gibt die Meldung DRI0101 aus.

– Sind temporäre, im Programm-Modus definierte SQL-Objekte (Programm-Cursor oder temporäre Views) vorhanden, werden sie von DRIVE/WINDOWS gelöscht.
Wenn ein SQL-Objekt nicht gelöscht werden kann, gibt DRIVE/WINDOWS die Meldung DRI0150 aus.

– Sind beim Zugriff auf SESAM V2.x dynamische, temporäre Views vorhanden, löscht DRIVE/WINDOWS sie und gibt die Meldung DRI0488 aus.

– Beim Zugriff auf SESAM V2.x setzt DRIVE/WINDOWS eine SET SESSION-, eine SET CATALOG- und eine SET SCHEMA-Anweisung ab, deren jeweiliger Operand durch die letzte vorausgegangene PARAMETER DYNAMIC AUTHORIZATION-, PARAMETER DYNAMIC CATALOG- und PARAMETER DYNAMIC SCHEMA-Anweisung bestimmt ist.

ENTER
Programm als UTM-Asynchronvorgang starten

Diese Anweisung ist gültig

– im UTM-Betrieb
– im Dialog- und Programm-Modus

ENTER stößt im UTM-Betrieb die asynchrone Ausführung eines Programms oder eines anwendereigenen Teilprogramms in lokalen oder entfernten UTM-Anwendungen an. ENTER-Anweisungen, die Programme in entfernten Systemen aufrufen, werden Remote-ENTER-Anweisungen genannt.

Wird ein DRIVE-Programm mit ENTER aufgerufen, so führt DRIVE/WINDOWS zunächst implizit ein COMPILE durch und startet anschließend das Programm.

Wurde für dieses Programm bereits ein Zwischencode erzeugt und in der aktuellen DRIVE-Bibliothek gespeichert, greift ENTER auf diesen Zwischencode direkt zu und startet nur noch das Programm. Die Syntax- und Semantiküberprüfung entfällt in diesem Fall. DRIVE/WINDOWS sucht nach dem zuletzt bearbeiteten Programm, unabhängig davon, ob dieses als Source (S-Element) oder als Zwischencode (X-Element) vorliegt.

Ist ENTER eine Anweisung innerhalb eines mit ENTER gestarteten Programms , wird ein weiteres Programm als UTM-Asynchronvorgang gestartet. Das rufende Programm läuft normal weiter.

Für den Aufruf von DRIVE-Programmen gilt folgendes:

– Eigenschaften, die für das startende Programm festgelegt wurden (z.B. über PARAMETER), werden bei lokaler Ausführung vom Folgeprogrammen übernommen, bei entfernter Ausführung nicht.

– Treten innerhalb des mit ENTER gestarteten Programms Fehler auf, wird das Programm abgebrochen. Eine Fehlerliste und die Meldung `DRI0087 programmname FEHLERHAFT ABGEBROCHEN` werden in die zentrale Druckdatei geschrieben (siehe DRIVE-Programmiersystem [1]).

Zusätzlich können über PERMIT eine Benutzeridentifikation angegeben und über SET TRANSACTION Transaktionsbedingungen vereinbart werden.

```
ENTER { bibliothek(elemname) |
        elemname |
        [ COBOL | C ] TAC tacname }

      [ USING { ausdruck | NULL [ INDICATOR ] }, ... ]
      [ PERMIT ]
      [ SET TRANSACTION ]
```

bibliothek
: Name der DRIVE-Bibliothek (max. 54 Zeichen), in der das DRIVE-Programm eingelesen wird.

  bibliothek kann auch der Dateikettungsname der DRIVE-Bibliothek (entsprechend den BS2000-Konventionen) sein.

  DRIVE/WINDOWS interpretiert *bibliothek* zuerst als Dateikettungsnamen, dann als Bibliotheksnamen.

  Wird die DRIVE-Bibliothek mit der Anweisung PARAMETER DYNAMIC LIBRARY voreingestellt, so kann die Angabe von *bibliothek* entfallen.

elemname
: Name des Elements (max. 31 Zeichen), das das DRIVE-Programm als Source oder als Zwischencode enthält.

  DRIVE/WINDOWS sucht nach dem zuletzt bearbeiteten Element, unabhängig davon, ob dieses eine Source enthält (S-Element) oder einen Zwischencode (X-Element).

  Im Dialog-Modus wird überprüft, ob mit der Anweisung PARAMETER DISTRIBUTION eine Verteilungsinformation festgelegt wurde. Im Programm-Modus wird nur dann die Verteilungsinformation überprüft, wenn die Übersetzungsoption OPTION DISTRIBUTION=ON definiert wurde.

  Gemäß der Verteilungsinformation wird ein lokales oder ein entferntes Programm aufgerufen. Ist mit der Anweisung PARAMETER DISTRIBUTION keine Verteilungsinformation definiert, wird das Programm lokal gesucht.

  Wird keine Angabe für *bibliothek* gemacht, wird die für das (lokale oder entfernte) System bei PARAMETER DYNAMIC LIBRARY angegebene Bibliothek eingesetzt.

COBOL	Vorbelegung Sprachspezifische Steuerung des Programmaufrufs für ein in der Programmiersprache COBOL geschriebenes Programm.
C	Sprachspezifische Steuerung des Programmaufrufs für ein in der Programmiersprache C geschriebenes Programm.
TAC	Bei der Angabe von ENTER TAC sind *permit* und *set transaction* (siehe unten) nicht erlaubt. Im Dialog-Modus wird überprüft, ob mit der Anweisung PARAMETER DISTRIBUTION eine Verteilungsinformation festgelegt wurde. Im Programm-Modus wird nur dann die Verteilungsinformation überprüft, wenn die Übersetzungsoption OPTION DISTRIBUTION=ON definiert wurde. Gemäß der Verteilungsinformation wird ein lokales oder ein entferntes Programm aufgerufen. Ist mit der Anweisung PARAMETER DISTRIBUTION keine Verteilungsinformation definiert, wird das Programm lokal gesucht. Es dürfen maximal soviele Variablen übergeben werden, daß die Länge der Gesamtdatenwerte und -beschreibungen 31 Kbyte nicht überschreitet.
tacname	Name des Transaktionscodes (max. 8 Zeichen) eines anwendereigenen UTM-Teilprogramms. Das UTM-Teilprogramm kann Teil der lokalen UTM-Anwendung sein oder Teil einer UTM-Anwendung in einem entfernten System (remote). Die UTM-Anwendung kann unter SINIX oder unter BS2000 laufen. *tacname* darf nicht das Präfix `dri`, `drt`, `drc` oder `sql` haben. Diese Namen sind von DRIVE/WINDOWS reserviert. Beim Aufruf eines anwendereigenen UTM-Teilprogramms muß die Programmiersprache, in der es geschrieben wurde, angegeben werden.
USING	Mit USING werden Parameter des rufenden Programms an das aufgerufene Programm übergeben. Die Parameter müssen zuweisungsverträglich sein (siehe DRIVE-Programmiersprache [2]). Ist das aufgerufene Programm ein DRIVE-Programm, muß es mit PROCEDURE...USING... definiert worden sein. Ist das aufgerufene Programm ein UTM-Teilprogramm, werden die übergebenen Parameter im Nachrichtenbereich hinterlegt. Die UTM-Anweisung MGET fordert die übergebenen Daten von UTM für das Teilprogramm an.

Im Dialog-Modus darf beim Aufruf eines UTM-Teilprogramms (ENTER TAC) die USING-Klausel nicht verwendet werden.

ausdruck

Angabe der zu übergebenden Parameter (Sendefelder).

Im Programm-Modus können an externe Unterprogrammen in DRIVE (ENTER *bibliothek(elemname)* ..., ENTER *elemname* ...) Variablen, Vektoren, Matrizen, Aggregate, Literale oder Rechenausdrücke übergeben werden.
An externe DRIVE-Unterprogramme im entfernten System können einfache Variablen, Vektoren, Matrizen, Literale oder Rechenausdrücke übergeben werden.

Im Dialog-Modus können an externe Unterprogramme in DRIVE Literale, Aggregate, deren Komponenten Literale sind, und Rechenausdrücke, die keine Variablen enthalten, übergeben werden.

An UTM-Teilprogramme (ENTER TAC) können einfache Variablen, Vektoren, Matrizen oder Komponenten von strukturierten Variablen übergeben werden.

ausdruck muß vom Typ *variable* sein.

Außerdem darf die Gesamtlänge der übergebenen Datenwerte und Datenbeschreibungen 31 Kbyte nicht überschreiten.

Hat bei ENTER TAC *ausdruck* den NULL-Wert, muß die INDICATOR-Klausel angegeben sein. Ansonsten wird das Programm abgebrochen.

NULL

Übergabe des NULL-Werts an das zu startende Programm.

INDICATOR

Mit INDICATOR wird eine Indikatorvariable angelegt. Der Wert der Indikatorvariablen gibt an, ob der Übergabeparameter den Wert NULL oder einen definierten Wert enthält. Die Angabe INDICATOR ist nur bei ENTER TAC erlaubt.

PERMIT

PERMIT-Anweisung. Siehe Anweisung PERMIT in den SQL-Lexika [4], [5] und [6].

Die Anweisung PERMIT wird bei Aufrufen von DRIVE-Programmen im lokalen System ausgewertet und bei Aufrufen von DRIVE-Programmen im entfernten System ignoriert.

Die Anweisung PERMIT ist beim Aufruf von UTM-Teilprogrammen nicht erlaubt.

SET TRANSACTION

SET TRANSACTION-Anweisung. Siehe Anweisung SET TRANSACTION in den SQL-Lexika [4], [5] und [6].

Die Anweisung SET TRANSACTION wird bei Aufrufen von DRIVE-Programmen im lokalen System ausgewertet und bei Aufrufen von DRIVE-Programmen im entfernten System ignoriert.

Die Anweisung SET TRANSACTION ist beim Aufruf von UTM-Teilprogrammen nicht erlaubt.

Ausführungszeitpunkt

Die maximale Anzahl parallel ablaufender UTM-Asynchronvorgänge ist durch die UTM-Generierung festgelegt (siehe DRIVE-Programmiersystem [1]).

Es ist daher möglich, daß das mit ENTER gestartete Programm nicht sofort ausgeführt wird.

Die Reihenfolge, nach der als UTM-Asynchronvorgänge gestartete Programme abgearbeitet werden, ist nicht immer identisch mit der Reihenfolge, in der die ENTER-Anweisungen durchlaufen werden.

Beziehungen zu anderen Anweisungen

– Innerhalb transaktionsgesicherter, als UTM-Asynchronvorgänge gestarteter Programme sind DO-Anweisungen nicht wirksam. Als Ersatz sind in diesem Fall jedoch ENTER-Anweisungen erlaubt.

– Ein als UTM-Asynchronvorgang gestartetes Programm darf keine CALL-Anweisungen enthalten, die ein UTM-Teilprogramm aufrufen (CALL TAC).

– Ein als UTM-Asynchronvorgang gestartetes Programm darf keine CALL-Anweisungen enthalten, die ein Old-Style-Programm aufrufen.

– Ein als UTM-Asynchronvorgang gestartetes Programm darf keine Remote-CALL-Anweisungen enthalten.

– Beim Aufruf eines Programms im lokalen System werden alle Angaben zur Parametrisierung des als UTM-Asynchronvorgang gestarteten Programms von der PARAMETER-Anweisung des aufrufenden Vorgangs übernommen.

– Wurde das Programm mit der Compiler-Option OPTION OBJECT=ON übersetzt, darf beim Programmaufruf *bibliothek* nicht angegeben werden.

EXECUTE
Anweisung dynamisch generieren und ausführen

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

Zum Ablaufzeitpunkt generiert EXECUTE genau eine Anweisung, analysiert sie und führt sie sofort aus.

Der dynamisch erzeugte Anweisungsstring muß eine syntaktisch korrekte DRIVE- oder SQL-Anweisung sein. Diese Anweisung heißt dann "dynamische" DRIVE- oder SQL-Anweisung.

Abweichend von der Regel, daß DRIVE-Anweisungen mit einem Semikolon abgeschlossen werden, brauchen Anweisungen in einer EXECUTE-Anweisung nicht mit einem Semikolon (;) abgeschlossen werden.
Sie können z.B. angeben: EXECUTE 'OPEN c1;'; oder EXECUTE 'OPEN c1';.
Werden zwei durch Semikolon getrennte Anweisungen angegeben, wird nur die erste dynamisch ausgeführt; die zweite wird ignoriert.

Soll in einer Anweisung eine deklarative Größe verwendet werden, muß sie im Programmablauf vorher definiert worden sein: entweder statisch im Deklarationsteil oder dynamisch in einem vorhergehenden EXECUTE. Für deklarative Größen, die erst dynamisch mit EXECUTE definiert werden, gilt: Beim statischen Aufbau eines Programms kann bei den nicht-deklarativen Anweisungen nicht auf dynamisch deklarierte Größen zugegriffen werden.

Werden in einem Programm temporäre Cursor und Views dynamisch deklariert, sind sie bis zum Ende des Programms oder bis zum explizit angegebenen dynamischen DROP gültig. Permanente Cursor sind bis zum Ende des Programm-Modus gültig.

Beim Zugriff auf das Datenbanksystem SESAM V2.x werden bei der Ausführung von dynamischen SQL-Anweisungen vorausgegangene SET SESSION-, SET CATALOG- und SET SCHEMA-Anweisungen ausgewertet. Falls diese fehlen, greift DRIVE/WINDOWS auf entsprechende PARAMETER DYNAMIC-Anweisungen (Operanden AUTHORIZATION, CATALOG, SCHEMA) zurück.

EXECUTE charausdruck

---

charausdruck

Das Ergebnis von *charausdruck* muß eine syntaktisch korrekte DRIVE-Anweisung sein (siehe Metavariable *charausdruck*).

Folgende Anweisungen können dynamisch ausgeführt werden:

SQL-Anweisungen	DRIVE-Anweisungen
ALTER TABLE	CALL
CLOSE cursor	DECLARE FORM
COMMIT WORK	DECLARE LIST
CREATE SCHEMA	DISPLAY
CREATE TABLE	DO
CREATE TEMPORARY VIEW	ENTER
CREATE VIEW	FILL
DECLARE ... CURSOR ...	LIST
DELETE	SEND MESSAGE
DROP CURSOR	SET
DROP CURSORS	UNSAVE
DROP SCHEMA	
DROP TABLE	
DROP VIEW	
DROP VIEWS	
FETCH	
GRANT	
INSERT	
OPEN	
PERMIT	
PRAGMA	
RESTORE	
REVOKE	
ROLLBACK WORK	
SELECT	
SET CATALOG	
SET SCHEMA	
SET SESSION	
SET TRANSACTION	
STORE	
UPDATE	

Wenn *charausdruck* ein Literal ist, muß die vollständige Anweisung (mit dem abschließenden Semikolon) in Hochkommas (') gesetzt werden.

Wenn in *charausdruck* Literale vorkommen, so dürfen diese max. 256 Zeichen lang sein.

Wird der DRIVE-Compiler DRIVE/WINDOWS-COMP eingesetzt, dürfen die Anweisungen CALL, DO oder ENTER nicht angegeben werden.

Im Fehlerfall wird eine Ablaufmeldung ausgegeben.
Im UTM-Betrieb wird der Vorgang abgebrochen.
Im TIAM-Betrieb wird das Programm abgebrochen.

In einer EXECUTE-Anweisung wird bei der Anweisung "CALL ..." ohne Angabe einer Bibliothek ein externes DRIVE-Unterprogramm aufgerufen.

Beispiel 1

Der Inhalt einer Cursortabelle "c1" wird erst zum Ablaufzeitpunkt bestimmt und kann variabel gestaltet werden:
Der *select-ausdruck* innerhalb der Cursor-Deklaration steht in der Variablen &s. Dieser Variablen können während des Programmablaufes unterschiedliche Werte zugewiesen werden. Durch eine dynamische Anweisungsausführung (EXECUTE) wird die Cursor-Deklaration zum Ablaufzeitpunkt generiert, analysiert und sofort ausgeführt. Analog dazu das Öffnen des Cursors und das Versorgen von Variablen.

Die Anweisungen nach EXECUTE müssen in Hochkommas geschrieben werden.

```
...
/* Deklaration der Variablen */
DECLARE VARIABLE &s CHAR (250);
DECLARE VARIABLE &c CHAR (20) INIT 'CLOSE c1';
DECLARE VARIABLE 1 &e,
                   2 e1 CHAR (20) INIT 'FETCH c1 INTO ',
                   2 e2 CHAR (20) INIT '&f1,&f2,&f3';

/* Ausfuehrungsteil */
...
SET &s='SELECT * FROM t1 WHERE a=0';
EXECUTE 'DECLARE c1 CURSOR FOR ' || &s;
EXECUTE 'OPEN c1';
EXECUTE CHARACTER (&e);
EXECUTE &c;
```

Beispiel 2

Zum Ablaufzeitpunkt werden zwei verkettete Variablen (CONCAT (&abfragebefehl,&abfragetext)) und eine einfache Variable (&fehlermeldung) dynamisch ausgeführt. Den Variablen wurden zuvor Werte zugewiesen.

```
DECLARE VARIABLE
   &abfragebefehl CHAR(15) INIT 'DISPLAY FORM',
   &abfragetext CHAR(74) INIT
      'NL 15,TAB 5,"Geben Sie die laufende Nummer ein: "RETURN &nummer;',
   &fehlermeldung CHAR(74) INIT
      'SEND MESSAGE "Der Datensatz ist nicht vorhanden. DUE-Taste";';
...
EXECUTE CONCAT (&abfragebefehl,&abfragetext);
...
EXECUTE &fehlermeldung;
```

Die folgenden Variablendeklarationen sind Grundlage für die Beispiele 3 bis 6.

```
DECLARE VARIABLE
    1 &e,
       2 table_n   CHAR (19) INIT 'T1',
       2 set_s     CHAR (04) INIT 'SET',
       2 set_c     CHAR (200),
       2 w,
         3 w1      CHAR (10) INIT 'WHERE',
         3 where_c CHAR (200);
DECLARE VARIABLE &opc  CHAR (30) INIT 'DELETE FROM UPDATE';
DECLARE VARIABLE &v    CHAR (17) INIT 'V1';
DISPLAY FORM LINE
        RETURN &table_n,
        RETURN &set_c INIT 'F1=100',
        RETURN &where_c INIT 'F2=2';
```

Beispiel 3

Das Öffnen des Cursors "c1" wird dynamisch ausgeführt.

Die Anweisung muß in Hochkommas geschrieben werden.

```
...
EXECUTE 'OPEN c1';
...
```

Beispiel 4

Die Anweisung "UPDATE T1 SET F1=100 WHERE F2=2;" wird mit Hilfe einer Verkettung (CONCAT) und Teilverkettung (SUBSTRING) dynamisch ausgeführt. Für die Teilkette (SUBSTRING) ist die Position 13 und die Länge 6 angegeben.

```
...
EXECUTE CONCAT (SUBSTRING (&opc,13,6) CHAR(&e));
...
```

Beispiel 5

Die Anweisung "DELETE FROM T1 WHERE F2=2;" wird mit Hilfe des Verkettungszeichens "||" und einer Teilverkettung (SUBSTRING) dynamisch ausgeführt.

```
...
EXECUTE SUBSTRING (&opc,1,12) || &table_n || CHARACTER (&w);
...
```

Beispiel 6

Die Anweisung "DECLARE c1 CURSOR FOR SELECT * FROM V1 WHERE F2=2;" wird dynamisch ausgeführt.

```
...
EXECUTE 'DECLARE c1 CURSOR FOR SELECT * FROM ' || UPDSTRING (&table_n,&v,3) ||
CHARACTER (&w);
...
```

EXIT
DRIVE-Sitzung beenden

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Dialog-Modus
– im Programm-Modus nur als Bildschirmeingabe in einem Programm

EXIT beendet die DRIVE-Sitzung. Eine evtl. noch offene Transaktion wird ohne Fehlermeldung zurückgesetzt. Der Inhalt der EDT-Arbeitsdatei 0 wird nicht gesichert.

EXIT hat dieselbe Wirkung wie eine K-/F-Taste, die mit ACTION=EXIT belegt wurde.

Im TIAM-Betrieb werden alle angeforderten Dateien geschlossen, alle Views sowie alle DRIVE-spezifischen Speicherbereiche werden freigegeben.

Im UTM-Betrieb werden alle Listen des Vorgangs ausgedruckt und in der zentralen Druckdatei gelöscht, wenn sie nicht explizit mit LIST * ... DELETE ausgedruckt wurden. Anschließend muß ein Transaktionscode (TAC) oder KDCOFF eingegeben werden.

```
EXIT
```

FILL formatname
DRIVE-Bildschirmformat aufbauen und füllen

Diese Anweisung ist gültig

– im TIAM-Betrieb
– im UTM-Betrieb, aber nicht im UTM-Asynchronbetrieb und bei VTV nicht in Auftragnehmer-Umgebung
– im Programm-Modus

FILL *formatname* definiert den Inhalt und das Layout von Bildschirmformaten. Die Anweisung füllt den mit DECLARE FORM definierten Speicherbereich des DRIVE-Formats.

Hat die Anweisung einen Bildschirm-Überlauf zur Folge, wird implizit ein DISPLAY ausgeführt.

Eingabefelder dürfen nur innerhalb einer Bildschirmseite verwendet werden, ansonsten meldet DRIVE/WINDOWS einen Fehler.

Eingabefelder werden an der Datensichtstation hell, Ausgabefelder halbhell dargestellt.

Soll eine Gruppe (= Datengruppe oder Wiederholungsgruppe) ausgegeben werden, werden immer die Namen der einfachsten Komponenten (= unterste Ebene) ausgegeben. Voraussetzung dafür ist, daß die Angabe *format* die Namensausgabe (NAMES) vorsieht. Wird für *format* LINE angegeben, werden auch die Strukturnamen ausgegeben.

Datenwerte, die über das Zeilenende hinausgehen, werden zu Beginn der folgenden Zeile fortgesetzt.

```
FILL formatname [ format ]

        { [ RETURN ] ausdruck [ INIT ausdruck1 [ NOCHECK ] ]
            [ ATTRIBUTE ( attribute, ... ) ] [ mask ] |
          NEWLINE n |
          NEWPAGE n |
          TABULATOR n |
          BLANK n }, ...
```

---

formatname — Name des DRIVE-Bildschirmformats (max. 31 Zeichen). Es muß im Deklarationsteil des Programms mit DECLARE FORM definiert sein.

format

Festlegen des Bildschirmformats für den FILL-Bereich (siehe Metavariable *format*).

Für *format*=TABLE gilt:

– NEWPAGE und NEWLINE dürfen nicht angegeben werden.

– Gehen Datenwerte über das Zeilenende hinaus, wird das Programm abgebrochen. Wird TABLE nicht vereinbart, werden diese Datenwerte zu Beginn der folgenden Zeile fortgesetzt.

Für *format*=NAMES gilt:

– RETURN darf nicht angegeben werden.

– Bei auszugebenden Literalen wird das Literal als Name ausgegeben.

Für *format*=LINE gilt:

– Auf die Angaben TABULATOR und BLANK erfolgt ein Zeilenvorschub mit Leerzeile.

RETURN

Mit RETURN wird festgelegt, daß eine Variable zum Eingabefeld wird, das vorbelegt werden kann. Eine Variable, die mit RETURN gekennzeichnet ist, kann pro Format nur einmal als Eingabefeld verwendet werden.

Paßt eine Eingabevariable (mit RETURN gekennzeichnet) nicht auf eine Bildschirmseite, so erfolgt zum Ausführungszeitpunkt Prozedurabbruch.

Paßt eine Ausgabevariable (ohne RETURN gekennzeichnet) nicht auf eine Bildschirmseite, so wird sie zur Ausgabe abgeschnitten. Die letzten drei Zeichen der Ausgabevariable werden mit ">>>" gekennzeichnet.

ausdruck

Definieren von Ausgabe- und/oder Eingabefeldern für das Bildschirmformat. *ausdruck* kann sein: Eine oder mehrere Variablen (auch Systemvariablen) und/oder ein oder mehrere Literale.

Wenn RETURN, INIT oder *mask* angegeben wird, muß *ausdruck* eine Variable sein, die nicht mit ".*" qualifiziert werden darf.

INIT

ausdruck wird ein Anfangswert zugewiesen. Die INIT-Klausel ist nur zulässig, wenn *ausdruck* eine Variable ist.
Ist *ausdruck* ein Vektor oder eine Matrix, erhalten alle Komponenten den entsprechenden Anfangswert *literal* oder NULL.

ausdruck1	*ausdruck1* darf nur *literal*, NULL oder Funktion sein, deren Argumente Literale (aber nicht CURRENT DATE/TIME/TIMESTAMP) sind. *ausdruck1* muß zum Übersetzungszeitpunkt berechenbar sein.
NOCHECK	Eine bei der Deklaration der Variablen *ausdruck* angegebene CHECK-Klausel (siehe Metavariable *check*) wird für die Anfangswertzuweisung nicht ausgewertet. NOCHECK ist nicht erlaubt bei redefinierten Variablen oder einer Variablen, die eine andere redefiniert.
ATTRIBUTE	Bildschirmformaten werden Feldeigenschaften zugewiesen.
attribute	Feldeigenschaft (siehe Metavariable *attribute*)
mask	Definieren der Darstellungsmöglichkeiten für maskierte Ein- und Ausgaben (= Ausgabeaufbereitung). Die Angabe *mask* ist nur erlaubt, wenn *ausdruck* eine einfache Variable oder eine einfache Komponente ist.
NEWLINE n	Mit NEWLINE wird ein Vorschub von *n* Zeilen durchgeführt. Mit NEWLINE 1 wird die aktuelle Zeile abgeschlossen. Falls Daten folgen, werden sie in die nächste Zeile geschrieben. Auch wenn die aktuelle Zeile schon ganz gefüllt ist, also schon soviele Zeichen enthält, wie in der COLUMNS-Angabe des entsprechenden DECLARE vereinbart wurden, bewirkt NEWLINE 1 nicht die Ausgabe einer Leerzeile, sondern nur eine Positionierung auf die nächste Zeile. Hat *n* den Wert "0", erfolgt ein bedingter Vorschub, d.h. wenn die aktuelle Zeile leer ist, wird NEWLINE 0 ignoriert, wenn die aktuelle Zeile nicht leer ist, wirkt NEWLINE 0 wie NEWLINE 1.
NEWPAGE n	Mit NEWPAGE wird ein Vorschub von *n* Seiten durchgeführt. Mit NEWPAGE 1 wird die aktuelle Seite abgeschlossen. Falls Daten folgen, werden sie auf die nächste Seite geschrieben. Auch wenn die aktuelle Seite schon ganz gefüllt ist, also schon soviele Zeilen enthält, wie in der LINES-Angabe des entsprechenden DECLARE vereinbart wurden, bewirkt NEWPAGE 1 nicht die Ausgabe einer Leerseite, sondern nur eine Positionierung auf die nächste Seite. Hat *n* den Wert "0", erfolgt ein bedingter Vorschub, d.h. wenn die aktuelle Seite leer ist, wird NEWPAGE 0 ignoriert, wenn die aktuelle Seite nicht leer ist, wirkt NEWPAGE 0 wie NEWPAGE 1.
TABULATOR n	Die Ausgabe wird ab Spalte *n* fortgeführt. Ist der Wert kleiner als die aktuelle Spaltenposition, so erfolgt ein Zeilen- oder Seitenvorschub. Der entstehende Zwischenbereich wird mit Leerzeichen gefüllt.

Eine Angabe von TABULATOR ohne einen nachfolgenden Wert *n* hat entweder keine Wirkung oder nur einen Zeilenvorschub zur Folge. Es gilt: 1 ≤ TABULATOR ≤ COLUMNS

BLANK n — BLANK bewirkt, daß *n* Leerzeichen eingefügt werden. Dabei kann ein Zeilen- oder Seitenvorschub erfolgen.

n ist eine ganze Zahl.

Beispiel

Die Definition des Bildschirmformats "mitarbeiterausgabe" befindet sich auf Seite 66.

Der FILL-Bereich dieses Bildschirmformats wird folgendermaßen mit Inhalt gefüllt: Die Namen der einfachsten Komponenten der Datengruppe &masatz werden zunächst ausgegeben. Dann werden die Dateninhalte der Datengruppe &masatz solange ausgegeben bis das Ende der Cursortabelle "macursor" erreicht ist.

```
1995-12-20                                                15:28:32

                              Mitarbeiter
                             ____________

nachname            vorname             gehalt
Winterberg          Hannelore             3500.00
Mitscherlich        Hermann               2700.00
.
.
.

================================================================
```

```
DECLARE macursor CURSOR ...
DECLARE VARIABLE &masatz,
                  2 nachname CHAR (20),
                  2 vorname  CHAR (20),
                  2 gehalt   NUM (7,2);
...
FILL mitarbeiterausgabe TABLE NAMES &masatz;
CYCLE macursor INTO &masatz.*;
   FILL mitarbeiterausgabe TABLE VALUES &masatz;
END CYCLE;
```

FILL listname
Listenformat aufbauen und füllen

Diese Anweisung ist gültig

– im TIAM-Betrieb
– im UTM-Betrieb, aber nicht im UTM-Asynchronbetrieb
– im Programm-Modus

FILL *listname* definiert den Inhalt und das Layout von Listen zur Ausgabe von Daten. Hat die Anweisung FILL *listname* einen Seiten-Überlauf zur Folge, wird implizit ein DISPLAY *listname* ausgeführt.

Bei Einsatz unter UTM muß die zentrale Druckdatei generiert sein. Für die Generierung gibt es drei Möglichkeiten:

– die Datei DRI.LIST.FILE ist bereits vorhanden
– die Datei wird über den Dateikettungsnamen DRILIST zugewiesen
– die Datei DRI.LIST.FILE wird bei der ersten DISPLAY-Anweisung von DRIVE/WINDOWS generiert

Die tatsächliche Druckausgabe auf den SPOOL-Drucker erfolgt beim nächsten STOP (siehe Anweisung STOP) oder LIST (siehe Anweisung LIST).

```
FILL listname [ format ]

     { ausdruck [ mask ] |
       NEWLINE n |
       NEWPAGE n |
       TABULATOR n |
       BLANK n }, ...
```

---

listname — Name des Listenformats (max. 31 Zeichen). Es muß im Deklarationsteil des Programms mit DECLARE LIST definiert sein.

format — Festlegen des Listenformats.

Wird für *format* TABLE angegeben, dürfen NEWPAGE und NEWLINE nicht angegeben werden.
Wird für *format* LINE angegeben, erfolgt auf die Angaben TABULATOR und BLANK ein Zeilenvorschub mit Leerzeile.

ausdruck	Definieren von Ausgabefeldern für das Listenformat. *ausdruck* kann sein: Eine oder mehrere Variablen (auch Systemvariablen) und/oder ein oder mehrere Literale. Paßt eine Ausgabevariable nicht auf eine Druckseite, so wird sie zur Ausgabe abgeschnitten. Die letzten drei Zeichen der Ausgabevariable werden mit ">>>" gekennzeichnet.
mask	Definieren der Darstellungsmöglichkeiten für maskierte Ausgaben (= Ausgabeaufbereitung). Die Angabe *mask* ist nur erlaubt, wenn *ausdruck* eine einfache Variable oder eine einfache Komponente ist.
NEWLINE n	Mit NEWLINE wird ein Vorschub von *n* Zeilen durchgeführt. Mit NEWLINE 1 wird die aktuelle Zeile abgeschlossen. Falls Daten folgen, werden sie in die nächste Zeile geschrieben. Auch wenn die aktuelle Zeile schon ganz gefüllt ist, also schon soviele Zeichen enthält, wie in der COLUMNS-Angabe des entsprechenden DECLARE vereinbart wurden, bewirkt NEWLINE 1 nicht die Ausgabe einer Leerzeile, sondern nur eine Positionierung auf die nächste Zeile. Hat *n* den Wert "0", erfolgt ein bedingter Vorschub, d.h. wenn die aktuelle Zeile leer ist, wird NEWLINE 0 ignoriert, wenn die aktuelle Zeile nicht leer ist, wirkt NEWLINE 0 wie NEWLINE 1.
NEWPAGE n	Mit NEWPAGE wird ein Vorschub von *n* Seiten durchgeführt. Mit NEWPAGE 1 wird die aktuelle Seite abgeschlossen. Falls Daten folgen, werden sie auf die nächste Seite geschrieben. Auch wenn die aktuelle Seite schon ganz gefüllt ist, also schon soviele Zeilen enthält, wie in der LINES-Angabe des entsprechenden DECLARE vereinbart wurden, bewirkt NEWPAGE 1 nicht die Ausgabe einer Leerseite, sondern nur eine Positionierung auf die nächste Seite. Hat *n* den Wert "0", erfolgt ein bedingter Vorschub, d.h. wenn die aktuelle Seite leer ist, wird NEWPAGE 0 ignoriert, wenn die aktuelle Seite nicht leer ist, wirkt NEWPAGE 0 wie NEWPAGE 1.
TABULATOR n	Die Ausgabe wird ab Spalte *n* fortgeführt. Ist der Wert kleiner als die aktuelle Spaltenposition, so erfolgt ein Zeilen- oder Seitenvorschub. Der entstehende Zwischenbereich wird mit Leerzeichen gefüllt. Eine Angabe von TABULATOR ohne einen nachfolgenden Wert *n* hat entweder keine Wirkung oder nur einen Zeilenvorschub zur Folge. Es gilt: 1 ≤ TABULATOR ≤ COLUMNS

BLANK n

BLANK bewirkt, daß *n* Leerzeichen eingefügt werden. Dabei kann ein Zeilen- oder Seitenvorschub erfolgen.

n ist eine ganze Zahl.

GET FILE POSITION
Dateiposition lesen

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

GET FILE POSITION liest in einer geöffneten Datei die aktuelle Dateiposition und überträgt diese in eine Variable.

In ISAM-Dateien positionieren Sie mit der Anweisung LOCATE FILE über den ISAM-Schlüssel.

```
GET FILE POSITION datei TO variable
```

---

datei	Logischer Name einer Datei, aus der die Dateiposition gelesen wird. Mit diesem Namen muß die Datei im Programm mit der Anweisung DECLARE FILE definiert sein.
variable	Bezeichnung der Variablen, in die die Dateiposition übertragen wird. (siehe Metavariable *variable*). Die Variable *variable* muß groß genug sein, um die Dateiposition (260 Byte) aufnehmen zu können.

GET MODIFIED INDEX
Modifizierte Listenzeile erfassen

Diese Anweisung ist gültig

– im UTM-Betrieb
– im Programm-Modus

GET MODIFIED INDEX erfaßt modifizierte Zeilen aus Listenbereichen in der Reihenfolge, in der sie vom Anwender verändert wurden.

```
GET { FIRST | NEXT } MODIFIED INDEX INTO variable FROM screenformat
```

FIRST	Diejenige Zeile im Listenbereich, die zuerst modifiziert wurde, wird erfaßt.
NEXT	Die nächste modifizierte Zeile im Listenbereich wird erfaßt. NEXT dürfen Sie nur angeben, wenn Sie zuvor eine Zeile desselben Listenbereichs mit FIRST erfaßt haben. Enthält ein Listenbereich keine weitere modifizierte Zeile, liefert NEXT den NULL-Wert.
variable	Name der Variablen, in der DRIVE/WINDOWS die Nummer der modifizierten Listenzeile ablegt. Der Datentyp von *variable* muß numerisch sein.
screenformat	FHS-DE-Teilformat mit einem Listenbereich, das mit DISPLAY SCREEN ausgegeben sein muß.

i Die parallele Bearbeitung zweier Listenbereiche mit GET MODIFIED INDEX ist nicht möglich.

GET SCREEN CURSOR
Position der Schreibmarke lesen

Diese Anweisung ist gültig

– im UTM-Betrieb
– im Programm-Modus

GET SCREEN CURSOR liest in einem Bildschirmformat die Position der Schreibmarke. Das Bildschirmformat kann ein Teilformat oder eine Dialog-Box sein.

Wenn sich die Schreibmarke in einem benannten Feld befindet, ermittelt DRIVE/WINDOWS den Feldnamen. Ist das Feld unbenannt, ermittelt DRIVE/WINDOWS die absolute Position innerhalb des Teilformats oder der Dialog-Box.

```
GET SCREEN CURSOR INTO variable1, variable2, variable3 FROM screenformat
```

---

variable1
: Name der Variablen, in der DRIVE/WINDOWS den Feldnamen eines FHS-DE-Teilformats oder einer FHS-DE-Dialog-Box ablegt, wenn sich die Schreibmarke in einem benannten Feld befindet.

  variable1 erhält den NULL-Wert, wenn sich die Schreibmarke in einem unbenannten Feld befindet.

  Der Datentyp von *variable1* muß alphanumerisch sein. *variable1* muß mindestens 8 Zeichen lang sein.

variable2
: Name der Variablen, in der DRIVE/WINDOWS die Zeile (absolute Position) ablegt, wenn sich die Schreibmarke in einem unbenannten Feld befindet.

  variable2 erhält den NULL-Wert, wenn sich die Schreibmarke in einem benannten Feld befindet.

  Der Datentyp von *variable2* muß numerisch sein.

variable3
: Name der Variablen, in der DRIVE/WINDOWS die Spalte (absolute Position) ablegt, wenn sich die Schreibmarke in einem unbenannten Feld befindet.

  variable3 erhält den NULL-Wert, wenn sich die Schreibmarke in einem benannten Feld befindet.

  Der Datentyp von *variable3* muß numerisch sein.

screenformat	FHS-DE-Format (Teilformat oder Dialog-Box), in dem sich die Schreibmarke befindet, deren Position gelesen werden soll. *screenformat* muß am Bildschirm ausgegeben sein.

IF
Bedingung programmieren

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

IF kennzeichnet den Beginn eines IF-Blocks, dessen Ende mit END IF festgelegt wird. Innerhalb eines IF-Blocks wird, abhängig vom Wahrheitswert einer Bedingung, zu alternativen Anweisungen verzweigt. Wenn die Bedingung den Wert TRUE liefert, wird der THEN-Zweig durchlaufen, ansonsten der ELSE-Zweig.

Stehen Bedingungen innerhalb eines internen Unterprogramms, müssen sie ebenfalls mit END IF abgeschlossen werden (siehe Anweisung SUBPROCEDURE).

Bedingungen dürfen geschachtelt werden. Die Schachtelungstiefe ist beliebig und nur abhängig vom Speicherplatz, den DRIVE/WINDOWS zur Bearbeitung benötigt.

Bedingungen, Schleifen (CYCLE), Verzweigungen (CASE ... OF) und parallele remote Verarbeitung (DISPATCH) dürfen sich nicht überlappen.

Bedingungen, Schleifen (CYCLE) und Verzweigungen (CASE ... OF) dürfen sich nicht überlappen.

In einem Programm darf nach der Anweisung IF kein Semikolon stehen. Das Semikolon steht nach der Anweisung im THEN-Zweig.

```
IF bedingung THEN { programming } ...
             [ ELSE { programming } ... ]
```

---

bedingung	Bedingungen an Dateninhalte von Attributen und/oder Variablen stellen. *bedingung* darf keine *projektion* enthalten.
THEN	Der THEN-Zweig wird durchlaufen, wenn *bedingung* den Wahrheitswert TRUE liefert.
ELSE	Der ELSE-Zweig wird durchlaufen, wenn *bedingung* die Wahrheitswerte FALSE oder UNKNOWN liefert. Wird ELSE nicht angegeben und liefert *bedingung* einen der Wahrheitswerte FALSE oder UNKNOWN, wird die nächste Anweisung nach END IF ausgeführt.
programming	Anweisungen für den Verarbeitungsteil eines Programms.

Beispiel

Wenn der Inhalt der Variablen &antwort "j" ist, wird der Datensatz &artikel.* in die Tabelle "v_artikel" eingefügt. Andernfalls wird das Unterprogramm "beenden" aufgerufen.

```
IF &antwort = 'j'
   THEN INSERT INTO v_artikel VALUES (&artikel.*);
   ELSE CALL beenden;
END IF;
```

Definition von Fehlerausgängen

Die Definition eines Fehlerausgangs über WHENEVER ist die einzige Möglichkeit, einen Programmabbruch wegen Semantikfehler bei der Auswertung von Bedingungen zu vermeiden. Die Definition muß im Deklarationsteil stehen. Das Programm wird entsprechend des bei WHENEVER definierten Fehlerausgangs nach dem zugehörenden END IF fortgesetzt (siehe Anweisung WHENEVER).

LIST
Liste ausgeben

Diese Anweisung ist gültig

– im UTM-Betrieb
– im Dialog- und Programm-Modus

LIST erteilt UTM-Druckaufträge sowohl im UTM-Dialog- als auch im UTM-Asynchronbetrieb. Der Druck selbst erfolgt asynchron. Zur Erzeugung von Druckaufträgen müssen mindestens ein Asynchron-Task und der Transaktionscode DRILIST und die zentrale Druckdatei generiert sein. LIST greift auf die benutzereigenen Daten in der zentralen Druckdatei zu und gibt sie aus.

LIST greift auf den druckaufbereiteten Inhalt der zentralen Druckdatei zu und gibt ihn entweder über einen Drucker (DEVICE) oder auf einem Datensichtgerät (LTERM) aus oder kopiert ihn in eine Datei (FILE)oder in ein P-Element der DRIVE-Bibliothek (LIB).

Mit dem Operanden DELETE können Sie bestimmen, ob nach der Druckausgabe der Inhalt der zentralen Druckdatei gelöscht wird.

Wird DRIVE/WINDOWS im Dialogbetrieb mit der Anweisung STOP beendet, wird der Inhalt der zentralen Druckdatei automatisch ausgedruckt und in der Druckdatei gelöscht. Ein automatischer Ausdruck von Fehlerlisten nach STOP unterbleibt, wenn der Parameter PARAMETER DIAGNOSIS INTTRACE ='NO_ERR_LIST' gesetzt wurde.

Bei einem fehlerhaften Vorgangsabbruch bleibt der Inhalt der zentralen Druckdatei erhalten. Sie müssen ihn beim nächsten UTM-Vorgang mit der Anweisung LIST ausgeben.

Unter UTM wird eine Startdatei erzeugt, in der die benutzerbezogenen DRIVE-Startparameter hinterlegt werden. Diese Datei kann mit der Anweisung LIST * nicht ausgegeben werden. Sie wird nach Eingabe der Anweisung STOP gelöscht. Im Fehlerfall werden auch Fehlermeldungen in der Datei gespeichert.

```
LIST * [ WHERE STATUS { ENTER | DIALOG | vorgang } ]

       [ INTO { FILE { filename | variable1 } |
                LTERM { ltermname | variable2 } |
                DEVICE { devname | variable3 } } ]
                LIB { bibliothek(elemname) | elemname }

       [ spoolparameter ] [ DELETE ]
```

---

*	Ausgabe aller Listen
WHERE STATUS	Alle noch vorhandenen Druckaufträge des angegebenen Vorgangs (Dialog-, Asynchronbetrieb) werden ausgegeben. Wird WHERE STATUS nicht angegeben, werden nur die Druckaufträge des aktuellen Vorgangs ausgegeben.
ENTER	Alle Druckaufträge des Asynchronbetriebs werden ausgegeben.
DIALOG	Alle Druckaufträge des Dialogbetriebs werden ausgegeben.
vorgang	Variablenname für den aktuellen Vorgang. *vorgang* muß eine alphanumerische Variable sein und darf nur einen der Werte ENTER oder DIALOG annehmen. *vorgang* darf nur im Programm-Modus angegeben werden.
INTO	Mit INTO wird das Medium der Ausgabe festgelegt. Wird INTO nicht angegeben, werden die unter *spoolparameter* angegebenen Spool-Parameter ausgewertet. Die Druckausgabe erfolgt auf den zentralen Schnelldrucker.
FILE	Die Ausgabe erfolgt in eine Datei.
filename	Namenserweiterung für die Datei, in die die Druckausgabe erfolgt (max. 20 Zeichen). *filename* muß den BS2000-Dateinamenskonventionen entsprechen. DRIVE/WINDOWS ergänzt zusätzlich noch den Präfix `DRI.LIST.utmname`. Der vollständige Name lautet: `DRI.LST.utmname.`*filename*.
variable1	Variablenname für die bei *filename* beschriebene Namenserweiterung. *variable1* muß eine alphanumerische Variable vom Typ CHAR(L) mit $1 \leq L \leq 21$ sein. *variable1* darf nur im Programm-Modus angegeben werden.
LTERM	Die Druckausgabe erfolgt auf ein in der KDCFILE eingetragenes UTM-Datensichtgerät oder einen UTM-Stationsdrucker oder auf das Ausgabemedium, das im KDCDEF bei LTERM angegeben wurde. Im TIAM-Betrieb ist diese Angabe nicht möglich.
ltermname	Name eines UTM-Datensichtgerätes oder eines UTM-Stationsdruckers (max. 8 Zeichen). *ltermname* muß als Literal angegeben werden.

variable2	Variablenname für ein UTM-Datensichtgerät. *variable2* muß eine alphanumerische Variable vom Typ CHAR(L) mit $1 \leq L \leq 8$ sein. *variable2* darf nur im Programm-Modus angegeben werden.
DEVICE	Die Druckausgabe erfolgt auf einen Drucker. Alle Druckaufträge des Vorgangs werden ausgedruckt.
devname	Name eines Druckers, der im PDN generiert sein muß (max. 8 Zeichen). *devname* muß als Literal angegeben werden.
variable3	Variablenname eines Druckers. *variable3* muß eine alphanumerische Variable vom Typ CHAR(L) mit $1 \leq L \leq 8$ sein. *variable3* darf nur im Programm-Modus angegeben werden.
LIB	Die Ausgabe erfolgt in ein P-Element der DRIVE-Bibliothek.
bibliothek	Name der DRIVE-Bibliothek (max. 54 Zeichen), in die die Datei ausgegeben wird. *bibliothek* kann auch der Dateikettungsname der DRIVE-Bibliothek (entsprechend den BS2000-Konventionen) sein. DRIVE/WINDOWS interpretiert *bibliothek* zuerst als Dateikettungsnamen, dann als Bibliotheksnamen. Wird die DRIVE-Bibliothek mit der Anweisung PARAMETER DYNAMIC LIBRARY voreingestellt, so kann die Angabe von *bibliothek* entfallen.
elemname	Name des Elements (max. 31 Zeichen), in die die Liste ausgegeben wird. Wird keine Angabe für *bibliothek* gemacht, wird die bei PARAMETER DYNAMIC LIBRARY angegebene Bibliothek eingesetzt.
spoolparameter	bezeichnet Optionen für die Druckerverwaltung. Diese Optionen müssen der Syntax des BS2000-Kommandos PRINT-FILE entsprechen (siehe BS2000-Kommandos [35]). Die Zeichenkette *spoolparameter* wird ungeprüft an die Druckerverwaltung übergeben. *spoolparameter* wird ignoriert, wenn INTO FILE, INTO LTERM oder INTO LIB angegeben werden. Die Zeichenkette darf max. 256 Zeichen lang sein und kann als Literal oder als Inhalt einer DRIVE-Variablen vom Typ CHARACTER oder VARCHAR angegeben werden.

Sind keine Optionen für die Druckerverwaltung angegeben, wird die Einstellung SPOOLDOPT des Benutzerprofils ausgewertet. Sind keine Angaben vorhanden, wird für das BS2000-Kommando PRINT-FILE der Operand
LAYOUT-CONTROL=PARAMETERS(CONTROL-CHARACTERS=PHYSICAL)
eingesetzt (siehe BS2000-Kommandos [35]).

Werden in einem Report mehrere Zeichensätze verwendet, müssen die Zeichensätze mit dem Operanden
LAYOUT-CONTROL=PARAMETERS(CONTROL-CHARACTERS=PHYSICAL,
CHARACTER-SETS=...)
angegeben werden. (Siehe BS2000-Kommandos [35], Kommando PRINT-FILE).

DELETE — Nach Abschluß der Druckausgabe wird der Inhalt der zentralen Druckdatei gelöscht.

LOCATE FILE
In einer ISAM-Datei positionieren

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

LOCATE FILE positioniert in einer geöffneten ISAM-Datei auf einen Datensatz.

Bei der Angabe POSITION=KEY wird auf den Datensatz mit dem angegebenen ISAM-Schlüssel positioniert. Wenn DRIVE/WINDOWS einen Satz mit dem angegebenen Schlüssel nicht findet, gibt DRIVE/WINDOWS eine Fehlermeldung aus.

Bei der Angabe POSITION>=KEY wird auf den Datensatz mit dem angegebenen oder dem nächsthöheren ISAM-Schlüssel positioniert.

```
LOCATE FILE datei TO charausdruck

                [ [ WITH ] POSITION { = | >= } KEY ]
```

---

datei	Logischer Name einer Datei, in der positioniert wird. Mit diesem Namen muß die Datei im Programm mit der Anweisung DECLARE FILE definiert sein.
charausdruck	Der Schlüsselwert (ISAM-Schlüssel) des Datensatzes wird festgelegt.
POSITION=KEY	Es wird auf den Satz mit dem ISAM-Schlüssel *charausdruck* positioniert.
POSITION>=KEY	Vorbelegung Es wird auf den Satz mit dem ISAM-Schlüssel *charausdruck* positioniert oder auf den ersten Satz, dessen ISAM-Schlüssel größer als *charausdruck* ist.

OPEN FILE
Datei öffnen

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

OPEN FILE öffnet eine Datei und weist dieser Datei einen logischen Namen zu. Der logische Name muß mit der Anweisung DECLARE FILE festgelegt worden sein. Mit ihm wird die Datei in DRIVE-Programmen angesprochen.

DRIVE/WINDOWS erlaubt es, dieselbe Datei unter verschiedenen logischen Namen zu öffnen, wenn das Betriebssystem BS2000 dies zuläßt.

Mit der Anweisung geben Sie den Dateityp an und in welcher Eröffnungsart (OPEN-Modus) die Datei geöffnet werden soll.

Beim Öffnen von Dateien mit INPUT, OUTPUT, UPDATE, INOUT und OUTIN wird auf den Dateianfang positioniert, beim Öffnen mit EXTEND wird auf das Dateiende positioniert.

Wenn eine Datei mit UPDATE, INOUT oder OUTIN zum Lesen und Schreiben geöffnet ist, dürfen lesende und schreibende Zugriffe nicht unmittelbar aufeinander folgen. Zwischen den Anweisungen READ FILE und WRITE FILE muß die Anweisung SET FILE POSITION stehen, es sei denn, DRIVE/WINDOWS erreicht bei einem lesenden Zugriff das Dateiende.

DRIVE/WINDOWS gibt eine Fehlermeldung aus, wenn eine Datei, die für Lesezugriffe geöffnet werden soll, nicht existiert oder die Angaben über den Dateityp nicht übereinstimmen.

Das Betriebssystem BS2000 überprüft die Zugriffsrechte auf Dateien. Falls der Zugriff nicht erlaubt ist, übernimmt DRIVE/WINDOWS die Fehlermeldung des Betriebssystems und gibt sie aus.

i Überschreiben Sie die Inhalte von Textdateien nicht. Weil die tatsächlich abgespeicherte Länge von Variablen nicht bekannt ist, können überschreibende Daten länger oder kürzer sein als die zu überschreibenden. In beiden Fällen wird der Aufbau der Textdatei geändert und die Textdatei kann nicht mehr über die (sie beschreibende) DRIVE-Variable gelesen werden.

```
OPEN FILE datei IN bsdatei
                [ SAM | ISAM | BIN | CHARACTER zeichen ]
                [ INPUT | OUTPUT | EXTEND | UPDATE | INOUT | OUTIN ]
```

datei	Logischer Name einer Datei (max. 31 Zeichen). Mit diesem Namen muß die Datei im Programm mit der Anweisung DECLARE FILE definiert sein. Es darf noch keine Datei mit diesem logischen Namen geöffnet sein.
IN	Angabe der Datei, die unter dem logischen Namen angesprochen wird.
bsdatei	Name einer Datei auf Betriebssystemebene (max. 54 Zeichen), die geöffnet wird. *bsdatei* kann auch der Dateikettungsname der Datei sein. DRIVE/WINDOWS interpretiert *bsdatei* zuerst als Dateikettungsnamen, dann als Dateinamen. *bsdatei* muß ein Dateikettungsname wenn eine ISAM-Datei SHARED-UPDATE bearbeitet werden soll. Dieser Name muß den Dateinamenskonventionen des Betriebssystems BS2000 entsprechen *bsdatei* muß vom Typ *charausdruck* sein.
SAM	Vorbelegung Die Datei ist eine SAM-Datei.
ISAM	Die Datei ist eine ISAM-Datei.
BIN	Die Datei ist eine Binärdatei (siehe DRIVE-Programmiersprache [2]).
CHARACTER	Die Datei ist eine Textdatei (siehe DRIVE-Programmiersprache [2]).
zeichen	Das Trennzeichen zwischen Feldern (ein Zeichen) wird festgelegt. Vorbelegung: ein Leerzeichen

Geben Sie für *zeichen* nicht an:
– das Zeilenendezeichen
– das Dateiendezeichen
– das Leerzeichen (%)
– die Zeichen: + - , oder .
Dies kann zu ungewollten Reaktionen führen.

Das Trennzeichen *zeichen* muß als alphanumerisches Literal (*charliteral*, siehe Metavariable *literal*) oder als sedezimales Zeichen (*sedecliteral*, siehe Metavariable *literal*) angegeben werden.

INPUT — Öffnen zum Lesen. Die Datei muß bereits vorhanden sein.

OUTPUT — Vorbelegung

Öffnen zum Schreiben. Ist bereits eine Datei mit dem Namen *bsname* vorhanden, wird der alte Inhalt gelöscht, ansonsten wird die Datei neu erstellt.

EXTEND — Öffnen zum Schreiben. Ist bereits eine Datei mit dem Namen *bsname* vorhanden, bleibt der alte Inhalt erhalten und die neuen Daten werden ans Ende der Datei angehängt. Ist die Datei nicht vorhanden, wird sie neu erstellt.

EXTEND darf bei ISAM-Dateien nicht angegeben werden.

UPDATE — Öffnen zum Lesen und Schreiben. Ist bereits eine Datei mit dem Namen *bsname* vorhanden, bleibt der alte Inhalt beim Öffnen erhalten und kann dann überschrieben werden.

INOUT — Öffnen zum Lesen und Schreiben. Die Datei muß bereits vorhanden sein. Der alte Inhalt bleibt beim Öffnen erhalten und kann dann überschrieben werden.

OUTIN — Öffnen zum Lesen und Schreiben. Ist bereits eine Datei mit dem Namen *bsname* vorhanden, wird der alte Inhalt gelöscht. Ist die Datei nicht vorhanden, wird sie neu erstellt.

Besonderheiten der Dateimerkmale

– Sie können Dateien mit dem K- und NK-Blockformat bearbeiten. (Siehe Handbuch Einführung in das DVS [37])

– Wenn eine Datei neu angelegt wird und Sie keine Angaben über Dateityp und Eröffnungsart gemacht haben, erhält sie die Merkmale SAM und OUTPUT. Alle weiteren Dateimerkmale entsprechen den Voreinstellungen des Betriebssystems BS2000.

– Wenn eine Datei neu angelegt wird, erhält sie abhängig vom angegebenen Dateityp folgende Merkmale:

  SAM: Satzformat: (V,N)

  ISAM: Satzformat: (V,N)
  Schlüsselposition: 5
  Schlüssellänge: 8

– Wenn bereits eine Datei vorhanden ist, gelten die Angaben aus dem Dateikatalog oder dem Dateikettungseintrag. Auch wenn der Inhalt einer bereits vorhandenen Datei gelöscht wird, bleiben die Katalogeigenschaften dieser Datei erhalten.

– Wenn Sie eine Datei beim Öffnen überschreiben wollen, müssen Sie in der OPEN FILE-Anweisung den gleichen Dateityp (SAM oder ISAM) angeben, den die zu überschreibende Datei hat.

– Sie bearbeiten mit DRIVE/WINDOWS nur Dateien, die die Blockkontrollinformation im Datenblock haben (Blockformat DATA). Auch wenn Sie auf Dateien ohne Blockkontrollinformation zugreifen (Blockformat NO), erhält die Datei das Blockformat DATA (siehe Handbuch Einführung in das DVS [37]).

– Verwenden Sie beim Öffnen einer Datei den Dateikettungsnamen, so lassen sich mit dem SET-FILE-LINK-Kommando im BS2000 folgende Dateimerkmale ändern: Zugriffsmethode, Satzformat, Satzlänge, Blockformat und Blocklänge.

  Das SET-FILE-LINK-Kommando muß eingegeben werden:

  – im TIAM-Betrieb vor dem Start des DRIVE-Programms oder im DRIVE-Programm mit der SYSTEM-Anweisung. Die Anweisung SYSTEM muß vor der OPEN-FILE-Anweisung stehen.
  – im UTM-Betrieb vor dem Start der DRIVE-Anwendung.

– Sollen ISAM-Dateien von mehreren Programmen gleichzeitig bearbeitet werden (SHARED-UPDATE=YES), so müssen Sie diese Dateien mit Dateikettungsnamen ansprechen. Geben Sie im BS2000 das SET-FILE-LINK-Kommando mit dem Operanden LINK-NAME= und der Struktur SUPPORT=...(SHARED-UPDATE=YES) ein.

  Das SET-FILE-LINK-Kommando muß eingegeben werden:

  – im TIAM-Betrieb vor dem Start des DRIVE-Programms oder im DRIVE-Programm mit der SYSTEM-Anweisung. Die Anweisung SYSTEM muß vor der OPEN-FILE-Anweisung stehen.
  – im UTM-Betrieb vor dem Start der DRIVE-Anwendung.

– Die ISAM-Schlüssel gehören zu den Satzdaten, die vom DRIVE-Programm aus geschrieben oder beim Lesen an das DRIVE-Programm geliefert werden.
Position und Länge des Schlüssels müssen mit dem entsprechenden SET-FILE-LINK-Kommando angegeben werden
(SET-FILE-LINK ... KEY-LENGTH=..., KEY-POSITION=...).

  Das SET-FILE-LINK-Kommando muß eingegeben werden:

  – im TIAM-Betrieb vor dem Start des DRIVE-Programms oder im DRIVE-Programm mit der SYSTEM-Anweisung. Die Anweisung SYSTEM muß vor der OPEN-FILE-Anweisung stehen.
  – im UTM-Betrieb vor dem Start der DRIVE-Anwendung.

– Im UTM-Betrieb gilt folgendes:
Dateien, die mit INPUT geöffnet werden und ISAM-Dateien, die mit der SET-FILE-LINK-Anweisung zu SHARED-UPDATE-Dateien erklärt wurden, werden während des UTM-Vorgangs nur einmal (bei der ersten OPEN FILE-Anweisung) geöffnet und dann erst bei Vorgangsende von DRIVE/WINDOWS geschlossen. Nach der ersten OPEN-Anweisung werden alle weiteren CLOSE FILE- und OPEN-FILE-Anweisungen ignoriert. Es ist auch nicht möglich, eine geöffnete Datei in einem anderen OPEN-Modus nochmals zu öffnen.

i Die Bearbeitung von ISAM-Dateien mit Schlüsselverdopplung (DUPKEY=YES) ist nicht erlaubt.

Beziehungen zu anderen Anweisungen

– Der Dateiname oder der Dateikettungsname *bsdatei* kann in Groß- oder Kleinbuchstaben geschrieben werden. Er wird wird im BS2000 automatisch in Großbuchstaben umgesetzt. Angaben, die Sie mit der Anweisung PARAMETER DYNAMIC LETTERS oder OPTION LETTERS gemacht haben, sind für das Betriebssystem BS2000 ungültig.

OPTION
Übersetzung eines Programms steuern

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Dialog- und Programm-Modus

OPTION steuert den Übersetzungslauf eines DRIVE-Programms. Zur Übersetzung von DRIVE-Programmen in Zwischencode werden Compiler-Optionen eingestellt.

OPTION kann in Sourcen (vor der Anweisung PROCEDURE und DECLARE TYPE) oder mit der Anweisung COMPILE angegeben werden.

Voreinstellungen von DRIVE/WINDOWS werden durch OPTION-Angaben in der Source überschrieben. OPTION-Angaben in der Source werden durch OPTION-Angaben bei COMPILE überschrieben.

Einige Operanden der Anweisung OPTION werden nicht auf jedem Betriebssystem (BS2000, SINIX und MS-Windows) unterstützt (siehe DRIVE-Lexika für SINIX [12] und MS-Windows[9]). Aus Kompatibilitätsgründen werden diese Operanden ohne Funktionalität von DRIVE/WINDOWS ignoriert.

```
OPTION { AUTHORIZATION=berechtigung |
         CATALOG=sesdbname |
         CODE={ OFF | ON } |
         DBSYSTEM={ OFF | SESAM | SESAMSQL | UDS } |
         DCSYSTEM={ TIAM | UTM | BOTH } |
         DECIMALSIGN={ . | , } |
         DISTRIBUTION={ OFF | ON } |
         LETTERS={ CAPITAL | BOTH | UNCHANGED } |
         LISTING={ OFF | LIBRARY | LIST | BOTH } |
         LISTTYPE={ OFF | USER | EXPERT } |
         MONINFO={ OFF | ON } |
         NULLVALUE={ OFF | ON } |
         OBJECT={ OFF | ON } |
         PERMIT={ OFF | ON } |
         SCHEMA=schemaname |
         SCREENCHECK={ OFF | ON }
         TASKTYPE={ DIALOG | ENTER | BOTH } |
         VERSIONMIX={ OFF | ON } |
         XREF={ OFF | ON } } ...
```

AUTHORIZATION	legt den Berechtigungsschlüssel *berechtigung* für eine SESAM-Datenbank (max. 18 Zeichen) fest (siehe SQL-Lexikon für SESAM V2 [5]). Vorbelegung: die aktuelle Einstellung aus PARAMETER DYNAMIC AUTHORIZATION
CATALOG	legt die Voreinstellung für eine SESAM-Datenbank *sesdbname* von SESAM V2.x (max. 18 Zeichen) fest (siehe SQL-Lexikon für SESAM V2 [5]). Vorbelegung: die aktuelle Einstellung aus PARAMETER DYNAMIC CATALOG oder Leerzeichen
CODE	legt fest, ob der erzeugte Zwischencode abgespeichert werden soll.
=OFF	Vorbelegung Der Zwischencode wird nicht abgespeichert.
=ON	Der Zwischencode wird in der DRIVE-Bibliothek gespeichert (Elementtyp X). Das Element mit dem Zwischencode heißt wie das Element mit der Source.
DBSYSTEM	legt das Datenbanksystem fest, auf das die SQL-Anweisungen in einem DRIVE-Programm zugreifen sollen. Vorbelegung: die geladene Variante, falls das zu übersetzende DRIVE-Programm SQL-Anweisungen enthält (außer COMMIT WORK und ROLLBACK WORK).
=OFF	Das zu übersetzende DRIVE-Programm enthält außer COMMIT WORK und ROLLBACK WORK keine weiteren SQL-Anweisungen, die auf eine Datenbank zugreifen. Vorbelegung, falls das zu übersetzende DRIVE-Programm außer COMMIT WORK und ROLLBACK WORK keine weiteren SQL-Anweisungen enthält. Wenn für ein Programm kein Datenbanksystem festgelegt ist (OPTION DBSYSTEM=OFF), wird zum Ablaufzeitpunkt das Datenbanksystem bestimmt, das für das rufende Programm festgelegt ist (siehe DRIVE-Programmiersprache [2]).
=SESAM	SQL-Anweisungen greifen auf eine SESAM-Datenbank von SESAM V1.x zu.
=SESAMSQL	SQL-Anweisungen greifen auf eine SESAM-Datenbank von SESAM V2.x zu.

=UDS	SQL-Anweisungen greifen auf eine UDS-Datenbank zu.
DCSYSTEM	legt das Zielkommunikationssystem fest, in dem das Programm ablauffähig sein soll.
	Vorbelegung: der aktuelle Wert zum Zeitpunkt des Übersetzens
=TIAM	für TIAM-Betrieb
=UTM	für UTM-Betrieb
=BOTH	für den Programm-Einsatz mit beiden Einsatzformen
DECIMALSIGN	legt das Dezimalzeichen in der Source fest.
	Vorbelegung: der Punkt (.)
	Diese Option wird auch bei der Analyse von dynamischen Anweisungen verwendet (siehe Anweisung EXECUTE).
DISTRIBUTION	legt fest, ob beim Aufruf von Programmen mit CALL oder ENTER die Verteilungsinformationen ausgewertet werden sollen.
=OFF	Vorbelegung
	Die Verteilungsinformationen werden nicht ausgewertet.
=ON	Die Verteilungsinformationen werden zum Ablaufzeitpunkt ausgewertet.
LETTERS	legt die Buchstabenverarbeitung des übersetzten Programms fest. Diese Angabe hat Auswirkungen auf Namen und Literale.
	Diese Option wird auch bei der Analyse von dynamischen Anweisungen verwendet (siehe Anweisung EXECUTE).
=CAPITAL	Vorbelegung
	Vom übersetzten Programm werden nur Großbuchstaben verarbeitet.
	Alle Kleinbuchstaben in Namen und Literalen werden in Großbuchstaben umgesetzt. Umlaute werden nicht umgesetzt.
=BOTH	Vom übersetzten Programm werden Großbuchstaben und Kleinbuchstaben verarbeitet.
	Kleinbuchstaben in Namen werden in Großbuchstaben umgesetzt. Kleinbuchstaben in Literalen werden nicht in Großbuchstaben umgesetzt.
	Schlüsselwörter und Metavariablen werden immer umgesetzt (gilt nur bei Erstellen von OLTP-Anwendung für das BS2000 und bei Remote-Zugriff).

=UNCHANGED	Bei Namen und Literalen erfolgt keine Umwandlung von Kleinbuchstaben in Großbuchstaben. Die Angabe LETTERS=UNCHANGED sollte nicht verwendet werden, weil Software-Produkte wie LMS, EDT, SESAM, UDS, mit denen DRIVE/WINDOWS zusammenarbeitet, klein geschriebene Buchstaben unterschiedlich behandeln.
LISTING	legt fest, ob eine Übersetzungliste erzeugt werden soll.
=OFF	Vorbelegung Es wird keine Übersetzungsliste erzeugt.
=LIBRARY	Es wird eine Übersetzungsliste in der DRIVE-Bibliothek gespeichert (Elementtyp P). Das Element mit der Übersetzungsliste heißt wie das Element mit der Source.
=LIST	Im TIAM-Betrieb wird eine Übersetzungsliste nach SYSLST ausgegeben. Im UTM-Betrieb wird die Übersetzungsliste in die zentrale Druckdatei geschrieben.
=BOTH	Die Übersetzungsliste wird in die DRIVE-Bibliothek ausgegeben. Außerdem wird die Übersetzungsliste im TIAM-Betrieb nach SYSLST und im UTM-Betrieb in die zentrale Druckdatei ausgegeben.
LISTTYPE	legt fest, ob eine Compiler-Übersetzungsliste erzeugt werden soll. LISTTYPE kann nur angegeben werden, wenn der DRIVE-Compiler DRIVE/WINDOWS-Comp eingesetzt wird (siehe DRIVE-Compiler [16]).
=OFF	Vorbelegung Es wird keine Übersetzungsliste erzeugt.
=USER	Es wird eine Übersetzungsliste mit dem generierten ASSEMBLER-Code nach SYSLST ausgegeben.
=EXPERT	Es wird eine Übersetzungsliste mit dem generierten ASSEMBLER-Code und eine Querverweisliste nach SYSLST ausgegeben.
MONINFO	legt fest, ob eine Montageinformation erzeugt werden soll. MONINFO kann nur angegeben werden, wenn der DRIVE-Compiler DRIVE/WINDOWS-Comp eingesetzt wird (siehe DRIVE-Compiler [16]).

=OFF	Vorbelegung
	Es wird keine Montageinformation erzeugt.
=ON	Es wird eine Montageinformation nach *SYSLST* ausgegeben.
NULLVALUE	legt fest, ob der NULL-Wert zu den Variablen im Programm berücksichtigt werden soll.
	NULLVALUE kann nur angegeben werden, wenn der DRIVE-Compiler DRIVE/WINDOWS-Comp eingesetzt wird (siehe DRIVE-Compiler [16]).
=OFF	Vorbelegung
	Der NULL-Wert wird nicht berücksichtigt. Das Programm darf keine Zuweisungen mit NULL enthalten.
=ON	Der NULL-Wert wird berücksichtigt.
OBJECT	legt fest, ob der Compiler gestartet und Objektcode (Elemnettyp R) erzeugt werden soll.
	OBJECT kann nur angegeben werden, wenn der DRIVE-Compiler DRIVE/WINDOWS-Comp eingesetzt wird (siehe DRIVE-Compiler [16]).
=OFF	Vorbelegung
	Kein Compilerlauf. Es wird kein Objektcode erzeugt.
=ON	Der DRIVE-Compiler DRIVE/WINDOWS-Comp wird gestartet. Es wird Objektcode erzeugt.
PERMIT	legt fest, ob das erzeugte Objekt einen Bildschirm zur Eingabe der Benutzeridentifikation bringen soll (siehe Anweisung PERMIT in den SQL-Lexika [4] und [6]).
	PERMIT kann nur angegeben werden, wenn der DRIVE-Compiler DRIVE/WINDOWS-Comp eingesetzt wird (siehe DRIVE-Compiler [16]).
=OFF	Vorbelegung
	Es soll kein Bildschirm zur Eingabe der Benutzeridentifikation ausgegeben werden.
=ON	Es soll ein Bildschirm zur Eingabe der Benutzeridentifikation ausgegeben werden.

SCHEMA=schemaname

Name einer SESAM-Datenbank von SESAM V1.x (max. 18 Zeichen), eines SESAM-Schemas von SESAM V2.x (max. 31 Zeichen) oder eines UDS-Schemas (max. 30 Zeichen), auf die zugegriffen wird, wenn in einem Programm in SQL-Anweisungen kein Name angegeben wurde (siehe SQL-Lexika [4], [5], [6]).

Vorbelegung bei SESAM V1.x und UDS: keine, bei SESAM V2.x: die aktuelle Einstellung aus PARAMETER DYNAMIC SCHEMA oder Leerzeichen

SCREENCHECK legt fest, ob die von IFG erzeugten CHECK-Klauseln in den Adressierungshilfen von DRIVE/WINDOWS ausgewertet werden sollen (siehe IFG [28]).

=OFF Die CHECK-Klauseln werden nicht ausgewertet.

=ON Vorbelegung

Die CHECK-Klauseln werden ausgewertet.

TASKTYPE legt fest, wie ein übersetztes Programm ablaufen soll.

Vorbelegung: der aktuelle Wert zum Zeitpunkt des Übersetzens

=DIALOG Das übersetzte Programm kann nur als ein Dialog-Programm ablaufen (siehe Anweisung DO).

=ENTER Das übersetzte Programm kann nur als UTM-Asynchronvorgang gestartetet werden (siehe Anweisung ENTER).

=BOTH Das übersetzte Programm kann sowohl als Dialog-Programm und als UTM-Asynchronvorgang ablaufen.

VERSIONMIX legt fest, ob das übersetzte Programm im Mischbetrieb eingesetzt werden soll.

VERSIONMIX kann nur angegeben werden, wenn der DRIVE-Compiler DRIVE/WINDOWS-Comp eingesetzt wird (siehe DRIVE-Compiler [16]).

=OFF Vorbelegung

Das übersetzte Programm ist nur ablauffähig im New-Style-Betrieb von DRIVE/WINDOWS.

=ON Das übersetzte Programm ist im Mischbetrieb von DRIVE-WINDOWS ablauffähig.

DO-Anweisungen erfordern für das gerufene Programm eine TAC-Definition bei der UTM-Generierung (siehe DRIVE-Compiler [16]).

XREF	legt fest, ob für das übersetzte Programm Querverweise ausgegeben werden sollen. Dies geschieht nur, wenn nicht die Option LISTING=OFF angegeben wurde.
=OFF	Vorbelegung Es werden keine Querverweise ausgegeben.
=ON	Für das übersetzte Programm werden Querverweise ausgegeben.

Regeln

– Bei Formateingaben zum Ablaufzeitpunkt haben die OPTION-Angaben LETTERS und DECIMALSIGN keine Auswirkung.

  Ausnahme: Formateingaben, die in einer dynamischen SQL-Anweisung umgesetzt werden, unterliegen der OPTION-Angabe.

Beziehungen zu anderen Anweisungen

– Bei DO oder COMPILE ohne Angabe eines Bibliothekelements werden folgende Übersetzungsoptionen in der Source ignoriert: LISTING=LIBRARY, CODE=ON und OBJECT=ON.

– Bei DO werden folgende Optionen, die in der Source angegeben sind, ignoriert: CODE=ON und OBJECT=ON.

– Wenn in einem Programm die Option TASKTYPE=DIALOG angegeben ist und dieses Programm mit der Anweisung ENTER gestartet wurde, bleibt selbstverständlich der UTM-Asynchronvorgang bestehen.

Übersicht über die vorbelegten Werte

Option	**Vorbelegung**
AUTHORIZATION	aktuelle Einstellung aus PARAMETER DYNAMIC AUTHORIZATION
CATALOG	aktuelle Einstellung aus PARAMETER DYNAMIC CATALOG oder Leerzeichen
CODE	OFF
DBSYSTEM	geladene Variante oder OFF
DCSYSTEM	der aktuelle Wert zum Zeitpunkt des Übersetzens
DECIMALSIGN	. (Punkt)
DISTRIBUTION	OFF
LETTERS	CAPITAL
LISTING	OFF
LISTTYPE	OFF
MONINFO	OFF
NULLVALUE	OFF
OBJECT	OFF
PERMIT	OFF
SCHEMA	aktuelle Einstellung aus PARAMETER DYNAMIC SCHEMA oder Leerzeichen
SCREENCHECK	ON
TASKTYPE	der aktuelle Wert zum Zeitpunkt des Übersetzens
VERSIONMIX	OFF
XREF	OFF

Beispiele

In der Source "AUFTRAGS" sollen folgende Optionen eingetragen sein:

– Bei SQL-Anweisungen, bei denen explizit kein SCHEMA-Name angegeben ist, soll der Name "TEST" verwendet werden.

– Es sollen Querverweise nach SYSLST ausgegeben werden.

In der Source müssen die OPTION-Angaben **vor** der Anweisung PROCEDURE stehen:

```
OPTION SCHEMA=TEST XREF=ON LISTING=LIST;
PROCEDURE AUFTRAGS;
...
```

COMPILE startet den Übersetzungslauf für die Source "AUFTRAGS". Folgende Optionen sollen dabei abgesetzt werden:

– Der erzeugte Zwischencode soll in die aktuelle DRIVE-Bibliothek unter dem Namen "AUFTRAGC" (Elementtyp X) abgespeichert werden.

– Es sollen keine Querverweise ausgedruckt werden. Da eine OPTION-Angabe bei COMPILE eine OPTION-Angabe aus der Source überschreibt gilt hier: XREF=OFF

– Eine Übersetzungsliste soll erzeugt und in der aktuellen DRIVE-Bibliothek (Elementtyp P) abgespeichert werden. Die Übersetzungsliste erhält denselben Namen wie der Zwischencode ("AUFTRAGC").

```
COMPILE AUFTRAGS INTO AUFTRAGC
        OPTION LISTING=LIBRARY CODE=ON XREF=OFF
```

PARAMETER
PARAMETER-Anweisung auswählen

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Dialog-Modus

PARAMETER gibt eine Menümaske aus, von der aus man in die Menümaske einer der folgenden PARAMETER-Anweisungen verzweigen kann:

– PARAMETER DIAGNOSIS
– PARAMETER DYNAMIC
– PARAMETER KFKEY
– PARAMETER LOCK
– PARAMETER STATIC

Die Menümasken der einzelnen Anweisungen können auch direkt aufgerufen werden. Dazu müssen Sie PARAMETER mit dem entsprechenden Operanden angeben.

```
PARAMETER [ DIAGNOSIS |
            DYNAMIC |
            KFKEY |
            LOCK { DIALOG | PROCEDURE } |
            STATIC ]
```

DIAGNOSIS	Die Menümaske mit den aktuellen Parameterwerten (außer dem Operanden INTTRACE) wird angezeigt. Die Parameterwerte können vom Anwender korrigiert werden. Fehlerhafte Angaben werden an der Datensichtstation wieder angezeigt und können ebenfalls vom Anwender korrigiert werden.
DYNAMIC	Die Menümaske mit den aktuellen Parameterwerten, die vom Anwender korrigiert werden können, wird angezeigt. Fehlerhafte Angaben werden an der Datensichtstation wieder angezeigt und können ebenfalls vom Anwender korrigiert werden.
KFKEY	Die Menümaske mit der aktuellen Tastenbelegung wird angezeigt.

	Im TIAM-Betrieb können die Tastenbelegungen jederzeit korrigiert werden. Fehlerhafte Angaben werden an der Datensichtstation wieder angezeigt und können ebenfalls vom Anwender korrigiert werden. Im UTM-Betrieb können die Tastenbelegungen nicht korrigiert werden.
LOCK	Die Menümaske mit den im Dialog- oder Programm-Modus zu sperrenden Anweisungen wird angezeigt. Die Anweisungen sind alphabetisch aufsteigend sortiert. Sie können vom Anwender gesperrt werden. Fehlerhafte Angaben werden an der Datensichtstation wieder angezeigt und können vom Anwender korrigiert werden.
DIALOG	Die Anweisungen für den Dialog-Modus werden ausgegeben.
PROCEDURE	Die Anweisungen für den Programm-Modus werden ausgegeben.
STATIC	Die Menümaske mit den aktuellen Parameterwerten, die vom Anwender nur beim ersten Aufruf korrigiert werden können, wird angezeigt. Bei jedem weiteren Aufruf können nur Informationen über die Parameterwerte eingeholt werden.

PARAMETER DIAGNOSIS
Protokollierung einschalten

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Dialog-Modus
– im Programm-Modus nur im Verabeitungsteil eines Dialog-Programms

PARAMETER DIAGNOSIS stellt die Protokollierung ein und legt Besonderheiten für die Programmausführung fest.

Die Protokollierung erfolgt in die DRIVE-interne Diagnosedatei DRI.INTTRACE.FILE. Die Datei DRI.INTTRACE.FILE ist eine ISAM-Datei. (Siehe DRIVE-Programmiersystem [1])

Die Anweisung kann sowohl ohne Angabe von Operanden maskenunterstützt als auch mit mindestens einem Operanden angegeben werden.

PARAMETER DIAGNOSIS ohne Operanden kann nur im Dialog-Modus verwendet werden.

Werden mehrere Operanden innerhalb einer Anweisung angegeben, werden sie gleichzeitig wirksam. Wird ein Operand im Dialog- und im Programm-Modus angegeben, gilt die letzte Angabe.

Treten bei der Ausführung Fehler auf, wird die gesamte Anweisung nicht ausgeführt und die fehlerhafte Anweisung markiert wieder ausgegeben. Zusätzlich erscheint in der Meldungszeile eine Fehlermeldung.

Die Anweisung PARAMETER DIAGNOSIS innerhalb eines DRIVE-Programms (Übersetzungseinheit) hat keinen Einfluß auf den Übersetzungslauf dieses Programms.

```
PARAMETER DIAGNOSIS [ ACCOUNT={ ON | OFF } |
                      DBTRACE={ ON | OFF } |
                      DMSTRACE={ ON | OFF } |
                      INTTRACE={ ON | OFF } |
                      MEMTRACE={ ON | OFF } | ...
                      { 'FILEON' | 'FILEOFF' } ]
```

ACCOUNT	Es wird festgelegt, ob eine Performance-Messung durchgeführt wird. Die Meßdaten werden in die Datei `DRI.ACCOUNT.DAT` geschreiben. Diese Datei ist eine ISAM-Datei. Vorbelegung: OFF
DBTRACE	Es wird festgelegt, ob Inhalte der Datenbankschnittstelle protokolliert werden. Die Angabe DBTRACE ist nur möglich, wenn die aktuelle DRIVE-Umgebung Anschluß an SESAM oder UDS bietet. Vorbelegung: OFF
DMSTRACE	Es wird festgelegt, ob die Inhalte der DVS-Schnittstelle protokolliert werden. Die Angabe DMSTRACE ist nur möglich, wenn die aktuelle DRIVE-Umgebung DMS-Anschluß bietet. Vorbelegung: OFF
INTTRACE	INTTRACE dient internen Testzwecken. INTTRACE=ON darf nur nach Aufforderung des Services der Siemens Nixdorf Informationssysteme AG angegeben werden. Vorbelegung: OFF
MEMTRACE	Es werden die Tabellen der Speicherverwaltung protokolliert. Vorbelegung: OFF
'FILEON'	Werden die Anweisungen von SYSDTA gelesen, wird bei einem Programmfehler weiter von SYSDTA gelesen und nicht – wie sonst nach Fehlern üblich – vom Bildschirm gelesen.
'FILEOFF'	Die Einstellung FILEON wird ausgeschaltet.

Zeitpunkt der Auswertung

Die folgende Tabelle enthält eine Übersicht über den Zeitpunkt, zu dem die Operanden der PARAMETER DIAGNOSIS-Anweisung ausgewertet werden.

Operand	**Übersetzungszeitpunkt**	**Ablaufzeitpunkt**
ACCOUNT	ja	ja
DBTRACE	ja	ja
DMSTRACE	-	-
INTTRACE	-	-
MEMTRACE	ja	ja

PARAMETER DISTRIBUTION
Zugriff im verteilten System festlegen

Diese Anweisung ist gültig

– im UTM-Betrieb, bei VTV nur, wenn alle Auftragnehmer-Vorgänge beendet sind
– im Dialog- und Programm-Modus

PARAMETER DISTRIBUTION legt für ENTER- und CALL-Anweisungen den Zugriff auf Programme im entfernten System fest. Die mit PARAMETER DISTRIBUTION gegebene Verteilungsinformation wird bei jeder ENTER-Anweisung im Dialog-Modus ausgewertet. Bei allen anderen Programmaufrufen wird die Verteilungsinformation nur dann ausgewertet, wenn die Anweisung OPTION DISTRIBUTION=ON angegeben wurde.

Die Anweisung muß mit mindestens einem Operanden (STATUS) angegeben werden.

Die Anweisung PARAMETER DISTRIBUTION wird zum Ablaufzeitpunkt eines Programms ausgewertet.

Wird PARAMETER DISTRIBUTION im Dialog- und im Programm-Modus angegeben, gilt die letzte Angabe.

```
PARAMETER DISTRIBUTION

        [ [ LIBRARY=bibliothek ] ELEMENT=elemname [ TYPE={ CODE | OBJECT } ] |
          TAC=tacname ]

        [ APPLICATION=anwendung ]

        STATUS={ OFF | ADD | REMOVE }
```

---

bibliothek

bezeichnet die DRIVE-Bibliothek im entfernten System (max. 54 Zeichen), aus der das Element eingelesen wird.

Wird die DRIVE-Bibliothek im entfernten System mit der Anweisung PARAMETER DYNAMIC LIBRARY voreingestellt, so kann die Angabe von *bibliothek* entfallen.

bibliothek muß angegeben werden, wenn bei der CALL- oder ENTER-Anweisung eine Angabe für die DRIVE-Bibliothek gemacht wird.

bibliothek darf nicht angegeben werden, wenn bei der CALL- oder ENTER-Anweisung keine Angabe für die DRIVE-Bibliothek gemacht wird.

Bei einem entfernten BS2000-System:

bibliothek kann der Dateikettungsname der DRIVE-Bibliothek (entsprechend den BS2000-Konventionen) sein.

DRIVE/WINDOWS interpretiert *bibliothek* zuerst als Dateikettungsnamen, dann als Bibliotheksnamen.

Bei einem entfernten SINIX-System:

Absoluter oder relativer Pfadname eines Dateiverzeichnisses, das die DRIVE-Bibliothek darstellt.

Ein relativer Pfadname bezieht sich auf das Dateiverzeichnis, in dem DRIVE/WINDOWS im entfernten System gestartet wurde.

Die Angaben *bibliothek* und *elemname* werden mit dem für das entfernte System angegebenen *class_name* aus der Anweisung PARAMETER DYNAMIC CLASS oder mit der Vorbelegung des entfernten Systems zu einem Datei-Pfadnamen ergänzt: *bibliothek/class_name/elemname* (siehe Anweisung PARAMETER DYNAMIC).

elemname bezeichnet das Element (max. 31 Zeichen) im entfernten System, das eingelesen wird.

Bei einem entfernten BS2000-System:

Name eines Bibliothekselements, das eine Source (= S-Element), einen Zwischencode (= X-Element) oder einen Objektcode (= R-Element) enthält.

Wird keine *bibliothek* angegeben, wird die für das entfernte System mit PARAMETER DYNAMIC LIBRARY voreingestellte Bibliothek eingesetzt.

Bei einem entfernten SINIX-System:

Name der Datei, die eine Source, einen Zwischencode oder einen Objektcode enthält..

Wird keine *bibliothek* angegeben, wird die für das entfernte System das mit PARAMETER DYNAMIC LIBRARY voreingestellte Dateiverzeichnis eingesetzt. Wenn mit PARAMETER DYNAMIC LIBRARY kein Dateiverzeichnis festgelegt ist, wird das Dateiverzeichnis eingesetzt, das mit der Umgebungsvariablen DRIVE_PROJECTLIB festgelegt ist.

	Die Datei muß im Dateiverzeichnis *bibliothek/class_name* vorhanden sein (siehe Anweisung PARAMETER DYNAMIC CLASS).
TYPE	gibt den Typ des Elements an.
CODE	Vorbelegung Das Element *elemname* ist eine Source oder ein Zwischencode. DRIVE/WINDOWS sucht in einem entfernten BS2000-System nach dem zuletzt bearbeiteten Element und in einem entfernten SINIX-System nach der zuletzt bearbeiteten Datei, unabhängig davon, ob dieses bzw. diese eine Source enthält oder einen Zwischencode.
OBJECT	Das Element *elemname* ist ein Objektcode. TYPE=OBJECT darf nur angegeben werden, wenn das Element *elemname* mit dem DRIVE-Compiler DRIVE/WINDOWS-Comp übersetzt wurde (siehe DRIVE-Compiler für BS2000 [16] oder SINIX [41]).
tacname	bezeichnet das UTM-Teilprogramm, das im entfernten System aufgerufen wird. Zum Ablaufzeitpunkt muß die UTM-Anwendung, zu der das Teilprogramm gehört, gestartet sein. *tacname* darf kein DRIVE-Interpreter-TAC oder DRIVE-Compiler-TAC sein.
anwendung	Name der UTM-Anwendung (max. 8 Zeichen) im entfernten System. Fehlt eine Angabe für *anwendung*, wird die bei der Generierung mit der KDCDEF-Steueranweisung LPAP angegebene entfernte Anwendung eingesetzt.
STATUS	legt fest, wie die Verteilungsinformation behandelt werden soll.
OFF	Die Verteilungsinformation wird insgesamt gelöscht. Bei STATUS=OFF ist die Angabe weiterer Operanden nicht notwendig.
ADD	Ein neuer Eintrag wird in die Verteilungsinformation aufgenommen. Ist bereits ein Element oder ein UTM-Teilprogramm gleichen Namens vorhanden, wird dieses überschrieben.
REMOVE	In der Verteilungsinformation wird das angegebene Element oder das UTM-Teilprogramm gelöscht.

PARAMETER DYNAMIC
Dynamischen Parameter festlegen

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Dialog-Modus
– im Programm-Modus nur im Verabeitungsteil eines Dialog-Programms

PARAMETER DYNAMIC legt Parameterwerte fest, die während der DRIVE-Sitzung beliebig oft verändert werden können. Die Parameterwerte bestimmen Eigenschaften der DRIVE-Sitzung, und zwar sowohl für den Dialog-Modus als auch für die Umgebung, in der DRIVE-Programme ablaufen.

Die Anweisung kann sowohl ohne Angabe von Operanden maskenunterstützt als auch mit mindestens einem Operanden angegeben werden.

PARAMETER DYNAMIC ohne Operanden kann nur im Dialog-Modus verwendet werden.

Werden mehrere Operanden innerhalb einer Anweisung angegeben, werden sie gleichzeitig wirksam. Wird ein Operand im Dialog- und im Programm-Modus angegeben, gilt die letzte Angabe.

Wenn Sie PARAMETER DYNAMIC maskenunterstützt eingeben und es treten bei der Ausführung von PARAMETER DYNAMIC Fehler auf, können bereits Angaben wirksam geworden sein, die nicht mehr rückgängig zu machen sind (z.B. LIBRARY). Die fehlerhafte Anweisung wird markiert wieder ausgegeben. Zusätzlich erscheint in der Meldungszeile eine Fehlermeldung.

Die Anweisung PARAMETER DYNAMIC innerhalb eines DRIVE-Programms (Übersetzungseinheit) hat keinen Einfluß auf den Übersetzungslauf dieses Programms.

```
PARAMETER DYNAMIC [ AUTHORIZATION=berechtigung |
                    CATALOG=sesdbname |
                    DBSYSTEM={ SESAM | SESAMSQL | UDS } |
                    DECIMALSIGN={ , | . } |
                    ERRORATTRIBUTE=( attribute, ... ) |
                    FORMAT={ LINE | TABLE | SEQUENCE } |
                    LETTERS={ CAPITAL | BOTH | UNCHANGED } |
                    LIBRARY=bibliothek |
                    LOG={ OFF | IN | OUT | INOUT } |
                    LOGFILE=filename |
                    LOGPASSWORD=passwort |
                    NORMSQL={ ON | OFF } |
                    NULL { FORM | LIST } nullwert |
                    SCHEMA=schemaname |
                    TEST={ STANDARD | ALL } |
                    USERMSGFILE=name ] ...
```

AUTHORIZATION — legt den aktuellen Berechtigungsschlüssel *berechtigung* (max. 18 Zeichen) für SESAM V2.x-Datenbanken fest (siehe SQL-Lexikon für SESAM V2 [5]).

AUTHORIZATION=*berechtigung* gilt für SQL-Anweisungen, die im Dialog-Modus eingegeben werden, und hat Wirkung auf SQL-Anweisungen in Programmen.

Wenn Sie AUTHORIZATION=*berechtigung* angeben, müssen Sie auch mit PARAMETER DYNAMIC DBSYSTEM=SESAMSQL arbeiten, d.h. die Datenbankfassung SESAM V2.x laden.

Vorbelegung: keine

AUTHORIZATION=*berechtigung* dürfen Sie nur angeben, wenn keine Transaktion offen ist.

CATALOG — Name einer SESAM-Datenbank *sesdbname* (max. 18 Zeichen), auf die zugegriffen wird, wenn in dynamischen SQL-Anweisungen kein Name angegeben wurde (siehe SQL-Lexikon für SESAM V2 [5]).

Wenn Sie CATALOG=*sesdbname* angeben, müssen Sie auch mit PARAMETER DYNAMIC DBSYSTEM=SESAMSQL arbeiten, d.h. die Datenbankfassung SESAM V2.x laden.

	CATALOG=*sesdbname* gilt für SQL-Anweisungen, die im Dialog-Modus eingegeben werden, und kann Wirkung auf dynamische SQL-Anweisungen in Programmen haben (Siehe Anweisung OPTION und SQL-Lexikon für SESAM V2 [5]). Vorbelegung: keine
DBSYSTEM	Angabe des Datenbanksystems, auf das SQL-Anweisungen, die im Dialog-Modus eingeben werden, zugreifen. Diese Angabe gilt auch für SQL-Anweisungen in Programmen, wenn diese Programme mit der Anweisung CALL, DEBUG oder DO aufgerufen werden (siehe Anweisung CALL, DEBUG oder DO). PARAMETER DBSYSTEM kann nur im Dialog-Modus eingegeben werden. Entspricht das angegebene Datenbanksystem nicht der geladenen Variante, so wird die Anweisung PARAMETER DYNAMIC DBSYSTEM abgewiesen. Vorbelegt ist die geladene Variante.
=SESAM	SQL-Anweisungen greifen auf eine SESAM-Datenbank von SESAM V1.x zu.
=SESAMSQL	SQL-Anweisungen greifen auf eine SESAM-Datenbank von SESAM V2.x zu. DBSYSTEM=SESAMSQL muß angegeben werden, wenn die Parameter AUTHORIZATION=*berechtigung* oder CATALOG=*sesdbname* eingestellt sind.
=UDS	SQL-Anweisungen greifen auf eine UDS-Datenbank zu.
DECIMALSIGN	Das Dezimalzeichen bei Eingaben im Dialog-Modus und bei DRIVE-Formaten innerhalb von numerischen Literalen wird festgelegt. Als Dezimalzeichen können das Komma (,) und der Punkt (.) verwendet werden. Vorbelegung: der Punkt (.)
ERRORATTRIBUTE	Zur Kennzeichnung von fehlerhaften Bildschirmfeldern werden Feldattribute zugewiesen.

attribute

Folgende globale Feldattribute zugewiesen werden:

- UNPROTECTED
- HIGHINTENSITY, NORMALINTENSITY
- VISIBLE, SIGN, INVISIBLE
- UNDERLINE, NOUNDERLINE
- GREEN, RED, WHITE, YELLOW

Vorbelegung: HIGHINTENSITY, UNDERLINE

Bei fehlerhaften Eingaben in FHS-Formate erscheint das festgelegte Feldattribut nur, wenn in der DISPLAY screenformat-Anweisung SCREENERROR REPEAT angegeben wird.

FORMAT

Das Ausgabeformat für Feldnamen und den dazugehörenden Werten wird festgelegt.

=LINE

Vorbelegung

Das Ausgabeformat enthält Feldnamen und Werte in einer Zeile.

Beispiel

```
KUNDENNUMMER : K03452
NAME : MUELLER
WOHNORT : BAD ORB
STRASSE : GOETHESTR. 10
```

=TABLE

Das Ausgabeformat enthält Feldnamen und Werte in Tabellenform. In der ersten Bildschirmzeile werden die Feldnamen, in den folgenden Zeilen werden die dazugehörenden Werte ausgegeben. Die Breite der Tabellenspalten wird durch das Maximum der Länge des Feldnamens und seinen Werten bestimmt. So wird weder der Feldname noch einer der dazugehörenden Werte abgeschnitten. Die Tabellenspalten werden jeweils durch ein Leerzeichen voneinander getrennt.

Beispiel

```
KUNDENNUMMER NAME    WOHNORT STRASSE
K03452       MUELLER BAD ORB GOETHESTR. 10
K05734       SCHULZE BERLIN  KANTSTR. 5
K18982       ZINKE   BONN    RHEINSTR. 10
```

TABLE darf nur verwendet werden, wenn die Tabelle vollständig auf den Bildschirm paßt (Zeilenlänge ≤ 80 Zeichen). Ist die Zeilenlänge > 80 Zeichen, wird eine Fehlermeldung ausgegeben.

=SEQUENCE Das Ausgabeformat enthält Feldnamen und Werte in sequentieller Folge.

Beispiel

```
KUNDENNUMMER: K03452  NAME: MUELLER  WOHNORT: BAD ORB  STRAS
SE: GOETHESTR. 10  KUNDENNUMMER: K05734 ...
```

LETTERS Die Behandlung von Kleinbuchstaben bei Eingaben im Dialog-Modus, in den EDT und bei DRIVE-Formaten. Diese Angabe hat Auswirkungen auf Namen und Literale.

=CAPITAL Vorbelegung

Alle Kleinbuchstaben in Namen und Literalen werden in Großbuchstaben umgesetzt.

=BOTH Kleinbuchstaben in Namen werden in Großbuchstaben umgesetzt. Kleinbuchstaben in Literalen werden nicht in Großbuchstaben umgesetzt.

=UNCHANGED Bei Namen und Literalen erfolgt keine Umwandlung von Kleinbuchstaben in Großbuchstaben.

Die Angabe LETTERS=UNCHANGED sollte nicht verwendet werden, weil Software-Produkte wie LMS, EDT, SESAM, UDS, mit denen DRIVE/WINDOWS zusammenarbeitet, klein geschriebene Buchstaben unterschiedlich behandeln.

LIBRARY=bibliothek *bibliothek* bezeichnet die DRIVE-Bibliothek, in der Programme verwaltet werden.

Die angegebene Bibliothek wird für ein System definiert und wird nicht als Parameter an entfernte Systeme weitergegeben.

Vorbelegung: keine

bibliothek kann auch der Dateikettungsname der DRIVE-Bibliothek (entsprechend den BS2000-Konventionen) sein.

DRIVE/WINDOWS interpretiert den angegebenen Namen zuerst als Dateikettungsnamen, dann als Bibliotheksnamen.

Ist die angegebene Bibliothek nicht vorhanden, gibt DRIVE/WINDOWS eine Fehlermeldung aus.

LOG Die Dialog-Protokollierung wird eingestellt. Falls die Dialog-Protokollierungs-Komponente `SYSPRG.DRIVE.021.DRILOG` noch nicht geladen ist, stößt DRIVE/WINDOWS den Batch-Prozeß `SYSENT.DRIVE.011.DRILOG` an (siehe DRIVE-Programmiersystem [1]).

=OFF	Vorbelegung Der Dialog wird nicht protokolliert.
=IN	Alle Eingaben werden protokolliert.
=OUT	Alle Ausgaben werden protokolliert.
=INOUT	Alle Ein- und Ausgaben werden protokolliert.
LOGFILE=filename	Namenserweiterung für den Namen der Dialog-Protokolldatei (max. 20 Zeichen). Der vollständige Dateiname lautet `DRILOG.filename`. Die Dialog-Protokolldatei ist eine SAM-Datei. Vorbelegung: `DRILOG.jjmmtt`.
LOGPASSWORD=passwort	Das Kennwort für die Dialog-Protokolldatei wird festgelegt (max. 4 Zeichen). Vorbelegung: Leerzeichen LOGPASSWORD kann nur in Verbindung mit LOGFILE angegeben werden.
NORMSQL	Es wird vereinbart, ob die Anweisungen den New- oder Old-Style-Anforderungen entsprechend interpretiert werden. NORMSQL darf nur außerhalb einer Transaktion im Dialog-Modus angegeben werden und wenn die aktuelle DRIVE-Umgebung Mischbetrieb erlaubt.
=ON	Vorbelegung Die folgenden DRIVE-Anweisungen werden entsprechend den New-Style-Anforderungen interpretiert.
=OFF	Die folgenden DRIVE-Anweisungen werden entsprechend den Old-Style-Anforderungen interpretiert.
NULL	Für die NULL-Wertdarstellung wird ein Zeichen festgelegt. Anwender können NULL-Werte nur eingeben, wenn ein Zeichen für die Darstellung festgelegt wurde. Wird bei DECLARE FORM oder DECLARE LIST keine NULL-Wertdarstellung festgelegt, gilt die mit PARAMETER vereinbarte Darstellung auch für Programme.
FORM	Die NULL-Wertdarstellung wird für die Bildschirmein-/ausgabe festgelegt. Vorbelegung: das Sonderzeichen @

LIST — Die NULL-Wertdarstellung wird für die Druckerausgabe festgelegt.

Vorbelegung: der Punkt (.)

nullwert — *nullwert* bestimmt die Darstellung des NULL-Wertzeichens für den alphanumerischen Datentypen CHARACTER sowie für die numerischen Datentypen NUMERIC, DECIMAL, INTEGER und SMALLINT (siehe Metavariable *nullwert*).

Die alphanumerische NULL-Wertdarstellung ist auch gültig für die Zeit-Datentypen.

Die numerische NULL-Wertdarstellung ist auch gültig für den Datentyp INTERVAL.

SCHEMA=schemaname

Name einer SESAM-Datenbank von SESAM V1.x (max. 18 Zeichen), eines SESAM-Schemas von SESAM V2.x (max. 31 Zeichen) oder eines UDS-Schemas (max. 30 Zeichen), auf den zugegriffen wird, wenn in dynamischen SQL-Anweisungen kein Name angegeben wurde (siehe SQL-Lexika [4], [5], [6]).

SCHEMA=*schemaname* gilt für SQL-Anweisungen, die im Dialog-Modus eingegeben werden und hat bei SESAM V1.x und UDS keine Wirkung auf SQL-Anweisungen in Programmen. Bei SESAM V2.x kann SCHEMA=*schemaname* Wirkung auf dynamische SQL-Anweisungen in Programmen haben (Siehe Anweisung OPTION und SQL-Lexikon für SESAM V2 [5]).

Ausnahme: Bei der PERMIT-Anweisung muß *schemaname* explizit angegeben werden.

Bei SESAM V1.x und UDS darf SCHEMA=*schemaname* nur im Dialog-Modus eingegeben werden.

Vorbelegung: keine

TEST — Es wird vereinbart, wie sich DRIVE/WINDOWS bei einem Programmabbruch verhält.

=STANDARD — Vorbelegung

Bei einem Programmabbruch wird in den Dialog-Modus verzweigt.

Im TIAM-Betrieb besteht die Möglichkeit, in den EDT zur Fehleranalyse zu verzweigen, wenn das Programm in der EDT-Arbeitsdatei 0 stand.

=ALL — Bei einem Programmabbruch wird DRIVE/WINDOWS beendet. DRIVE/WINDOWS stößt intern eine Druckausgabe an.

USERMSGFILE=name

Meldungsklasse (= erster Teil des Meldungsschlüssels). *name* darf maximal 3 Zeichen lang sein.

Vorbelegung: keine

Einsatzmöglichkeiten der Operanden

Operand	TIAM-Betrieb	UTM-Betrieb	UTM-Startprozedur
AUTHORIZATION	ja	ja	ja
CATALOG	ja	ja	ja
DBSYSTEM	ja	ja	ja
DECIMALSIGN	ja	ja	nein
ERRORATTRIBUTE	ja	ja	ja
FORMAT	ja	ja	nein
LETTERS	ja	ja	nein
LIBRARY	ja	ja	ja
LOG	ja	ja	nein
LOGFILE	ja	ja	nein
LOGPASSWORD	ja	ja	nein
NORMSQL	ja	ja	nein
NULL	ja	ja	nein
SCHEMA	ja	ja	nein
TEST	ja	ja	nein
USERMSGFILE	ja	ja	ja

Zeitpunkt der Auswertung

Die folgende Tabelle enthält eine Übersicht über den Zeitpunkt, zu dem die Operanden der PARAMETER DYNAMIC-Anweisung ausgewertet werden.

Operand	**Übersetzungszeitpunkt**	**Ablaufzeitpunkt**
AUTHORIZATION	ja, wenn nicht durch OPTION anders festgelegt	ja
CATALOG	ja, wenn nicht durch OPTION anders festgelegt	ja
DBSYSTEM	-	-
DECIMALSIGN	nein	ja
ERRORATTRIBUTE	nein	ja
FORMAT	nein	ja
LETTERS	nein	ja
LIBRARY	ja	ja
LOG	nein	nein
LOGFILE	nein	nein
LOGPASSWORD	nein	nein
NORMSQL	-	-
NULL	nein	ja
SCHEMA	ja, wenn nicht durch OPTION anders festgelegt	ja
TEST	ja	nein
USERMSGFILE	ja	ja

PARAMETER KFKEY
K- oder F-Taste belegen

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Dialog-Modus
– im Programm-Modus nur im Verabeitungsteil eines Dialog-Programms

PARAMETER KFKEY belegt K- (Kurznachrichten-) oder F- (Funktions-) Tasten.

Die Anweisung kann sowohl ohne Angabe von Operanden maskenunterstützt als auch mit mindestens einem Operanden angegeben werden.

PARAMETER KFKEY ohne Operanden kann nur im Dialog-Modus verwendet werden.

Werden mehrere Operanden innerhalb einer Anweisung angegeben, werden sie gleichzeitig wirksam. Wird ein Operand im Dialog- und im Programm-Modus angegeben, gilt die letzte Angabe.

Treten bei der Ausführung Fehler auf, wird die gesamte Anweisung nicht ausgeführt und die fehlerhafte Anweisung markiert wieder ausgegeben. Zusätzlich erscheint in der Meldungszeile eine Fehlermeldung.

Im TIAM-Betrieb können die Tastenbelegungen jederzeit korrigiert werden. Fehlerhafte Angaben werden an der Datensichtstation wieder angezeigt und können korrigiert werden.

Die Anweisung PARAMETER KFKEY innerhalb eines DRIVE-Programms (Übersetzungseinheit) hat keinen Einfluß auf den Übersetzungslauf dieses Programms.

```
PARAMETER { KFKEY [=literal [ ACTION={ BREAK | EXIT | DELETE } ]
          [ UTMRC=literal ] ] } ...
```

KFKEY=literal — K- oder F-Tasten werden festgelegt und dieser Wert in die Systemvariable &KFKEY geschrieben.

literal darf die Werte K1 und K3 bis K14 oder F1 bis F20 annehmen. *literal* darf auch sedezimal angegeben werden.

Die Belegung der Systemvariablen &KFKEY steht für die Verarbeitung in Dialog-Programmen bis zum Quittieren der nächsten Bildschirmausgabe mit der DUE-Taste zur Verfügung. Nach dem Quittieren wird die Systemvariable &KFKEY wieder mit Leerzeichen belegt.

ACTION

Den Tasten werden Funktionen zugewiesen.

Im TIAM-Betrieb kann der K2-Taste keine Funktion zugeordnet werden. Die K2-Taste dient im TIAM-Betrieb dazu, von DRIVE/WINDOWS in den BS2000-Systemmodus zu wechseln.

Im UTM-Betrieb können die Tasten nur in der UTM-Startprozedur belegt werden. Die Belegung muß eindeutig sein, d.h. Tasten und UTM-Returncodes dürfen jeweils nur einmal bei der Eingabe vorkommen.

Der Anwender ist für die korrekte Behandlung der Tastenbelegung und im UTM-Betrieb für die Übereinstimmung der Tastenbelegung von KDCDEF und der über PARAMETER vorgenommenen Tastenbelegung verantwortlich.

Wird im Programm-Modus zur Ablaufsteuerung eines DRIVE-Programms eine Taste betätigt, der keine Funktion zugewiesen wurde, wird die Systemvariable &KFKEY mit Leerzeichen versorgt.

=BREAK

Die mit KFKEY zugewiesene Taste erhält die Funktion BREAK (siehe Anweisung BREAK).

Vorbelegung: BREAK ist der K1-Taste zugewiesen. Diese Belegung kann vom Anwender aufgehoben werden.

=EXIT

Die mit KFKEY zugewiesene Taste erhält die Funktion EXIT. Wird diese Taste gedrückt, wird die DRIVE-Sitzung abgebrochen und alle offenen Transaktionen zurückgesetzt (siehe Anweisung EXIT).

=DELETE

Die Belegung der Taste wird gelöscht.

UTMRC=literal

Den Tasten werden UTM-Returncodes zugewiesen. *literal* darf die Werte 20Z bis 39Z annehmen.

UTMRC=*literal* darf nur in der UTM-Startprozedur verwendet werden. Dort muß es aber auch angegeben werden.

Beziehungen zu anderen Anweisungen

– Im UTM-Betrieb dürfen Programme, die mit der Compiler-Option OPTION OBJECT=ON übersetzt werden, nicht die Anweisung PARAMETER KFKEY enthalten. Statt dessen kann die KDCDEF-Steueranweisung SFUNC sowie ein DRIVE-Parameter in der UTM-Startprozedur angegeben werden (siehe DRIVE-Programmiersystem [1]).

– Im TIAM-Betrieb dürfen Programme, die mit der Compiler-Option OPTION OBJECT=ON übersetzt werden, nur dann Anweisung PARAMETER KFKEY enthalten, wenn sie mit DO aufgerufen werden.

PARAMETER LOCK
Anweisung sperren

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Dialog-Modus
– im Programm-Modus nur im Verabeitungsteil eines Dialog-Programms

PARAMETER LOCK sperrt Anweisungen für eine DRIVE-Sitzung. In derselben Sitzung kann diese Sperre nicht wieder aufgehoben werden.

Die Anweisung kann sowohl ohne Angabe von Operanden maskenunterstützt als auch mit mindestens einem Operanden angegeben werden.

PARAMETER LOCK ohne Operanden kann nur im Dialog-Modus verwendet werden.

Werden mehrere Operanden innerhalb einer Anweisung angegeben, werden sie gleichzeitig wirksam. Wird ein Operand im Dialog- und im Programm-Modus angegeben, gilt die letzte Angabe.

Treten bei der Ausführung Fehler auf, wird die gesamte Anweisung nicht ausgeführt und die fehlerhafte Anweisung markiert wieder ausgegeben. Zusätzlich erscheint in der Meldungszeile eine Fehlermeldung.

Die Anweisung PARAMETER LOCK PROCEDURE wird zum Ablaufzeitpunkt eines Programms ausgewertet.

```
PARAMETER LOCK { DIALOG | PROCEDURE }

               { ALL | { anweisung={ ON | OFF } } ... }
```

DIALOG	Die Anweisungen werden für den Dialog-Modus gesperrt.
PROCEDURE	Die Anweisungen werden für den Programm-Modus gesperrt.
ALL	Alle Anweisungen des Dialog- oder des Programm-Modus werden gesperrt (Ausnahme: EXIT).
anweisung	Folgende Anweisungen können gesperrt werden: ACQUIRE ALTER TABLE BREAK [CYCLE | PROCEDURE | SUBPROCEDURE]

```
CALL
CASE
CLEAR
CLOSE { CURSOR | REPORT }
COMMIT
CONTINUE
CREATE { SCHEMA | TABLE | TEMPORARY VIEW | VIEW }
CYCLE [ CURSOR | FOR | WHILE ]
DEBUG
DECLARE { CONSTANT | CURSOR | FILE | FORM | LIST |
   REPORT | SCREEN | TYPE | VARIABLE }
DELETE
DETAIL
DISPATCH
DISPLAY [ FORM | LIST ]
DO
DROP { CURSOR(S) | SCHEMA | TABLE | TEMPORARY VIEW(S) | VIEW }
EDT
ENTER
EXECUTE
FETCH
FILL { FORM | LIST | REPORT }
GLOBAL
GRANT
GROUP
IF
INSERT
LIST
OPEN { CURSOR | REPORT }
PAGE
PARAMETER
PERMIT
PRINT
PROCEDURE
REPEAT
REPORT
RESTORE
REVOKE
ROLLBACK
SAVE
SELECT
SEND  MESSAGE
SET [ CATALOG |  SCHEMA | SESSION | TRANSACTION ]
SHOW
```

SOURCE
STANDARD
STOP
STORE
SUBPROCEDURE
SYSTEM
UNSAVE
UPDATE
WHENEVER

=ON	Die Anweisung ist für den Anwender gesperrt.
=OFF	Vorbelegung Die Anweisung ist für den Anwender erlaubt.

PARAMETER STATIC
Statischen Parameter festlegen

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Dialog-Modus
– im Programm-Modus nur im Verabeitungsteil eines Dialog-Programms

PARAMETER STATIC legt Parameterwerte fest, die für die gesamte DRIVE-Sitzung gelten. Die Parameterwerte bestimmen Eigenschaften der DRIVE-Sitzung, und zwar sowohl für den Dialog-Modus als auch für die Umgebung, in der DRIVE-Programme ablaufen.

Die Anweisung kann sowohl ohne Angabe von Operanden maskenunterstützt als auch mit mindestens einem Operanden angegeben werden.

PARAMETER STATIC ohne Operanden kann nur im Dialog-Modus verwendet werden.

Werden mehrere Operanden innerhalb einer Anweisung angegeben, werden sie gleichzeitig wirksam. Wird ein Operand im Dialog- und im Programm-Modus angegeben, dürfen nur Operanden festgelegt werden, die noch nicht vorher versorgt worden sind.

Treten bei der Ausführung von PARAMETER STATIC Fehler auf, können bereits Angaben wirksam geworden sein, die nicht mehr rückgängig zu machen sind.

Die Anweisung PARAMETER STATIC innerhalb eines DRIVE-Programms (Übersetzungseinheit) hat keinen Einfluß auf den Übersetzungslauf dieses Programms.

```
PARAMETER STATIC [ FIRSTPAGE={ ON | OFF } |
                   FORMLIB=formatbibliothek |
                   LASTPAGE={ ON | OFF } |
                   OLDSTYLE={ SESAM | SESAMSQL | LEASY | DMS } |
                   USER=username ] ...
```

FIRSTPAGE

Die Ausgabe der mit LIST * erzeugten ersten Seite mit dem Listenkopf wird bei Angabe von OFF unterdrückt.

Vorbelegung: ON

FIRSTPAGE darf nur in der UTM-Startprozedur angegeben werden.

FORMLIB=formatbibliothek

Name der Formatbibliothek, in der die FHS-Formate gespeichert sind.

Vorbelegung: keine

LASTPAGE

Die Ausgabe der mit LIST * erzeugten letzten Seite mit dem Listenfuß wird bei Angabe von OFF unterdrückt.

Vorbelegung: ON

LASTPAGE darf nur in der UTM-Startprozedur angegeben werden.

OLDSTYLE

Angabe des Datenhaltungssystems, auf das eine DRIVE-Old-Style-Anwendung zugreift.

OLDSTYLE darf nur im TIAM-Betrieb angegeben werden.

Vorbelegung: die geladene (New-Style-)Variante

=SESAM

Die DRIVE-Old-Style-Anwendung greift auf eine SESAM-Datenbank von SESAM V1.x zu.

=SESAMSQL

Die DRIVE-Old-Style-Anwendung greift auf eine SESAM-Datenbank von SESAM V2.x zu.

=LEASY

Die DRIVE-Old-Style-Anwendung greift auf LEASY-Dateien zu.

=DMS

Die DRIVE-Old-Style-Anwendung greift auf DMS-Dateien zu.

USER=username

USER darf nur im TIAM-Betrieb angegeben werden.

Mit *username* (max. 8 Zeichen) wird ein Benutzername festgelegt.

username darf die Sonderzeichen / \ * ? [ ] und das Leerzeichen (%) nicht enthalten.

username muß als Literal angegeben werden.

Solange noch keine SQL-Anweisung eingegeben wurde, kann *username* vergeben werden. Danach wird für *username* die TSN eingetragen.

Vorbelegung: Leerzeichen

Einsatzmöglichkeiten der Operanden

Operand	TIAM-Betrieb	UTM-Betrieb	UTM-Startprozedur
FIRSTPAGE	nein	ja	ja
FORMLIB	ja	nein	nein
LASTPAGE	ja	nein	ja
OLDSTYLE	ja	nein	nein
USER	ja	nein	nein

Zeitpunkt der Auswertung

Die folgende Tabelle enthält eine Übersicht über den Zeitpunkt, zu dem die Operanden der PARAMETER STATIC-Anweisung ausgewertet werden.

Operand	Übersetzungszeitpunkt	Ablaufzeitpunkt
FIRSTPAGE	nein	nein
FORMLIB	ja	ja
LASTPAGE	nein	nein
OLDSTYLE	nein	ja
USER	ja	ja

PROCEDURE
Programm beginnen

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

PROCEDURE kennzeichnet den Anfang eines Programms. Über PROCEDURE kann ein Programm von einem anderen Programm Parameter übernehmen. Dazu muß für jeden übernommenen Parameter in der USING-Klausel eine Variable definiert werden. Wird ein Programm im Dialog-Modus aufgerufen, bei dem nach USING die RETURN-Anweisung folgt, dann wird RETURN ignoriert.

Die Parameter des rufenden Programms werden einzeln in die Variablen des Folgeprogramms übertragen, nachdem im Folgeprogramm für jeden Parameter eine Variable definiert wurde. Anzahl und Format der Variablen im rufenden und im gerufenen Programm müssen zuweisungsverträglich sein. Bei der Übergabe darf die Länge aller Parameter (Beschreibung und Werte) 31 Kbyte nicht überschreiten, wenn das Programm mit DO oder ENTER aufgerufen wurde.

In Programmen, die mit dem DRIVE-Compiler DRIVE/WINDOWS-Comp übersetzt werden sollen, ist die USING-Klausel nicht erlaubt, wenn das Programm als Hauptprogramm im TIAM-Betrieboder wenn es als First-TAC im im UTM-Betrieb eingesetzt werden soll.

Innerhalb des Programms müssen zuerst die deklarativen Anweisungen stehen. Danach müssen sämtliche Subprozeduren definiert werden. Erst danach kommt der Verarbeitungsteil.

Stellt DRIVE/WINDOWS bei einem Programm Syntax- oder Semantikfehler fest, werden alle bisher in diesem Programm definierten Größen zurückgesetzt.

Das Ende eines Programms wird durch END PROCEDURE festgelegt.

```
PROCEDURE progname

    [ USING { [ RETURN ] [ level ] varname datendef }, ... ]
```

---

progname — Name des Programms (max. 31 Zeichen). Dieser Name muß nicht identisch mit dem Elementnamen sein, unter dem die Source gespeichert ist.

USING	Mit USING werden Parameter an ein Programm übergeben. USING muß angegeben werden, wenn beim Aufruf des Programms (CALL, DO, ENTER oder DEBUG) die USING-Klausel verwendet werden soll.
RETURN	Mit RETURN werden die Parameter gekennzeichnet, die vom gerufenen Programm wieder an das rufende Programm zurückgegeben werden sollen. Der Aufruf muß über CALL oder DEBUG erfolgen. Der entsprechende Parameter muß bei CALL ebenfalls in der USING-Klausel mit RETURN gekennzeichnet sein.
level	Stufennummer. Die Stufennummer muß 1 sein.
varname	Der Name einer einfachen Variablen. *varname* muß mit dem "&"-Zeichen beginnen und darf insgesamt max. 32 Zeichen lang sein. Es können einfache Variablen, Vektoren, Matrizen, Datengruppen und Wiederholungsgruppen definiert werden. Der Wertebereich einer Variablen darf ausschließlich seines Indikatorwertebereiches nicht größer als 32 Kbyte sein.
datendef	Datentyp der Variablen (siehe Metavariable *datendef*). *datendef* darf keine INIT- und keine REDEFINES-Klausel enthalten (siehe Anweisung DECLARE VARIABLE).

Beispiel

An das Programm "mitarb1" wird die alphanumerische Variable &vgl1, die 4 Zeichen lang ist, übergeben.

```
PROCEDURE mitarb1 USING &vgl1 CHAR(4);
```

READ FILE
Datei lesen

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

READ FILE liest in einer zum Lesen geöffneten Datei an der aktuellen Dateiposition einen Datensatz und überträgt diesen Datensatz in eine Variable oder in eine Liste von Variablen.

Ist der Datensatz länger als die Variable oder die Liste von Variablen, schneidet DRIVE/WINDOWS den Rest ab. Die Systemvariable &ERROR erhält den Eintrag "TOO LONG".

Ist der Datensatz kürzer als die Variable oder die Variablenliste, so gilt:

– Numerische Felder, die nicht vollständig belegt werden können, erhalten den NULL-Wert. Die Systemvariable &ERROR erhält den Eintrag "TOO SHORT".
– Alphanumerische Felder, die nicht belegt werden können, erhalten den NULL-Wert. Die Systemvariable &ERROR erhält den Eintrag "TOO SHORT".
– Alphanumerische Felder mit fester Länge (CHARACTER), die nicht vollständig belegt werden können, erhalten die Teilwerte und werden mit Leerzeichen aufgefüllt. Die Systemvariable &ERROR erhält den Eintrag "TOO SHORT".
– Alphanumerische Felder mit variabler Länge (VARCHAR) erhalten die Teilwerte. Die Systemvariable &ERROR erhält den Eintrag "TOO SHORT".

Nach dem Lesen weist die aktuelle Dateiposition auf den nächsten Satz.

Außerdem trägt DRIVE/WINDOWS die physikalische Satzlänge in die Systemvariable &PHYS_REC_LENGTH ein und die DRIVE-Satzlänge in die Systemvariable &DRIVE_REC_LENGTH.

Wenn die READ FILE-Anweisung bei einem Leseversuch keine Zeichen in der Datei findet, ist das Dateiende erreicht. Bei Erreichen des Dateiendes erhält die Systemvariable &ERROR den Eintrag "OK END".

Wenn ein Programm auf Dateien zugreift, die mit dem OPEN-Modus UPDATE, INOUT oder OUTIN geöffnet sind, darf die Anweisung READ FILE nicht unmittelbar auf die Anweisung WRITE FILE folgen. Zwischen der READ FILE- und der WRITE FILE-Anweisung muß mindestens eine Anweisungen zur Positionierung (SET FILE POSITION) stehen.

```
READ FILE datei INTO variable, ...
```

datei	Logischer Name einer Datei, aus der gelesen wird. Mit diesem Namen muß die Datei im Programm mit der Anweisung DECLARE FILE definiert sein.
variable	Bezeichnung der Variablen, in die der gelesene Datensatz übertragen wird (siehe Metavariable *variable*).

REMOVE
Testpunkt und Aktion löschen

Diese Anweisung ist gültig

– im TIAM-Betrieb
– im Debug-Modus

REMOVE löscht Testpunkte und Aktionen an Testpunkten, die mit AT gesetzt wurden. Außerdem werden Durchlaufzähler gelöscht, die mit der Anweisung AT ... COUNT vereinbart wurden.

Wird keine Aktion angegeben, werden die angegebenen Testpunkte vollständig gelöscht.

```
REMOVE

   { [ bibliothek(elemname) | elemname ] { zeile ... | zeile1 - zeile2 | ALL } |
     * }

   [ COUNT | DISPLAY | SET ]
```

---

bibliothek — Name der DRIVE-Bibliothek (max. 54 Zeichen), aus der das Programm eingelesen wird (gilt nur für externe Programme in DRIVE).

bibliothek kann auch der Dateikettungsname der DRIVE-Bibliothek (entsprechend den BS2000-Konventionen) sein.

DRIVE/WINDOWS interpretiert *bibliothek* zuerst als Dateikettungsnamen, dann als Bibliotheksnamen.

Wird die DRIVE-Bibliothek mit der Anweisung PARAMETER DYNAMIC LIBRARY voreingestellt, so kann die Angabe von *bibliothek* entfallen.

elemname — Name des Elements (max. 31 Zeichen), das das Programm enthält.

DRIVE/WINDOWS sucht nach dem zuletzt bearbeiteten Element, unabhängig davon, ob dieses eine Source enthält (S-Element) oder einen Zwischencode (X-Element).

Wird keine Angabe für *bibliothek* gemacht, wird die bei PARAMETER DYNAMIC LIBRARY angegebene Bibliothek eingesetzt.

*	Der zuletzt eingegebene Testpunkt wird gelöscht (siehe auch Anweisung AT).
zeile	Die Angabe von *zeile* bezieht sich auf eine Zeilennummer in der Übersetzungsliste. Alle Testpunkte in dieser Zeile werden gelöscht. Es können mehrere Zeilennummern angegeben werden.
zeile1-zeile2	Die Angabe bezieht sich auf Zeilennummern in der Übersetzungsliste. Alle Testpunkte in diesem Bereich werden gelöscht. *zeile1* muß kleiner als *zeile2* sein.
ALL	Alle im Programm gesetzten Testpunkte werden gelöscht.
COUNT	Nur Aktionen vom Typ COUNT werden gelöscht.
DISPLAY	Nur Aktionen vom Typ DISPLAY werden gelöscht.
SET	Nur Aktionen vom Typ SET werden gelöscht.

REMOVE BOX
Dialog-Box entfernen

Diese Anweisung ist gültig

– im UTM-Betrieb, aber nicht im UTM-Asynchronbetrieb und bei VTV nicht in Auftragnehmer-Umgebung
– im Programm-Modus

REMOVE BOX entfernt eine, mehrere oder alle ausgegebenen Dialog-Boxen. Es dürfen nur so viele Dialog-Boxen entfernt werden, wie ausgegeben sind, sonst bricht DRIVE/WINDOWS das Programm ab.

REMOVE BOX wird erst bei der nächsten Bildschirmausgabe wirksam. Abhängig von der Anweisung für die nächste Bildschirmausgabe, hat REMOVE BOX ein unterschiedliches Verhalten:

– DISPLAY screenformat
  Vor der Ausgabe des FHS-Formats werden alle ausgegebenen Dialog-Boxen entfernt.
– ADD BOX Vor der Ausgabe der Dialog-Box wird die gewünschte Anzahl der ausgegebenen Dialog-Boxen entfernt.
– REPLACE BOX Vor der Ausgabe der Dialog-Box wird die gewünschte Anzahl der ausgegebenen Dialog-Boxen (= Summe aus der REMOVE BOX- und REPLACE BOX-Anweisung) entfernt.
– SEND MESSAGE Wenn die Meldung in einer Meldungsbox ausgegeben wird, wird vor der Ausgabe der Meldungsbox die gewünschte Anzahl der ausgegebenen Dialog-Boxen entfernt.
  Wenn die Meldung im Meldebereich des Teilformats ausgegeben wird, werden vor der Ausgabe des Teilformats alle ausgegebenen Dialog-Boxen entfernt.
– DISPLAY formatname
  Vor der Ausgabe des dynamischen Formats werden alle ausgegebenen Dialog-Boxen entfernt.
– DRIVE-Meldungen
  Vor der Ausgabe von DRIVE-Meldungen z.B. bei Programmende oder -abbruch werden alle ausgegebenen Dialog-Boxen entfernt.

Wenn vor einer Bildschirmausgabe eine weitere REMOVE BOX-Anweisung angegeben wird, addiert sich die Anzahl der zu entfernenden Dialog-Boxen.

REMOVE BOX ohne Angabe eines Operanden entfernt die zuletzt ausgegebene Dialog-Box.

```
REMOVE [ n | ALL ] BOX
```

---

n	Die *n* obersten Dialog-Boxen werden entfernt. *n* muß eine ganze positive Zahl sein. *n* kann als Variable angegeben werden (siehe Metavariable *variable*).
ALL	Alle ausgegebenen Dialog-Boxen werden entfernt.

REPEAT
Anweisung wiederholen

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Dialog-Modus

REPEAT zeigt die zuletzt eingegebene Anweisung nochmals am Bildschirm an. Der Anwender kann dann entscheiden, ob die Anweisung ausgeführt oder abgeändert werden soll.

Im UTM-Betrieb wird nach der ROLLBACK-Anweisung der Sicherstellungsbereich auf den letzten Konsistenzpunkt zurückgesetzt.

REPEAT wirkt nur bei folgenden SQL-Anweisungen:

– CREATE
– DECLARE
– DELETE { POSITIONED | SEARCHED }
– FETCH
– INSERT INTO
– SELECT
– UPDATE { POSITIONED | SEARCHED }

REPEAT

REPLACE BOX
Dialog-Box ersetzen

Diese Anweisung ist gültig

– im UTM-Betrieb, aber nicht im UTM-Asynchronbetrieb und bei VTV nicht in Auftragnehmer-Umgebung
– im Programm-Modus

REPLACE BOX ersetzt eine, mehrere oder alle ausgegebenen Dialog-Boxen durch eine neue Dialog-Box, die Sie zuvor mit IFG erstellt haben (siehe IFG [28]). REPLACE BOX wirkt wie die ADD BOX- und REMOVE BOX-Anweisung zusammen.

Es dürfen nur so viele Dialog-Boxen ersetzt werden, wie ausgegeben sind, sonst bricht DRIVE/WINDOWS das Programm ab.

Bereits ausgegebene Bildschirmformate (Teilformate und Dialog-Boxen), die nicht ersetzt werden, bleiben erhalten, werden aber von der neuen Dialog-Box überlagert und sind gesperrt, d.h. es sind keine Benutzereingaben in diese Bildschirmformate möglich.

Die zuletzt ausgegebene Dialog-Box ist die aktuelle Dialog-Box. Benutzereingaben sind nur in dieser aktuellen Dialog-Box möglich.

REPLACE BOX ohne Angabe eines Operanden entfernt die zuletzt ausgegebene Dialog-Box.

```
REPLACE { n | ALL } BOX BY dialogbox
        [ POSITION ( zeile1 , spalte1 ) ] [ TO feld1 ]
        [ CURSOR { POSITION ( zeile2 , spalte2 ) | TO feld2 }
        [ MESSAGE schluessel [ POSITION ( zeile3 , spalte3 ) | TO feld3 ]
```

---

n	Die *n* obersten Dialog-Boxen werden entfernt. *n* muß eine ganze positive Zahl sein. *n* kann als Variable angegeben werden (siehe Metavariable *variable*).
ALL	Alle ausgegebenen Dialog-Boxen werden entfernt.
dialogbox	Name des FHS-DE-Formats (max. 7 Zeichen). Das Format muß mit IFG mit der Eigenschaft "Anzeige in einer Box" erstellt worden sein. Das Format muß im Deklarationsteil des Programms mit der Anweisung DECLARE SCREEN definiert sein.

POSITION	legt die Position der Dialog-Box fest. Die Position wird über den Start- oder Bezugspunkt festgelegt. Der Startpunkt ist das erste Zeichen (links oben) der Dialog-Box. Der Bezugspunkt ist das erste Zeichen von *feld1*, falls *feld1* angegeben ist, oder die linke obere Ecke der darunterliegenden Dialog-Box / des darunterliegenden FHS-DE-Teilformats. Wenn Sie POSITION nicht oder mit (0,0) angeben, versucht DRIVE/WINDOWS, die Dialog-Box mit der Standardverschiebung (+2,+2) zum Bezugspunkt zu positionieren. Ist dies nicht möglich, wird die Dialog-Box so verschoben, daß sie auf den Bildschirm paßt. Wenn POSITION angegeben ist, aber der Platz an der vorgegebenen Position für die Dialog-Box nicht ausreicht, bricht UTM den Vorgang mit PEND ER ab.
zeile1	Zeilenabstand zwischen Bezugspunkt und Startpunkt der Dialog-Box. *zeile1* muß eine ganze Zahl sein.
spalte1	Spaltenabstand zwischen Bezugspunkt und Startpunkt der Dialog-Box. *spalte1* muß eine ganze Zahl sein.
feld1	Feld im zuletzt ausgegebenen FHS-DE-Format (Teilformat und Dialog-Box). *feld1* muß eine einfache Komponente der zugehörigen SCREEN-Variablen sein. Wenn *variable* nicht die Komponente der SCREEN-Variablen zum zuletzt ausgegebenen Bildschirmformat ist, bricht UTM den Vorgang mit PEND ER ab. Damit die Komponenten von SCREEN-Variablen eindeutig zugeordnet werden können, müssen sich die ersten 8 Zeichen der Feldnamen im zuletzt ausgegebenen Bildschirmformat unterscheiden.
CURSOR	Die Schreibmarke wird an eine bestimmte Stelle auf den Bildschirm gesetzt. CURSOR dürfen Sie nur angeben, wenn mit IFG für *dialogbox* das Globalattribut "Schreibmarkenposition" gesetzt wurde (siehe IFG [28]).
POSITION	legt die absolute Position (Zeile / Spalte) der Schreibmarke fest.
zeile2	Zeile (1 ≤ *zeile2* ≤ Anzahl der Bildschirmzeilen). *zeile2* muß eine ganze Zahl sein.
spalte2	Spalte (1 ≤ *spalte2* ≤ Anzahl der Bildschirmspalten). *spalte2* muß eine ganze Zahl sein.

TO	Die Schreibmarke wird auf das erste Zeichen des Felds *feld2* gesetzt. Bei Listen wird die Schreibmarke auf die erste Spalte und die erste Zeile des Listenbereichs gesetzt.
feld2	Feld in der Dialog-Box, die ausgegeben werden soll. *feld2* muß eine Komponente der SCREEN-Variablen zu *dialogbox* sein. Damit die Komponenten von SCREEN-Variablen eindeutig zugeordnet werden können, müssen sich die ersten 8 Zeichen der Feldnamen in der aktuellen Dialog-Box unterscheiden.
MESSAGE	Die FHS-DE-Meldung mit Meldungsschlüssel *schluessel*, die Sie IFG erstellt haben, wird ausgegeben. Abhängig von der Festlegung mit IFG erfolgt die Ausgabe entweder in einer Meldungsbox oder im Meldebereich der Dialog-Box. MESSAGE dürfen Sie nur angeben, wenn mit IFG für *dialogbox* das Globalattribut "Meldungskennzeichen" gesetzt wurde (siehe IFG [28]).
schluessel	Meldungsschlüssel der FHS-DE-Meldung. *schluessel* kann als Variable (siehe Metavariable *variable*) oder als alphanumerisches Literal (siehe *charliteral* bei Metavariable *literal*) angegeben werden. *schluessel* muß in der Form AAAAnnn angegeben werden. A steht für einen Buchstaben (A-Z), n für eine Ziffer (0-9). Für AAAA darf nicht IDHS stehen.
POSITION	legt die absolute Position der Meldungsbox fest. Die Meldungsbox wird mit einer zusätzlichen Verschiebung (+2,+2) zu *zeile3*, *spalte3* positioniert. POSITION wird nur ausgewertet, wenn mit IFG festgelegt wurde, daß die Meldung in eine Meldungsbox ausgegeben werden soll. Wenn eine Meldung in einer Meldungsbox ausgegeben werden soll, und Sie weder POSITION noch TO angeben, wird die Meldungsbox in die Mitte des Bildschirms plaziert. Wenn eine mit MESSAGE POSITION positionierte Meldungsbox eine Schreibmarke überdeckt, die mit CURSOR POSITION gesetzt ist, wird MESSAGE POSITION ignoriert.
zeile3	Zeile ($1 \leq$ *zeile3* $\leq$ Bildschirmzeilen). *zeile3* muß eine ganze Zahl sein.
spalte3	Spalte ($1 \leq$ *spalte3* $\leq$ Bildschirmspalten). *spalte3* muß eine ganze Zahl sein.

TO	legt fest, daß die Meldungsbox mit der Standardverschiebung (+2,+2) zu *feld3* positioniert werden soll. TO wird nur ausgewertet, wenn mit IFG festgelegt wurde, daß die Meldung in eine Meldungsbox ausgegeben werden soll. Wenn eine Meldung in einer Meldungsbox ausgegeben werden soll, und Sie weder TO noch POSITION angeben, wird die Meldungsbox in die Mitte des Bildschirms plaziert.
feld3	Feld in der Dialog-Box, die ausgegeben werden soll. *feld3* muß eine Komponente der SCREEN-Variablen zu *dialogbox* sein. Damit die Komponenten von SCREEN-Variablen eindeutig zugeordnet werden können, müssen sich die ersten 8 Zeichen der Feldnamen in der aktuellen Dialog-Box unterscheiden.

SAVE
EDT-Arbeitsdatei 0 speichern

Diese Anweisung ist gültig

– im TIAM-Betrieb
– im Dialog-Modus

SAVE speichert den Inhalt der EDT-Arbeitsdatei 0 als Element vom Typ S in eine DRIVE-Bibliothek (z.B Sourcen, COPY-Elemente, Benutzerkennsätze).

Ist unter diesem Namen bereits ein Element vom Typ S in der DRIVE-Bibliothek vorhanden, wird die Meldung `DRI0046 <elemname> UEBERSCHREIBEN? ANTWORT: (Y=JA, N=NEIN)` ausgegeben.

```
SAVE { bibliothek(elemname) | elemname }
```

---

bibliothek
: Name der DRIVE-Bibliothek (max. 54 Zeichen), in der der Inhalt der EDT-Arbeitsdatei 0 gespeichert wird.

  bibliothek kann auch der Dateikettungsname der DRIVE-Bibliothek (entsprechend den BS2000-Konventionen) sein.

  DRIVE/WINDOWS interpretiert *bibliothek* zuerst als Dateikettungsnamen, dann als Bibliotheksnamen.

  Wird die DRIVE-Bibliothek mit der Anweisung PARAMETER DYNAMIC LIBRARY voreingestellt, so kann die Angabe von *bibliothek* entfallen.

elemname
: Name des Elements (max. 31 Zeichen), in das der Inhalt der EDT-Arbeitsdatei 0 gespeichert wird. Dieser Name muß nicht identisch mit dem Programmnamen sein, der bei PROCEDURE angegeben ist.

  Das Element wird in die angegebene DRIVE-Bibliothek gespeichert. Wird keine *bibliothek* angegeben, wird die bei PARAMETER DYNAMIC LIBRARY angegebene Bibliothek eingesetzt.

SEND MESSAGE
Meldung ausgeben

Diese Anweisung ist gültig

- im TIAM-Betrieb
- im UTM-Betrieb, aber nicht im UTM-Asynchronbetrieb und bei VTV nicht in Auftragnehmer-Umgebung
- im Programm-Modus

Mit der Anweisung SEND MESSAGE gibt man eine Nachricht auf dem Bildschirm aus.

Die Nachricht wird in die Meldungszeile (letzte Bildschirmzeile) geschrieben. Ist die Nachricht länger als die Meldungszeile, wird sie verkürzt mit ">>>" ausgegeben. Der übrige Bildschirminhalt bleibt erhalten.

Falls sich eine Dialog-Box auf dem Bildschirm befindet und die Anweisung SEND MESSAGE ausgeführt werden soll, wird die Meldung in einer Dialog-Box, die DRIVE/WINDOWS zur Verfügung stellt, ausgegeben.

```
SEND MESSAGE { ausdruck [ mask ] | BLANK n | TABULATOR n }, ...

             [ [ WITHOUT ] WAIT ]
```

---

ausdruck	Meldung, die ausgegeben werden soll.
mask	Angaben zur Ausgabeaufbereitung der Meldung (siehe Metavariable *mask*).
BLANK n	Es werden *n* Leerzeichen ausgegeben. *n* muß eine vorzeichenlose ganze Zahl sein. Die Nachricht darf einschließlich Leerzeichen maximal 79 Zeichen lang sein. Ist die Nachricht länger, gilt: Wird nur ein *ausdruck* ausgegeben, wird die Nachricht abgeschnitten und mit der Zeichenfolge ">>>" beendet. Werden mehrere Ausdrücke ausgegeben, wird das DRIVE-Programm mit einer Fehlermeldung abgebrochen.

TABULATOR n	Angabe der Spaltenposition, auf die die Schreibmarke gesetzt werden soll. Ein Zwischenraum zwischen Meldung und Schreibmarkenposition wird dabei mit Leerzeichen aufgefüllt. *n* muß eine vorzeichenlose ganze Zahl > 0 sein. Ist die aktuelle Spaltenposition größer als die angegebene TABULATOR-Position, wird eine Fehlermeldung ausgegeben.
WAIT	Nach Ausgabe der Meldung wartet DRIVE/WINDOWS auf eine Eingabe von der Datensichtstation (z.B. Taste RETURN). Erst nach der Eingabe wird die nächste DRIVE-Anweisung ausgeführt.
WITHOUT WAIT	Nur zulässig im TIAM-Betrieb. Die Angabe WITHOUT WAIT wird ignoriert, wenn SEND MESSAGE auf DISPLAY screenformat folgt. Nach Ausgabe der Meldung wird sofort die nächste DRIVE-Anweisung ausgeführt. WITHOUT WAIT ist nicht sinnvoll, wenn der Meldung unmittelbar eine weitere Ausgabe folgt.

Beispiel

Die Nachricht "Die Liste wird ausgedruckt" wird in die Meldungszeile ausgegeben.

```
SEND MESSAGE 'Die Liste wird ausgedruckt';
```

SET
Wert und Feldattribut zuweisen

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm- und Debug-Modus

SET weist Variablen Werte zu. Jeder Variablen darf jeweils nur ein Wert zugewiesen werden. Falls eine Variable mit DECLARE SCREEN definiert wurde, können dieser Variablen (SCREEN-Variable) Feld- und Globalattribute zugewiesen werden. Wird einer Variablen gleichzeitig ein Wert und ein Feldattribut zugewiesen, muß sie (Teil einer) eine SCREEN-Variable(n) sein.

Es stehen 3 Varianten der SET-Anweisung zur Verfügung.

1. Einer Variablen wird ein Wert, der NULL-Wert und/oder Feldattribute zugewiesen.

```
SET { variable { = { ausdruck | NULL } [ NOCHECK ]
                                  [ [ WITH ] ATTRIBUTE ( attribute1, ... ) ] |
                                  [ WITH ] ATTRIBUTE ( attribute1, ... ) } }, ...
```

2. Mit der 2. SET-Variante werden allen Feldern einer SCREEN-Variablen Globalattribute zugewiesen.

```
SET { screenformat [ WITH ] ATTRIBUTE ( attribute2, ... ) }, ...
```

3. Mit der 3. SET-Variante werden allen Feldern einer SCREEN-Variablen, deren Feldwert EDIT_STATE≠"V" ist (Feld nicht fehlerfrei), Feldattribute zugewiesen.

```
SET { screenformat [ WITH ] ERRORATTRIBUTE ( attribute3, ... ) }, ...
```

variable	Name einer Variablen, der ein Wert oder lokale Feldattribute zugewiesen werden. Der Datentyp von *variable* muß mit dem Datentyp von *ausdruck* zuweisungsverträglich sein. Wird einer strukturierten Variablen ein ebenfalls strukturierter Wert zugewiesen, erfolgt die Zuweisung komponentenweise. Die einzelnen Komponenten müssen jeweils von demselben Datentyp sein. Kommen in einer strukturierten Variablen Redefinitionen vor, erhalten nur die nicht redefinierten Komponenten diesen Wert. Wird einer strukturierten Variablen nur ein einziger Wert zugewiesen, erhalten alle Komponenten diesen Wert.
ausdruck	Wert, der der Variablen zugewiesen wird. Der Datentyp von *ausdruck* muß mit dem Datentyp von *variable* zuweisungsverträglich sein. Beim Zuweisen von Werten auf numerische Variablen werden Nachkommastellen kaufmännisch gerundet, wenn die Variablendeklaration nicht genügend Nachkommastellen zuläßt.
NULL	Der Variablen wird der NULL-Wert zugewiesen.
NOCHECK	Bei der Wertzuweisung wird eine der Variablen *variable* zugeordnete CHECK-Klausel ignoriert, wenn eine CHECK-Klausel festgelegt wurde. Ist *variable* strukturiert, gilt NOCHECK für alle Komponentenzuweisungen.
WITH ATTRIBUTE	Einer SCREEN-Variablen werden Feldattribute zugewiesen.
attribute1	Feldattribut. Es können folgende Werte zugewiesen werden: – MUST, NORMALINPUT, POTMUST – UNPROTECTED, PROTECTED – HIGHINTENSITY, NORMALINTENSITY – VISIBLE, SIGN, INVISIBLE – UNDERLINE, NOUNDERLINE – INVERSE, NOINVERSE – BLUE, CYAN, GREEN, MAGENTA, RED, WHITE, YELLOW, NOCOLOUR – CURSOR, NOCURSOR – VALID, INVALID SET ... ATTRIBUTE (attribute1, ...) darf im Debug-Modus nicht angegeben werden.

screenformat — Name eines Formats der zugehörigen SCREEN-Variablen, in der das FHS-Format abgebildet ist und der Globalattribute zugewiesen werden.

WITH ATTRIBUTE — Allen Feldern einer SCREEN-Variablen werden Globalattribute zugewiesen.

attribute2 — Globalattribut. Es können folgende Werte zugewiesen werden:

- ALARM
- HARDCOPY
- INIT, NOINIT
- CURSOR, NOCURSOR

Ein weiterer möglicher Wert ist DEFAULT. Wird er angegeben, werden alle Feldattribute auf den vordefinierten Stand (Leerzeichen) zurückgesetzt. DEFAULT muß bei einer Kombination von Globalattributen als erstes in der Liste stehen.

Im Debug-Modus darf für *attribute2* nur DEFAULT angegeben werden.

WITH ERRORATTRIBUTE — Den Feldern einer SCREEN-Variablen, deren Feldwert EDIT_STATE="F" oder "M" ist, werden Feldattribute zugewiesen.

attribute3 — Feldattribut. Es können folgende Werte zugewiesen werden:

- MUST, NORMALINPUT, POTMUST
- UNPROTECTED
- HIGHINTENSITY, NORMALINTENSITY
- VISIBLE, SIGN, INVISIBLE
- UNDERLINE, NOUNDERLINE
- INVERSE, NOINVERSE
- BLUE, CYAN, GREEN, MAGENTA, RED, WHITE, YELLOW, NOCOLOUR
- CURSOR, NOCURSOR

Beispiele

Die Variable &index wird hochgezählt.

```
SET &index = index + 1;
```

Der Variablen &datum wird das aktuelle Datum in der Form "*jahr-monat-tag*" zugewiesen.

```
SET &datum = CURRENT DATE
```

Jedem der drei Felder des Vektors &fremdsprache(3) wird der NULL-Wert zugewiesen.

```
SET &fremdsprache = NULL
```

Allen Feldern des FHS-Teilformats "maske" wird das Globalattribut CURSOR zugewiesen.

```
SET maske ATTRIBUTE (CURSOR)
```

Ein fehlerhaftes Feld des FHS-Teilformats "maske" wird mit hoher Helligkeitseinstellung dargestellt.

```
SET maske ERR0RATTRIBUTE (HIGHINTENSITY)
```

SET FILE POSITION
In einer Datei positionieren

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

SET FILE POSITION positioniert in einer geöffneten SAM-Datei an den Dateianfang, an das Dateiende oder auf eine beliebige Position, die Sie vorher mit der Anweisung GET FILE POSITION gelesen haben.

In ISAM-Dateien positionieren Sie mit der Anweisung LOCATE FILE über den ISAM-Schlüssel.

```
SET FILE POSITION datei TO { variable | BEGIN | END }
```

---

datei	Logischer Name einer Datei, in der positioniert wird. Mit diesem Namen muß die Datei im Programm mit der Anweisung DECLARE FILE definiert sein.
TO	Es wird festgelegt, wohin positioniert wird.
variable	Bezeichnung der Stelle, wohin positioniert wird (siehe Metavariable *variable*). *variable* muß zuvor mit der Anweisung GET FILE POSITION belegt worden sein.
BEGIN	Es wird an den Dateianfang positioniert.
END	Es wird an das Dateiende positioniert.

SET SCREEN ATTRIBUTE
Formatattribut zuweisen

Diese Anweisung ist gültig

– im UTM-Betrieb
– im Programm-Modus

SET SCREEN ATTRIBUTE weist FHS-DE-Formaten Attribute zu. Sie können folgendes festlegen:

– die Anzahl der auszugebenden Zeilen in Listenbereichen
– welches Blätter- oder Verschiebekommando bei der nächsten Formatausgabe voreingestellt sein soll
– welche Auswahlmöglichkeit in Auswahlfeldern gesperrt ausgegeben werden soll
– welche Auswahlmöglichkeit in Auswahlfeldern vorausgewählt ausgegeben werden soll
– welche Zeile in Listenbereichen vorausgewählt ausgegeben werden soll

---

```
SET SCREEN ATTRIBUTE

   { { { LOCK | PRESELECT }  { ON | OFF } [ ITEM ( i, ...) ] } |
     LINES n |
     SCROLL charausdruck }

     FOR { screenformat | feld }
```

LOCK	Auswahlmöglichkeiten werden gesperrt (ON) oder entsperrt (OFF). *feld* muß die Komponente einer SCREEN-Variablen sein.
PRESELECT	Auswahlmöglichkeiten oder Zeilen werden als vorausgewählt (ON) oder als nicht vorausgewählt (OFF) gekennzeichnet.
ON	Auswahlmöglichkeiten / Zeilen werden gesperrt (bei LOCK) oder als vorausgewählt gekennzeichnet (bei PRESELECT).
OFF	Auswahlmöglichkeiten / Zeilen werden entsperrt (bei LOCK) oder als nicht vorausgewählt gekennzeichnet (bei PRESELECT).

ITEM	bei LOCK: sperrt/entsperrt die *i*-te Auswahlmöglichkeit eines Einfachauswahlfeldes. LOCK ... ITEM ist bei Mehrfachauswahlfeldern und Listenbereichen nicht erlaubt. Fehlt ITEM bei Einfachauswahlfeldern, werden alle Auswahlmöglichkeiten ge- oder entsperrt. bei PRESELECT: kennzeichnet die *i*-te Auswahlmöglichkeit eines Einfachauswahlfelds oder die *i*-te Zeile eines Listenbereichs. PRESELECT ... ITEM ist bei Mehrfachauswahlfeldern nicht erlaubt. Bei Einfachauswahlfeldern muß ITEM angegeben werden. Es darf aber nur eine Auswahlmöglichkeit *i* angegeben werden. Fehlt ITEM bei Listenbereichen, werden alle Zeilen gekennzeichnet.
i	Auswahlmöglichkeit / Zeile ($i \leq$ Anzahl der Auswahlmöglichkeiten oder Listenzeilen) *i* muß eine ganze positive Zahl sein. *i* kann als numerischer Ausdruck angegeben werden (siehe Metavariable *numausdruck*).
LINES	Zeilen des Listenbereichs von *screenformat* werden ausgegeben.
n	Anzahl der Zeilen, die ausgegeben werden sollen ($n \leq$ Maximalanzahl, die mit IFG für den Listenbereich festgelegt wurde). *n* muß eine ganze positive Zahl sein.
SCROLL	Die Zeichen für die Blätterinfo werden festgelegt.
charausdruck	Zeichen für die Blätterinfo. Für *charausdruck* sind die Zeichen + – < > und das Leerzeichen (%) zulässig. Enthält *charausdruck* nur Leerzeichen, wird keine Blätterinfo ausgegeben.
screenformat	FHS-DE-Format mit einem Listenbereich. Das FHS-DE-Format muß mit DECLARE SCREEN im Programm definiert sein.
feld	Auswahlfeld, für das ein Attribut gesetzt wird. Das Feld muß Teil eines FHS-DE-Formats (= Komponente einer SCREEN-Variablen) sein. Das zugehörige FHS-DE-Format muß mit DECLARE SCREEN im Programm definiert sein.

STOP
DRIVE-Sitzung beenden

Diese Anweisung ist gültig

- im TIAM-Betrieb
- im UTM-Betrieb, aber bei VTV nur auf oberster Programmstufe in Auftragnehmer-Umgebung
- im Dialog- und Programm-Modus

STOP beendet die DRIVE-Sitzung.

Im TIAM-Betrieb werden alle angeforderten Dateien geschlossen, alle Views sowie alle DRIVE-spezifischen Speicherbereiche werden freigegeben.

Befindet sich im TIAM-Betrieb bei der Eingabe von STOP oder STOP WITH DISPLAY noch eine ungesicherte Datei in der EDT-Arbeitsdatei 0, wird die Meldung `EDT-BEREICH NICHT LEER. 'DRIVE' BEENDEN? ANTWORT (Y=JA;N=NEIN)` ausgegeben.

- Bei der Antwort "N" wird STOP nicht ausgeführt. Am Bildschirm wird die SAVE-Anweisung ausgegeben, so daß die Datei gesichert werden kann.
- Bei der Antwort "Y" wird STOP ausgeführt. Änderungen in der ungesicherten Datei gehen verloren.

Im UTM-Betrieb werden alle Listen des Vorgangs ausgedruckt und in der zentralen Druckdatei gelöscht, wenn sie nicht explizit mit LIST * ... DELETE ausgedruckt wurden. Anschließend muß ein Transaktionscode (TAC) oder KDCOFF eingegeben werden (bei VTV nur in Auftraggeber-Umgebung).

Wenn noch Transaktionen geöffnet sind, folgt auf STOP eine DRIVE-Fehlermeldung. Die DRIVE-Sitzung wird nicht beendet. Sie müssen noch offene Transaktionen mit COMMIT, ROLLBACK oder EXIT beenden. EXIT beendet die DRIVE-Sitzung.

Einschränkungen

- Im lokalen Betrieb ist STOP nur im Dialog-Modus, und im Programm-Modus nur in einem Dialog-Programm erlaubt.
- STOP WITH DISPLAY ist nur in einem Dialog-Programm erlaubt.
  STOP WITH DISPLAY ist im UTM-Asynchronbetrieb und bei VTV in Auftragnehmer-Umgebung nicht erlaubt.
- STOP WITH *charausdruck* ist im UTM-Asynchronbetrieb und bei VTV in Auftragnehmer-Umgebung nicht erlaubt.

```
STOP [ WITH { DISPLAY formatname |
              DISPLAY screenformat, ... [ SCREENERROR { REPEAT | CONTINUE } ] |
              DISPLAY form |
              charausdruck } ]
```

DISPLAY formatname

formatname wird an der Datensichtstation ausgegeben (siehe Anweisung DISPLAY *formatname*).

DISPLAY screenformat

screenformat wird an der Datensichtstation ausgegeben (siehe Anweisung DISPLAY *screenformat*).

SCREENERROR Mit SCREENERROR wird das Verhalten von DRIVE/WINDOWS festgelegt, wenn fehlerhafte Feldeingaben vorliegen.

REPEAT Vorbelegung

Die Bildschirmmaske wird solange wiederholt ausgegeben, bis die Eingabe fehlerfrei ist oder ein gewollter Abbruch durch BREAK erfolgt. Danach wird die Durchführung der STOP-Anweisung fortgesetzt.

CONTINUE Die Durchführung der STOP-Anweisung wird nach DISPLAY fortgesetzt, auch wenn Eingaben fehlerhaft waren.

DISPLAY form Definieren und Ausgeben eines Kompakt-Bildschirmformats (siehe Anweisung DISPLAY FORM).

charausdruck Name des UTM-Folgetacs (max. 8 Zeichen).

charausdruck darf nur im Programm-Modus im UTM-Betrieb angegeben werden.

charausdruck darf im UTM-Asynchronbetrieb und in Auftragnehmer-Umgebung bei VTV nicht angegeben werden.

Existiert der angegebene UTM-Folgetac nicht, wird das Programm abgebrochen, aber der Vorgang nicht beendet.

Regeln bei Verteilter Transaktionsverarbeitung

– In Auftragnehmer-Umgebung müssen bei STOP alle Auftragnehmer-Vorgänge beendet sein.

– In Auftragnehmer-Umgebung wird bei STOP zum Auftraggeber-Vorgang zurückgesprungen. RETURN-Parameter werden an den Auftraggeber-Vorgang zurückgegeben.

SUBPROCEDURE
Internes Unterprogramm beginnen

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

SUBPROCEDURE kennzeichnet den Anfang eines internen Unterprogramms. Ein internes Unterprogramm ist eine Anweisungsfolge, die innerhalb eines Programms im Deklarationsteil hinter den DECLARE-Anweisungen definiert wird. Geschachtelte Definitionen von internen Unterprogrammen sind nicht erlaubt. Ein internes Unterprogramm darf selbst keine DECLARE-Anweisungen enthalten.

Innerhalb eines Programms können interne Unterprogramme beliebig oft aufgerufen werden. Der Aufruf erfolgt mit CALL *subprogname*.

Interne Unterprogramme dürfen nur aufgerufen werden, wenn sie vorher definiert wurden. In einem internen Unterprogramm müssen alle Strukturierungsanweisungen (CASE, CYCLE, DISPATCH, IF) abgeschlossen sein, d.h. das zugehörige END muß in demselben internen Unterprogramm stehen. Ein END für eine Strukturierungsanweisung, die außerhalb dieses internen Unterprogramms gegeben wurde, darf nicht vorkommen.

Ein BREAK CYCLE kann keine außerhalb dieses internen Unterprogramms beginnende Schleife abbrechen. Ein BREAK PROCEDURE beendet sowohl das interne Unterprogramm als auch das gesamte Programm. Wird ein BREAK SUBPROCEDURE erreicht, wird das Programm mit der Anweisung fortgesetzt, die dem CALL-Aufruf für dieses interne Unterprogramm folgt.

Das Ende eines internen Unterprogramms wird durch END SUBPROCEDURE festgelegt.

```
SUBPROCEDURE subprogname
```

---

subprogname Name des internen Unterprogramms (maximal 31 Zeichen).

SYSTEM
BS2000-Kommando eingeben

Diese Anweisung ist gültig

– im TIAM-Betrieb
– im Dialog- und Programm-Modus

Mit SYSTEM gibt man im laufenden DRIVE-Betrieb BS2000-Kommandos ein.

```
SYSTEM charausdruck
```

---

charausdruck — Name des auszuführenden BS2000-Kommandos.

charausdruck muß eine alphanumerische Zeichenfolge (max. 254 Zeichen) sein. Siehe Metavariable *charausdruck*.

Das Ergebnis von *charausdruck* muß ein syntaktisch richtiges BS2000-Kommando sein.

Beispiele

```
SYSTEM 'PRINT-FILE *SYSLST'

SYSTEM 'SHOW-USER-STATUS'

SET &FS='SHOW-FILE-ATTRIBUTES'
SYSTEM &FS
```

TRACE
Ablaufverfolgung einschalten

Diese Anweisung ist gültig

– im TIAM-Betrieb
– im Debug-Modus

TRACE schaltet im Debug-Modus die Ablaufverfolgung eines Programms ein. Das bedeutet: ein Programm wird in Einzelschritten kontrolliert ausgeführt und die zugehörigen Zeilen der Übersetzungsliste werden ausgegeben.

Mit der TRACE-Anweisung wird festgelegt, wieviel Programmanweisungen mitverfolgt werden sollen. Nach der vereinbarten Anzahl von Anweisungen hält das Programm an, wenn die Anweisungen fehlerfrei ausgeführt wurden. Der Tracepunkt ist erreicht. Die Ablaufverfolgung wird automatisch ausgeschaltet und der Tracepunkt gelöscht.

Eingegebene Parameter werden zur Vorbelegung für die nächste TRACE-Anweisung, wenn diese TRACE-Anweisung ohne Operanden eingegeben wird.

Die vom Anwender definierten Tracepunkte setzt DRIVE/WINDOWS immer hinter eine Anweisung des DRIVE-Programms.

```
TRACE [ n | ALL ] [ OUT | LIST | BOTH ]
```

---

n	*n* ist eine positive ganze Zahl. Die nächsten *n* Anweisungen werden in Einzelschritten kontrolliert ausgeführt und die zugehörigen Zeilen der Übersetzungsliste(n) ausgegeben (Trace-Ausgaben). Wird ein Haltepunkt erreicht, bevor *n* Anweisungen ausgeführt sind, wird die Ablaufverfolgung ausgeschaltet und der Tracepunkt gelöscht. Vorbelegung beim erstmaligen Aufruf von TRACE: *n*=1. Bei weiteren Aufrufen ohne Angabe von *n* wird *n* aus der vorangegangenen TRACE-Anweisung übernommen.
ALL	Alle Anweisungen bis zum nächsten Haltepunkt werden in Einzelschritten kontrolliert ausgeführt und die zugehörigen Zeilen der Übersetzungsliste(n) ausgegeben (Trace-Ausgaben).
OUT	Vorbelegung Die Angabe OUT bewirkt Trace-Ausgaben am Bildschirm.

	Vorbelegung beim erstmaligen Aufruf von TRACE: OUT. Bei weiteren Aufrufen ohne Angabe eines Operanden wird der Operand aus der vorangegangenen TRACE-Anweisung übernommen.
LIST	Die Angabe LIST bewirkt Trace-Ausgaben auf *SYSLST*.
BOTH	Die Angabe BOTH wirkt wie LIST und OUT zusammen.

Beziehungen zu anderen Anweisungen

Greift die DEBUG-Anweisung auf ein Programm zu, das in der EDT-Arbeitsdatei 0 steht, kann die Ablaufverfolgung nicht eingeschaltet werden.

UNSAVE
Programm, COPY-Element oder Benutzerkennsatz löschen

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Dialog- und Programm-Modus

UNSAVE löscht aus einer DRIVE-Bibliothek

– Sourcen, COPY-Elemente, Benutzerkennsätze (S-Elemente)
– Zwischencodes (X-Elemente)
– Objektcodes (R-Elemente)
– Übersetzungslisten (P-Elemente)

Die angegeben Objekte werden in folgender Reihenfolge gelöscht:

1. Verwendungsnachweise
2. Zwischencodes
3. Objektcodes
4. Übersetzungslisten
5. Sourcen
6. COPY-Elemente

Tritt beim Löschen mehrerer Objekte ein Fehler auf, wird eine Fehlermeldung ausgegeben, bei welchem Einzelobjekt der Fehler auftrat. Die in der Reihenfolge vorangegangenen Einzelobjekte sind ordnungsgemäß gelöscht, die folgenden nicht.

Ist eines der angegebenen Einzelobjekte nicht vorhanden, wird dies wie ein Fehlerfall behandelt.

Wird UNSAVE ohne optionale Operanden angegeben, werden die gespeicherten Sourcen, Zwischencodes, Objectcodes, Übersetzungslisten und Verwendungsnachweise mit dem Namen *elemname* gelöscht. DRIVE/WINDOWS gibt keine Fehlermeldung aus, wenn ein Einzelobjekt nicht existiert (Ausnahme: das S-Element existiert nicht).

Pro Anweisung darf jeder der Operanden SOURCE, OBJECT, CODE, LIST, COPYSOURCE und USERLABEL nur einmal angegeben werden. Eine Kombination von unterschiedlichen Operanden ist erlaubt.

Die Anweisung UNSAVE ist nicht erlaubt in Programmen, die mit dem DRIVE-Compiler DRIVE/WINDOWS-Comp übersetzt werden sollen.

```
UNSAVE { bibliothek(elemname) | elemname }

       [ { SOURCE | CODE | OBJECT | LIST | COPYSOURCE | USERLABEL }, ... ]
```

bibliothek	Name der DRIVE-Bibliothek (max. 54 Zeichen), aus der ein Element oder ein Verwendungsnachweis gelöscht wird. *bibliothek* kann auch der Dateikettungsname der DRIVE-Bibliothek (entsprechend den BS2000-Konventionen) sein. DRIVE/WINDOWS interpretiert *bibliothek* zuerst als Dateikettungsnamen, dann als Bibliotheksnamen. Wird die DRIVE-Bibliothek mit der Anweisung PARAMETER DYNAMIC LIBRARY voreingestellt, so kann die Angabe von *bibliothek* entfallen.
elemname	Name des Elements (max. 31 Zeichen), das gelöscht wird. Wird keine Angabe für *bibliothek* gemacht, wird die bei PARAMETER DYNAMIC LIBRARY angegebene Bibliothek eingesetzt.
SOURCE	Das DRIVE-Programm (S-Element) in der DRIVE-Bibliothek wird gelöscht.
CODE	Der Zwischencode (X-Element) in der DRIVE-Bibliothek wird gelöscht.
OBJECT	Der Objektcode (R-Element) in der DRIVE-Bibliothek wird gelöscht.
LIST	Die Übersetzungsliste (P-Element) in der DRIVE-Bibliothek wird gelöscht.
COPYSOURCE	Das COPY-Element (S-Element) in der DRIVE-Bibliothek wird gelöscht.
USERLABEL	Der Benutzerkennsatz (S-Element) in der DRIVE-Bibliothek wird gelöscht.

WHENEVER
Fehlerausgang definieren

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

WHENEVER definiert einen Fehlerausgang, wenn in einem Programm ein Fehler auftritt. WHENEVER muß im Programm im Deklarationsteil hinter den Definitionen von internen Unterprogrammen stehen. Werden für ein Ereignis mehrere WHENEVER angegeben, gilt die letzte Anweisung.

Mit der Anweisung WHENEVER werden die Einträge der Systemvariablen &KFKEY, &ERROR (= &ERROR_STATE.ERROR) und &DML_STATE (= &ERROR_STATE.DML_STATE) abgefragt und Fehlerausgänge definiert. Die Beschreibung der Systemvariablen und ihrer Einträge finden Sie in der DRIVE-Programmiersprache [2].

Wird sowohl für &ERROR als auch für &DML_STATE ein Eintrag abgefragt und treten beide Ereignisse gleichzeitig ein, wird der Fehlerausgang für &ERROR ausgeführt, falls &SQL_CODE > 0, ansonsten wird der Fehlerausgang für &DML_STATE ausgeführt.

Tritt ein Fehler ein, wird das entsprechende Zählfeld hochgezählt (siehe DRIVE-Programmiersprache [2], Abschnitt Systemvariablen).

Wird kein Fehlerausgang definiert, bricht DRIVE/WINDOWS das Programm ab. (Ausnahme: Bei den &ERROR-Einträgen "OK END", "TOO LONG" und "TOO SHORT" und bei den &DML_STATE-Einträgen "TABLE END" und "DIRTY READ" wird das Programm fortgesetzt.)

```
WHENEVER { &KFKEY [ IN ( literal, ... ) ] |
           &ERROR [ IN ( error, ... ) ] |
           &DML_STATE [ IN ( status, ... ) ] }

         { CONTINUE | CALL subprogname | BREAK }
```

&KFKEY IN	Die Bedingung tritt beim Betätigen der Taste *literal* ein. Sie wird nur bei Programmen ohne Fensteroberfläche ausgewertet. Dabei wird von DRIVE/WINDOWS die Aktion CONTINUE gesetzt. Die Bedingung wird nur ausgeführt, wenn keine andere Fehlerbedingung auftritt. Bei einer DISPLAY-Anweisung mit Überlauf führt das Betätigen der Taste *literal* zum Abbruch der Ausgabe. Als nächste Anweisung wird entweder CONTINUE, CALL oder BREAK ausgeführt.
literal	Bezeichnung einer Taste (K1, K3 - K14 oder F1 - F20)
&ERROR IN	Der Eintrag von &ERROR kann nach folgenden Anweisungen abgefragt werden: CALL (nicht CALL *subprogname*) CASE CYCLE FOR / WHILE DISPLAY [FORM / LIST / SCREEN] DO ENTER END CYCLE einer CYCLE WHILE- oder CYCLE FOR-Schleife END CYCLE einer CYCLE *cursorname*-Schleife bei Remote-Zugriff END DISPATCH (Nicht abgefragt werden kann &ERROR nach CALL-Anweisungen, die Programme im entfernten System aufrufen.) END IF EXECUTE (nach EXECUTE und nach EXECUTE mit einer der aufgeführten Anweisungen) FILL {FORM / LIST} IF PROCEDURE SEND MESSAGE SET SYSTEM SQL-Anweisungen mit INTO-Klausel Anweisungen für die Dateibearbeitung Remote-Zugriffen auf eine SESAM- oder UDS-Datenbank Wird IN (*error*, ...) nicht angegeben, ist dies gleichbedeutend mit der Angabe aller möglichen Angaben für *error*.
error	Mit *error* wird der Eintrag festgelegt, für den ein Fehlerausgang definiert wird.

Welche Einträge von &ERROR abgefragt werden können, entnehmen Sie der DRIVE-Programmiersprache [2], Abschnitt Systemvariablen.

Für *error* muß ein Literal angegeben werden.

&DML_STATE IN — Der Eintrag von &DML_STATE kann für alle SQL-Anweisungen und nach EXECUTE, das eine SQL-Anweisung ausführt, abgefragt werden. Außerdem nach END CYCLE innerhalb einer "CYCLE cursorname INTO"-Schleife, wenn der Wert von SQLCODE < 0 ist.

Wird IN (*status*, ...) nicht angegeben, ist dies gleichbedeutend mit der Angabe aller möglichen Angaben für *status*.

status — Mit *status* wird der Eintrag festgelegt, für den ein Fehlerausgang definiert wird.

Welche Einträge von &DML_STATE abgefragt werden können, entnehmen Sie der DRIVE-Programmiersprache [2], Abschnitt Systemvariablen.

Für *status* muß ein Literal angegeben werden.

CONTINUE — Bei Eintritt eines definierten Fehlerereignisses wird das Programm fortgesetzt. Die Systemvariable &ERROR_STATE wird dann mit den oben beschriebenen Fehlerinformationen versorgt.

CALL subprogname — Bei Eintritt eines definierten Fehlerereignisses wird das interne Unterprogramm *subprogname* aufgerufen. Die Systemvariable &ERROR_STATE wird mit den oben beschriebenen Fehlerinformationen versorgt.

Tritt bei der Abarbeitung des internen Unterprogramms erneut ein Fehler auf, für den ein Fehlerausgang bei WHENEVER definiert wurde, wird das Programm abgebrochen. Die Systemvariable &ERROR_STATE wird in dem internen Unterprogramm nicht verändert.

BREAK — Bei Eintritt eines definierten Fehlerereignisses, wird das Programm abgebrochen.

Beispiel

Anweisung	**Ereignis**	**&ERROR=**	**&DML_STATE=**
`SET &v=&a(&i)`	INDEX ERROR	`'INDEX ERROR'`	unverändert
`OPEN` *cursorname*	SQL ERROR	unverändert	`'SQL ERROR'`
`FETCH` *cursorname* `INTO ...`	DIRTY READ	`'OK'`	`'DIRTY READ'`

WRITE FILE
In Datei schreiben

Diese Anweisung ist gültig

– im TIAM- und UTM-Betrieb
– im Programm-Modus

WRITE FILE schreibt in eine zum Schreiben geöffnete Datei einen Datensatz. Der Aufbau und der Inhalt des Datensatzes sind durch die WITH-Klausel definiert.

In SAM-Dateien wird der Datensatz an die aktuelle Dateiposition geschrieben. Existiert schon ein Datensatz, der überschrieben werden soll, muß der neue Satz die gleiche Länge haben wie der zu überschreibende Satz.

In ISAM-Dateien wird der Datensatz an die Stelle geschrieben, die durch den ISAM-Schlüssel festgelegt ist. Der ISAM-Schlüssel gehört zu den Satzdaten.

Ist der zu schreibende Datensatz länger als die erlaubte Datensatzlänge im BS2000, erhält die Systemvariable &ERROR den Eintrag "TOO LONG".

Nach dem Schreiben weist die aktuelle Dateiposition auf den nächsten Satz.

Wenn ein Programm auf Dateien zugreift, die mit dem OPEN-Modus UPDATE, INOUT oder OUTIN geöffnet sind, darf die Anweisung WRITE FILE nicht unmittelbar auf die Anweisung READ FILE folgen. Zwischen der WRITE FILE- und der READ FILE-Anweisung muß mindestens eine Anweisung zur Positionierung (SET FILE POSITION) stehen.

```
WRITE FILE datei WITH { ausdruck | NULL }, ...
```

---

datei	Logischer Name einer Datei, in die geschrieben wird. Mit diesem Namen muß die Datei im Programm mit der Anweisung DECLARE FILE definiert sein.
ausdruck	Bezeichnung des Ausdrucks, der in die Datei geschrieben wird (siehe Metavariable *ausdruck*). *ausdruck* beschreibt den Datensatz.
NULL	In die Datei wird das Zeichen geschrieben, das für die NULL-Wertdarstellung festgelegt wurde (siehe Anweisung DECLARE FILE).

Besonderheiten der Dateimerkmale

– Die Dateimerkmale entsprechen den Voreinstellungen des Betriebssystems BS2000. Wenn Sie mit Dateimerkmalen arbeiten wollen, die den Voreinstellungen nicht entsprechen (z.B. längeren Datensätzen), müssen Sie die Dateimerkmale mit dem entsprechenden ADD-FILE-LINK-Kommando festlegen. Die betroffene Datei müssen Sie dann mit dem Dateikettungsnamen ansprechen (SET-FILE-LINK... LINK-NAME=)

  Die Kommandos ADD-FILE-LINK und SET-FILE-LINK müssen eingegeben werden:

  – im TIAM-Betrieb vor dem Start des DRIVE-Programms oder im DRIVE-Programm mit der SYSTEM-Anweisung. Die Anweisung SYSTEM muß vor der OPEN-FILE-Anweisung stehen.
  – im UTM-Betrieb vor dem Start der DRIVE-Anwendung.

4 Report-Anweisungen

Dieses Kapitel enthält zunächst einen Überblick über alle Report-Anweisungen. Dem folgt eine Liste der DRIVE-Anweisungen, die bei einer Report-Generierung zulässig sind. Daran schließt sich eine Beschreibung der Einschränkungen für Report-Parameter und eine Beschreibung der Report-Mengenfunktionen an.

Im weiteren sind in diesem Kapitel alle Report-Anweisungen in alphabetischer Reihenfolge ausführlich beschrieben.

Informationen zur Anwendung des Report-Generators sind in DRIVE-Programmiersprache [2] zu finden.

Report-Anweisungen sind gültig

– im TIAM-Betrieb
– im UTM-Betrieb, allerdings dürfen zwischen OPEN REPORT und CLOSE REPORT keine Bildschirmein- und -ausgaben erfolgen
– im Programm- Modus und im Debug-Modus. Im Debug-Modus können zwischen DECLARE REPORT und END REPORT keine Haltepunkte gesetzt werden

Report-Anweisungen im Überblick

Zur Generierung eines Reports sind zwei Schritte notwendig:

1. Definieren des Reports
2. Ausführen und Erzeugen des Reports

Alle Anweisungen zur Report-Definition stehen im deklarativen Teil eines DRIVE-Programms. Die Anweisungen zum Ausführen eines Reports befinden sich im ausführbaren Teil eines DRIVE-Programms.

Der Beginn und das Ende einer Report-Definition werden mit folgenden Anweisungen gekennzeichnet:

DECLARE REPORT	Report-Definition beginnen, siehe Seite 226
END REPORT	Report-Definition beenden, siehe Seite 233

Die zwischen DECLARE REPORT und END REPORT stehenden Anweisungen dienen zur Beschreibung der Daten und des Layouts. Dazu können folgende Report-Anweisungen verwendet werden:

DECLARE VARIABLE	Report-Variablen definieren, siehe Seite 230
STANDARD LAYOUT	Layout eines Standard-Reports beschreiben, siehe Seite 274
GLOBAL LAYOUT	Layout eines individuellen Reports beschreiben, siehe Seite 236
GLOBAL LINE BASE	Zeilenhintergrundmuster definieren, siehe Seite 240
GLOBAL PAGE BASE	Definition eines Seitenhintergrundmusters beginnen, siehe Seite 242
PAGE PRINT	Seitenhintergrundmuster beschreiben, siehe Seite 252
OVERLAY PAGE BASE	Seitenhintergrundmuster aktivieren, siehe Seite 251
REPORT	Listenkontrollblock definieren, siehe Seite 251
PAGE	Seitenkontrollblock definieren, siehe Seite 252
GROUP	Gruppenkontrollblock definieren, siehe Seite 243
DETAIL	Detailkontrollblock definieren, siehe Seite 231
PRINT	Report-Ausgabe definieren, siehe Seite 259
SOURCE	Textdateien einfügen, siehe Seite 272

Zum Ausführen eines Reports sind die folgenden Report-Anweisungen erlaubt:

OPEN REPORT	Report-Ausführung beginnen, siehe Seite 246
FILL REPORT	Report mit Daten versorgen, siehe Seite 234
CLOSE REPORT	Report-Ausführung beenden, siehe Seite 225

Zulässige DRIVE-Anweisungen

Folgende DRIVE-Anweisungen sind bei der Report-Generierung zulässig:

BREAK — Schleife abbrechen. Diese Anweisung ist nur in folgender Form erlaubt:
BREAK CYCLE

CALL — C-Module aufrufen. Die Anweisung CALL ist nur in folgender Form erlaubt:
CALL C MODULE ... RETURN
Es werden keine Nullwertanzeigen übergeben. Die Angabe INDICATOR ist **nicht** erlaubt.

CYCLE — Schleife programmieren. Die Anweisung CYCLE *cursorname* ist bei der Report-Generierung nicht sinnvoll, da es im Report keinen Cursor gibt. Die Anweisung CYCLE FOR ist **nicht** erlaubt.

IF — Bedingungen programmieren. Bei einer IF-Anweisung sind für *bedingung* nur folgende Vergleiche erlaubt:

- Ausdrücke mit Vergleichsoperatoren
- Satzelement mit dem NULL-Wert

(Siehe Metavariable *bedingung*).

SET — Werte zuweisen. In einer SET-Anweisung ist nur die Angabe SET *variable* ... ohne die Zuordnung von Attributen erlaubt. Zuweisungen sind nur an Startparameter und lokale Variablen erlaubt.

Einschränkungen für Report-Parameter

Werden in einer Report-Definition in Report- oder DRIVE-Anweisungen Variablen, Systemvariablen, Maskensteuerzeichen, Ausdrücke und Übergabeparameter verwendet, gelten folgende Einschränkungen:

Literale — Literale vom Datentyp INTERVAL müssen sekundengenau angegeben werden.

Variablen — Die innerhalb einer Report-Definition vereinbarten Variablen gelten nur lokal. Es müssen einfache Variablen sein.

Nicht erlaubt ist das Verwenden

- der Datentypen TIME(3) und TIMESTAMP(3)
- von INIT-, CHECK- oder REDEFINES-Klauseln
- von LIKE-Klauseln
- von Variablen, die außerhalb der Report-Definition vereinbart sind

Systemvariablen	Als Systemvariablen sind nur &LINES und &PAGES erlaubt. Sie gelten lokal innerhalb der Report-Definition.
Maskensteuerzeichen	Folgende Maskensteuerzeichen dürfen **nicht** verwendet werden: ZZZY, ZI, ZS, ZW, BWZ, YYY, ZZY, JJJ, ZZJ. Ebenfalls **nicht** erlaubt sind CHAR-Masken. Bei der Angabe von Q(3), QQQ oder R(3), RRR werden Namen abgekürzt. Bei allen andern Q- und R-Angaben werden Namen voll ausgegeben (siehe Metavariable *mask*).
Ausdrücke	In Ausdrücken sind nur folgende Funktionen erlaubt: – CURRENT DATE – CURRENT TIME (siehe Metavariablen *charprim* und *format*) Der Potenzierungsoperator (**) ist **nicht** erlaubt. Es empfiehlt sich, bei der Ausgabe von Rechenoperationen und Report-Mengenfunktionen Masken anzugeben.
Übergabeparameter	In der USING-Klausel übergebene Parameter dürfen nicht verändert werden. Sie dürfen z.B. bei der SET-Anweisung nicht vor dem Gleichheitszeichen stehen.

Report-Mengenfunktionen in Ausdrücken

Ausdrücke können die folgenden Report-Mengenfunktionen enthalten:

```
{ COUNT | MIN | MAX | AVG | SUM }
 ( [ REPORT | PAGE | GROUP ] [ GROUPNUM n ] [ TOTAL ] [ ALL ] [ DISTINCT ] ausdruck )
```

Dabei bedeutet:

COUNT	Die Anzahl von *ausdruck* ermitteln.
MIN	Das Minimum von *ausdruck* ermitteln.
MAX	Das Maximum von *ausdruck* ermitteln.
AVG	Den Durchschnitt von *ausdruck* ermitteln.
SUM	Die Summe von *ausdruck* bilden.

Der Anwendungsbereich der oben genannten Funktionen kann spezifiziert werden mit:

REPORT — Für den ganzen Report.

Die Mengenwerte werden bei jedem Auftreten der Felder *ausdruck* aktualisiert und bis zum Ende der Liste nicht zurückgesetzt.

PAGE — Für eine Seite.

Die Berechnung der Mengenwerte von *ausdruck* beginnt auf jeder Seite neu. Die aktuellen Werte der Report-Mengenfunktion stehen zur Verfügung

- in allen Kontrollblöcken der aktuellen Seite, mit Ausnahme des Seitenkopfs,
- auf der nachfolgenden Seite im Seitenkopf zum Übertragen der Werte.

Die Mengenwerte werden nach dem Seitenkopf und vor dem ersten Gruppen- oder Detailkontrollblock zurückgesetzt.

GROUP — Für eine Gruppe.

Die Berechnung der Mengenwerte von *ausdruck* beginnt mit den ersten Report-Daten einer Gruppenstufe. Die aktuellen Werte einer gruppenspezifischen Report-Mengenfunktion sind vom Gruppenkopf bis zum Gruppenfuß der aktuellen Gruppenstufe verfügbar. Die Mengenwerte werden nach dem zugeordneten Gruppenfuß zurückgesetzt.

Fehlt die Angabe REPORT, PAGE oder GROUP, dann bezieht sich eine Report-Mengenfunktion auf den Kontrollblock, in dem sie spezifiziert ist.

Der Anwendungsbereich einer Report-Mengenfunktionen kann noch differenzierter angegeben werden mit:

GROUPNUM n — Für eine bestimmte Gruppe.

Die Report-Mengenfunktion bezieht sich nur auf die Gruppe mit der Nummer *n*. Die Gruppennummer *n* wird in der Anweisung GROUP zugewiesen (siehe Anweisung GROUP). *n* darf nicht größer als 32767 sein.

TOTAL — Im voraus für alle Feldinhalte des Kontrollblocks, für den die Mengenfunktion definiert ist.

Die Vorausberechnung bietet die Möglichkeit, den ermittelten Wert schon innerhalb des Kontrollblocks, zum Beispiel in Ausdrücken, zu verwenden.

ALL	Für alle Feldinhalte. Unabhängig davon, ob die Feldinhalte von *ausdruck* ausgegeben werden oder nicht, wird die Report-Mengenfunktionen auf alle Feldinhalte von *ausdruck* angewendet. Vorbelegt ist, daß nur die auszugebenden Werte zur Berechnung herangezogen werden.
DISTINCT	Für voneinander abweichende Feldinhalte. Zur Berechnung werden nur die Feldinhalte von *ausdruck* herangezogen, die sich vom Inhalt des Feldes im vorangegangenen Satz unterscheiden.

CLOSE REPORT
Report-Ausführung beenden

CLOSE REPORT beendet die Ausführung eines Reports. Dabei schließt der Report-Generator den mit der zugehörigen OPEN REPORT-Anweisung bereitgestellten Report-Puffer und die verwendete Report-Definition (siehe Anweisung OPEN REPORT).

Nach dem Schließen der Report-Definition und des Report-Puffers wird der Report auf dem in OPEN REPORT angegebenen Ausgabegerät erzeugt.

Eine CLOSE REPORT-Anweisung ist nur im ausführbaren Teil von Programmen erlaubt.

```
CLOSE REPORT report_name [ variable ]
```

---

report_name	bezeichnet den Namen der Report-Definition, die verwendet wurde. Der Name muß mit einer DECLARE REPORT-Anweisung vereinbart worden sein (siehe Anweisung DECLARE REPORT). Der entsprechende Report muß offen sein (siehe Anweisung OPEN REPORT).
variable	bezeichnet die Variable oder Variablenkomponente, die zur Identifizierung des Report-Puffers in der Anweisung OPEN REPORT verwendet wurde (siehe Anweisung OPEN REPORT).

DECLARE REPORT
Report definieren

DECLARE REPORT kennzeichnet den Beginn und END REPORT das Ende einer Report-Definition.

Mit der Anweisung DECLARE REPORT erhält eine Report-Definition ihren Namen. Unter diesem Namen wird die Report-Definition bei der Report-Ausführung verwendet.

In der Anweisung DECLARE REPORT sind die Parameter zu definieren, die in einer Report-Definition verarbeitet werden sollen. Diese Parameter werden beim Ausführen des Reports mit Werten versorgt (siehe Anweisung FILL REPORT). Bei der Definition dieser Parameter besteht die Möglichkeit, unterschiedliche Satzarten zu berücksichtigen.

Außerdem können Startparameter spezifiziert werden, denen der Report-Generator beim Start der Report-Ausführung einmalig Werte zuweist (siehe Anweisung OPEN REPORT).

Variablen, die in der Anweisung DECLARE REPORT definiert werden, erscheinen nicht in der Querverweisliste.

Die Anweisung DECLARE REPORT ist nur im deklarativen Teil von Programmen erlaubt. In einem Programm können mehrere Reports definiert werden.

```
DECLARE REPORT report_name
    [ FOR START USING { varname1 grunddatentyp [ mask ] }, ... ]

    { USING { [ level ] varname2 { datendef |
                                LIKE { CURSOR cursor | TABLE tabelle } } }, ... |

      { RECORD TYPE charliteral USING { [ level ] varname2 { datendef |
                          LIKE { CURSOR cursor | TABLE tabelle } } }, ... }, ... }
```

---

report_name — bezeichnet den Namen einer Report-Definition. Der Name muß eindeutig sein und darf maximal aus sieben Zeichen bestehen. Da mit dem Namen der Report-Definition zur Übersetzungs- oder Laufzeit Dateinamen gebildet werden, muß der Name den BS2000-Konventionen für Dateinamen entsprechen.

FOR START USING — definiert Startparameter, für die bei Beginn einer Report-Ausführung mittels der Anweisung OPEN REPORT einmalig aktuelle Werte übergeben werden. Diese Werte können z.B. ausgegeben werden oder in Ausdrücken, die Ergebnisse von Report-Mengenfunktionen enthalten, zur Layout-Steuerung dienen.

Die Anzahl der verwendeten Startparameter muß übereinstimmen mit der Anzahl der Parameter, die in der USING-Klausel der Anweisung OPEN REPORT stehen (siehe Anweisung OPEN REPORT).

Sind die korrespondierenden Parameter in den Anweisungen DECLARE REPORT und OPEN REPORT von unterschiedlichem Typ *grunddatentyp*, so müssen deren Werte in den entsprechenden Typ konvertierbar sein.

Die Gesamtlänge der Datenwerte darf inklusive Nullwertanzeigen 31K nicht überschreiben.

varname1 bezeichnet eine einfache Variable vom Typ *grunddatentyp*.

Werden in einem Programm mehrere Reports definiert, müssen die Namen der Reportparameter über alle Reports eindeutig sein.

Die Namen von Reportparametern dürfen gleich den Namen von DRIVE-Variablen sein. Sie werden trotzdem verschieden behandelt.

grunddatentyp bezeichnet den Typ der Variablen *varname1*. *grunddatentyp* kann ein vom Benutzer definierter Typ *usertyp* oder einer der folgenden Typen sein:

CHARACTER, DECIMAL, NUMERIC, INTEGER, SMALLINT, DATE, TIME, INTERVAL, CHARACTER VARYING, VARCHAR, REAL, DOUBLE PRECISION, FLOAT

Nicht erlaubt ist das Verwenden der Typen TIME(3) und TIMESTAMP(3) , von INIT-, CHECK- oder REDEFINES-Klauseln und von LIKE-Klauseln auf Report-Parameter.

mask bezeichnet eine Darstellungsmöglichkeit (Maskensteuerzeichen) für die Ausgabe eines Startparameters *varname1*. Für die Angabe von *mask* gelten die beschriebenen Einschränkungen (siehe Abschnitt „Einschränkungen für Report-Parameter“ auf Seite 221).

USING definiert Parameter, die bei der Report-Ausführung durch die Anweisung FILL REPORT mit Daten versorgt werden ("Nettodaten").

Die Anzahl der verwendeten Parameter muß mit der Anzahl der Ausdrücke übereinstimmen, die in der USING-Report-Klausel der Anweisung FILL REPORT stehen (siehe Anweisung FILL REPORT).

Sind die korrespondierenden Parameter in den Anweisungen DECLARE REPORT und FILL REPORT von unterschiedlichem Typ, so müssen deren Werte in den entsprechenden Typ konvertierbar sein.

	Die Gesamtlänge der Datenwerte darf inklusive Nullwertanzeigen 31K nicht überschreiben.
level	Stufennummer. Die Stufennummer muß 1 sein.
varname2	bezeichnet eine Variable vom Typ *datendef* oder von einem Typ gemäß einer LIKE-Klausel (siehe DRIVE-Anweisung DECLARE VARIABLE). Es dürfen maximal soviele Variablen angegeben werden, daß die Länge der Werte aller Parameter einschließlich der Nullanzeigen 31 Kbyte nicht überschreitet.
datendef	bezeichnet den Typ der Variablen *varname2* (siehe Metavariable *datendef*). Folgende Angaben sind für *datendef* **nicht** erlaubt: INIT-, CHECK-, REDEFINES-Klauseln und LIKE-Klauseln für Report-Parameter. Die Datentypen TIME(3) und TIMESTAMP(3) sind **nicht** zulässig.
LIKE	kopiert die Struktur eines Cursors oder einer Tabelle komponentenweise in die Variable *varname2* (siehe DRIVE-Anweisung DECLARE VARIABLE).
cursor	bezeichnet einen Cursor, der im DRIVE-Programm deklariert sein muß.
tabelle	bezeichnet eine Basistabelle, die in einer abgeschlossenen Datenbank deklariert sein muß.
RECORD TYPE	definiert eine Satzart *charliteral*. In der zugehörigen USING-Klausel wird der Aufbau der Satzart beschrieben. Es können mehrere Satzarten definiert werden (siehe Anweisung DETAIL).

Beispiel 1

In der Report-Deklaration wird als Startparameter die Variable &s2 definiert, die der Variablen &kd_name entspricht. Vor dem Öffnen des Reports wird der Variablen &kd_name der Wert 'VERTRETER' zugewiesen. Dieser Wert wird beim Öffnen des Reports übergeben.

```
PROCEDURE "t1.2007";

DECLARE VARIABLE &12      VARCHAR(256) INIT '9012-2';
DECLARE VARIABLE &v11     DEC (8,2);
DECLARE VARIABLE &repv    CHAR (8);
DECLARE VARIABLE &kd_name CHAR (10);
```

```
DECLARE REPORT drirep FOR START
                      USING &s2 CHAR (10)
                      USING &repdat,
                              2 schluessel CHAR (6),
                              2 artnam     CHAR (20),
                              2 preis      NUM (8,2),
                              2 bestand    NUM (5);
...
END REPORT;

SET &kd_name='VERTRETER';

OPEN REPORT drirep USING &kd_name RESULT list 'G207' DEVICETABLE &12;
...
CLOSE REPORT drirep;
```

Beispiel 2

Es wird die Report-Definition mit dem Namen "statist" vereinbart. In dieser Report-Definition werden die Daten des Parameters &daten verarbeitet, die als Variable &cvar im DRIVE-Programm vereinbart wurden:

```
...
/* Variable, die dem Report-Puffer später bei der Verarbeitung   */
/* übergeben wird                                                */

DECLARE VARIABLE  1 &cvar
                    2 vertreter       CHAR (20),
                    2 artikel         NUM (6),
                    2 anzahl          NUM,
                    2 einzelbetrag    NUM,
                    2 betrag          NUM;

/* Deklaration des Reports                                       */

DECLARE REPORT statist USING &daten LIKE &cvar;

   /* Beschreibung der Daten                                     */
   /* Beschreibung des Layouts                                   */

END REPORT;
...
```

DECLARE VARIABLE
Report-Variablen definieren

DECLARE VARIABLE dient zur Vereinbarung von Report-Variablen. Diese Variablen gelten nur lokal innerhalb der Report-Definition. Eine lokale Variable kann z.B. in Ausdrücken zur Steuerung des Layouts verwendet werden.
Report-Variablen erscheinen nicht in der Querverweisliste.

Die Systemvariablen &PAGES und &LINES stehen implizit in folgender Form zur Verfügung:

```
DECLARE VARIABLE &PAGES INTEGER,
                 &LINES INTEGER;
```

Diese Systemvariablen sind mit dem NULL-Wert vorbelegt und müssen nicht explizit mit der Anweisung DECLARE VARIABLE vereinbart werden; das Zuweisen eines Wertes ist nicht möglich. Die Systemvariablen &PAGES und &LINES gelten innerhalb der Report-Definition als lokale Variable.

```
DECLARE VARIABLE varname grunddatentyp [ mask ]
```

---

varname	bezeichnet eine einfache Variable vom Typ *grunddatentyp*, die mit dem NULL-Wert vorbelegt wird.
grunddatentyp	bezeichnet den Typ der Variablen *varname*. *grunddatentyp* kann einer der folgenden Typen sein: CHARACTER, INTEGER, SMALLINT, DATE, TIME, INTERVAL, CHARACTER VARYING oder VARCHAR, REAL, DOUBLE PRECISION, FLOAT, XDEC oder EXTENDED DECIMAL, DECIMAL, NUMERIC **Nicht** erlaubt ist das Verwenden der Datentypen TIME(3) und TIMESTAMP(3).
mask	bezeichnet eine Darstellungsmöglichkeit (Maskensteuerzeichen) für die Ausgabe einer Variablen *varname* (siehe Metavariable *mask*). Für die Angabe von *mask* gelten die beschriebenen Einschränkungen (siehe Abschnitt „Einschränkungen für Report-Parameter“ auf Seite 221).

Im Gegensatz zur Deklaration von DRIVE-Variablen darf die Deklaration von Report-Variablen keine INIT-, CHECK-, REDEFINES oder LIKE-Klauseln enthalten.

DETAIL
Detailkontrollblock definieren

DETAIL dient zur Beschreibung der Detailzeilen. In diesem Kontrollblock werden die Anweisungen zusammengefaßt, die die Bearbeitung der auszugebenden Datensätze steuern. Im Detailkontrollblock wird z.B. relativ zur aktuellen Ausgabeposition definiert, an welcher Stelle konstante Textteile und variable Daten einzufügen sind.

Die Anweisung DETAIL ist nur innerhalb einer Report-Definition erlaubt. Sollen unterschiedliche Satzarten bearbeitet werden, so ist für jede Satzart ein eigener Detailkontrollblock zu definieren.

Liegen mehrere Satzarten vor, identifiziert der Report-Generator eine Satzart in einer Zeichenfolge und dem Inhalt eines Kennfeldes. Nimmt das Kennfeld den Wert der Zeichenfolge an, wird in den entsprechenden Detailkontrollblock verzweigt. Das Kennfeld einer Satzart muß in allen Satzarten an derselben Position vorhanden sein. Als Kennfeld vorbelegt ist das erste Feld eines Datensatzes.

Wird in einem Detailkontrollblock eine andere Satzart angesprochen als für diesen Kontrollblock definiert, so müssen die Felder mit der angegebenen Zeichenkette qualifiziert werden.

Sind mehrere Detailkontrollblöcke für verschiedene Satzarten definiert, stehen in allen Detailkontrollblöcken zunächst die Felder aller Satzarten zur Verfügung. Ein Satz einer Satzart ist dann nicht mehr verfügbar, wenn der aktuelle Satz bearbeitet und der nachfolgende Satz von der gleichen Satzart ist.

Eine Detailzeile kann folgende Ausgaben enthalten:

– Literale und mit SOURCE eingefügte Texte,
– die Systemvariablen &PAGES und &LINES,
– Startparameter,
– Report-Mengenwerte,
– Nettodaten des aktuellen Datensatzes und anderer Satzarten,
– arithmetische Ausdrücke,
– lokale Variable.

```
DETAIL [ RECORD TYPE literal [ IN variable ] ] [ zeilenmuster ] { anweisung; } ...
```

RECORD TYPE literal dient zur Zuordnung einer Satzart *literal*. Die Anweisungen in diesem Kontrollblock werden dann nur für diese Satzart ausgeführt.

IN variable bezeichnet den Namen eines Kennfeldes für die Satzart *literal*. Als Kennfeld vorbelegt ist das erste Feld eines Datensatzes.

zeilenmuster Name eines Zeilenhintergrundmusters, das mit der Anweisung GLOBAL LINE BASE definiert worden sein muß.
Den Detailzeilen wird das entsprechende Zeilenmuster als Hintergrundmuster zugewiesen.

anweisung bezeichnet eine der DRIVE-Anweisungen BREAK, CALL, CYCLE, IF, SET oder eine der Report-Anweisungen PRINT oder SOURCE. Diese Anweisungen dienen zur Beschreibung der Detailzeilen. Für die Anweisungen gelten die beschriebenen Einschränkungen (siehe Abschnitt „Zulässige DRIVE-Anweisungen“ auf Seite 221 und Abschnitt „Einschränkungen für Report-Parameter“ auf Seite 221).

Beispiel

Für die Detailzeilen eines Reports wird definiert, daß der Schrifttyp Courier 12 verwendet werden soll. In diesem Schrifttyp sollen an den Tabulatorpositionen 20 und 30 die Variablen &daten.name und &daten.alter ausgegeben werden. Nach einem Zeilenvorschub von zwei Zeilen soll an Tabulatorposition 25 "Miete: " folgen und anschließend an Position 34 die Variable &daten.miete im Ausgabeformat Z9.

```
...
DETAIL
  PRINT SET (FONT 'COURIER', CHARARACTER DENSITY 12),
    TAB 20, &daten.name,
    TAB 30, &daten.alter,
    NL 2,

    TAB 25, 'Miete: ',
    TAB 34, &daten.miete MASK 'Z9', '% ';
...
```

END REPORT
Report-Definition beenden

END REPORT schließt eine Report-Definition ab.

```
END REPORT
```

FILL REPORT
Report mit Daten versorgen

FILL REPORT übergibt einen Datensatz an einen Report-Puffer. Dort bereitet der Report-Generator die Daten gemäß der Report-Definition für das Ausgabegerät auf, das in OPEN REPORT gewählt wurde.

Zwischen einer OPEN REPORT- und der zugehörigen CLOSE REPORT-Anweisung können so viele FILL REPORT-Anweisungen stehen, wie für die Datenübergabe nötig sind.

Mit der Anweisung FILL REPORT kann auch eine in DECLARE REPORT spezifizierte Satzart übergeben werden.

Eine FILL REPORT-Anweisung ist nur im ausführbaren Teil eines Programms und nur im Programm-Modus erlaubt.

```
FILL REPORT report-name [ variable ] [ RECORD TYPE charliteral ]
                        USING { ausdruck |  NULL }, ...
```

---

report-name — bezeichnet den Namen der Report-Definition, die zum Generieren des Reports verwendet wird. Der Name muß mit einer DECLARE REPORT-Anweisung vereinbart sein (siehe Anweisung DECLARE REPORT).

variable — bezeichnet eine Variable oder Variablenkomponente, die zur Identifizierung des Report-Puffers in der Anweisung OPEN REPORT spezifiziert wurde (siehe Anweisung OPEN REPORT).

RECORD TYPE charliteral

übergibt die Daten einer Satzart *charliteral* an den Report-Puffer. Die Satzart *charliteral* und ihr Aufbau wurde in der Anweisung DECLARE REPORT definiert (siehe Anweisung DECLARE REPORT).

DRIVE/WINDOWS prüft, ob die Ausdrücke der USING-Leiste in die Datentypen konvertierbar sind, die in der DECLARE REPORT-Anweisung für die Parameter angeben wurden.

USING — übergibt die Parameter, die in der USING-Klausel der Anweisung DECLARE REPORT definiert wurden.

ausdruck — bezeichnet die Parameter.

NULL — spezifiziert den NULL-Wert als Parameter.

Beispiel

Mit der Report-Definition "stat" soll ein Report mit den Daten "drive_daten" erzeugt werden. Die Daten entsprechen der Satzart "satz", die in der Anweisung DECLARE REPORT definiert wurde.

```
...
DECLARE REPORT stat RECORD TYPE 'satz' USING &daten LIKE &drive_daten;
...
END REPORT;
...
FILL REPORT stat RECORD TYPE 'satz' USING &drive_daten;
...
```

GLOBAL LAYOUT
Globale Vorbelegungen für einen Report festlegen

GLOBAL LAYOUT dient zur individuellen Gestaltung eines Reports und ist nur innerhalb einer Report-Definition erlaubt.

Mit einer Anweisung GLOBAL LAYOUT werden globale Vorbelegungen vereinbart, die für den gesamten Report Gültigkeit haben. Folgende Vorbelegungen sind möglich:

– Sortierkriterien für die Übergabeparameter festlegen,
– Breite der Seitenränder bestimmen,
– Größe des Bereichs für Seitenkopf und -fuß bestimmen,
– Darstellungsattribute wählen.

Bei der Angabe der Seitenränder und der Breite des Seitenkopf- oder Seitenfußbereichs wird von dem Standard ausgegangen, der im Geräteprofil des gewählten Ausgabegeräts eingetragen ist (siehe DRIVE-Programmiersprache [2], Kapitel Der Report-Generator).

Die mit der Anweisung GLOBAL LAYOUT vereinbarten Vorbelegungen für die Darstellungsattribute können mit der Anweisung PRINT jederzeit verändert oder zurückgesetzt werden (siehe Anweisung PRINT).

```
GLOBAL LAYOUT
       { ORDER BY { { variable [ ASCENDING | DESCENDING ] }, ... } [ EXTERNAL ] |
         { TOP MARGIN n | BOTTOM MARGIN n | LEFT MARGIN n | RIGHT MARGIN n } |
         { HEADER LINES n | TRAILER LINES n | MINIMUM LINES n } |
         format-klausel } ...
```

ORDER BY

legt die Sortierreihenfolge der Feldinhalte des Parameters *variable* fest.

Sind Gruppen definiert, sind die Feldinhalte nach den Gruppenwechselfeldern entsprechend ihrer Hierarchie zu sortieren.

Enthält eine Report-Definition keine ORDER-Klausel, wird implizit eine Sortierung durchgeführt, wenn Gruppenkontrollblöcke definiert sind (siehe Anweisung GROUP) und als Gruppenwechselfeld ein Einzelfeld angegeben wurde. In diesem Fall wird nach den Gruppenwechselfeldern in aufsteigender Reihenfolge sortiert.

variable	bezeichnet eine Komponente des Parameters *varname2*, der in der USING-Klausel der Anweisung DECLARE REPORT definiert wurde. Die Komponente *variable* muß von einfachem Typ und eindeutig qualifiziert sein. Die zu verarbeitenden Sätze werden nach den Feldinhalten dieser Komponenten sortiert.
ASCENDING	Vorbelegung. Es wird in aufsteigender Reihenfolge sortiert.
DESCENDING	Es wird in absteigender Reihenfolge sortiert.
EXTERNAL	Gibt an, daß die Daten, die dem Report-Generator übergeben werden, bereits sortiert sind. Wird diese Angabe gemacht, prüft der Report-Generator nicht mehr, ob die Daten tatsächlich sortiert vorliegen.
TOP MARGIN n	spezifiziert die Anzahl *n* der Zeilen, die der Report-Generator für den oberen Rand einer Ausgabeseite reservieren soll. *n* darf nicht größer als 32767 sein.
BOTTOM MARGIN n	Anzahl *n* der Zeilen, die der Report-Generator für den unteren Rand einer Ausgabeseite reservieren soll. *n* darf nicht größer als 32767 sein.
LEFT MARGIN n	Anzahl *n* der Spalten, die der Report-Generator für den linken Rand einer Ausgabeseite reservieren soll. *n* darf nicht größer als 32767 sein.
RIGHT MARGIN n	Anzahl *n* der Spalten, die der Report-Generator für den rechten Rand einer Ausgabeseite reservieren soll. *n* darf nicht größer als 32767 sein.
HEADER LINES n	Anzahl *n* der Zeilen, die der Report-Generator für den Seitenkopf einer Ausgabeseite reservieren soll. *n* darf nicht größer als 32767 sein.
TRAILER LINES n	Anzahl *n* der Zeilen, die der Report-Generator für den Seitenfuß einer Ausgabeseite reservieren soll. *n* darf nicht größer als 32767 sein.
MINIMUM LINES n	Anzahl *n* der Zeilen, die für eine Reportseite mindestens zur Verfügung stehen müssen. *n* darf nicht größer als 32767 sein. Diese Angabe ist dann sinnvoll, wenn bei der Report-Definition die Größe einer Ausgabeseite noch nicht bekannt ist. Dadurch wird vermieden, daß auf den Ausgabeseiten z.B. nur Seitenköpfe und -füße ausgegeben werden.

format-klausel

Die Beschreibung der Darstellungsattribute, die in der *format-klausel* angegeben werden können, ist bei der Anweisung PRINT zu finden.

In der Anweisung GLOBAL LAYOUT ist das Verwenden der Darstellungsattribute SUBSCRIPT und SUPERSCRIPT **nicht** erlaubt.

Beispiel

Es soll ein Report über das Konsumverhalten von Personen erzeugt werden. Für die dazu benötigten Daten wird im DRIVE-Programm die strukturierte Variable &person bereitgestellt.

Die Report-Definition erhält den Namen "verhalt". Dort wird auch die DRIVE-Variable &person über den Parameter &daten an die Report-Definition übergeben.

In der Report-Definition wird die lokale Variable &i deklariert und anschließend die Sortierfolge festgelegt. Sortiert wird aufsteigend nach den Komponenten &daten.geschlecht, &daten.sozial und &daten.name. Außerdem werden die Ränder und der Bereich für Kopf- und Fußzeilen bestimmt. Die danach gewählten Darstellungsattribute wie Schriftart, Zeichendichte und Schrifttyp gelten für den ganzen Report. Sie können jedoch mit der Anweisung PRINT für einzelne Ausgabefelder oder für den ganzen Report geändert werden.

```
...
/* DRIVE-Variablen zur Aufnahme der Report-Daten deklarieren */

DECLARE VARIABLE  1 &person,
                    2 name       VARCHAR(30),
                    2 alter      SMALLINT,
                    2 geschlecht CHAR,
                    2 sozial     CHAR,
                    2 miete      INTEGER,
                    2 auto       INTEGER,
                    2 konsum     INTEGER;

/* Report deklarieren */

DECLARE REPORT verhalt USING &daten LIKE &person;

   DECLARE VARIABLE &i INTEGER;              /* Report-Variable deklarieren */

   GLOBAL LAYOUT                             /* Layout voreinstellen  */
      ORDER BY &daten.geschlecht ASCENDING,  /* Sortierkriterien      */
               &daten.sozial ASCENDING,
               &daten.name ASCENDING
```

```
      TOP MARGIN 2                          /* Ränder                 */
      BOTTOM MARGIN 3
      HEADER LINES 2
      TRAILER LINES 5

      FONT 'CENTURY CONDENSED'              /* Darstellungsattribute */
      CHARACTER DENSITY 10
      ITALIC 1;
      ...
END REPORT;
```

GLOBAL LINE BASE
Zeilenhintergrundmuster definieren

Mit der Anweisung GLOBAL LINE BASE definieren Sie ein benanntes Hintergrundmuster für eine Druckzeile. Das Hintergrundmuster kann aus Tabulatoren und aus positionierten Textteilen bestehen, wobei Texte entweder direkt angegeben oder aus einer Meldungsdatei über MSGSTRING eingelesen werden können.

Zeilenhintergrundmuster können jeweils verschiedenen Kontrollblöcken zugewiesen werden. Die Texte werden als Hintergrundmuster ausgegeben, das durch aktuelle Ausgaben überschrieben wird.
In der PRINT-Anweisung können die mit GLOBAL LINE BASE vereinbarten Tabulatoren sequentiell von links nach rechts abgearbeitet und zur Positionierung innerhalb der Zeile verwendet werden.

Die Anweisung GLOBAL LINE BASE ist nur innerhalb einer Report-Definition erlaubt, wo sie vor der ersten REPORT DIRECTIVE-Anweisung stehen muß. Enthält die Report-Definition die Anweisung GLOBAL LAYOUT, muß GLOBAL LINE BASE zwischen der Anweisung GLOBAL LAYOUT und der ersten REPORT DIRECTIVE-Anweisung stehen.

```
GLOBAL LINE BASE zeilenmuster

       { tab_position |
         PATTERN position { charliteral |
                            MSGSTRING ( numausdruck1 [ [, numausdruck2 ], name ] ) }
                          [ format-klausel ] ... }, ...
```

---

zeilenmuster	Name des Zeilenhintergrundmusters (max. 54 Zeichen).
tab_position	Definiert einen Tabulator mit der Tabulatorposition *tab_position*. *tab_position* muß eine vorzeichenlose Zahl vom Typ INTEGER sein. Die Position wird in der Einheit (cm, inch, 1/300 Zoll) berechnet, die in der Anweisung GLOBAL LAYOUT vereinbart wurde. *tab_position* muß innerhalb der Druckzeile liegen.
PATTERN	Als Hintergrundmuster wird Text verwendet, der als Literal angegeben oder aus einer Meldungsdatei eingelesen werden kann.

position
: Angabe der Textposition als vorzeichenlose Zahl vom Typ INTEGER.
Die Position wird in der Einheit (cm, inch, 1/300 Zoll) berechnet, die in der Anweisung GLOBAL LAYOUT vereinbart wurde.

  position muß innerhalb der Druckzeile liegen.

charliteral
: Angabe eines Textes, der als Hintergrundmuster erscheinen soll (*charliteral* siehe Metavariable *literal*).

MSGSTRING (numausdruck1 [[, numausdruck2], name])
: Der Text, der als Hintergrrundmuster erscheinen soll, wird aus einer Meldungsdatei eingelesen (*MSGSTRING* siehe Metavariable *charprim*).

format-klausel
: Angabe von Darstellungsattributen für den Text des Hintergrundmusters. Unterschiedliche Textteile können unterschiedliche Darstellungsattribute haben.
Folgende Darstellungsattribute können Sie in *format-klausel* angeben:
FONT, NATIONAL SET, SIZE, BOLD, INVERSE, ITALIC, UNDERLINE, COLOUR FOREGROUND, COLOUR BACKGROUND.
Die Bedeutung der einzelnen Darstellungsattribute ist bei der PRINT-Anweisung unter *format-klausel* erklärt.

GLOBAL PAGE BASE
Seitenhintergrundmuster definieren

Mit der Anweisung GLOBAL PAGE BASE definieren Sie ein benanntes Hintergrundmuster für eine Druckseite. Das Hintergrundmuster kann neben positionierten Texten, Linien und Rechtecken auch gedrehte Texte enthalten. Für bestimmte Drucker können auch Seitenhintergrundmuster verwendet werden, die nicht mit DRIVE/Report-Mitteln erstellt wurden (siehe SOURCE-Anweisung).

Seitenhintergrundmuster ermöglichen es Ihnen

– Nettodaten in ein vorgedrucktes Formular auszugeben,

– Formularvordrucke mit Report-Mitteln innerhalb von DRIVE zu erstellen,

– Druckseiten mit vorbelegten Mustern zu überlagern,

Aktiviert und deaktiviert wird ein Seitenhintergrundmuster durch die Anweisung OVERLAY PAGE BASE. Ist das Seitenhintergrundmuster aktiviert, überlagert es die Ausgabe der Nettodaten.

Die Anweisung GLOBAL PAGE BASE ist nur innerhalb einer Report-Definition erlaubt, wo sie vor der ersten REPORT DIRECTIVE-Anweisung stehen muß. Enthält die Report-Definition die Anweisung GLOBAL LAYOUT, muß GLOBAL PAGE BASE zwischen der Anweisung GLOBAL LAYOUT und der ersten REPORT DIRECTIVE-Anweisung stehen.

```
GLOBAL PAGE BASE seitenmuster_name { anweisung; } ...
```

---

seitenmuster_name	Name des Seitenhintergrundmusters (max. 54 Zeichen).
anweisung	bezeichnet eine der DRIVE-Anweisungen CYCLE, IF, SET oder eine der Report-Anweisungen PAGE PRINT oder SOURCE. Diese Anweisungen dienen zur Beschreibung des Seitenhintergrundmusters. Für sie gelten die beschriebenen Einschränkungen (siehe Abschnitt „Zulässige DRIVE-Anweisungen“ auf Seite 221 und Abschnitt „Einschränkungen für Report-Parameter“ auf Seite 221). Die Anweisung SOURCE darf nur alleine angegeben werden, d.h. als erste und einzige *anweisung*.

GROUP
Gruppenkontrollblock definieren

GROUP beschreibt einen Gruppenkopf oder Gruppenfuß für eine Gruppe. Dazu muß ein Gruppenwechselfeld angegeben werden, das die Gruppe identifiziert. Zur Kennzeichnung einer Gruppe kann ein Gruppenkopf z.B. eine Überschrift enthalten. In einem Gruppenfuß können beispielsweise gruppenspezifische Zwischenergebnisse stehen.

Die Anweisung GROUP ist nur innerhalb einer Report-Definition erlaubt. Sie darf mehrmals angegeben werden. Die Reihenfolge der Definitionen gibt die Hierarchie an, nach der die Gruppenverarbeitung ausgeführt wird:

```
gruppenkopf a
   gruppenkopf b
      gruppenkopf c
         detailzeilen
      gruppenfuß c
   gruppenfuß b
gruppenfuß a
```

Sind mehrere Gruppen definiert und findet im nächsten zu bearbeitenden Datensatz ein Gruppenwechsel statt, dann werden die Kontrollblöcke in der Reihenfolge bearbeitet:

– alle Gruppenfüße von der niedrigsten bis zu der Kontrollhierarchie, in welcher der Gruppenwechsel stattfindet,

– alle Gruppenköpfe von der aktuellen bis zur niedrigsten Kontrollhierarchie,

– alle Anweisungen für den Detailbereich, mit den Daten des nächsten Datensatzes.

Die zu verarbeitenden Daten müssen entweder bereits sortiert an den Report-Generator übergeben oder durch die Angabe ORDER in der Anweisung GLOBAL LAYOUT sortiert werden (siehe Anweisung GLOBAL LAYOUT). Im Beispiel oben wurde die Sortierung von *a*, *b* und *c* vorausgesetzt. Sind die Daten nicht sortiert und enthält die Anweisung GLOBAL LAYOUT keine ORDER-Klausel, wird implizit eine Sortierung durchgeführt, wenn Gruppenkontrollblöcke definiert sind. In diesem Fall wird nach den Gruppenwechselfeldern in aufsteigender Reihenfolge sortiert.

Gruppenköpfe und Gruppenfüße dürfen mehrzeilig sein und folgende Ausgaben enthalten:

– Literale und mit SOURCE eingefügte Texte
– die Systemvariablen &PAGES und &LINES
– Startparameter
– Report-Mengenwerte
– arithmetische Ausdrücke
– Nettodaten des aktuellen Datensatzes
– lokale Variablen

```
GROUP { HEADER | TRAILER } { variable }, ... [ GROUPNUM n ] [ zeilenmuster ]
      { anweisung; } ...
```

HEADER	definiert den Gruppenkopf für die Gruppe *variable*. Der Gruppenkopf wird mit den nachfolgenden Anweisungen beschrieben.
TRAILER	definiert den Gruppenfuß für die Gruppe *variable*. Der Gruppenfuß wird mit den nachfolgenden Anweisungen beschrieben.
variable	bezeichnet ein Gruppenwechselfeld, das zur Identifikation einer Gruppe dient. Ein Gruppenwechsel wird dann ausgelöst, wenn sich der Inhalt des Gruppenwechselfeldes ändert. *variable* muß eine einfache Komponente aus der Menge der zu bearbeitenden Daten sein. Es können auch mehrere Felder (Feldkombinationen) angegeben werden. Dann findet ein Gruppenwechsel statt, wenn sich der Inhalt eines dieser Felder ändert.
GROUPNUM n	spezifiziert eine Gruppennummer *n* für die Gruppe *variable*. Diese Gruppennummer kann bei Report-Mengenfunktionen angegeben werden, wenn sich eine Berechnung nur auf diese Gruppe beziehen soll (siehe Abschnitt „Report-Mengenfunktionen in Ausdrükken“ auf Seite 222). *n* darf nicht größer als 32767 sein.
zeilenmuster	Name eines Zeilenhintergrundmusters, das mit der Anweisung GLOBAL LINE BASE definiert worden sein muß. Dem Gruppenkopf bzw. -fuß wird das entsprechende Zeilenmuster als Hintergrundmuster zugewiesen.
anweisung	bezeichnet eine der DRIVE-Anweisungen BREAK, CALL, CYCLE, IF, SET oder eine der Report-Anweisungen PRINT oder SOURCE. Diese Anweisungen dienen zur Beschreibung von Gruppenkopf oder -fuß. Für sie gelten die beschriebenen Einschränkungen (siehe Abschnitt „Zulässige DRIVE-Anweisungen“ auf Seite 221 und Abschnitt „Einschränkungen für Report-Parameter“ auf Seite 221).

Beispiel

Für die Gruppe &daten.geschlecht soll der Gruppenkopf die Überschrift "Männer: " oder "Frauen: " enthalten.

```
...
GROUP HEADER &daten.geschlecht

   PRINT NEWLINE 1, SET (NATIONAL SET 'GERMAN', BOLD 1,
                         FONT 'COURIER', CHAR DENSITY 12),
         NEED LINES 5;

   IF (&daten.geschlecht = 'm') THEN
      PRINT TAB 10, 'Männer: ',
            NEWLINE 1;
   ELSE
      PRINT TAB 10, 'Frauen: ',
            NEWLINE 1;
   END IF;
...
```

OPEN REPORT
Report-Ausführung beginnen

Mit OPEN REPORT beginnt die Ausführung eines Reports. Dazu wird die Report-Definition angegeben, die den Report beschreibt und ein Report-Puffer bereitgestellt und initialisiert, der die Report-Daten aufnimmt. Außerdem können hier die in DECLARE REPORT definierten Startparameter übergeben werden.

Mit der Anweisung OPEN REPORT wird auch das Ausgabegerät für einen Report spezifiziert; das kann ein Drucker oder eine Datei sein.

Mit einer zugehörigen Anweisung CLOSE REPORT werden der Report-Puffer und die Report-Definition wieder geschlossen und die Ausführung beendet. Nach dem Schließen der Report-Definition und des Puffers wird der Report auf dem gewählten Ausgabegerät erzeugt.

Unter UTM dürfen zwischen der Ausführung einer OPEN REPORT-Anweisung und der entsprechenden CLOSE REPORT-Anweisung keine Bildschirmein- und Ausgaben erfolgen.

Es dürfen mehrere OPEN REPORT-Anweisungen angegeben werden.

Die Anweisung ist nur im ausführbaren Teil eines Programms erlaubt.

```
OPEN REPORT report_name [ variable ] [ USING { ausdruck | NULL }, ... ]

     RESULT { LIST [ geraet ] [ DEVICETABLE geraete-tabelle [ spoolparameter ] ] |
              FILE datei [ DEVICETABLE geraete-tabelle ] }
```

---

report_name — bezeichnet den Namen der Report-Definition, die verwendet werden soll. Der Name muß mit einer DECLARE REPORT-Anweisung vereinbart sein (siehe Anweisung DECLARE REPORT).

variable — bezeichnet eine Variable oder Komponente vom Typ CHAR(8), mit der der Report-Puffer identifiziert wird. Sie müssen *variable* angeben, wenn Sie gleichzeitig mehrere Reports erzeugen wollen, die dieselbe Report-Definition *report_name* verwenden. Unterschiedliche *variablen* bezeichnen unterschiedliche Reports. DRIVE/WINDOWS merkt sich in *variable* eine interne Reportidentifikation.

Wenn *variable* angegeben wird, muß sie explizit im DRIVE-Programm vereinbart sein und darf zum Ablaufzeitpunkt, zwischen OPEN und CLOSE, nicht mit einer SET-Anweisung oder anderen Anweisungen verändert werden.

	Geben Sie *variable* nicht an, verwendet DRIVE eine Variable mit dem Namen *report_name*, die für jeden Report automatisch bei der Ausführung von DECLARE REPORT angelegt wird.
USING	übergibt Werte für die Startparameter, die in der Anweisung DECLARE REPORT mit der Klausel FOR START USING definiert wurden.
ausdruck	der Wert von *ausdruck* wird übergeben (siehe Metavariable *ausdruck*).
NULL	der NULL-Wert wird übergeben.
RESULT LIST	legt fest, daß der Report auf einem Drucker ausgegeben werden soll. Soll der Report auf einem Remote Spoolout (RSO)-Drucker ausgegeben werden, müssen Sie *geraet* und DEVICETABLE *geraete-tabelle* angeben.
geraet	Symbolischer Name für einen Drucker (max. 256 Zeichen). Der Name kann als Literal oder als Inhalt einer DRIVE-Variablen vom Typ CHARACTER oder VARCHAR angegeben werden. Erfolgt die Ausgabe auf dem Systemdrucker, wird die Angabe für *geraet* nicht ausgewertet. Bei Ausgabe auf einem RSO-Drucker muß *geraet* angegeben werden.
RESULT FILE	legt fest, daß der Report in eine Datei ausgegeben werden soll. Die Ausgabe in eine Datei erfolgt in einem geräteabhängigen Format gemäß *geraete-tabelle*.
datei	Name der Datei, in die der Report ausgegeben wird. Der Name muß den BS2000-Konventionen für Dateinamen entsprechen und kann als Literal oder als Inhalt einer DRIVE-Variablen vom Typ CHARACTER oder VARCHAR angegeben werden.
DEVICETABLE	Zuordnung einer Gerätetabelle. Bei Ausgabe auf einem RSO-Drucker muß DEVICETABLE angegeben werden.
geraete-tabelle	ist der logische Name des Ausgabegeräts. Der Name kann als Literal oder als Inhalt einer DRIVE-Variablen vom Typ CHARACTER oder VARCHAR angegeben werden (max. 256 Zeichen).

In der Datei PROFILE.*geraete-tabelle* ist das Geräteprofil hinterlegt. Dieses Profil enthält die Beschreibung einer Ausgabeseite und die Zuordnung der geräteunabhängigen zu den gerätespezifischen Steuerzeichen (siehe DRIVE-Programmiersprache [2], Kapitel Der Report-Generator).

Erfolgt die Ausgabe auf dem Systemdrucker, muß für geraete-tabelle der Wert "ND" bei Ausgabe auf einem ND-Laserdrucker oder der Wert "HP" bei Ausgabe auf einem HP-Laserdrucker angegeben werden. Wird für *geraetetabelle* keine Angabe gemacht, wird "ND" eingesetzt.

spoolparameter

gibt Optionen für die Druckerverwaltung an.
spoolparameter kann als Literal oder als Inhalt einer DRIVE-Variablen vom Typ CHARACTER oder VARCHAR angegeben werden (max. 256 Zeichen).

DRIVE/WINDOWS setzt automatisch ein PRINT-DOCUMENT-Kommando ab, das folgendermaßen aufgebaut ist:

```
PRINT-DOCUMENT FROM-FILE=tempdatei,-
   TO-PRINTER=*PARAMETERS(PRINTER-TYPE=*HP-PRINTER),-
   DELETE-AFTER-PRINT=*YES
```

oder

```
PRINT-DOCUMENT FROM-FILE=tempdatei,-
   DELETE-AFTER-PRINT=*YES
```

Vom Anwender können weitere Optionen für den Druckauftrag angegeben werden, die an das PRINT-DOCUMENT-Kommando angehängt werden. Diese Optionen müssen der Syntax des BS2000-Kommandos PRINT-DOCUMENT im SDF-Format entsprechen (siehe BS2000-Kommandos [35]).

Die Zeichenkette *spoolparameter* wird ungeprüft an die Druckerverwaltung übergeben.

Wollen Sie einen Report mit mehreren Zeichensätzen auf dem Systemdrucker ausgeben, müssen diese Zeichensätze im PRINT-DOCUMENT-Kommando angegeben werden (siehe unten).

Sind keine Optionen für die Druckerverwaltung angegeben, wird die Einstellung `SPOOLDOPT` des Benutzerprofils ausgewertet. Dabei wird das benutzerspezifische Benutzerprofil vor dem systemspezifischen Benutzerprofil ausgewertet.

Sind keine Angaben in den Benutzerprofilen vorhanden, wird für das BS2000-Kommando PRINT-DOCUMENT der Operand LAYOUT-CONTROL=PARAMETERS(CONTROL-CHARACTERS=PHYSICAL) eingesetzt (siehe BS2000-Kommandos [35]).

Dieser vorbelegte Wert entspricht den Mindestanforderungen.

Alle übrigen Einstellungen des Benutzerprofils, die den Spool betreffen, sind ohne Bedeutung.

Ausgeben von Reports mit mehreren Zeichensätzen auf Systemdrucker:

Wollen Sie beim Ausdrucken eines Reports mehrere Zeichensätze verwenden, müssen folgende Voraussetzungen geschaffen sein:

– Es müssen entsprechend viele Zeichensätze im Geräteprofil (PROFILE.ND bzw. PROFILE.HP) definiert sein. Für ND-Laserdrucker können maximal 4, für HP-Laserdrucker maximal 64 Zeichensätze definiert werden.

– In der Report-Definition müssen die Zeichensätze mit den im Geräteprofil definierten Namen verwendet werden. Bei ND-Laserdruckern sind die Namen der Zeichensätze vorbelegt mit ABSCHNITT1 bis ABSCHNITT4, bei HP-Laserdruckern mit ABSCHNITT1 bis ABSCHNITT64. Sie geben also zum Beispiel an:
```
PRINT ausdruck1 ATTRIBUT(FONT ABSCHNITT1);          und
PRINT SET(FONT ABSCHNITT2);
```

– Im PRINT-DOCUMENT-Kommando (siehe BS2000-Kommandos [35]) müssen die Zeichensätze angegeben werden, die in den Drucker geladen werden sollen. Das geschieht mit der Angabe CHARACTER-SETS des Operanden LAYOUT-CONTROL:
```
LAYOUT-CONTROL=PARAMETERS(CONTROL-CHARACTERS=PHYSICAL,
   CHARACTER-SETS=(font1,font2, ...) )
```

  Der bei CHARACTER-SETS an erster Stelle genannte Zeichensatz wird dem Zeichensatz ABSCHNITT1 zugeordnet, der an zweiter Stelle genannte Zeichensatz dem Zeichensatz ABSCHNITT2 usw. (es handelt sich um Stellungsparameter).

  Verwenden Sie im Report die Zeichensätze ABSCHNITT1 und ABSCHNITT2, könnten Sie angeben
```
LAYOUT-CONTROL=PARAMETERS(CONTROL-CHARACTERS=PHYSICAL,
   CHARACTER-SETS=(105,203) )
```

Der im Report in der FONT-Klausel genannte Zeichensatz "ABSCHNITT1" wird mit Zeichensatz 105 ausgedruckt, "ABSCHNITT2" mit Zeichensatz 203

Beispiel

Es wird die Ausführung mehrerer Reports mit der Report-Definition stat gestartet. Zur Identifizierung der zu öffnenden Report-Puffer werden die Variablen &puf1 und &puf2 bereitgestellt. Als einmaliger Startparameter wird die Variable &start übergeben. Das Ausgabegerät soll ein HP-Laserdrucker sein.

```
...
OPEN REPORT stat &puf1 USING &start RESULT LIST DEVICETABLE 'HP';
OPEN REPORT stat &puf2 USING &start RESULT LIST DEVICETABLE 'HP';
...
```

OVERLAY PAGE BASE
Seitenhintergrundmuster aktivieren

Mit der Anweisung OVERLAY PAGE BASE aktivieren Sie ein zuvor mit der Anweisung GLOBAL PAGE BASE definiertes Seitenhintergrundmuster.

Die Anweisung OVERLAY PAGE BASE ist nur innerhalb einer Report-Definition erlaubt.

```
OVERLAY PAGE BASE seitenmuster_name { ON | OFF }
```

---

seitenmuster_name	Name eines Seitenhintergrundmusters, das mit der Anweisung GLOBAL PAGE BASE definiert wurde.

Die Anweisung OVERLAY PAGE BASE wirkt auf die aktuelle Druckseite, wenn sie als erste Anweisung bezüglich der Ausgabeseite angegeben ist (Druckposition erste Zeile, erste Spalte). Sie wirkt zum Beispiel auf die aktuelle Seite, wenn sie als erste Anweisung eines PAGE HEADER Kontrollblockes angegeben wird.
Ansonsten wirkt die Anweisung erst auf die nächste Druckseite.

PAGE
Seitenkontrollblock definieren

PAGE beschreibt den Kopf oder Fuß für alle Report-Ausgabeseiten. Ein Seitenkopf kann beispielsweise als Übertrag von der vorangegangenen Seite, Report-Mengenwerte enthalten. In einem Seitenfuß kann z.B. ein Zwischenergebnis oder eine Seitennummer ausgegeben werden.

Die Anweisung PAGE ist nur innerhalb einer Report-Definition erlaubt und darf nur je einmal zur Definition von Seitenkopf und Seitenfuß angegeben werden. Wird ein Seitenkopf oder Seitenfuß definiert, muß in der Anweisung GLOBAL LAYOUT auch ein Kopf- oder Fußbereich definiert sein (siehe Anweisung GLOBAL LAYOUT).

Der Seitenkopf wird am Beginn und der Seitenfuß am Ende jeder Seite eines Reports ausgegeben. Eine Ausnahme davon ist, wenn ein Listenkopf oder Listenfuß eine ganze Ausgabeseite beansprucht. Dann erscheinen weder Seitenkopf noch -fuß auf der ersten oder letzten Seite eines Reports.

Ein Seitenkopf oder Seitenfuß wird dann ausgegeben, wenn der Report-Generator erkennt, daß die nächste freie Zeile identisch ist

– mit der ersten Zeile des Fußbereichs, der in der Anweisung GLOBAL LAYOUT spezifiziert wurde oder
– mit der ersten Zeile, die über die Blatthöhe hinausgeht.

Ein Seitenkopf oder -fuß kann folgende Ausgaben enthalten:

– Literale und mit SOURCE eingefügte Texte,
– die Systemvariablen &PAGES und &LINES,
– Startparameter,
– Report-Mengenwerte,
– arithmetische Ausdrücke,
– Nettodaten des aktuellen Datensatzes,
– lokale Variablen.

```
PAGE { HEADER |  TRAILER } [ zeilenmuster ] { anweisung; } ...
```

HEADER	definiert den Seitenkopf, der mit den nachfolgenden Anweisungen beschrieben wird.
TRAILER	definiert den Seitenfuß, der mit den nachfolgenden Anweisungen beschrieben wird.

zeilenmuster	Name eines Zeilenhintergrundmusters, das mit der Anweisung GLOBAL LINE BASE definiert worden sein muß. Dem Seitenkopf bzw. -fuß wird das entsprechende Zeilenmuster als Hintergrundmuster zugewiesen.
anweisung	bezeichnet eine der DRIVE-Anweisungen BREAK, CALL, CYCLE, IF, SET oder eine der Report-Anweisungen PRINT oder SOURCE. Diese Anweisungen dienen zur Beschreibung von Seitenkopf oder -fuß. Für sie gelten die beschriebenen Einschränkungen (siehe Abschnitt „Zulässige DRIVE-Anweisungen“ auf Seite 221 und Abschnitt „Einschränkungen für Report-Parameter“ auf Seite 221).

PAGE PRINT
Seitenhintergrundmuster beschreiben

Mit der Anweisung PAGE PRINT definieren Sie Texte, Linien oder Rechtecke für ein Seitenhintergrundmuster.

Die Anweisung PAGE PRINT ist nur in der Definition eines Seitenhintergrundmusters in der Anweisung GLOBAL PAGE BASE erlaubt.

```
PAGE PRINT { ausdruck x y [ CM | INCH | UNITS ]  [ mask ]  [ ANGLE n ]
                          [ format-klausel ] ... |

             SET ( { format-klausel }, ... ) |

             RESET ( { FONT | NATIONAL SET | SIZE |
                       CHARACTER DENSITY | CHARACTER DISTANCE |
                       EXPANSION HORIZONTAL | EXPANSION VERTICAL |
                       INVERSE | ITALIC | BOLD | UNDERLINE |
                       COLOUR FOREGROUND | COLOUR BACKGROUND }, ... ) |

             LINE x1 y1 x2 y2 [ CM | INCH | UNITS ]
                              [ LTYPE charliteral ]
                              [ LWIDTH x [ CM | INCH | UNITS ] ] |

             BOX x1 y1 x2 y2 [ CM | INCH | UNITS ]
                             [ BTYPE { charliteral | EMPTY } ]
                             BWIDTH x
                             [ LINE [ LTYPE charliteral ]
                                    [ LWIDTH x [ CM | INCH | UNITS ] ]  ]   |

             IMAGE LENGTH n [ WIDTH n ] [ RESOLUTION n ]
                   { [ COMPRESS charliteral ] DATA sedecliteral } ... }, ...
```

---

ausdruck x y [CM | INCH | UNITS]

Angabe des Hintergrundmusters *ausdruck* mit der Ausgabeposition *x y*. Die Positionskoordinaten beziehen sich auf einen Nullpunkt in der linken oberen Ecke des maximalen Druckbereichs (die Achsen liegen auf der linken und der oberen Kante). Berechnet wird die Ausgabeposition in der angegebenen Maßeinheit (Zentimeter, Inch oder 1/300 Zoll (UNITS)). Geben Sie keine Maßeinheit an, wird in UNITS gerechnet.
x und *y* können mit Nachkommastellen angegeben werden. Sie

müssen positiv und nicht größer als 32767 sein.
Bezeichnet *ausdruck* Variablen, müssen die angegebenen Variablen einfache Variablen sein.

mask

bezeichnet eine Darstellungsmöglichkeit (Maskensteuerzeichen) für die Ausgabe von *ausdruck* (siehe Metavariable *mask*). Für die Angabe von *mask* gelten die beschriebenen Einschränkungen (siehe Abschnitt „Einschränkungen für Report-Parameter“ auf Seite 221).

ANGLE n

Angabe eines Winkels, um den *ausdruck* gedreht weden soll. Der Winkel wird als positive, ganze Zahl angegeben. Erlaubt sind Werte zwischen 0° und 360°.
Auf HP-LaserJet-Druckern sind nur die Werte 0, 90, 180 und 270 wirksam, alle anderen Angaben werden ignoriert.

format-klausel

legt Darstellungsattribute für den aktuellen *ausdruck* fest.Folgende Darstellungsattribute sind erlaubt:

FONT, NATIONAL SET, SIZE, CHARACTER DENSITY, CHARACTER DISTANCE, EXPANSION HORIZONTAL, EXPANSION VERTICAL, BOLD, INVERSE, ITALIC, UNDERLINE, COLOUR FOREGROUND, COLOUR BACKGROUND.

Die einzelnen Darstellungsattribute sind bei der Anweisung PRINT unter *format-klausel* beschrieben.

SET ({ format-klausel }, ...)

setzt oder verändert vorbelegte Darstellungsattribute, die mit der Anweisung GLOBAL LAYOUT festgelegt wurden.

Folgende Darstellungsattribute sind in der SET-Angabe erlaubt:

FONT, NATIONAL SET, SIZE, CHARACTER DENSITY, CHARACTER DISTANCE, EXPANSION HORIZONTAL, EXPANSION VERTICAL, BOLD, INVERSE, ITALIC, UNDERLINE, COLOUR FOREGROUND, COLOUR BACKGROUND.

Beschrieben sind die einzelnen Darstellungsattribute unter *format-klausel*. Die in *format-klausel* spezifizierten Darstellungsattribute gelten ab der aktuellen Ausgabeposition bis zu der Position, an der sie zurückgesetzt werden.

RESET (...)

setzt die mit SET angebenen Darstellungsattribute wieder auf ihre Vorbelegungen gemäß der Anweisung GLOBAL LAYOUT zurück. Wurden keine globalen Vorbelegungen definiert, wird auf die gerätespezifischen Vorbelegungen umgeschaltet.

Die Bedeutung der Schlüsselworte ist unter *format-klausel* beschrieben.

LINE x1 y1 x2 y2 [CM | INCH | UNITS]

Als Hintergrundmuster wird eine Linie in Vektorgrafik gezeichnet. Die Linie wird definiert durch die Anfangskoordinaten *x1 y1* und die Endkoordinaten *x2 y2*. Koordinaten beziehen sich auf einen Nullpunkt in der linken oberen Ecke des maximalen Druckbereichs (die Achsen liegen auf der linken und der oberen Kante).

x und *y* können mit Nachkommastellen angegeben werden. Sie müssen positiv und nicht größer als 32767 sein.

Berechnet werden die Positionen in der angegebenen Maßeinheit (Zentimeter, Inch oder 1/300 Zoll (UNITS)). Geben Sie keine Maßeinheit an, wird in UNITS gerechnet.

LTYPE charliteral

legt fest, ob die Linie durchgehend, gestrichelt oder punktiert gezeichnet wird. Die für den verwendeten Drucker erlaubten Angaben *charliteral* finden Sie im jeweiligen Geräteprofil PROFILE.*geraete-tabelle*.

Geben Sie LTYPE nicht an, wird die Vorbelegung aus dem Geräteprofil verwendet.

LWIDTH x [CM | INCH | UNITS]

legt die Linienstärke fest.

Berechnet wird die Linienstärke in der angegebenen Maßeinheit (Zentimeter, Inch oder 1/300 Zoll (UNITS). Geben Sie keine Maßeinheit an, wird in UNITS gerechnet.

x kann mit Nachkommastellen angegeben werden, muß positiv und nicht größer als 32767 sein.

Geben Sie LWIDTH nicht an, wird die Vorbelegung aus dem Geräteprofil PROFILE.*geraete-tabelle* verwendet.

BOX x1 y1 x2 y2 [CM | INCH | UNITS]

Als Hintergrundmuster wird ein Rechteck gezeichnet. Das Rechteck wird definiert durch eine Begrenzungslinie und eine Ausdehnung (BWIDTH). Als Füllmuster kann ein Grauwert angegeben werden (BTYPE). Soll das Rechteck mit Umrandung gezeichnet werden, müssen Sie dem Rechteck mit LINE eine Linienstärke zuweisen (siehe LWIDTH).

Die Begrenzungslinie wird definiert durch die Anfangskoordinaten *x1 y1* und die Endkoordinaten *x2 y2*. Koordinaten beziehen sich auf einen Nullpunkt in der linken oberen Ecke des maximalen Druckbereichs (die Achsen liegen auf der linken und der oberen Kante). *x* und *y* können mit Nachkommastellen angegeben werden. Sie müssen positiv und nicht größer als 32767 sein.

Berechnet werden die Positionen in der angegebenen Maßeinheit (Zentimeter, Inch oder 1/300 Zoll (UNITS)). Geben Sie keine Maßeinheit an, wird in UNITS gerechnet.

BTYPE { charliteral | EMPTY }

legt den Grauwert des Füllmuster fest.

Die für den verwendeten Drucker erlaubten Angaben *charliteral* finden Sie im jeweiligen Geräteprofil PROFILE.*geraete-tabelle*.

EMPTY legt fest, daß dem Rechteck kein Füllmuster zugewiesen wird.

Geben Sie BTYPE nicht an, wird die Vorbelegung aus dem Geräteprofil verwendet.

BWIDTH x

legt die Ausdehnung des Rechtecks in die andere Richtung fest (im Uhrzeigersinn).
Berechnet wird die Ausdehnung in der bei BOX angegebenen Maßeinheit.

x kann mit Nachkommastellen angegeben werden, muß positiv und nicht größer als 32767 sein.
Es wird nur dann ein Rechteck ausgegeben, wenn gilt $x > 0$.

BOX ... LINE

mit der Angabe LINE weisen Sie einem parallel zum Blattrand gezeichneten Rechteck eine Umrandung zu. Sie können die Linienart festlegen (siehe LTYPE) und die Linienstärke (siehe LWIDTH).

Es wird nur dann eine Umrandungslinie gezeichnet, wenn Sie eine Linienstärke angeben.

Bei „schräg“ gezeichneten Rechtecken wird die Angabe LINE ignoriert.

IMAGE

Diese Angabe dient zum Einbinden von Rastergrafiken in das Hintergrundmuster.

LENGTH n

gibt an, wieviele Zeilen die Rastergrafik umfaßt.

n muß eine vorzeichenlose, ganze Zahl sein und darf nicht größer als 32767 sein.

WIDTH n

gibt die Briete der Rastergrafik in Pixel an.

n muß eine vorzeichenlose, ganze Zahl sein und darf nicht größer als 32767 sein.

Geben Sie WIDTH nicht an, wird für WIDTH die Breite der Druckseite eingesetzt.

RESOLUTION n

gibt die Auflösung der Rastergrafik in dots-per-inch an.

n muß eine vorzeichenlose, ganze Zahl sein und darf nicht größer als 32767 sein.
Für HP-LaserJet-Drucker sind nur die Angaben 75, 100, 150 und 300 erlaubt.

Geben Sie RESOLUTION nicht an, wird die Vorbelegung aus dem Geräteprofil PROFILE.*geraete-tabelle* verwendet.

COMPRESS charliteral

liegt die Rastergrafik in komprimierter Form vor, geben Sie mit COMPRESS die Komprimierungsart an. Mögliche Werte für *charliteral* sind:
UNENCODED: nicht komprimiert (Vorbelegung)
RLE: Run-Length-Encoding
TIFF: Tagged Imaged File Format rev. 4.0
DELTA ROW: Delta Row Compression

Die einzelnen Zeilen der Rastergrafik können auch in unterschiedlicher Komprimierungsart vorliegen.

DATA sedecliteral

Angabe der Daten, die das Rastergrafikbild beschreiben, als sedezimale Zeichenfolge (*sedecliteral* siehe Metavariable *literal*).

Ist für eine Rasterzeile keine Komprimierungsart angegeben, wird für diese Zeile der zuletzt bei COMPRESS angegebene Wert verwendet.

PRINT
Report-Ausgabe definieren

PRINT legt in jedem Kontrollblock fest,

– welche Felder an welcher Position auszugeben sind
– wieviel Zeilen vorhanden sein müssen, um die Ausgabe auf der aktuellen Seite fortzusetzen
– wieviel Leerzeilen ab der aktuellen Zeile einzufügen sind
– ob die Ausgabe auf einer neuen Seite fortzusetzen ist

Die Darstellungsattribute, die mit der Anweisung GLOBAL LAYOUT vorbelegt sind, lassen sich mit der PRINT-Anweisung verändern oder auf diese Vorbelegungen zurücksetzen.

Fehlt in einer PRINT-Anweisung die Positionsangabe, wird die Ausgabe an der aktuellen Position fortgesetzt. Wird eine Ausgabeposition durch die Spaltenanzahl spezifiziert, muß von dem Standard ausgegangen werden, der im Geräteprofil des gewählten Ausgabegeräts eingetragen ist (siehe DRIVE-Programmiersprache [2], Kapitel Der Report-Generator)

PRINT-Anweisungen sind nur innerhalb einer Report-Definition erlaubt.

```
PRINT { NEED LINES n |
        NEED SPACE n { CM | INCH | UNITS } |
        NEWLINE n |
        NEWPAGE |
        TABULATOR [ [ + ] n ] |
        POSITION x { CM | INCH | UNITS } |

        ausdruck [ mask | CLIPPED ] [ ATTRIBUT ( { format-klausel }, ... ) ]
                 [ MANDATORY | DISTINCT ]
                 [ CENTER [ x y [ CM | INCH | UNITS ] ] |
                   RIGHT x [ CM | INCH | UNITS ] ] |

        SET ( { format-klausel }, ... ) |

        RESET ( { FONT | NATIONAL SET | SIZE |
                  CHARACTER DENSITY | CHARACTER DISTANCE |
                  LINE DISTANCE | EXPANSION HORIZONTAL |
                  EXPANSION VERTICAL | SIGN | BOLD |
                  NORMALINTENSITY | INVERSE | ITALIC |
                  PROPORTIONAL | SUBSCRIPT | SUPERSCRIPT |
                  UNDERLINE | COLOUR FOREGROUND | COLOUR BACKGROUND |
                  PAPER SOURCE }, ... ) |

        PAGE POSITION x y [ CM | INCH | UNITS ] }, ...
```

NEED LINES n

Anzahl *n* der Zeilen, die auf einer Seite für den Detailbereich noch vorhanden sein müssen, um die Ausgabe fortzusetzen. Sind *n* Zeilen vorhanden, wird auf der aktuellen Seite weiter ausgegeben; sind es weniger als *n* Zeilen, wird eine neue Seite erzeugt. *n* darf nicht größer als 32767 sein.

Mit dieser Angabe wird z.B. verhindert, daß eine Überschrift auf der letzten Zeile der aktuellen Seite und die zugehörigen Listenelemente auf einer neuen Seite ausgegeben werden.

Im Seitenkopf und im Seitenfuß darf kein Seitenumbruch stattfinden.

NEED SPACE n { CM | INCH | UNITS }

NEED SPACE wirkt wie NEED LINES, nur daß die Größe des benötigten Bereichs nicht in Zeilen, sondern in CM, INCH oder 1/300 Zoll (UNITS) angegeben wird (und damit auf unterschiedlichen Druckern zu unterschiedlichen Ergebnissen führen kann).

NEWLINE n

bewirkt einen Vorschub um *n* Zeilen. Wenn *n* gleich 0 ist, wird ein Wagenrücklauf (CR) durchgeführt. *n* darf nicht größer als 32767 sein.

NEWPAGE

bewirkt einen Seitenvorschub und ist bei der Definition von Seitenkopf oder -fuß nicht erlaubt.

TABULATOR

Verwendet Tabulatoren, die mit GLOBAL LINE BASE als Zeilenhintergrundmuster definiert wurden.
Die Ausgabe wird an der nächsten im Zeilenhintergrund definierten Tabulatorposition fortgesetzt, wobei von links nach rechts positioniert wird.

TABULATOR [+] n

spezifiziert zeichenabhängig die Position, an der die Ausgabe fortgesetzt wird. Dabei gibt es zwei Möglichkeiten der Positionsangabe:

- absolut: TABULATOR *n*
- relativ: TABULATOR + *n*

Bei der absoluten Positionierung wird ab der Spalte *n* innerhalb der aktuellen Zeile ausgegeben.

Bei der relativen Positionierung wird die Anzahl *n* der Spalten angegeben, die vom Ende des vorher ausgegebenen Feldes (Bezugsfeld) aus gezählt werden, um die Ausgabeposition zu finden. *n* darf nicht größer als 32767 sein.

Ist bei der relativen Positionierung das Bezugsfeld ein alphanumerisches Feld, werden zwei Fälle unterschieden:

- Das Bezugsfeld wurde unabhängig von abschließenden Leerzeichen in der Länge ausgegeben, in der es definiert wurde. Dann wird zur Positionierung der aktuellen Ausgabe ab der letzten Position des Bezugsfeldes gerechnet, auch wenn dort ein Leerzeichen steht.
- Das Bezugsfeld wurde abhängig von abschließenden Leerzeichen nur bis zum letzten Zeichen ausgegeben, das kein Leerzeichen ist. Dann wird zur Positionierung der aktuellen Ausgabe ab der zuletzt ausgegebenen Position im Bezugsfeld gerechnet (siehe CLIPPED).

Wird die Ausgabeposition mit der Angabe TABULATOR spezifiziert, sind die Spaltenbreiten abhängig von den Einheiten des Zeichensatzes, der mit der Anweisung GLOBAL LAYOUT festgelegt wurde (siehe Anweisung GLOBAL LAYOUT).

POSITION x { CM | INCH | UNITS }

Spezifiziert die Ausgabeposition *x* in der angegebenen Maßeinheit (Zentimeter, Inch oder 1/300 Zoll (UNITS)), gemessen vom linken Seitenrand aus. *x* kann mit Nachkommastellen angegeben werden und darf nicht größer als 32767 sein.

Das Spezifizieren der Ausgabeposition mit POSITION ist dann vorteilhaft, wenn unterschiedliche Zeichengrößen oder Zeichendichten innerhalb einer Ausgabezeile verwendet werden.

ausdruck — bezeichnet einen auszugebenden Parameter *ausdruck*. Für diesen Parameter können eine Darstellungsmöglichkeit, eine Ausgabeposition, bestimmte Attribute und die Ausgabeart für Gruppenwechselfelder gewählt werden.

Bezeichnet *ausdruck* Variablen, müssen die angegebenen Variablen einfache Variablen sein.

mask — bezeichnet eine Darstellungsmöglichkeit (Maskensteuerzeichen) für die Ausgabe von *ausdruck* (siehe Metavariable *mask*). Für die Angabe von *mask* gelten die beschriebenen Einschränkungen (siehe Abschnitt „Einschränkungen für Report-Parameter“ auf Seite 221).

CLIPPED — unterdrückt die Ausgabe abschließender Leerzeichen bei Feldern vom Typ CHARACTER und VARCHAR.

Diese Angabe ist nur bei fehlender Positionsangabe oder bei relativer Positionierung der nachfolgenden PRINT-Anweisung wirkungsvoll (siehe TABULATOR).

ATTRIBUT format-klausel

spezifiziert Darstellungsattribute für *ausdruck* gemäß *format-klausel* (siehe unten, Abschnitt format-klausel). Da die angegebenen Attribute nur für *ausdruck* gelten, sind die Angaben LINE DISTANCE , PROPORTIONAL und ROTATION hier **nicht** erlaubt.

MANDATORY — Bezeichnet *ausdruck* ein Gruppenwechselfeld, so erzwingt diese Angabe, daß der Inhalt des Feldes jedesmal ausgegeben wird. Vorbelegt ist, daß der Inhalt eines Gruppenwechselfeldes nur beim erstenmal nach einem Gruppenwechsel und am Beginn neuer Seiten ausgegeben wird.

DISTINCT — Zur Ausgabe werden nur die Felder von *ausdruck* verwendet, die sich vom Inhalt des Feldes im vorangegangenen Satz unterscheiden.

CENTER [x y [CM | INCH | UNITS]]

Bewirkt, daß das Feld zentriert wird, und zwar bezüglich eines rechten und linken Randes. Die Position der Ränder bestimmen Sie durch die Werte *x* (linker Rand) und *y* (rechter Rand). Berechnet wird die Position in der angegebenen Maßeinheit (Zentimeter, Inch oder 1/300 Zoll (UNITS)) als Abstand vom linken bzw. rechten Seitenrand. *x* kann mit Nachkommastellen angegeben werden und darf nicht größer als 32767 sein.
Geben Sie keine Maßeinheit an, wird in Spalten gerechnet.
Exakt wird die Position nur für Nichtproportionalfonts berechnet.

Geben Sie keine Ränder an, wird bezüglich des linken und rechten Blattrandes zentriert.

Wenn Sie CENTER angeben, darf das PRINT-Kommando keine anderen Angaben zur Positionierung enthalten.

RIGHT [x [CM | INCH | UNITS]]

Bewirkt, daß das Feld rechtbündig bezüglich eines rechten Randes ausgegeben wird. Die Position des Randes bestimmen Sie durch den Wert *x* . Berechnet wird die Position in der angegebenen Maßeinheit (Zentimeter, Inch oder 1/300 Zoll (UNITS)) als Abstand vom rechten Seitenrand. *x* kann mit Nachkommastellen angegeben werden und darf nicht größer als 32767 sein.
Geben Sie keine Maßeinheit an, wird in Spalten gerechnet.
Exakt wird die Position nur für Nichtproportionalfonts berechnet.

Geben Sie keinen Rand an, wird das Feld am rechten Blattrand ausgerichtet.

Wenn Sie RIGHT angeben, darf das PRINT-Kommando keine anderen Angaben zur Positionierung enthalten.

SET format-klausel

setzt oder verändert vorbelegte Darstellungsattribute, die mit der Anweisung GLOBAL LAYOUT festgelegt wurden. Beschrieben sind die einzelnen Darstellungsattribute unter *format-klausel*. Die Angabe ROTATION ist nicht erlaubt. Die in *format-klausel* spezifizierten Darstellungsattribute gelten ab der aktuellen Ausgabeposition bis

- zum Ende des Kontrollblocks, in dem sie gesetzt werden oder
- zu der Position, an der sie zurückgesetzt werden.

RESET

setzt die mit SET angebenen Darstellungsattribute wieder auf ihre Vorbelegungen gemäß der Anweisung GLOBAL LAYOUT zurück. Wurden keine globalen Vorbelegungen definiert, wird auf die gerätespezifischen Vorbelegungen umgeschaltet.

Die Bedeutung der Schlüsselworte ist unter *format-klausel* beschrieben.

PAGE POSITION x y { CM | INCH | UNITS]

Bewirkt eine absolute Positionierung auf die Koordinaten *x y*. Die Koordinaten beziehen sich auf einen Nullpunkt in der linken oberen Ecke des maximalen Druckbereichs (die Achsen liegen auf der linken und der oberen Kante). Berechnet wird die Position in der angegebenen Maßeinheit (Zentimeter, Inch oder 1/300 Zoll (UNITS)). Geben Sie keine Maßeinheit an, wird in UNITS gerechnet.
x und *y* können mit Nachkommastellen angegeben werden und dürfen nicht größer als 32767 sein.
y muß einen positiven Wert haben. Hat *x* ein Vorzeichen, werden die Koordinaten bezüglich der relativen Ausgabeposition berechnet, nicht bezüglich des Nullpunkts.

Die Angabe PAGE POSITION ist nur in einem DETAIL-Kontrollblock erlaubt.
Außerdem darf die Report-Definition bei Verwendung von PAGE POSITION keine PAGE DIRECTIVEs oder GROUP DIRECTIVEs enthalten.

PAGE POSITION ist im Zusammenhang mit der Definition eines Seitenhintergrundmusters (siehe Anweisung GLOBAL PAGE BASE) gedacht für den Formulardruck.

format-klausel

Mit den Angaben der *format-klausel* können in der Anweisung GLOBAL LAYOUT Darstellungsattribute für den gesamten Report gesetzt werden. Sind keine globalen Vorbelegungen definiert, verwendet der Report-Generator die gerätespezifischen Vorbelegungen. In der PRINT-Anweisung werden die Vorbelegungen mit

ausdruck ATTRIBUT — nur für *ausdruck* verändert,

SET — ab der aktuellen Position bis zum Ende des Kontrollblocks oder bis zu einem entsprechenden RESET verändert,

RESET — auf alle oder einen bestimmten Voreinstellwert zurückgesetzt.

Bei der Angabe von Darstellungsattributen ist zu beachten, daß die Darstellungsattribute nur dann ausgeführt werden können, wenn das mit OPEN REPORT gewählt Gerät dies unterstützt. Ebenfalls zu berücksichtigen ist, daß es voneinander abhängige Attribute gibt (siehe FONT). Solche Attribute können nur dann ausgeführt werden, wenn auch die abhängigen Darstellungsattribute verändert werden. Nicht unterstützte Darstellungsattribute werden ignoriert.

```
{ ROTATION { PORTRAIT | LANDSCAPE } |
  FONT charliteral |
  NATIONAL SET charliteral |
  SIZE x |
  CHARACTER { DENSITY | DISTANCE } x |
  LINE DISTANCE x |
  EXPANSION { HORIZONTAL | VERTICAL } n |
  SIGN |
  BOLD n |
  NORMALINTENSITY n |
  INVERSE n |
  ITALIC n |
  PROPORTIONAL |
  SUBSCRIPT |
  SUPERSCRIPT |
  UNDERLINE n |
  COLOUR { FOREGROUND |  BACKGROUND } charliteral |
  PAPER SOURCE  charliteral } ...
```

ROTATION — spezifiziert die Schreibrichtung (Hoch- oder Querformat).

PORTRAIT — es wird im Hochformat geschrieben.

LANDSCAPE — es wird im Querformat geschrieben.

FONT charliteral — spezifiziert die Schriftart oder den Font *charliteral* für die Druckerausgabe des Reports. Abhängig vom Drucker kann für *charliteral* beispielsweise COURIER, ROMAN oder HELVETICA angegeben werden. Die verfügbaren Schriftarten sind in den jeweiligen Drukkerhandbüchern verzeichnet.

Die Zeichensätze bei ND-Laserdruckern heißen ABSCHNITT1 bis ABSCHNITT4 und bei HP-Laserdruckern ABSCHNITT1 bis ABSCHNITT64 (= Vorbelegung). (Siehe Anweisung OPEN REPORT, Operand spoolparameter)

Wird eine neue Schriftart gewählt, müssen ebenfalls verändert werden:

- die Zeichendichte CHARACTER DENSITY *n*
- Proportionalschrift (ein/aus) PROPORTIONAL

Bei einigen Druckern muß auch die Schriftgröße SIZE *x* neu angegeben werden. Die passenden Angaben sind ebenfalls den Drukkerhandbüchern zu entnehmen.

Fehlt die Angabe FONT *charliteral* oder verfügt der Drucker nicht über die angegebene Schriftart, wird der Report mit der vorbelegten Schriftart des Druckers ausgegeben.

NATIONAL SET charliteral

spezifiziert einen Zeichensatz, der bei der Ausgabe verwendet werden soll. Der Zeichensatz hat den Namen *charliteral*. Diese Angabe dient dazu, daß nationale Sonderzeichen wie z.B. die deutschen Umlaute "richtig" ausgegeben werden:

charliteral	sedezimal												
	23	24	40	5B	5C	5D	5E	5F	60	7B	7C	7D	7E
INT003	#	¤	@	[	\	]	^	_	`	{	|	}	¯
INT303	#	$	@	[	\	]	^	_	`	{	|	}	~
BELGIAN	#	$	à	¨	ç	°	^	_	`	é	ù	è	~
DANISH	#	$	@	Æ	Ø	Å	Ü	_	`	æ	ø	å	ü
DUTCH	#	$	@	[	\	]	^	_	`	{	|	}	~
FRENCH	#	$	à	°	ç	§	^	_	`	é	ù	è	¨
GERMAN	#	$	§	Ä	Ö	Ü	^	_	`	ä	ö	ü	ß
ITALIAN	£	$	§	°	ç	é	^	_	ù	à	ò	è	ì
NORWEGIAN	#	$	@	Æ	Ø	Å	^	_	`	æ	ø	å	¨
SPANISH	#	$	@	¡	Ñ	¿	^	_	`	´	ñ	ç	¨
SWEDISH	#	¤	É	Ä	Ö	Å	Ü	_	é	ä	ö	å	ü
SWISS	ç	$	§	à	é	è	^	¨	`	ä	ö	ü	_
UK-ENGLISH	£	$	@	[	\	]	^	_	`	{	|	}	¯

Wenn der spezifizierte Zeichensatz nicht vorhanden ist oder NATIONAL SET *charliteral* nicht angegeben wird, verwendet der Report-Generator bei der Ausgabe den vorbelegten Zeichensatz des Druckers oder den bisherigen Zeichensatz.

SIZE x

spezifiziert die Schriftgröße *x* der mit FONT *charliteral* gewählten Schriftart. *x* wird in Einheiten von 1/300 Zoll (0,0085 cm) angegeben und darf nicht größer als 32767 sein.

Bei der Angabe einer unzulässigen Größe *x* wird bei der Ausgabe auf einen druckerspezifischen Standard umgeschaltet. Wenn SIZE *x* nicht spezifiziert wird, schaltet der Drucker ebenfalls auf seinen vorbelegten Standard um.

CHARACTER	spezifiziert für die mit FONT *charliteral* gewählte Schriftart die Zeichendichte oder den Abstand zwischen den auszugebenden Zeichen. Wenn die Zeichendichte oder der Zeichenabstand nicht spezifiziert wird oder unzulässig ist, verwendet der Drucker bei der Ausgabe des Reports einen vorbelegten Standard.
DENSITY x	Die Zeichendichte wird in Anzahl *x* Zeichen pro Zoll (CPI) angegeben. *x* darf nicht größer als 32767 sein. Bei einer nicht-proportionalen Schriftart ist die Zeichendichte je Schriftgröße für alle Zeichen konstant. Ist die gewählte Schriftart *charliteral* eine Proportionalschrift, ist diese Angabe wirkungslos.
DISTANCE x	Der Zeichenabstand *x* wird in Einheiten von 1/300 Zoll (0,0085 cm) angegeben. *x* darf nicht größer als 32767 sein. Mit einem großen Zeichenabstand kann z.B. Sperrschrift erzeugt werden.
LINE DISTANCE x	spezifiziert den Abstand zwischen zwei auszugebenden Zeilen, die durch einen Zeilenvorschub (LF) voneinander getrennt sind. Der Abstand *x* wird in Einheiten von 1/300 Zoll (0,0085 cm) angegeben. *x* darf nicht größer als 32767 sein. Wenn der Zeilenabstand nicht angegeben wird oder unzulässig ist, behält der Drucker seine bisherige Einstellung bei.
EXPANSION	spezifiziert einen Faktor *n*, um den alle Zeichen in horizontaler oder vertikaler Richtung gedehnt werden. Der Faktor *n* kann einen Wert von 0 bis einschließlich 6 sein. Dabei bedeutet: 0: keine Dehnung 6: Dehnung auf doppelte Breite oder Höhe. Sowohl bei einer Dehnung in horizontaler als auch in vertikaler Richtung bleibt die Zeichengrundlinie in ihrer ursprünglichen Lage. Bei fehlender Angabe oder bei einem unzulässigen Faktor *n* behält der Drucker seine bisherige Einstellung bei.
HORIZONTAL n	Die Zeichen werden in der Breite um den Faktor *n* gedehnt.
VERTIKAL n	Die Zeichen werden in der Höhe um den Faktor *n* gedehnt.
SIGN	schaltet bei der Bildschirmausgabe das "Blinken" ein oder wieder aus (da derzeit eine Bildschirmausgabe nicht möglich ist - ohne Wirkung).

BOLD n — schaltet beim Drucken den Fettdruck ein oder aus.

n = 0: schaltet den Fettdruck aus
n = 1: schaltet normalen Fettdruck ein
n = 2: schaltet doppelten Fettdruck ein

Steht auf einem Drucker der doppelte Fettdruck nicht zur Verfügung, wird in normalem Fettdruck ausgegeben.

Wenn ein Drucker keinen Fettdruck kennt, wird normal gedruckt.

NORMALINTENSITY n — schaltet bei der Bildschirmausgabe auf einen hellen oder halbhellen Bildschirm um.

n = 0: schaltet auf den normalen hellen Bildschirm
n = 1: schaltet auf den halbhellen Bildschirm

(Da derzeit eine Bildschirmausgabe nicht möglich ist - ohne Wirkung).

INVERSE n — schaltet für die Ausgabe den inversen Modus ein oder aus.

n = 0: schaltet den Inversmodus aus
n = 1: schaltet den Inversmodus ein

Wenn ein Drucker keinen Inversmodus kennt, wird normal gedruckt.

ITALIC n — schaltet beim Drucken den Kursivdruck ein oder aus.

n = 0: schaltet den Kursivdruck aus
n = 1: schaltet normalen Kursivdruck ein
n = 2: schaltet stark geneigten Kursivdruck ein

Steht auf einem Drucker der Kursivdruck mit starker Neigung nicht zur Verfügung, wird in normalem Kursivdruck ausgegeben. Wenn ein Drucker keinen Kursivdruck kennt, wird normal gedruckt.

PROPORTIONAL — schaltet zum Drucken die Proportionalschrift ein oder wieder aus.

Ist die Proportionalschrift eingeschaltet, wird ein mit CHARACTER DISTANCE *x* angegebener Zeichenabstand ignoriert.

SUBSCRIPT — schaltet zum Drucken eine halbe Zeile tiefer oder setzt das Tiefstellen wieder zurück. Diese Angabe wird in der Anweisung GLOBAL LAYOUT **nicht** unterstützt.

Wenn der Drucker kein Tiefstellen kennt, wird normal gedruckt.

SUPERSCRIPT — schaltet zum Drucken eine halbe Zeile höher oder setzt das Hochstellen wieder zurück. Diese Angabe wird in der Anweisung GLOBAL LAYOUT **nicht** unterstützt.

Wenn der Drucker kein Hochstellen kennt, wird normal gedruckt.

UNDERLINE n schaltet beim Drucken das Unterstreichen ein oder aus.

n = 0: schaltet das Unterstreichen aus
n = 1: schaltet das Unterstreichen ein.

Ist das Unterstreichen eingeschaltet, werden alle Zeichen einschließlich Leerzeichen unterstrichen. Die Lage und die Stärke des Unterstrichs sind von der gewählten Schriftart *charliteral* abhängig und können nicht verändert werden.

Wenn der Drucker kein Unterstreichen kennt, wird normal gedruckt.

COLOUR schaltet eine Vordergrund- oder Hintergrundfarbe. Mit Vordergundfarbe ist die Farbe gemeint, in der z.B. Texte und Linien ausgegeben werden. Bereiche, die weder Text noch Linien enthalten, können mit einer Hintergrundfarbe "eingefärbt" werden.

FOREGROUND charliteral

spezifiziert die Farbe mit dem Namen *charliteral* als Vordergundfarbe.

BACKGROUND charliteral

spezifiziert die Farbe mit dem Namen *charliteral* als Hintergrundfarbe.

Besitzt ein Drucker einen Rastergenerator, kann mit *charliteral* auch der Name für eine Graustufe oder eine Schraffur angegeben werden.

PAPER SOURCE charliteral

spezifiziert den Papierschacht des Druckers.
Beim Ausdrucken wird für den Papiereinzug der Papierschacht *charliteral* verwendet.
Die beim jeweiligen Drucker zulässigen Angaben können Sie dem entsprechenden Geräteprofil entnehmen.

REPORT
Listenkontrollblock definieren

Die Anweisung REPORT beschreibt den Listenkopf oder Listenfuß eines Reports. Ein Listenkopf erscheint genau einmal am Beginn eines Reports; er kann z.B. eine Überschrift für den ganzen Report enthalten. Ein Listenfuß wird ebenfalls nur einmal am Ende eines Reports ausgegeben; in ihm können z.B. Report-Ergebnisse zusammengefaßt sein.

Die Anweisung REPORT ist nur innerhalb einer Report-Definition erlaubt und darf nur einmal je Listenkopf und Listenfuß angegeben werden.

Ein Listenkopf oder -fuß kann folgende Ausgaben enthalten:

– Literale und mit SOURCE eingefügte Texte
– die Systemvariablen &PAGES und &LINES
– Startparameter
– arithmetische Ausdrücke
– lokale Variablen

Zusätzlich können im Listenkopf die Daten des ersten Datensatzes und im Listenfuß Report-Mengenwerte sowie die Daten des letzten Datensatzes enthalten sein.

```
REPORT { HEADER | TRAILER } [ zeilenmuster ] { anweisung; } ...
```

HEADER	definiert den Listenkopf, der mit den nachfolgenden Anweisungen beschrieben wird.
TRAILER	definiert den Listenfuß, der mit den nachfolgenden Anweisungen beschrieben wird.
zeilenmuster	Name eines Zeilenhintergrundmusters, das mit der Anweisung GLOBAL LINE BASE definiert worden sein muß. Dem Listenkopf bzw. -fuß wird das entsprechende Zeilenmuster als Hintergrundmuster zugewiesen.
anweisung	bezeichnet eine der DRIVE-Anweisungen BREAK, CALL, CYCLE, IF, SET oder eine der Report-Anweisungen PRINT oder SOURCE. Diese Anweisungen dienen zur Beschreibung von Listenkopf oder Listenfuß. Für sie gelten die beschriebenen Einschränkungen (siehe Abschnitt „Zulässige DRIVE-Anweisungen“ auf Seite 221 und Abschnitt „Einschränkungen für Report-Parameter“ auf Seite 221).

Beispiel

In einem Listenkopf soll am linken oberen Rand die Adresse einer "Meinungsforschungs AG", am rechten oberen Rand das aktuelle Datum und darunter zentriert die Überschrift des Reports ausgegeben werden:

```
...
REPORT HEADER
      PRINT NEWLINE 1,
            SET (ITALIC 1, FONT 'CENTURY CONDENSED', CHAR DENSITY 10),

            TAB 5,  'Meinungsforschungs A.G.',
            TAB 55, 'Datum: ', CURRENT DATE MASK 'DD''.''MO''.''YYYY',
            NEWLINE 1,
            TAB 5,  'Hans Sachs Gasse 14',
            NEWLINE 1,
            TAB 5,  '1180 Wien',
            NEWLINE 4,

            SET (ITALIC 1, FONT 'COURIER', NATIONAL SET 'GERMAN',
                 EXPANSION HORIZONTAL 1, EXPANSION VERTICAL 1, BOLD 1),

            TAB 8,  '*** ',
                    'KONSUMVERHALTEN' ATTRIBUTE (UNDERLINE 1),
                    ' ***',
            NEWLINE 3;
...
```

Ausgabe:

```
Meinungsforschungs A.G.                              Datum: 13.03.1992
Hans Sachs Gasse 14
1180 Wien

                        *** KONSUMVERHALTEN ***
```

SOURCE
Textdatei einfügen

SOURCE gibt Textdateien innerhalb eines Reports aus. Der Report-Generator fügt den Inhalt der angegebenen Datei unverändert an der aktuellen Ausgabeposition ein. Erstreckt sich der Text über mehr als eine Seite, werden keine Seitenköpfe oder -füße ausgegeben.

```
SOURCE datei [ n [ n ] ]
```

---

datei — bezeichnet die Datei, die den einzufügenden Text enthält.

Handelt es sich um eine PostScript-Datei, muß diese Datei im Encapsulated PostScript-Format (EPS-Format) vorliegen. Die Datei wird so eingefügt, daß die linke obere Ecke der in der Datei definierten Bounding Box an der aktuellen Ausgabeposition liegt.

n — Das erste *n* steht für die Anzahl der Zeilen der letzten Seite des Textes in der Datei *name*. Das zweite *n* steht für die Anzahl der Seiten. Wird die Anzahl der Seiten nicht angegeben, interpretiert der Report-Generator das erste *n* als Anzahl der Zeilen in der gesamten Datei.

Mit dieser Angabe ist der Report-Generator in der Lage, interne Seiten- und Zeilenzähler um die Werte *n* zu verändern. Dadurch können nach der Ausgabe des Textes, Seitenumbrüche richtig durchgeführt und Seitennummern aktualisiert werden.

Beispiel

In einer Liste werden nebeneinander die Daten der Variablen &schluessel, &artnam und &preis ausgegeben. Jeder dieser Ausgabezeilen soll der Inhalt der Datei "d.report.source" folgen.

```
...
DETAIL
  PRINT NEWLINE 1,
        TAB  2, &schluessel,
        TAB 10, &artname,
        TAB 40, &preis;

  SOURCE "d.report.source";
...
```

Der Name der Datei muß in Anführungszeichen stehen, weil er Sonderzeichen enthält.

STANDARD LAYOUT
Layout eines Standard-Reports beschreiben

STANDARD LAYOUT veranlaßt das Generieren eines Standard-Reports. Die Anweisung muß zwischen den Anweisungen DECLARE REPORT und END REPORT angegeben werden. Bei einem Standard-Report wird die Report-Definition aus der Satzbeschreibung der Parameter abgeleitet, die der Report-Generator aus dem DRIVE-Programm übernimmt.

Zur Gestaltung eines Standard-Reports stehen drei vorgegebene Formate zur Verfügung:

TABLE	Ausgabe im Tabellenformat
LINE	Ausgabe in vertikalem Format
SEQUENCE	Ausgabe in horizontalem Format

```
STANDARD LAYOUT { TABLE [ FILL charliteral ] |
                  LINE |
                  SEQUENCE }
```

TABLE	fordert das Tabellenformat an. Im Tabellenformat wird eine Überschriftszeile mit den Namen der Komponenten der niedrigsten Stufe erzeugt. Unter dem jeweiligen Namen werden die zugehörigen Feldinhalte aufgelistet. Wieviel Platz in einer Zeile für den Feldinhalt einer Komponenten zur Verfügung steht, richtet sich entweder nach der Länge des Namens oder des längsten Feldinhalts. Die Spaltenbreite wird nach dem längeren von beiden bemessen. Ist die Summe aller Spaltenbreiten einer Tabellenzeile größer als auf dem gewünschten Gerät ausgebbar, wird die Ausgabezeile rechts abgeschnitten und das DRIVE-Programm wird nach der Anweisung CLOSE REPORT mit einem Fehler abgebrochen. Matrizen und Wiederholungsgruppen dürfen im Format TABLE nicht ausgegeben werden.
FILL charliteral	definiert die Zeichen *charliteral*, die zur Trennung nebeneinander liegender Ausgabefelder verwendet werden. Vorbelegung: ein Leerzeichen

LINE

fordert das vertikale Format an. In diesem Format werden Komponenten von Datengruppen (siehe Metavariable *datengruppe*) und Wiederholungsgruppen (siehe Metavariable *wiederholungsgruppe*) untereinander ausgegeben. Komponenten vom Typ grunddatentyp mit Wiederholungsfaktor werden einzeilig in der Form *komponentenname: wert1 wert2 ...* ausgegeben. Die Komponenten von Wiederholungsgruppen dürfen keinen Wiederholungsfaktor enthalten, müssen also eindimensional sein. Matrizen werden in Matrixform ausgegeben, ihr Name erscheint an jedem Matrixzeilenanfang.

Nebeneinander liegende Ausgabefelder werden durch ein Leerzeichen voneinander getrennt (Vorbelegung).

Bei einem Zeilenüberlauf wird am rechten Zeilenrand abgeschnitten, und das DRIVE-Programm wird nach der Anweisung CLOSE REPORT mit einem Fehler abgebrochen.

SEQUENCE

fordert das horizontale Format an. Die Ausgabedaten werden in der Form *name: wert* bzw. *komponentenname[(n)]: wert* hintereinander in eine Zeile geschrieben.

Die Ausgabe von Matrizen und von Wiederholungsgruppen wird nicht unterstützt.

Nebeneinander liegende Ausgabefelder werden durch ein Leerzeichen voneinander getrennt (Vorbelegung).

Bei einem Zeilenüberlauf wird am rechten Zeilenrand abgeschnitten, und das DRIVE-Programm wird nach der Anweisung CLOSE REPORT mit einem Fehler abgebrochen.

Beispiele

Zur Demonstration der drei Formate, die Sie mit der Anweisung STANDARD LAYOUT anfordern können, wird folgende Variablendefinition gegeben:

```
/* Im DRIVE-Programm werden die zu verarbeitenden Variablen deklariert */

DECLARE VARIABLE
  1 &anschrift(2),
    2 ort(2)      CHAR(8),
    2 tel         CHAR(7);
.
.
/* Die Beschreibung der DRIVE-Daten wird an die Report-Definition    */
/* übergeben. Der Wiederholungsfaktor wird nicht übernommen.        */

DECLARE REPORT stand USING &daten LIKE &anschrift;

   STANDARD LAYOUT TABLE FILL '|';    /* für Ausgabe im Tabellenformat      */

   oder

   STANDARD LAYOUT LINE;              /* für Ausgabe im vertikalen Format   */

   oder

   STANDARD LAYOUT SEQUENCE;          /* für Ausgabe im horizontalen Format */

END REPORT;

/* Der Report wird im DRIVE-Programm geöffnet, mit Daten gefüllt     */
/* und geschlossen.                                                  */

OPEN REPORT stand RESULT LIST 'druck' DEVICETABLE '9002';
   FILL REPORT stand USING &anschrift(1);
   FILL REPORT stand USING &anschrift(2);
CLOSE REPORT stand;
```

Ausgabe im Fall STANDARD LAYOUT TABLE

```
1995 Sep 12  16 : 08   -   Standard Report TABLE   -   Page:   1

ORT(1)    ORT(2)    TEL
--------------------------
MUENCHEN| BERLIN  | 1234567
BERLIN  | HAMBURG | 1234567
```

Ausgabe im Fall STANDARD LAYOUT LINE

```
1995 Sep 12  16 : 15   -   Standard Report LINE   -   Page:   1

1 DATEN
2 ORT: MUENCHEN BERLIN
2 TEL: 1234567
1 DATEN
2 ORT: BERLIN   HAMBURG
2 TEL: 1234567
```

Ausgabe im Fall STANDARD LAYOUT SEQUENCE

```
1995 Sep 12  16 : 23   -   Standard Report SEQUENCE   -   Page:   1

ORT(1): MUENCHEN ORT(2): BERLIN  TEL: 1234567
ORT(1): BERLIN   ORT(2): HAMBURG TEL: 1234567
```

Im Format STANDARD LAYOUT LINE können Sie auch Wiederholungsgruppen ausgeben. Die Komponenten dieser Wiederholungsgruppen dürfen allerdings keine Wiederholungsfaktoren enthalten (d.h. die Komponenten müssen eindimensional sein).

```
DECLARE VARIABLE
  1 &anschrift(2),
    2 ort         CHAR(20),
    2 tel         CHAR(7);
.
.

DECLARE REPORT stand USING 1 &daten(2),
                             2 ort    CHAR(20),
                             2 tel    CHAR(7);
      STANDARD LAYOUT LINE;
END REPORT;

OPEN REPORT stand RESULT LIST 'druck' DEVICETABLE '9002';
   FILL REPORT stand USING &anschrift;
CLOSE REPORT stand;
```

Ausgabe

```
1995 Sep 12  17 : 19  -   Standard Report LINE   -   Page:   1

1 DATEN (1)
2 ORT: MUENCHEN    BERLIN
2 TEL: 1234567
1 DATEN (2)
2 ORT: BERLIN      HAMBURG
2 TEL: 1234567
```

5 DRIVE-Metavariablen

attribute
Feldattribut beschreiben

attribute beschreibt die Feldeigenschaften für Formate. (Umfassende Information hierzu siehe IFG für FHS [28], FHS [29] sowie FORMANT [39] und UTM Formatierungssystem [32].)

```
attribute::=[ UNPROTECTED | PROTECTED ] |
            [ HIGHINTENSITY | NORMALINTENSITY ] |
            [ VISIBLE | SIGN | INVISIBLE ] |
            [ INIT | NOINIT ] |
            [ VALID | INVALID ] |
            [ HARDCOPY | ALARM | DEFAULT ] |
            [ MUST | POTMUST | NORMALINPUT ] |
            [ INVERSE | NOINVERSE ] |
            [ UNDERLINE | NOUNDERLINE ] |
            [ CURSOR | NOCURSOR ] |
            [ BLUE | CYAN | GREEN | MAGENTA | RED | WHITE | YELLOW | NOCOLOUR ]
```

Die Feldeigenschaften lassen sich in zwei Gruppen einteilen: In Feldattribute, die sich auf ein Bildschirmfeld beziehen, und in Globalattribute, die sich auf einen kompletten Bildschirm beziehen. Feldattribute sind zu Feldattributgruppen zusammengefaßt.

Feldattribute

Feldattribute beziehen sich auf jeweils ein Bildschirmfeld oder auf eine Variable, da Bildschirmfelder ja nur über Variablen ansprechbar sind.

Feldattributgruppe Art der Eingabe

MUST	Eingabepflicht: In das Feld muß ein Eintrag gemacht werden.
POTMUST	einmalige Eingabepflicht, d.h. bei einer Eingabewiederholung muß der Eintrag nicht noch einmal gemacht werden.
NORMALINPUT	keine Eingabepflicht.
UNPROTECTED	Ungeschütztes Feld: Der Feldinhalt kann über die Tastatur verändert werden. Alle Zeichen dürfen eingegeben werden.
PROTECTED	Geschütztes Feld: Der Feldinhalt kann über Tastatur nicht verändert werden.

Feldattributgruppe Darstellung

HIGHINTENSITY	Hohe Intensität: Am Bildschirm bewirkt dieser Wert eine hohe Helligkeitseinstellung, bei Druckerstationen Fettdruck (falls das Gerät diese Funktion unterstützt). Vorbelegung zur Kennzeichnung fehlerhafter Bildschirmfelder.
NORMALINTENSITY	Normale Intensität.
VISIBLE	Der Feldinhalt ist voll sichtbar.
SIGN	Der Feldinhalt ist blinkend sichtbar; bei Druckerstationen bewirkt dieser Wert Schattendruck (falls das Gerät diese Funktion unterstützt).
INVISIBLE	Der Feldinhalt ist nicht sichtbar und bei Hardcopy nicht abdruckbar.
UNDERLINE	Der Feldinhalt wird unterstrichen dargestellt. Vorbelegung zur Kennzeichnung fehlerhafter Bildschirmfelder.
NOUNDERLINE	Der Feldinhalt wird nicht unterstrichen dargestellt.
INVERSE	Der Feldinhalt wird invers dargestellt.
NOINVERSE	Der Feldinhalt wird nicht invers dargestellt.

Feldattributgruppe Schreibmarke

CURSOR — Die Schreibmarke (Cursor) wird auf die erste Eingabestelle des Feldes positioniert. CURSOR kann auch als Globalattribut angegeben werden.

FHS wertet das Feldattribut CURSOR nur aus, wenn gleichzeitig das Globalattribut CURSOR gesetzt ist.

NOCURSOR — Die Schreibmarke wird im Feld nicht positioniert. NOCURSOR kann auch als Globalattribut angegeben werden.

Feldattributgruppe Farbe

BLUE bis NOCOLOUR

Bei mehrfarbigen Datensichtstationen kann für die Datenausgabe eine Farbe festgelegt werden:

BLUE - blau
CYAN - cyan (blaugrün)
GREEN - grün
MAGENTA - magenta (lila)
RED - rot
WHITE - weiß
YELLOW - gelb
NOCOLOUR - keine Farbe, d.h. Normaldarstellung des Bildschirms

Weitere Feldattribute

VALID — Der Feldinhalt ist geprüft und fehlerfrei, oder es ist keine Editroutine verlangt.

INVALID — Der Feldinhalt ist geprüft und fehlerhaft.

Globalattribute

Globalattribute beziehen sich auf jeweils ein Bildschirmformat, d.h. die Eigenschaften der Globalattribute wirken sich auf den kompletten Bildschirm aus.

INIT	Datenübergabebereiche, die bereits mit Daten versorgt sind, werden auf den Grundzustand zurückgesetzt: Somit werden alle Feldattribute entsprechend ihren vorbelegten Werten im Format versorgt.
NOINIT	Ausgabe ohne Initialisierung.
HARDCOPY	Automatischer Hardcopy-Betrieb: Der gesamte Bildschirminhalt wird nach Ausgabe der Nachricht automatisch auf das Hardcopy-Gerät ausgegeben (falls die Datensichtstation mit Hardcopy generiert ist).
ALARM	Ausgabe mit Alarm. Dieses Globalattribut wirkt nur auf Geräten, die eine Alarmfunktion (optisch und/oder akustisch) unterstützen.
DEFAULT	Rücksetzen aller Attribute auf ihren vorbelegten Wert. DEFAULT wirkt auf alle Feldattribute der Adressierungshilfe. Es werden (außer den Feldattributen VALID und INVALID) alle Feldattributgruppen auf ihren vorbelegten Wert zurückgesetzt (Informationen zum vorbelegten Wert der einzelnen Feldattribute finden Sie im Handbuch "UTM. Formatierungssystem" [32]). Globalattribute werden nicht auf ihren vorbelegten Wert zurückgesetzt.
CURSOR	Das Feldattribut CURSOR wird ausgewertet.
NOCURSOR	Das Feldattribut CURSOR wird nicht ausgewertet.

Kombinationsmöglichkeiten der Feldattribute (für dynamische Attribute)

UNPROTECTED kann mit allen anderen Feldattributen beliebig kombiniert werden. Für die anderen Feldattribute gelten folgende Kombinationsmöglichkeiten:

Feldattribut	**zulässige Kombinationen**							**Gruppe**
HIGHINTENSITY	*	*				*		1
NORMALINTENSITY			*	*			*	
VISIBLE	*	*	*	*				
SIGN						*	*	
INVISIBLE					*			
UNDERLINE	*		*					
NOUNDERLINE		*		*				
GREEN		*						2
RED	*							
WHITE			*					
YELLOW				*				

An einem Farbbildschirm werden die Angaben über Intensität, Sichtbarkeit und Unterstreichung (Gruppe 1) in den aus der Tabelle ersichtlichen Farben dargestellt (Gruppe 2). Die Angabe HIGHINTENSITY, VISIBLE, UNDERLINE hat somit dieselbe Wirkung wie die Angabe RED.

Umgekehrt wird die Angabe RED an einem Nicht-Farbgerät in die Kombination HIGHINTENSITY, VISIBLE, UNDERLINE umgesetzt.

Es dürfen nur Feldattribute der Gruppe 1 oder der Gruppe 2 angegeben werden. Bei doppelter Angabe haben die Feldattributangaben der Gruppe 1 Vorrang vor den Feldattributangaben der Gruppe 2.

ausdruck
Ausdrücke

ausdruck beschreibt numerische, Charakter-, Datumzeit- und Intervallausdrücke.

```
ausdruck::={ charausdruck | datumzeitausdruck | intervallausdruck | numausdruck }
```

---

charausdruck — siehe Metavariable *charausdruck*:

```
charausdruck::={ charprim | charausdruck || charprim }
```

---

datumzeitausdruck — siehe Metavariable *datumzeitausdruck*:

```
datumzeitausdruck::=
 { datumzeitterm | datumzeitausdruck { + | - } intervallterm }
```

---

intervallausdruck — siehe Metavariable *intervallausdruck*:

```
intervallausdruck::=
 { intervallterm | ( datumzeitausdruck - datumzeitterm ) }
```

---

numausdruck siehe Metavariable *numausdruck*:

```
numausdruck::=
 numterm1 [ [ * | / | % | ** ] [ + | - ] numterm2 ]
```

basistyp
Klausel definieren

basistyp legt folgende Klauseln fest:

INIT-Klausel	Einer Variablen wird ein Anfangswert zugewiesen.
REDEFINES-Klausel	Für den Speicherbereich einer Variablen werden mehrere Beschreibungen angegeben.
CHECK-Klausel	Für eine Variable wird eine Bedingung vereinbart und während des Programmlaufes überprüft (siehe Metavariable *check*).
MASK-Klausel	Für Ein- und Ausgabefelder wird die Darstellung von Datenwerten festgelegt (siehe Metavariable *mask*).

Weitere Datendefinitionen siehe Metavariable *datendef*.

```
basistyp::={ INIT ausdruck [ NOCHECK ] |
             REDEFINES { variable | komponente [ suffix ] } }
           [ check ] [ mask ]
```

INIT	Mit INIT wird einer Variablen ein Anfangswert zugewiesen.
ausdruck	*ausdruck* darf nur Literale, NULL oder Funktionen, deren Argumente Literale sind, enthalten. *ausdruck* darf nicht CURRENT DATE / TIME / TIMESTAMP enthalten. *ausdruck* muß zum Übersetzungszeitpunkt berechenbar sein.
NOCHECK	Die Angabe NOCHECK bewirkt, daß für den mit INIT *ausdruck* zugewiesenen Anfangswert die CHECK-Klausel nicht berücksichtigt wird (siehe Metavariable *check*). Bei redefinierten Variablen oder einer Variablen, die eine andere definiert, ist die Angabe NOCHECK nicht erlaubt.
REDEFINES	Mit REDEFINES wird die Variable gekennzeichnet, die redefiniert wird (Basisvariable). Die redefinierende Variable darf nicht länger als die Basisvariable sein. Ist die redefinierende Variable Komponente einer Struktur, muß die Basisvariable eine Komponente derselben Hauptstruktur (Struktur mit der Stufennummer 1) sein.

Die redefinierende Variable darf nicht den Datentyp VARCHAR haben.

variable — Name der Basisvariablen, die redefiniert wird. Sie muß bereits definiert worden sein und darf selbst keine redefinierende Variable sein. Ist die Basisvariable eine Struktur, darf die redefinierende Variable keine Komponente dieser Struktur sein.

Zwischen dem Beginn der kleinsten Teilstruktur, die sowohl die Basisvariable als auch die redefinierende Variable enthält, und der Basisvariablen oder der redefinierenden Variablen dürfen keine Wiederholungsgruppen liegen.

komponente — Alphanumerische Zeichen für den Namen einer Komponente.

0 < Anzahl(*komponente*) < 32

suffix — siehe Metavariable *variable*

check — siehe Metavariable *check*

mask — siehe Metavariable *mask*

Beispiele

REDEFINES auf eine andere Variable:

```
DECLARE VARIABLE &a CHAR (8),
                 &b CHAR (2) REDEFINES &a;
```

&b belegt die ersten 2 Byte des &a zugeordneten Bereichs neu.

REDEFINES und INIT:

```
DECLARE VARIABLE &a CHAR (8),
                 &b CHAR (2) INIT 'VERBOTEN' REDEFINES &a;
                                             *
```

□ Fehler: bei INIT verboten

bedingung
Bedingungen

Eine Bedingung besteht aus einem oder mehreren logischen Ausdrücken und den logischen Operatoren AND, OR oder NOT.

Folgende Bedingungen können gestellt werden:

- Vergleichen von Ausdrücken durch Vergleichsoperatoren (siehe Seite 291)
- Vergleichen eines Ausdrucks mit einem Wertebereich (siehe Seite 292)
- Vergleichen eines Ausdrucks mit einer Liste von Werten (siehe Seite 293)
- Vergleichen eines Wertes mit dem NULL-Wert (siehe Seite 295)
- Prüfen, ob eine Zeichenfolge numerisch ist (siehe Seite 296)

Hinweise für die Auswertung von Bedingungen

- NULL-Werte in Bedingungen

  Wenn NULL-Werte in Bedingungen vorkommen, kann das Ergebnis von Bedingungen neben erfüllt und nicht erfüllt auch unbestimmt sein. Wann das Ergebnis einer Bedingung erfüllt, nicht erfüllt oder unbestimmt ist, ist jeweils bei der Bedingung beschrieben.

- Vergleich alphanumerischer Werte

  Zwei Zeichenketten werden von links nach rechts verglichen. Bei unterschiedlich langen Zeichenketten wird die kürzere mit Leerzeichen aufgefüllt. Zwei Zeichenketten sind gleich, wenn sie an jeder Position das gleiche Zeichen haben. Andernfalls legt das erste unterschiedliche Zeichen fest, welche Zeichenkette größer oder kleiner ist.

- Vergleich numerischer Werte

  Zwei numerische Werte sind gleich, wenn sie dasselbe Vorzeichen und denselben Betrag haben.

Es gilt folgende Syntax für *bedingung*:

```
bedingung::=[ NOT ] { ( bedingung1 [ { AND | OR } bedingung2 ) ] ... ) |
                      ( ausdruck1 { = | < | > | <> | <= | >= } ausdruck2 ) |
                      ( ausdruck3 [ NOT ] BETWEEN ausdruck4 AND ausdruck5 ) |
                      ( ausdruck6 [ NOT ] IN ( wert, ... ) ) |
                      ( wert IS [ NOT ] NULL ) |
                      ( charausdruck IS [ NOT ] NUMERIC ) }
```

Die folgenden Ergebnisse von *bedingung* sind für AND, OR und NOT möglich:

AND logisches UND: beide mit AND verknüpften Bedingungen müssen erfüllt sein, damit die gesamte Bedingung erfüllt ist.

	bedingung2		
bedingung1	erfüllt	nicht erfüllt	unbestimmt
erfüllt	erfüllt	nicht erfüllt	unbestimmt
nicht erfüllt	nicht erfüllt	nicht erfüllt	nicht erfüllt
unbestimmt	unbestimmt	nicht erfüllt	unbestimmt

OR logisches ODER: mindestens eine der beiden mit OR verknüpften Bedingungen muß erfüllt sein, damit die gesamte Bedingung erfüllt ist.

	bedingung2		
bedingung1	erfüllt	nicht erfüllt	unbestimmt
erfüllt	erfüllt	erfüllt	erfüllt
nicht erfüllt	erfüllt	nicht erfüllt	unbestimmt
unbestimmt	erfüllt	unbestimmt	unbestimmt

NOT

Negation: die mit NOT verknüpfte Bedingung darf nicht erfüllt sein, damit die gesamte Bedingung erfüllt ist.
Ist das Ergebnis der mit NOT verknüpften Bedingung unbestimmt, so ist auch das Ergebnis der gesamten Bedingung unbestimmt.

bedingung	NOT *bedingung*
erfüllt	nicht erfüllt
nicht erfüllt	erfüllt
unbestimmt	unbestimmt

bedingung

Solange *bedingung* erfüllt ist, wird die gewünschte Funktion ausgeführt.

Regeln

– Die Datentypen von *ausdruck1* und *ausdruck2* müssen vergleichbar sein (beide numerischer, alphanumerischer, Zeit-Datentyp oder Datentyp INTERVAL).

– Beim Vergleich von Zeit-Datentypen müssen beide Ausdrücke ein Datum, eine Uhrzeit oder ein Zeitstempel sein.

– Ist *ausdruck* ein strukturierter Wert, darf nur ein Vergleich mit = oder <> vorkommen.

– Die Konstante NULL darf nicht angegeben werden.

– Wenn Sie die logischen Operatoren AND, OR und NOT kombinieren, so gelten die üblichen Vorrangregeln für die Auswertung:

  NOT vor AND vor OR

  Wenn Sie die beschriebene Reihenfolge ändern wollen, so müssen Sie entsprechend Klammern setzen. Operatoren innerhalb der Klammern haben Vorrang.
  Operatoren mit derselben Priorität werden von links nach rechts abgearbeitet.

  Nimmt *bedingung* in der Anweisung IF den Wahrheitswert UNBESTIMMT oder FALSCH an, wird in den ELSE-Pfad verzweigt.

  Nimmt *bedingung* in der Anweisung CYCLE den Wahrheitswert UNBESTIMMT oder FALSCH an, wird die Ausführung der Anweisung beendet.

Vergleichen von Ausdrücken mit Vergleichsoperatoren

Mit Vergleichsoperatoren können Sie die Werte zweier Ausdrücke miteinander vergleichen.

Vergleichsoperator	Bedeutung
=	gleich
<	kleiner
>	größer
<=	kleiner oder gleich
>=	größer oder gleich
<>	ungleich

Die Bedingung ist erfüllt, wenn der Vergleich zutrifft.

Das Ergebnis der Bedingung ist unbestimmt, wenn mindestens ein Ausdruck den NULL-Wert annimmt. Andernfalls ist die Bedingung nicht erfüllt.

Regeln

– Die Datentypen von *ausdruck1* und *ausdruck2* müssen vergleichbar sein (beide numerischer, alphanumerischer, Zeit-Datentyp oder Datentyp INTERVAL).

– Beim Vergleich von Zeit-Datentypen müssen beide Ausdrücke ein Datum, eine Uhrzeit oder ein Zeitstempel sein.

– Vektoren dürfen nicht in einem Vergleich mit < > <= oder >= vorkommen.

– Die Konstante NULL darf nicht angegeben werden.

Beispiel

```
SELECT bezeichnung
  FROM projekt
  WHERE budget >= &minbudget

IF &budget < &var ... THEN
```

Vergleichen eines Ausdrucks mit einem Wertebereich

Es wird geprüft, ob der Wert des Ausdrucks innerhalb oder außerhalb des Wertebereichs liegt.

```
ausdruck1 [ NOT ] BETWEEN ausdruck2 AND ausdruck3
```

BETWEEN ... AND
: Das Ergebnis der Bedingung ist dasselbe wie für die Bedingung

ausdruck2 <= *ausdruck1* AND *ausdruck1* <= *ausdruck3*.

Die Bedingung ist erfüllt, wenn der Wert von *ausdruck1* innerhalb des Wertebereichs liegt.

NOT BETWEEN ... AND
: Das Ergebnis der Bedingung ist dasselbe wie für die Bedingung

ausdruck1 < *ausdruck2* OR *ausdruck1* > *ausdruck3*.

Die Bedingung ist erfüllt, wenn der Wert von *ausdruck1* außerhalb des Wertebereichs liegt.

Regeln

- Die Datentypen von *ausdruck1*, *ausdruck2* und *ausdruck3* müssen verträglich sein (alle numerischer, alphanumerischer, Zeit-Datentyp oder Datentyp INTERVAL).
- Beim Vergleich von Zeit-Datentypen müssen alle Ausdrücke ein Datum, eine Uhrzeit oder ein Zeitstempel sein.
- Die Konstante NULL darf nicht angegeben werden.
- Vektoren sind bei einem Vergleich mit BETWEEN ... AND nicht zulässig.

Beispiel

```
SELECT nachname, gehalt
   FROM mitarbeiter
   WHERE gehalt BETWEEN &untergrenze AND 6000

IF &gehalt BETWEEN 1000 AND &obergrenze THEN ...
```

Vergleichen eines Ausdrucks mit einer Liste von Werten

Der Ausdruck wird verglichen mit einem Wert oder einer Liste von Werten.

```
ausdruck [ NOT ] IN ( wert, ... )
```

IN	Die Bedingung ist erfüllt, wenn der Vergleich für mindestens einen Wert zutrifft. Die Bedingung ist nicht erfüllt, wenn der Vergleich für keinen Wert zutrifft. Andernfalls ist das Ergebnis der Bedingung unbestimmt.
NOT IN	Die Bedingung ist erfüllt, wenn der Vergleich für jeden Wert von *ausdruck* zutrifft. Die Bedingung ist nicht erfüllt, wenn der Vergleich für mindestens einen Wert nicht zutrifft. Andernfalls ist das Ergebnis des Vergleichs unbestimmt.
wert	Wert mit numerischem, alphanumerischem oder Datentyp INTERVAL, der durch eine Konstante oder eine Variable angegeben wird (siehe Metavariable *wert*).

Regeln

– Der Datentyp von *ausdruck* muß verträglich sein mit dem Datentyp der nach IN spezifizierten Werte (numerischer, alphanumerischer oder Datentyp INTERVAL).

– Beim Vergleich von Zeit-Datentypen müssen alle nach IN spezifizierten Werte entsprechend *ausdruck* ein Datum, eine Uhrzeit oder ein Zeitstempel sein.

– Vektoren dürfen in Vergleichen mit IN nicht vorkommen.

Beispiele

```
SELECT abt_mit_nr, lfd_nr, nachname
       FROM mitarbeiter
       WHERE ort IN ('Muenchen','Frankfurt', &varort)
```

```
SELECT abt_mit_nr, lfd_nr, nachname
       FROM mitarbeiter INTO &var
       WHERE ort NOT IN ('Hamburg','Kiel','Hannover', &varort)
```

```
SELECT personal_nr, nachname
       FROM mitarbeiter INTO &var
       WHERE gehalt IN (SELECT gehalt
                        FROM mitarbeiter
                        WHERE gehalt > 600000 AND gehalt < &max)

IF &stadt IN ('Hamburg','Kiel','Hannover') ... THEN
```

Vergleichen eines Wertes mit dem NULL-Wert

```
wert IS [ NOT ] NULL
```

wert	Wert, der auf den NULL-Wert überprüft werden soll.
IS NULL	Die Bedingung ist erfüllt, wenn *wert* den NULL-Wert enthält. Andernfalls ist die Bedingung nicht erfüllt.
IS NOT NULL	Die Bedingung ist erfüllt, wenn *wert* nicht den NULL-Wert enthält. Andernfalls ist die Bedingung nicht erfüllt.

Beispiel

```
SELECT abt_mit_nr, lfd_nr, nachname
       FROM mitarbeiter
       WHERE proj_mit IS NULL

IF &var IS NULL THEN ...
```

Prüfen, ob eine Zeichenfolge numerisch ist

Eine Zeichenfolge wird überprüft, ob sie einen numerischen Wert repräsentiert.

Diese Überprüfung darf nur bei CYCLE, IF und CASE, aber nicht in SQL-Anweisungen angegeben werden.

```
charausdruck IS [ NOT ] NUMERIC
```

---

charausdruck	Zeichenfolge, die überprüft werden soll.
IS NUMERIC	Die Bedingung ist erfüllt, wenn die *wertefunktion* NUMERIC ohne Angabe von *[, charliteral]* (d.h. mit Standardmaske) auf die Zeichenkette angewendet wird und keinen Konvertierungsfehler ergibt.
IS NOT NUMERIC	Die Bedingung ist erfüllt, wenn die *wertefunktion* NUMERIC ohne Angabe von *[, charliteral]* (d.h. mit Standardmaske) auf die Zeichenkette angewendet wird und einen Konvertierungsfehler ergibt.

Beispiele

```
SET &c = 'ABCDEF';

  IF &c IS NOT NUMERIC
     THEN DISPLAY FORM 'Nicht numerisch';
  END IF;

SET &c = '1000';

  IF &c IS NUMERIC
     THEN SET &n = NUM(&c)
     ELSE SET &n = 0;
  END IF;
```

charausdruck
Charakterausdruck definieren

charausdruck legt Charakterausdrücke fest. Der Datentyp von *charausdruck* muß alphanumerisch sein.

charausdruck kann auch eine leere Zeichenkette sein. Die Höchstlänge beträgt 32000 Zeichen.

Weitere Ausdrücke siehe Metavariable *ausdruck*.

```
charausdruck::={ charprim | charausdruck || charprim }
```

---

charprim	siehe Metavariable *charprim*
charausdruck	*charausdruck* kann auch ein Literal oder eine Variable sein. Hat *charausdruck* den Wert NULL, hat auch *charprim* den Wert NULL.
||	Mit dem Verknüpfungsoperator || werden Zeichenketten miteinander verknüpft. Dazu wird die zweite Zeichenkette direkt an die erste angehängt. Die maximale Länge einer Verknüpfung ist 32000 Byte. Die verwendeten Ausdrücke dürfen weder strukturierte Variablen noch Aggregate enthalten.

charprim
Stringfunktionen

Mit *charprim* werden folgende Funktionen ausgeführt:

- Verknüpfen von Zeichenketten
- Auswählen von Teilketten
- Ersetzen und Ändern von Teilketten
- Löschen von Teilketten
- Linksbündiges Ausgeben von Zeichenketten
- Umsetzen von Zeichen in Zeichenketten
- Meldungen aus einer Meldungsdatei ausgeben
- Verknüpfen aller atomaren Felder von strukturierten Variablen

---

```
charprim::={ wert1 |
             satzelement |
             ( charausdruck ) |
             CONCAT ( charausdruck1, charausdruck2 ) |
             SUBSTRING ( charausdruck3, startpos1 [, länge1 ] ) |
             UPDSTRING ( charausdruck4, charausdruck5, updpos ) |
             DELSTRING ( charausdruck6, startpos2 [, länge2 ] ) |
             SHIFTLEFTSTRING ( charausdruck7 ) |
             UPPERSTRING ( charausdruck8 ) |
             LOWERSTRING ( charausdruck9 ) |
             TRSTRING ( charausdruck10, charausdruck11, charausdruck12) |
             MSGSTRING ( numausdruck1 [ [, numausdruck2 ], name ] ) |
             CHARACTER ( { datumzeitausdruck [, charliteral1 ] |
                           wert2 |
                           numausdruck3 [, charliteral2 ] } ) }
```

---

wert1	Der Datentyp von *wert1* muß alphanumerisch sein (siehe Metavariable *wert*). Für *wert1* darf kein Aggregat angegeben werden.
satzelement	Name eines Satzelements vom Typ CHARACTER oder VARCHAR.
charausdruck	Hat *charausdruck* den Wert NULL, hat auch *charprim* den Wert NULL (siehe Metavariable *charausdruck*).
CONCAT	Zeichenketten werden miteinander verknüpft. Die maximale Länge einer Verknüpfung ist 32000 Byte. Die verwendeten Ausdrücke dürfen weder strukturierte Variablen noch Aggregate enthalten.

Hat eines der Argumente den NULL-Wert, dann liefert CONCAT den NULL-Wert.

charausdruck1, charausdruck2
: Die Zeichenkette *charausdruck2* wird direkt an die Zeichenkette *charausdruck1* angehängt.

  Hat einer der beiden Ausdrücke (*charausdruck1* oder *charausdruck2*) den Wert NULL, hat auch der gesamte Ausdruck den Wert NULL.

SUBSTRING
: Aus einer Zeichenkette wird eine Teilkette ausgewählt. Die Teilkette beginnt mit dem Zeichen, das an der Stelle *startpos1* steht, und endet mit dem Ende der gesamten Zeichenkette oder nach der angegebenen Länge. (*startpos1* + *länge1* - 1) darf nicht größer als die Gesamtlänge der Zeichenkette sein.

  Die verwendeten Ausdrücke dürfen weder strukturierte Variablen noch Aggregate enthalten.

  Hat eines der Argumente den NULL-Wert, dann liefert SUBSTRING den NULL-Wert.

charausdruck3
: siehe Metavariable *charausdruck*

startpos1
: Anfang der Teilkette. *startpos1* muß ein numerischer Ausdruck ohne Nachkommastellen sein.

  startpos1 muß > 0 sein.

länge1
: Länge der Teilkette. Wird *länge1* nicht angegeben, berechnet sie sich aus der (Gesamtlänge der Zeichenkette - *startpos1* + 1). *länge1* muß ein numerischer Ausdruck ohne Nachkommastellen sein.

  länge1 muß > 0 sein und es muß gelten:
  0 < *startpos1* ≤ Länge (*charausdruck3*)

UPDSTRING
: Ein Teil einer Zeichenkette wird durch eine andere Zeichenkette ersetzt. Die Länge von *charausdruck4* muß größer gleich der Länge von *charausdruck5* + *updpos* -1 sein.

  Die verwendeten Ausdrücke dürfen weder strukturierte Variablen noch Aggregate enthalten.

  Hat eines der Argumente den NULL-Wert, dann liefert UPDSTRING den NULL-Wert.

charausdruck4, charausdruck5
: siehe Metavariable *charausdruck*

updpos	Änderungsposition. *updpos* muß ein numerischer Ausdruck ohne Nachkommastellen sein. (0 < *updpos* ≤ Länge (*charausdruck4*) - Länge (*charausdruck5*) + 1.
DELSTRING	Eine Teilkette wird gelöscht. Die verwendeten Ausdrücke dürfen weder strukturierte Variable noch Aggregate enthalten. Hat eines der Argumente den NULL-Wert, dann liefert DELSTRING den NULL-Wert.
charausdruck6	siehe Metavariable *charausdruck*
startpos2	Anfang der Teilkette. *startpos2* muß ein numerischer Ausdruck ohne Nachkommastellen sein. Der Wert von *startpos2* muß größer als Null sein und darf den Wert der Länge von *ausdruck* nicht überschreiten.
länge2	*länge2* muß ein numerischer Ausdruck ohne Nachkommastellen sein. Die Zeichen ab *startpos2* bis zum Wert von *länge2* werden gelöscht DELSTRING (*charausdruck6*, *startpos2*, *länge2*) ist äquivalent zu SUBSTRING (*charausdruck3*, 1, *startpos1 - 1*) || SUBSTRING (*charausdruck3*, *startpos1* + *länge1*, *länge von charausdruck3* - *länge1* - *startpos1 + 1*). Ohne Angabe von *länge2* ist das Ergebnis eine alphanumerische Zeichenkette, die nur bis *startpos2* geht, d.h. es werden alle Zeichen ab *startpos2* gelöscht. DELSTRING (*charausdruck6*, *startpos2*) ist äquivalent zu SUBSTRING (*charausdruck3*, 1, *startpos1 -1*).
SHIFTLEFTSTRING	Alle linksbündigen Zeichen, deren hexadezimale Darstellung kleiner gleich dem Leerzeichen (≤ X'40') ist, werden gelöscht. Hat *charausdruck7* den NULL-Wert, dann liefert SHIFTLEFTSTRING den NULL-Wert.
charausdruck7	Ist *charausdruck7* vom Datentyp CHARACTER, dann wird von rechts mit Leerzeichen aufgefüllt. Ist *charausdruck7* vom Datentyp VARCHAR oder CHARACTER VARYING, dann wird die Länge entsprechend der Längenangabe dieser Datentypen verringert.

UPPERSTRING

Alle Kleinbuchstaben in *charausdruck8* werden durch Großbuchstaben ersetzt. Alle anderen Zeichen bleiben unverändert. Die Umsetzung erfolgt gemäß der länderspezifischen Einstellung in der aktuellen Systemumgebung.

Die verwendeten Ausdrücke dürfen weder strukturierte Variablen noch Aggregate enthalten.

Hat *charausdruck8* den NULL-Wert, dann liefert UPPERSTRING den NULL-Wert.

charausdruck8

siehe Metavariable *charausdruck*

LOWERSTRING

Alle Großbuchstaben in *charausdruck9* werden durch Kleinbuchstaben ersetzt. Alle anderen Zeichen bleiben unverändert. Die Umsetzung erfolgt gemäß der länderspezifischen Einstellung in der aktuellen Systemumgebung.

Die verwendeten Ausdrücke dürfen weder strukturierte Variablen noch Aggregate enthalten.

Hat *charausdruck9* den NULL-Wert, dann liefert LOWERSTRING den NULL-Wert.

charausdruck9

siehe Metavariable *charausdruck*

TRSTRING

Die Zeichen in *charausdruck10* werden folgendermaßen umgesetzt: Nur wenn ein Zeichen *c* aus *charausdruck10* auch in *charausdruck11* vorkommt, wird dieses Zeichen umgesetzt. Die Position *n* des Zeichens in *charausdruck11* bestimmt das Zeichen, durch das es ersetzt wird. Das ersetzende Zeichen steht an Position *n* in *charausdruck12*.

Ein Zeichen wird nicht umgesetzt, wenn seine Position in *charausdruck11* größer ist als die Länge von *charausdruck12*.

Die verwendeten Ausdrücke dürfen weder strukturierte Variablen noch Aggregate enthalten.

Hat eines der Argumente den NULL-Wert, dann liefert TRSTRING den NULL-Wert.

charausdruck10

Zeichen in *charausdruck10* werden zeichenweise umgesetzt. Zeichen werden umgesetzt, wenn sie in *charausdruck11* vorkommen und wenn sie an einer Position *n* in *charausdruck11* stehen, die kleiner ist als die Anzahl der Zeichen in *charausdruck12*.

Zeichen bleiben unverändert, wenn sie nicht in *charausdruck11* vorkommen oder wenn sie an einer Position *n* in *charausdruck11* stehen, die größer ist als die die Anzahl der Zeichen in *charausdruck12*.

charausdruck11

charausdruck11 definiert die Zeichen, die in *charausdruck10* ersetzt werden. Die Zeichenposition in *charausdruck11* legt das ersetzende Zeichen fest.

Zeichen in *charausdruck11* dürfen sich nicht wiederholen.

charausdruck11 mit den Zeichen, die ersetzt werden sollen, und *charausdruck12* mit den ersetzenden Zeichen sollen gleich lang sein. Ist *charausdruck11* länger als *charausdruck12*, werden Zeichen, deren Position in *charausdruck11* größer ist als die Länge von *charausdruck12*, nicht umgesetzt.

charausdruck11 darf höchstens 256 Zeichen lang sein.

charausdruck12

charausdruck12 definiert die ersetzenden Zeichen. Die Zeichenposition des zu ersetzenden Zeichens in *charausdruck11* bestimmt die Zeichenposition des ersetzenden Zeichens in *charausdruck12*.

charausdruck12 mit den ersetzenden Zeichen und *charausdruck11* mit den Zeichen, die ersetzt werden sollen, sollen gleich lang sein. Ist *charausdruck12* kürzer als *charausdruck11*, werden Zeichen, deren Position in *charausdruck11* größer ist als die Länge von *charausdruck12*, nicht umgesetzt.

charausdruck12 darf höchstens 256 Zeichen lang sein.

MSGSTRING

Auf eine Meldung in der Meldungsdatei (= aktuelle MIP-Datei) wird zugegriffen. Die Meldung wird durch die Parameter bestimmt.

Wenn drei Parameter (*numausdruck1, numausdruck2, name*) angegeben werden, wird *numausdruck2* ignoriert.
Wenn zwei Parameter angegeben werden, muß der zweite Parameter *name* sein (*numausdruck1, name*).

Wenn keine eindeutige Meldung gefunden werden kann, gibt DRIVE/WINDOWS zurück: `MELDUNG NICHT GEFUNDEN`.

Die verwendeten Ausdrücke dürfen weder strukturierte Variablen noch Aggregate enthalten.

Hat eines der Argumente den NULL-Wert, dann liefert MSGSTRING den NULL-Wert.

Bei DECLARE-Anweisungen (z.B. DECLARE VARIABLE, DECLARE CONSTANT etc.) erfolgt der Zugriff zum Übersetzungszeitpunkt, bei ausführbaren Anweisungen (z.B. SET) zum Ablaufzeitpunkt. Für vollständig sprachunabhängige Programme muß der Zugriff zum Ablaufzeitpunkt erfolgen.

numausdruck1

Meldungsnummer (= zweiter Teil des Meldungsschlüssels).

Führende Nullen in der Meldungsnummer brauchen nicht angegeben werden.

numausdruck1 muß ein numerischer Ausdruck ohne Nachkommastellen sein.

numausdruck2 *numausdruck2* wird ignoriert, falls ein Wert angegeben wird.

numausdruck2 wird nur wegen der Kompatibilität von DRIVE/WINDOWS (BS2000) und DRIVE/WINDOWS (SINIX) unterstützt.

name Meldungsklasse (= erster Teil des Meldungsschlüssels). *name* darf maximal 3 Zeichen lang sein.

Wird keine Angabe für *name* gemacht, setzt DRIVE/WINDOWS `DRI` ein.

CHARACTER Das Argument (*datumzeitausdruck*,*wert2*,*numausdruck3*) wird in ein Ergebnis vom Typ CHARACTER umgewandelt. Der Datentyp des Ausdrucks ist alphanumerisch, d.h. Ziffern und Trennzeichen werden als Zeichen dargestellt.

Für ein Datum ist die Länge der Zeichenkette zehn Zeichen (z.B. '1911-11-11'),
für eine Uhrzeit ohne Sekundenbruchteile acht Zeichen (z.B. '17:35:12'),
für eine Uhrzeit mit Sekundenbruchteilen zwölf Zeichen (z.B. '17:35:12.361') und
für einen Zeitstempel 23 Zeichen (z.B. '1911-11-11%17:35:12.361').

Hat eines der Argumente den NULL-Wert, dann liefert CHARACTER den NULL-Wert.

datumzeitausdruck siehe Metavariable *datumzeitausdruck*

charliteral1, charliteral2

charliteralx muß die Bedingungen von *mask* erfüllen. Wenn die erste Komponente des Ausdrucks ein *datumzeitausdruck* ist, dann muß *charliteralx* Steuerzeichen für Zeit-Datentypen enthalten. Entfällt die Angabe von *charliteralx*, dann wird eine Standardmaske verwendet (siehe auch Metavariable *literal*).

wert2 *wert* muß eine Datengruppe oder ein Aggregat sein. Die Datengruppe darf weder ein Teil einer Wiederholungsgruppe sein noch eine Wiederholungsgruppe enthalten. Der Datentyp aller Basisfelder muß alphanumerisch sein. Wird für *wert* eine Variable angegeben, ist *charprim* eine Verknüpfung aller Basisfelder (siehe oben, CONCAT).

numausdruck3 siehe Metavariable *numausdruck*

Beispiel 1

```
DECLARE VARIABLE &a CHAR (5)  INIT 'DONAU',
                 &b CHAR (11) INIT 'DAMPFSCHIFF';

CONCAT (&a,&b)   → 'DONAUDAMPFSCHIFF'

SUBSTRING (&b,6)   → 'SCHIFF'

UPDSTRING (&b,'LOK%%%',6)   → 'DAMPFLOK%%%'
```

Beispiel 2

```
DECLARE VARIABLE &a CHAR (250),
                 &b CHAR (250);
...
SET &a = SUBSTRING(CONCAT(&a,&b), 180, 200);
```

Beispiel 3

Der Wert von "MSGSTR (17)" wird zum Übersetzungszeitpunkt zugewiesen.

```
DECLARE CONSTANT &c MSGSTR (17);
SET &v = &c;
```

Beispiel 4

Der Wert von "MSGSTR (17)" wird zum Ablaufzeitpunkt zugewiesen.

```
...
SET &v = MSGSTR (17);
...
```

Beispiel 5

Der Datentyp des Ergebnisses einer Rechenoperation ist alphanumerisch.

```
DECLARE VARIABLE &e CHAR (8);
...
SET &e = CHARACTER (5967 / 17);
```

Beispiel 6

Die Zeichen aus der Variablen &text sollen umgesetzt werden. Die Variable mit den zu ersetzenden Zeichen heißt &zeichenalt, die Variable mit den ersetzenden Zeichen heißt &zeichenneu.

```
DECLARE VARIABLE &text CHAR (6)  INIT 'PEKING';
DECLARE VARIABLE &zeichenalt CHAR (5)  INIT 'JKPTU',
                 &zeichenneu CHAR (5)  INIT 'IJBDO';
...
SET &text = TRSTRING(&text,&zeichenalt,&zeichenneu);
```

Der ursprüngliche Inhalt der Variablen &text ("PEKING") wurde umgesetzt und ist nun "BEJING".

check
CHECK-Klausel definieren

Die CHECK-Klausel vereinbart eine Prüfbedingung. Ist zum Ausführungszeitpunkt des Programms bei einer Zuweisung (SET, CYCLE ... INTO, SELECT ... INTO, FETCH ... INTO, CALL ... RETURN, FILL ... RETURN etc.) die Prüfbedingung zu einer Variablen nicht erfüllt, erhält &ERROR einen Eintrag (siehe Anweisung WHENEVER).

Ist die Prüfbedingung für den Input einer DISPLAY-Anweisung nicht erfüllt, wird der Bildschirm wiederholt ausgegeben.

check auf eine Komponente einer strukturierten Variablen

Ist die Komponente Teil einer Wiederholungsgruppe, gilt die CHECK-Klausel für alle Ausprägungen der Komponente.
Einzelne Vektor- oder Matrixkomponenten dürfen in der CHECK-Klausel nicht angegeben werden.

```
check::=CHECK [ ( ] bedingung [ ) ] [ MESSAGE charausdruck ]
```

CHECK	Die CHECK-Klausel muß mit der INIT-Klausel oder der Standardinitialisierung verträglich sein oder bei *basistyp* muß die Angabe NOCHECK stehen.
( )	Die runden Klammern müssen in SQL-Anweisungen für SESAM V2.x angegeben werden
bedingung	Vereinbarte Prüfbedingung. *bedingung* darf nur die Variable enthalten, zu der die CHECK-Klausel gehört. Die Variable darf keine Indexangaben enthalten. Sie braucht nicht qualifiziert zu sein, auch wenn sie nicht eindeutig ist. Für diese Variable ist kein strukturierter Vergleich möglich. Die *bedingung* darf anstelle der Variablen auch das Schlüsselwort VALUE enthalten. Zur Verwendung innerhalb einer DECLARE TYPE-Anweisung siehe Anweisung DECLARE TYPE.

MESSAGE

Mit der MESSAGE-Klausel wird bei Formulareingabe eine Meldung vereinbart, die anstelle der Standardmeldung ausgegeben wird, wenn *bedingung* bei der dynamischen Formatierung oder der Anweisung MOVE DATA nicht erfüllt ist.

Bei SET-Anweisungen und Parameterübergaben wird eine Standardmeldung ausgegeben. Eine MESSAGE-Angabe wird nicht berücksichtigt.

In SQL-Anweisungen für SESAM V2.x darf die MESSAGE-Klausel nicht verwendet werden.

charausdruck

Mit *charausdruck* wird die auszugebende Meldung festgelegt (maximal 79 Zeichen).

Die Meldung wird nur beim Verstoß gegen die *bedingung* ausgegeben.
Im *charausdruck* von MESSAGE darf für *wert* nur das Schlüsselwort VALUE oder ein Literal stehen.

datendef
Datentyp definieren

datendef legt den Datentyp für Variablen und benutzereigene Datentypen fest.

```
datendef::={ grunddatentyp [ basistyp ] |
             strukturtyp |
             ( zeilen, spalten ) grunddatentyp [ basistyp ] }
```

---

grunddatentyp	siehe Metavariable *grunddatentyp*
basistyp	siehe Metavariable *basistyp*
strukturtyp	siehe Metavariable *strukturtyp*
zeilen	Mit *zeilen* wird die Anzahl der Komponenten einer Variablen mit zweifacher Ausprägung (Matrix) in der Vertikalen festgelegt (0 < *zeilen* < 256).
spalten	Mit *spalten* wird die Anzahl der Komponenten einer Variablen mit zweifacher Ausprägung (Matrix) in der Horizontalen festgelegt (0 < *spalten* < 256).

datengruppe
Datengruppe definieren

datengruppe legt für eine Variable den Datentyp "Datengruppe" fest. *datengruppe* besteht aus Komponenten, deren Datentypen beliebig sein können. Die Struktur von *datengruppe* wird bestimmt durch die Reihenfolge der Komponenten. Die Komponenten selbst sind dann Datengruppen, wenn sie ihrerseits wieder aus Komponenten bestehen.

Bei der Definition können mit *level* Stufennummern vergeben werden, die durch ihren Wert die Strukturstufe bestimmen. Man spricht genau dann von einer Datengruppe, wenn eine Definition folgt, deren Stufennummer größer ist. Umgekehrt spricht man genau dann von der Komponente einer Datengruppe, wenn eine Definition vorangeht, deren Stufennummer kleiner ist. Die Festlegung mit der höchsten Stufennummer ist eine einfache Komponente.

Die Schachtelungstiefe von Datengruppen ist 49, wobei maximal drei Wiederholungsgruppen ineinander geschachtelt sein dürfen.

In SQL-Anweisungen darf *datengruppe* nur bei UDS-Datenbanken angegeben werden.

Weitere Strukturtypen siehe Metavariable *strukturtyp*.

```
datengruppe::=[ REDEFINES { variable | komponente1 [ suffix ] } ]
              {, level { komponente2 | FILLER }
                       { grunddatentyp [ basistyp ] | strukturtyp } }
```

REDEFINES

Mit REDEFINES wird die Variable gekennzeichnet, die redefiniert wird (Basisvariable).

Die redefinierende Variable darf nicht länger als die Basisvariable sein. Ist die redefinierende Variable Komponente einer Struktur, muß die Basisvariable eine Komponente derselben Hauptstruktur (Struktur mit der Stufennummer 1) sein.

variable

Name der Basisvariablen, die redefiniert wird. Sie muß bereits definiert worden sein und darf selbst keine mit LIKE oder REDEFINES definierte Variable sein. Ist die Basisvariable eine Struktur, darf die redefinierende Variable keine Komponente dieser Struktur sein.

Zwischen dem Beginn der kleinsten Teilstruktur, die sowohl die Basisvariable als auch die redefinierende Variable enthält, und der Basisvariablen oder der redefinierenden Variablen dürfen keine Wiederholungsgruppen liegen.

komponente1	Alphanumerische Zeichen für den Namen einer Komponente. 0 < Anzahl(*komponente1*) < 32
suffix	siehe Metavariable *variable*
level	Mit *level* werden die Stufennummern festgelegt. Die erste Stufennummer muß immer "1" sein.
komponente2	Alphanumerische Zeichen für den Namen einer Komponente. 0 < Anzahl(*komponente2*) < 32
FILLER	FILLER kann anstelle von *zeichen* für Stufen > 1 angegeben werden. Ein mit FILLER benanntes Feld kann einzeln nicht angesprochen werden, sondern nur innerhalb von Aggregat (siehe Metavariable *wert*) verwendet werden.
grunddatentyp	siehe Metavariable *grunddatentyp*
basistyp	siehe Metavariable *basistyp*
strukturtyp	siehe Metavariable *strukturtyp*

Beispiel

Die Variable "a" ist eine Datengruppe. Komponenten sind "b", "b1" bis "b31" und "c". Einfache Komponenten sind "b1", "b2", "b31" und "c".

```
DECLARE 1 &a,
          2 b,
            3 b1 INTEGER,
            3 b2 NUM (7,2),
            3 b3,
              4 b31 CHAR (10),
          2 c CHAR (8);
```

datumzeitausdruck
Datum oder Zeitpunkt berechnen

datumzeitausdruck legt ein gültiges Datum oder einen gültigen Zeitpunkt fest. Der Datentyp von *datumzeitausdruck* ist ein Zeit-Datentyp (DATE, TIME, TIME(3) oder TIMESTAMP(3)).

In SQL-Anweisungen für SESAM V1.x und UDS darf *datumzeitausdruck* nicht verwendet werden. In SQL-Anweisungen für SESAM V2.x darf *datumzeitausdruck* verwendet werden, wenn *datumzeitausdruck* nur CURRENT DATE / TIME / TIMESTAMP enthält.

Weitere Ausdrücke siehe Metavariable *ausdruck*.

```
datumzeitausdruck::={ datumzeitterm1 |
                      datumzeitausdruck1 { + | - } intervallterm |
                      datumzeitausdruck2 || datumzeitterm2 }
```

---

datumzeitterm1	siehe Metavariable *datumzeitterm*
datumzeitausdruck1	Der Datentyp von *datumzeitausdruck1* muß ein Zeit-Datentyp sein. *datumzeitausdruck1* darf weder strukturierte Variablen noch Aggregate enthalten.
+	Summenoperator
–	Differenzoperator
intervallterm	siehe Metavariable *intervallterm*. *intervallterm* muß die Intervalleinheit YEARS, MONTHS oder DAYS haben, wenn für *datumzeitausdruck* ein Datum (Datentyp: DATE) angegeben wird. *intervallterm* muß die Intervalleinheit HOURS, MINUTES, SECONDS oder FRACTIONS haben, wenn für *datumzeitausdruck* eine Uhrzeit (Datentyp: TIME oder TIME(3)) angegeben wird.
datumzeitausdruck2	*datumzeitausdruck2* muß den Datentyp DATE, TIME oder TIME(3) haben. Wenn *datumzeitausdruck2* den Datentyp DATE hat, muß *datumzeitterm2* den Datentyp TIME oder TIME(3) haben. Wenn *datumzeitausdruck2* den Datentyp TIME oder TIME(3) hat, muß *datumzeitterm2* den Datentyp DATE haben.

Wenn *datumzeitausdruck2* den Typ TIME hat, werden die Sekundenbruchteile mit Nullen aufgefüllt (HH:MI:SS.000).

||

Mit dem Verknüpfungsoperator || werden die Zeichenketten *datumzeitausdruck2* und *datumzeitterm2* miteinander verknüpft. Dazu wird die zweite Zeichenkette direkt an die erste angehängt. Das Ergebnis hat den Datentyp TIMESTAMP(3).

datumzeitterm2

datumzeitterm2 muß den Datentyp DATE, TIME oder TIME(3) haben.

Wenn *datumzeitterm2* den Datentyp TIME oder TIME(3) hat, muß *datumzeitausdruck2* den Datentyp DATE haben.

Wenn *datumzeitterm2* den Datentyp DATE hat, muß *datumzeitausdruck2* den Datentyp TIME oder TIME(3) haben.

Wenn *datumzeitterm2* den Typ TIME hat, werden die Sekundenbruchteile mit Nullen aufgefüllt (HH:MI:SS.000).

Beispiele

Der Variablen &plus30 soll das Datum zugewiesen werden, das man erhält, wenn dem aktuellen Datum 30 Tage hinzugezählt werden.

```
DECLARE VARIABLE &datum DATE;
DECLARE VARIABLE &plus30 DATE;
...
SET &datum=CURRENT DATE;
SET &plus30=&datum + 30 DAYS;
```

Der Variablen &plus soll das Datum zugewiesen werden, das man erhält, wenn dem aktuellen Datum soviel Tage hinzugezählt werden wie seit der Mondlandung (20.07.1969) vergangen sind.

```
DECLARE VARIABLE &datum DATE;
DECLARE VARIABLE &landung DATE INIT DATE(1969-07-20);
DECLARE VARIABLE &plus DATE;
...
SET &datum=CURRENT DATE;
SET &plus=&datum + (&datum - &landung) DAYS;
```

datumzeiteinheit
Einheit für Zeitspanne definieren

datumzeiteinheit legt die Einheit für Zeitspannen (Intervalle) wie z.B. Jahre, Tage oder Minuten fest.

```
datumzeiteinheit::={ datumzeitfeld1 TO datumzeitfeld2 |
                     UNITS datumzeitfeld3 |
                     YEARS |
                     MONTHS |
                     DAYS |
                     HOURS |
                     MINUTES |
                     SECONDS |
                     FRACTIONS }
```

---

datumzeitfeld1, datumzeitfeld2

datumzeitfeld1 muß immer identisch sein mit *datumzeitfeld2*.

Es dürfen nur Intervalleinheiten mit einer Komponente angegeben werden, z.B. MONTH TO MONTH.

UNITS	Vereinbarung der Einheiten für Datum und Zeit. Es kann abgekürzt werden mit z.B. YEARS für UNITS YEAR (siehe auch Metavariable *datumzeitfeld*).
datumzeitfeld3	siehe Metavariable *datumzeitfeld*
YEARS	Die Einheit "Jahre" wird vereinbart.
MONTHS	Die Einheit "Monate" wird vereinbart.
DAYS	Die Einheit "Tage" wird vereinbart.
HOURS	Die Einheit "Stunden" wird vereinbart.
MINUTES	Die Einheit "Minuten" wird vereinbart.
SECONDS	Die Einheit "Sekunden" wird vereinbart.
FRACTIONS	Die Einheit "Sekundenbruchteile" wird vereinbart.

Beispiel

Der Variablen &dauer wird die Dauer bis zur Jahrtausendwende in Tagen zugewiesen.

```
DECLARE VARIABLE &datum DATE;
DECLARE VARIABLE &wende DATE INIT DATE(2000-01-01);
DECLARE VARIABLE &dauer INTERVAL DAYS;
...
SET &datum=CURRENT DATE;
SET &dauer=&wende - &datum;
```

datumzeitfeld
Komponente eines Datums oder Zeitpunkts definieren

datumzeitfeld legt die Komponenten eines Datums (Tag, Monat, Jahr) oder eines Zeitpunkts (Stunde, Minute, Sekunde, Sekundenbruchteil) fest.

```
datumzeitfeld::={ YEAR | MONTH | DAY | HOUR | MINUTE | SECOND | FRACTION }
```

YEAR	Die Komponente "Jahr" wird vereinbart.
MONTH	Die Komponente "Monat" wird vereinbart.
DAY	Die Komponente "Tag" wird vereinbart.
HOUR	Die Komponente "Stunde" wird vereinbart.
MINUTE	Die Komponente "Minute" wird vereinbart.
SECOND	Die Komponente "Sekunde" wird vereinbart.
FRACTION	Die Komponente "Sekundenbruchteil" wird vereinbart.

datumzeitterm
Datum oder Zeitpunkt definieren

datumzeitterm legt einen Zeitpunkt (Datum, Uhrzeit oder Zeitstempel) fest oder wandelt einen Charakterausdruck in ein Ergebnis vom Typ DATE, TIME(3) oder TIMESTAMP(3) um.

Der Datentyp von *datumzeitterm* ist ein Zeit-Datentyp (siehe Metavariable *grunddatentyp*).

```
datumzeitterm::=
        { wert |
        ( datumzeitausdruck ) |
        { DATE | TIME | TIMESTAMP } { ( charausdruck ) | ( datumzeitausdruck1 ) } |
        CURRENT { DATE | TIME | TIMESTAMP } |
        CONCAT { datumzeitausdruck2, datumzeitausdruck3 } }
```

---

wert	Der Datentyp von *wert* muß ein Zeit-Datentyp sein. Hat *wert* den Wert NULL, hat *datumzeitterm* auch den Wert NULL. *wert* darf keine strukturierte Variable und kein Aggregat sein.
datumzeitausdruck	Der Datentyp von *datumzeitausdruck* muß ein Zeit-Datentyp sein. Hat *datumzeitausdruck* den Wert NULL, hat *datumzeitterm* auch den Wert Null.
DATE	*charausdruck* oder *datumzeitausdruck1* wird in einen Wert vom Typ DATE umgewandelt.
TIME	*charausdruck* oder *datumzeitausdruck1* wird in einen Wert vom Typ TIME(3) umgewandelt.
TIMESTAMP	*charausdruck* oder *datumzeitausdruck1* wird in einen Wert vom Typ TIMESTAMP(3) umgewandelt.
charausdruck	Das Ergebnis von *charausdruck* muß die abdruckbare Form eines gültigen Zeitwertes sein (siehe *datumzeitliteral* bei Metavariable *literal*). Hat *charausdruck* den Wert NULL, hat *datumzeitterm* auch den Wert Null. *charausdruck* darf keine strukturierte Variable oder ein Aggregat sein und muß die entsprechenden Trennzeichen (siehe *datumzeitliteral* bei Metavariable *literal*) enthalten.

datumzeitausdruck1 Der Datentyp von *datumzeitausdruck1* muß ein Zeit-Datentyp sein.

– Bei DATE muß *datumzeitausdruck1* den Datentyp DATE oder TIMESTAMP(3) haben.

  Wenn *datumzeitausdruck1* vom Typ TIMESTAMP(3) ist, wird bei der Konvertierung *stunde*:*minute*:*sekunde.bruchteil* (HH:MI:SS.FFF) abgeschnitten.

– Bei TIME muß *datumzeitausdruck1* den Datentyp TIME, TIME(3) oder TIMESTAMP(3) haben.

  Wenn *datumzeitausdruck1* vom Typ TIME ist, werden bei der Konvertierung die Sekundenbruchteile mit Nullen aufgefüllt (HH:MI:SS.000).

  Wenn *datumzeitausdruck1* vom Typ TIMESTAMP(3) ist, wird bei der Konvertierung *jahr-monat-tag* (YYYY-MO-DD) abgeschnitten.

– Bei TIMESTAMP(3) kann *datumzeitausdruck1* den Datentyp DATE, TIME, TIME(3) oder TIMESTAMP(3) haben.

  Wenn *datumzeitausdruck1* vom Typ DATE ist, wird bei der Konvertierung die Uhrzeit *stunde*:*minute*:*sekunde.bruchteil* mit Nullen aufgefüllt (00:00:00.000).

  Wenn *datumzeitausdruck1* vom Typ TIME ist, wird bei der Konvertierung das aktuelle Datum *jahr-monat-tag* (YYYY-MO-DD) ergänzt und die Sekundenbruchteile mit Nullen aufgefüllt (HH:MI:SS.000).

  Wenn *datumzeitausdruck1* vom Typ TIME(3) ist, wird bei der Konvertierung das aktuelle Datum *jahr-monat-tag* (YYYY-MO-DD) ergänzt.

Hat *datumzeitausdruck1* den Wert NULL, hat *datumzeitterm* auch den Wert Null.

CURRENT DATE Ergibt das aktuelle Datum in der Form *jahr-monat-tag* (Zur Form siehe *datumzeitliteral* bei Metavariable *literal*).

CURRENT TIME Ergibt die aktuelle Uhrzeit in der Form *stunde:minute:sekunde.bruchteil* (Zur Form siehe *datumzeitliteral* bei Metavariable *literal*). Wird CURRENT TIME für ein Datenfeld mit dem Datentyp TIME angegeben, schneidet DRIVE/WINDOWS die Sekundenbruchteile ab.

CURRENT TIMESTAMP

Ergibt den aktuellen Zeitstempel in der Form *jahr-monat-tag% stunde:minute:sekunde.bruchteil* (Zur Form siehe *datumzeitliteral* bei Metavariable *literal*).

CONCAT

Die Zeichenketten *datumzeitausdruck1* und *datumzeitausdruck2* werden miteinander verknüpft (siehe Metavariable *datumzeitausdruck*). Das Ergebnis hat den Datentyp TIMESTAMP(3).

datumzeitausdruck2

datumzeitausdruck2 muß den Datentyp DATE, TIME oder TIME(3) haben.

Wenn *datumzeitausdruck2* den Datentyp DATE hat, muß *datumzeitausdruck3* den Datentyp TIME oder TIME(3) haben.

Wenn *datumzeitausdruck2* den Datentyp TIME oder TIME(3) hat, muß *datumzeitausdruck3* den Datentyp DATE haben.

Wenn *datumzeitausdruck2* den Typ TIME hat, werden die Sekundenbruchteile mit Nullen aufgefüllt (HH:MI:SS.000).

datumzeitausdruck3

datumzeitausdruck3 muß den Datentyp DATE, TIME oder TIME(3) haben.

Wenn *datumzeitausdruck3* den Datentyp TIME oder TIME(3) hat, muß *datumzeitausdruck2* den Datentyp DATE haben.

Wenn *datumzeitausdruck3* den Datentyp DATE hat, muß *datumzeitausdruck2* den Datentyp TIME oder TIME(3) haben.

Wenn *datumzeitausdruck3* den Typ TIME hat, werden die Sekundenbruchteile mit Nullen aufgefüllt (HH:MI:SS.000).

Beispiele

Die Variable &datum enthält das aktuelle Datum, die Variable &zeit die aktuelle Uhrzeit.

```
DECLARE VARIABLE &datum DATE;
DECLARE VARIABLE &zeit TIME;
...
SET &datum=CURRENT DATE;
SET &zeit=CURRENT TIME;
```

Der Variablen &jahrhundertende wird die Zeichenkette "1999-12-31" zugewiesen.

```
DECLARE VARIABLE &jahrhundertende DATE;
...
SET &jahrhundertende=DATE(1999-12-31)        oder
SET &jahrhundertende=DATE('1999-12-31')
```

format
Format für Bildschirm oder Liste definieren

format legt das Format für Bildschirme oder Listen fest.

```
format::={ FREE |
          { TABLE | LINE | SEQUENCE } [ NAMES [ VALUES ] | VALUES ] } }
```

FREE — Vorbelegung

FREE oder keine Formatangabe bewirkt eine formatfreie Ausgabe: Die Dateninhalte der auszugebenden Elemente werden lückenlos hintereinander ausgegeben, ohne Rücksicht auf das Zeilenende. Die Namen der Dateninhalte werden nicht ausgegeben.

Beispiel

```
HUBER          MAX            AUENSTR.       80469MUENC
HEN
```

– Die Ausgabe erfolgt sequentiell ohne Leerzeichen zwischen den einzelnen Feldern. Sie kann aber beliebig mit Hilfe von NEWLINE, NEWPAGE, TABULATOR und BLANK gesteuert werden.
– Zeilenüberlauf bewirkt automatisch einen Zeilenumbruch.

TABLE — Die auszugebenden Elemente werden in Tabellenform ausgegeben.

Die Breite der Tabellenspalten wird bestimmt durch den längsten Elementnamen oder den längsten Datenwert. Weder der Elementname noch der Dateninhalt werden abgeschnitten. Die Tabellenspalten sind durch ein Leerzeichen voneinander getrennt.

TABLE ist nur erlaubt, wenn das Ausgabeformat vollständig in eine Zeile paßt. Sonst erfolgt Prozedurabbruch mit der Fehlermeldung `DRI0049 MAXIMALE ZEILENLAENGE ((&00)) UEBERSCHRITTEN`.

Im Programm-Modus bedeutet TABLE, daß immer eine Überschrift und eine Datenzeile gedruckt wird. Dies gilt nicht für Matrizen.

Im Dialog-Modus kommen in die nach der Titel-Zeile folgenden Zeilen nur die Daten-Werte der Elemente.

Wird für *format* TABLE angegeben, sind nachfolgend die Angaben NEWLINE und NEWPAGE nicht erlaubt.

NAMES VALUES NAMES VALUES ist beim Operanden TABLE vorbelegt.

In der ersten Zeile werden die Namen ausgegeben. In den folgenden Zeilen stehen die Dateninhalte der auszugebenden Elemente.

Beispiel

```
NAME        VORNAME     STRASSE
HUBER       MAX         AUENSTR.
```

– Es werden nur die Namen der Basiselemente ausgegeben. Matrizen werden im Matrix-Format ausgegeben, der Name erscheint nur einmal.
– Literale werden in jeder Zeile wiederholt.

NAMES Die Angabe TABLE NAMES ergibt ein tabellarisches Format mit Ausgabe der Namen, ohne Dateninhalte.

VALUES Die Angabe TABLE VALUES ergibt ein tabellarisches Format mit Ausgabe der Dateninhalte, ohne Namen.

LINE Die auszugebenden Elemente werden zeilenweise ausgegeben.

Wird für *format* LINE angegeben, erfolgt bei den Angaben TABULATOR und BLANK ein Zeilenvorschub mit Leerzeile.

NAMES VALUES NAMES VALUES ist beim Operanden LINE vorbelegt.

In einer Zeile stehen jeweils Stufennummer und Name und dahinter der zugehörige Dateninhalt.

Beispiel

```
1 NAME: HUBER
1 VORNAME: MAX
1 STRASSE: AUENSTR.
```

– Wiederholungsgruppen werden untereinander ausgegeben.
– Zeilenüberlauf (abhängig vom COLUMNS-Wert) bewirkt automatischen Umbruch.
– Elemente von Vektoren werden nebeneinander ausgegeben.
– Matrizen werden in Matrix-Format ausgegeben, der Name erscheint an jedem Matrix-Zeilenanfang.
– Zwischen je zwei Feldern wird ein Leerzeichen ausgegeben.

NAMES Ohne Angabe von VALUES wird ein vertikales Format ausgegeben, das nur die Namen, nicht die Dateninhalte ausgibt.

VALUES — LINE VALUES ergibt ein vertikales Format mit Ausgabe der Dateninhalte, ohne Namen.

SEQUENCE — Die auszugebenden Elemente werden hintereinander ausgegeben.

NAMES VALUES — NAMES VALUES ist beim Operanden SEQUENCE vorbelegt.

Die auszugebenden Elemente bestehen jeweils aus einem Namen, sowie dem zugehörigen Dateninhalt. Die auszugebenden Elemente werden jeweils durch ein Leerzeichen voneinander getrennt.

Beispiel

```
NAME: HUBER          %VORNAME: MAX          %STRASSE: AU
ENSTR.       %
```

- Wiederholungsgruppen werden nebeneinander ausgegeben.
- Zeilenüberlauf (abhängig vom COLUMNS-Wert) bewirkt automatischen Umbruch.
- Elemente von Vektoren werden nebeneinander ausgegeben.
- Elemente von Matrizen werden nebeneinander ausgegeben, der Name erscheint an jedem Matrix-Zeilenanfang.
- Es werden nur die Namen der Basiselemente ausgegeben.
- Zwischen je zwei Feldern wird ein Leerzeichen ausgegeben.

NAMES — Ohne Angabe von VALUES wird ein horizontales Format ausgegeben, das nur die Namen, nicht die Dateninhalte anzeigt.

VALUES — SEQUENCE VALUES ergibt ein horizontales Format mit Ausgabe der Dateninhalte, ohne Namen.

grunddatentyp
Datentypen

DRIVE/WINDOWS unterscheidet vier Arten von Grunddatentypen:

– alphanumerische Datentypen (CHARACTER, CHARACTER VARYING, VARCHAR)
– numerische Datentypen (DECIMAL, EXTENDED DECIMAL, XDEC, NUMERIC, INTEGER, SMALLINT, REAL, FLOAT, DOUBLE PRECISION)
– Zeit-Datentypen (DATE, TIME, TIME(3), TIMESTAMP(3))
– Datentyp INTERVAL
– benutzerdefinierter Datentyp

Diese Grunddatentypen werden auch als "atomare Typen" bezeichnet.

Weitere Datendefinitionen siehe Metavariable *datendef*.

```
grunddatentyp::={ CHARACTER [ ( länge ) ] |

                  { DECIMAL | NUMERIC | EXTENDED DECIMAL | XDEC }
                                  [ ( stellenanzahl [, nachkommastellen] ) ] |

                  INTEGER |
                  SMALLINT |
                  DATE |
                  TIME [ (3) ] |
                  TIMESTAMP(3) |
                  INTERVAL datumzeiteinheit |
                  CHARACTER VARYING ( länge ) |
                  VARCHAR ( länge ) |
                  REAL |
                  DOUBLE PRECISION |
                  FLOAT |
                  usertyp }
```

CHARACTER	Für eine Zeichenfolge wird der Datentyp alphanumerisch festgelegt.
länge	Für den Datentyp CHARACTER wird die Länge in Byte festgelegt (0 < *länge* ≤ 32000). Vorbelegung: 1

DECIMAL
Für eine Zeichenfolge wird der Datentyp numerisch gepackt festgelegt.

NUMERIC
Für eine Zeichenfolge wird der Datentyp numerisch ungepackt festgelegt.

EXTENDED DECIMAL, XDEC
Für eine Zeichenfolge wird der Datentyp erweitert numerisch festgelegt.

stellenanzahl
Für die Datentypen DECIMAL, NUMERIC, EXTENDED DECIMAL und XDEC werden die Anzahl der Stellen (= Genauigkeit) festgelegt.

Für DECIMAL und NUMERIC gilt: 0 < *stellenzahl* ≤ 15)
Vorbelegung für DECIMAL: 15
Vorbelegung für NUMERIC: 8

Für EXTENDED DECIMAL und XDEC gilt: 0 < *stellenzahl* ≤ 32
Vorbelegung für EXTENDED DECIMAL und XDEC: 32

nachkommastellen
Für die Datentypen DECIMAL, NUMERIC, EXTENDED DECIMAL und XDEC wird die Anzahl der Nachkommastellen festgelegt.

Für DECIMAL und NUMERIC gilt:
0 ≤ *nachkommastellen* ≤ *stellenanzahl*.
Ist *stellenzahl* = 15, muß *nachkommastellen* < *stellenzahl* sein.

Für EXTENDED DECIMAL und XDEC gilt:
0 ≤ *nachkommastellen* ≤ *stellenanzahl*.
Ist *stellenzahl* = 32, muß *nachkommastellen* < *stellenzahl* sein.

Vorbelegung: 0

INTEGER, SMALLINT
Für eine Zeichenfolge wird der Datentyp ganzzahlig festgelegt.

Der Wertebereich von INTEGER liegt zwischen -2^{31} und $2^{31} - 1$, d.h. zwischen -2147483648 und 2147483647.
Der Wertebereich von SMALLINT liegt zwischen -2^{15} und $2^{15} - 1$, d.h. zwischen -32768 und 32767.

DATE
Für eine Zeichenfolge wird der Datentyp "DATE" festgelegt. Die Zeichenfolge darf nur gültige Datumsangaben in der Form *jahr-monat-tag* enthalten (0001-01-01 bis 9999-12-31, siehe *datumzeitliteral* bei Metavariable *literal*).

Negative Jahreszahlen sind nicht gültig.

TIME — Für eine Zeichenfolge wird der Datentyp "TIME" festgelegt. Die Zeichenfolge darf nur gültige Uhrzeitangaben in der Form *stunde*:*minute*:*sekunde* enthalten (00:00:00 bis 23:59:59, siehe *datumzeitliteral* bei Metavariable *literal*).

TIME(3) — Für eine Zeichenfolge wird der Datentyp "TIME(3)" festgelegt. Die Zeichenfolge darf nur gültige Uhrzeitangaben in der Form *stunde*:*minute*:*sekunde.bruchteil* enthalten (00:00:00.000 bis 23:59:59.999, siehe *datumzeitliteral* bei Metavariable *literal*).

TIMESTAMP(3) — Für eine Zeichenfolge wird der Datentyp "TIMESTAMP(3)" festgelegt. Die Zeichenfolge darf nur gültige Zeitstempelangaben in der Form *jahr-monat-tag stunde*:*minute*:*sekunde.bruchteil* enthalten (0001-01-01 00:00:00.000 bis 9999-12-31 23:59:59.999, siehe *datumzeitliteral* bei Metavariable *literal*).

Negative Jahreszahlen sind nicht gültig.

INTERVAL — Für eine Datenfolge (max. 32 Zeichen) wird der Datentyp "INTERVAL" festgelegt.

Eine Darstellung der Ein- und Ausgabefelder mit *mask* ist nicht erlaubt.

datumzeiteinheit — Die Intervalleinheit wird festgelegt (siehe Metavariable *datumzeiteinheit*).

Es darf nur eine Komponente eines Datums oder Zeitpunkts angegeben werden (*datumzeitfeld2* muß identisch mit *datumzeitfeld1* sein, siehe Metavariable *datumzeiteinheit*).

CHARACTER VARYING, VARCHAR — Für eine Zeichenfolge wird der Datentyp alphanumerisch mit einer variablen Anzahl von Zeichen festgelegt.

länge — Für den Datentyp wird die Länge festgelegt ($0 < länge \leq 32000$).

REAL — Für eine Zeichenfolge wird der numerische Datentyp Gleitpunktzahl mit einer Genauigkeit von ca. sieben Dezimalstellen festgelegt. Der Wertebereich von REAL ist Hardware-abhängig.

Wenn die Rechengenauigkeit einer Operation im Vordergrund steht, wählen Sie nicht den Datentyp REAL. Beim Zuweisen und Ausgeben eines Datenwerts kann es zu Ungenauigkeiten kommen.

DOUBLE PRECISION, FLOAT — Für eine Zeichenfolge wird der numerische Datentyp Gleitpunktzahl mit einer Genauigkeit von ca. 15 Dezimalstellen festgelegt. Der Wertebereich von DOUBLE PRECISION ist Hardware-abhängig.

usertyp	Benutzerdefinierter Datentyp, der mit DECLARE TYPE im DRIVE-Programm definiert sein muß. Durch *usertyp* wird nicht automatisch eine Struktur erzeugt. In SQL-Anweisungen ist *usertyp* nicht erlaubt.

intervallausdruck
Zeitspanne berechnen

intervallausdruck legt eine gültige Zeitspanne (Intervall) fest.

Der Datentyp von *intervallausdruck* ist INTERVAL (siehe Metavariable *grunddatentyp*).

In SQL-Anweisungen für SESAM und UDS darf *intervallausdruck* nicht angegeben werden.

```
intervallausdruck::={ intervallterm |
                    ( datumzeitausdruck - datumzeitterm ) |
                    intervallausdruck { + | - } intervallterm }
```

---

intervallterm siehe Metavariable *intervallterm*

datumzeitausdruck – datumzeitterm
Zeitspanne als Differenz von Zeitpunkten

datumzeitausdruck und *datumzeitterm* müssen beide entweder Datums- oder Zeitangaben enthalten.

datumzeitausdruck und *datumzeitterm* dürfen weder strukturierte Variablen noch Aggregate enthalten.

Das Ergebnis hat folgenden Datentyp:

Datentyp	**bei Differenz von:**
INTERVAL DAYS	DATE - DATE DATE - TIMESTAMP(3) TIMESTAMP(3) - DATE
INTERVAL SECONDS	TIME - TIME
INTERVAL FRACTIONS	TIME - TIME(3) TIME - TIMESTAMP(3) TIME(3) - TIME TIME(3) - TIME(3) TIME(3) - TIMESTAMP(3) TIMESTAMP(3) - TIME TIMESTAMP(3) - TIME(3) TIMESTAMP(3) - TIMESTAMP(3)

intervallausdruck	Wenn *intervallausdruck* den Datentyp INTERVAL mit der Einheit YEARS oder MONTHS hat, muß *intervallterm* den Datentyp INTERVAL mit der Einheit YEARS oder MONTHS haben. Das Ergebnis hat stets die Einheit MONTHS. Wenn *intervallausdruck* den Datentyp INTERVAL mit der Einheit DAYS, HOURS, MINUTES, SECONDS oder FRACTIONS hat, muß *intervallterm* den Datentyp INTERVAL mit der Einheit DAYS, HOURS, MINUTES, SECONDS oder FRACTIONS haben. Das Ergebnis hat die Einheit SECONDS, falls weder *intervallausdruck* oder *intervallterm* die Einheit FRACTIONS hat. In allen anderen Fällen hat das Ergebnis die Einheit FRACTIONS.
intervallterm	Wenn *intervallterm* den Datentyp INTERVAL mit der Einheit YEARS oder MONTHS hat, muß *intervallausdruck* den Datentyp INTERVAL mit der Einheit YEARS oder MONTHS haben. Wenn *intervallterm* den Datentyp INTERVAL mit der Einheit DAYS, HOURS, MINUTES, SECONDS oder FRACTIONS hat, muß *intervallausdruck* den Datentyp INTERVAL mit der Einheit DAYS, HOURS, MINUTES, SECONDS oder FRACTIONS haben. Das Ergebnis hat den Datentyp INTERVAL mit der bei *intervallausdruck* beschriebenen Einheit.
–	Differenzoperator
+	Summenoperator

Bei der Berechnung von Zeitspannen als Differenz von Zeitpunkten rechnet DRIVE/WINDOWS nicht über Tagesgrenzen hinweg, sondern gibt negative Einheiten aus (siehe Beispiel).

Beispiele

Der Variablen &zeitspanne werden die Tage bis zur Jahrtausendwende zugewiesen.

```
DECLARE VARIABLE &datum DATE;
DECLARE VARIABLE &wende DATE INIT DATE(2000-01-01);
DECLARE VARIABLE &zeitspanne INTERVAL DAYS;
...
SET &datum=CURRENT DATE;
SET &zeitspanne=&wende - &datum;
```

Der Variablen &dauer wird die Anzahl der Sekunden zugewiesen, die am heutigen Tag schon vergangen sind.

```
DECLARE VARIABLE &zeit TIME;
DECLARE VARIABLE &anfang TIME INIT TIME(00:00:00);
DECLARE VARIABLE &dauer INTERVAL SECONDS;
...
SET &zeit=CURRENT TIME;
SET &dauer=&zeit - &anfang;
```

Der Variablen &dauer wird -3 (Stunden) zugewiesen. Es wird nicht über Tagesgrenzen hinweg (09:00:00 - 12:00:00 =21:00:00) gerechnet.

```
DECLARE VARIABLE &dauer INTERVAL HOURS;
...
SET &dauer=TIME(09:00:00) - TIME(12:00:00);
```

intervallterm
Zeitspanne definieren

intervallterm legt eine gültige Zeitspanne (Intervall) fest. Der Datentyp ist INTERVAL und hat die Intervalleinheit, die mit UNITS oder durch die zugrundeliegenden Variablen festgelegt wird.

In SQL-Anweisungen für SESAM und UDS darf *intervallterm* nicht angegeben werden.

```
intervallterm::={ numterm { UNITS datumzeitfeld | YEARS | MONTHS | DAYS |
                            HOURS | MINUTES | SECONDS | FRACTIONS } |

                  ( intervallausdruck ) |
                  intervallterm { * | / } numterm }
```

---

numterm — siehe Metavariable *numterm*

datumzeitfeld — siehe Metavariable *datumzeitfeld*

intervallausdruck — *intervallausdruck w*ird als numerisch mit Genauigkeit 15 und Skalenfaktor 0 behandelt (siehe Metavariable *intervallausdruck*).

intervallterm — Zeitspanne als Faktor oder Dividend

Abhängig von der Einheit des Datentyps INTERVAL hat das Ergebnis folgenden Datentyp:

Datentyp von *intervallterm*	**Datentyp des Ergebnisses**
INTERVAL YEARS	INTERVAL MONTHS
INTERVAL MONTHS	INTERVAL MONTHS
INTERVAL HOURS	INTERVAL SECONDS
INTERVAL MINUTES	INTERVAL SECONDS
INTERVAL SECONDS	INTERVAL SECONDS
INTERVAL FRACTIONS	INTERVAL FRACTIONS

* — Multiplikationsoperator

/ — Divisionsoperator

Beispiel

Zur Zinsberechnung wird der Variablen &monat der Wert 30 Tage zugewiesen.

```
DECLARE VARIABLE &monat INTERVAL DAYS;
...
SET &monat=30 DAYS;
```

In der Anweisung SET kann das Schlüsselwort "DAYS" weggelassen werden, weil die Intervalleinheit für die Variable &monat mit der Anweisung DECLARE festgelegt ist:

```
...
SET &monat=30;
```

literal
Literal definieren

literal enthält eine datentypspezifische Zeichenfolge mit einem konstanten Wert.

```
literal::={ charliteral | numliteral  | datumzeitliteral | intervalliteral |
            sedecliteral }
```

---

charliteral

```
charliteral::='string' [ (n) ]
```

```
string::=[ zeichen ] ...
```

string
: Zeichenfolge, deren Datentyp alphanumerisch ist.
string muß von Hochkommas (') eingeschlossen werden. Wird ein Hochkomma in *string* verwendet, muß es doppelt angegeben werden. Das verdoppelte Hochkomma wird als ein Zeichen gewertet.
string kann leer sein und darf max. 256 Zeichen enthalten.

n Wiederholungsfaktor ($1 \leq n \leq 256$)

zeichen
: EBCDIC-Zeichen

numliteral

```
numliteral::=
{ [ + | - ] zahl [ { . | , } zahl ] |
  [ + | - ] zahl [ { . | , } zahl ] E [ + | - ] zahl }
```

numliteral enthält eine Festpunktzahl, deren Datentyp numerisch ist (max. 32 Ziffern). Unter der Genauigkeit einer Festpunktzahl versteht man die Anzahl der Ziffern. Unter dem Skalenfaktor einer Festpunktzahl versteht man die Anzahl der Nachkommastellen (max. 31).

zahl
: Für *zahl* dürfen nur Ziffern eingesetzt werden.

E Exponentialdarstellung zur Basis 10
Vor oder hinter *E* müssen die vorausgehenden und nachfolgenden Zeichen direkt anschließen. Es darf kein Leerzeichen (%) stehen.

datumzeitliteral

```
datumzeitliteral::=
{ DATE (jahr-monat-tag) |
  TIME (stunde:minute:sekunde [.bruchteil ] ) |
  TIMESTAMP (jahr-monat-tag stunde:minute:sekunde.bruchteil ) }
```

datumzeitliteral enthält ein gültiges Datum (Datentyp: DATE), eine gültige Uhrzeit (Datentyp: TIME oder TIME(3)) oder einen gültigen Zeitstempel (Datentyp: TIMESTAMP(3)) .

jahr
: Vierstellige Ganzzahl zwischen 0000 und 9999, die das Jahr angibt

monat
: Zweistellige Ganzzahl zwischen 1 und 12, die den Monat angibt

tag
: Zweistellige Ganzzahl zwischen 1 und 31 (passend zu Monat), die den Tag angibt

stunde
: Zweistellige Ganzzahl zwischen 00 und 23, die die Stunde angibt

minute
: Zweistellige Ganzzahl zwischen 00 und 59, die die Minute angibt

sekunde
: Zweistellige Ganzzahl zwischen 00 und 59, die die Sekunde angibt

bruchteil
: Dreistellige Ganzzahl zwischen 000 und 999, die die Sekundenbruchteile (1/1000 Sekunden) angibt

Die Trennzeichen zwischen den Komponenten müssen genau eingehalten werden:
Bindestrich (-) zwischen Jahr, Monat und Tag
Leerzeichen (%) zwischen Tag und Stunde
Doppelpunkt (:) zwischen Stunde, Minute und Sekunde
Punkt (.) zwischen Sekunde und Sekundenbruchteilen

intervalliteral

```
intervalliteral::=INTERVAL ({ + | - } zahl) datumzeiteinheit
```

zahl
: Für *zahl* dürfen nur Ziffern eingesetzt werden (maximal 32 Ziffern).

datumzeiteinheit
: siehe Metavariable *datumzeiteinheit*

sedecliteral

```
sedecliteral::=X'string' [ (n) ]
```

string
: sedezimale Zeichenfolge (max 512 Zeichen). Für *string* dürfen nur die Ziffern "0" bis "9" und die Zeichen "A" bis "F" eingesetzt werden.

n Wiederholungsfaktor ($1 \leq n \leq 256$)

mask
MASK-KLausel definieren

Die MASK-Klausel legt die Darstellung von Datenwerten in Ein- und Ausgabefeldern fest.

```
mask::=MASK charliteral
```

MASK — Mit MASK wird die Aufbereitung von Datenwerten in Ein- und Ausgabefeldern festgelegt.

charliteral — siehe Metavariable *literal*

Eine Maske hat zwei Hauptbestandteile, die in *charliteral* angegeben werden können: Maskensteuerzeichen und User-Text.

Maskensteuerzeichen regeln die Darstellung des aktuell auszugebenden Datenwertes in der Ausgabe. Es gibt Steuerzeichen für numerische, alphanumerische und Zeit-Datentypen.

User-Texte sind Textteile, die vom Anwender frei definiert werden können. Sie können an beliebigen Stellen innerhalb einer Maske angegeben werden. Sie müssen in zwei Hochkommas (' ') eingeschlossen werden, die jedoch bei der Ausgabe nicht erscheinen. Bei der Dateneingabe werden die User-Texte ignoriert.

Maskensteuerzeichen und User-Text können in beliebiger Reihenfolge in einer Maske definiert werden (Ausnahme: die Reihenfolge einiger numerischer Maskensteuerzeichen ist festgelegt). Es gibt jedoch feste Folgen von Steuerzeichen, die nicht durch User-Text unterbrochen werden dürfen.

Der Inhalt einer Maske muß so festgelegt sein, daß das für die Ein- oder Ausgabe aufbereitete Ergebnis nicht länger als 256 Zeichen ist. Die Maske selbst darf maximal 256 Zeichen lang sein.

Maskensteuerzeichen für alphanumerische Datentypen:

Masken-steuerzei-chen	Ausgabe
X	Beliebiges Zeichen aus dem EBCDIC-Zeichenvorrat. Die Stelle wird immer ausgegeben.
X(n)	Darstellung für die n-fache Angabe des Maskensteuerzeichens X unmittelbar hintereinander (0 < n).

Der ein- oder auszugebende Wert muß zur Maske passen. Ansonsten werden rechtsbündige Leerzeichen abgeschnitten oder ergänzt.

Ein eingegebener Wert muß im Datenfeld abgelegt werden können.

Maskensteuerzeichen für numerische Datentypen:

Masken-steuerzei-chen	Ausgabe
9	Zeichenstelle für eine Ziffer; wird immer ausgegeben
Z	Führende numerische Ziffernstelle. Enthält diese Stelle eine führende Null, so wird ein Leerzeichen ausgegeben.
*	Schecksicherungssymbol: führende, numerische Ziffernstelle. Enthält diese Stelle eine führende Null, so wird "*" ausgegeben.
P	Zeichenstelle für Dezimalpunkt. Entsprechend der globalen Einstellung wird ein ". " oder "," ausgegeben. "P" darf maximal einmal als Steuerzeichen auftreten.
+	Zeichenstelle für das Vorzeichen Plus "+" oder Minus "-". Das Vorzeichen wird, entsprechend dem Datenwert des Feldes, immer ausgegeben.
–	Zeichenstelle für ein negatives Vorzeichen. Bei negativem Datenwert wird ein Minus, bei positivem Datenwert ein Leerzeichen ausgegeben.

Masken-steuerzei-chen	**Ausgabe**
S	Führende numerische Ziffernstelle. In diese Stelle kann das Vorzeichen eines Datenwertes gleiten, wenn das erste Steuerzeichen der Maske ein "+" oder "-" und diese Stelle eine führende Null enthält. Wird "S" angegeben, muß es immer vor eventuellen "Z"- oder "*"-Steuerzeichen angegeben werden. Ist das Steuerzeichen der Maske kein "+" oder "-", so hat "S" dieselbe Wirkung wie "Z" oder "*".
E	Kennzeichen für Gleitpunktdarstellung. Es folgt eine Längenangabe, über die die Anzahl der auszugebenden Nachkommastellen zu steuern ist. Für eine Gleitpunktmaske gilt: En (8 < n < 23) D.h. die Anzahl der auszugebenden Nachkommastellen muß zwischen 1 und 14 liegen. "E" darf maximal einmal als Steuerzeichen auftreten.
BWZ	Steuerzeichen für ein gesamtes Feld. Ist der Datenwert=0 oder eine leere Zeichenkette, dann wird bei der Bildschirmausgabe das gesamte Feld mit Leerzeichen belegt. Bei der Eingabe führt ein mit (einem oder mehreren) Leerzeichen belegtes Feld zusammen mit dem Steuerzeichen "BWZ" zu dem Datenwert 0. "BWZ" muß am Ende der Maske durch mindestens 1 Leerzeichen von der restlichen Maske getrennt angegeben werden.

Masken-steuerzei-chen	**Ausgabe**
, (Komma) . (Punkt) B (= Einfüge-Steuerzei-chen)	Einfüge-Steuerzeichen sind Zeichenstellen, in die ein Komma (,), Punkt (.) oder Leerzeichen (B) eingefügt wird. Für Einfüge-Steuerzeichen, die in der Folge von "Z"-, "*"- oder "S"-Steuerzeichen eingebettet sind oder die unmittelbar rechts an dieser Folge anschließen gilt: Sie sind Teil dieser Folge von Steuerzeichen. Führt diese Folge von Steuerzeichen zur Ausgabe von Leer- oder "*"-Zeichen durch Unterdrückung führender Nullen, so gilt weiter: An den Stellen eingebetteter oder unmittelbar rechts davon stehender Einfüge-Steuerzeichen wird ebenfalls ein Leer- oder "*"-Zeichen ausgegeben. Analog gilt dann: In Einfüge-Steuerzeichen, die innerhalb oder rechts von Folgen von "S"-Steuerzeichen stehen, gleiten Vorzeichen. Einfüge-Steuerzeichen dürfen nur links von "P" stehen.
9(n) Z(n) *(n) S(n)	Ersatzdarstellungen für die Angabe des jeweiligen Steuerzeichens das n-mal unmittelbar hintereinander dargestellt werden soll (0 < n).

Der ein- oder auszugebende Wert muß entsprechend den Konvertierungsregeln zur Maske passen. Ansonsten können Nachkommastellen abgeschnitten oder nicht belegte Stellen aufgefüllt werden.

Maskensteuerzeichen für Zeit-Datentypen:

Masken-steuerzei-chen	**Ausgabe**
YYYY	4-stellige Jahresangabe
ZZZY	4-stellige Jahresangabe; führende Nullen werden als Leerzeichen ausgegeben
MO	2-stelliger Monatswert
ZO	2-stelliger Monatswert; eine führende Null wird als Leerzeichen ausgegeben

Masken-steuerzei-chen	**Ausgabe**
DD	2-stelliger Tageswert
ZD	2-stelliger Tageswert; eine führende Null wird als Leerzeichen ausgegeben
HH	2-stelliger Stundenwert
ZH	2-stelliger Stundenwert; eine führende Null wird als Leerzeichen ausgegeben
MI	2-stelliger Minutenwert
ZI	2-stelliger Minutenwert; eine führende Null wird als Leerzeichen ausgegeben
SS	2-stelliger Sekundenwert
ZS	2-stelliger Sekundenwert; eine führende Null wird als Leerzeichen ausgegeben
FFF	3-stelliger Sekundenbruchteilswert
ZZF	3-stelliger Sekundenbruchteilswert; führende Nullen werden als Leerzeichen ausgegeben
WW	2-stelliger Wochenwert
ZW	2-stelliger Wochenwert; eine führende Null wird als Leerzeichen ausgegeben
JJJ	3-stellige, julianische Tagesangabe (= Tag im Jahr)
ZZJ	3-stellige, julianische Tagesangabe; führende Nullen werden als Leerzeichen ausgegeben
Q...Q	abdruckbarer Tagesname
Q(n)	abdruckbarer Tagesname mit n Stellen (0 < n)
R...R	abdruckbarer Monatsname
R(n)	abdruckbarer Monatsname mit n Stellen (0 < n)
AP	Amerikanische Angabe für a.m. (ante meridiem = vormittags) oder für p.m. (post meridiem = nachmittags)

Der eingegebende Wert muß einer Variablen mit einem Zeit-Datentyp (DATE, TIME, TIME(3) oder TIMESTAMP(3) eindeutig zugeordnet werden können. Ansonsten können die Steuerzeichen beliebig mit User-Text gemischt werden.

Maskensteuerzeichen für den Datentyp INTERVAL:

Masken-steuerzei-chen	**Ausgabe**
9	Zeichenstelle für eine Ziffer; wird immer ausgegeben
Z	Führende numerische Ziffernstelle. Enthält diese Stelle eine führende Null, so wird ein Leerzeichen ausgegeben.
+	Zeichenstelle für das Vorzeichen Plus "+" oder Minus "-". Das Vorzeichen wird, entsprechend dem Datenwert des Feldes, immer ausgegeben.
–	Zeichenstelle für ein negatives Vorzeichen. Bei negativem Datenwert wird ein Minus, bei positivem Datenwert ein Leerzeichen ausgegeben.
*	Schecksicherungssymbol: führende, numerische Ziffernstelle. Enthält diese Stelle eine führende Null, so wird "*" ausgegeben.
S	Führende numerische Ziffernstelle. In diese Stelle kann das Vorzeichen eines Datenwertes gleiten, wenn das erste Steuerzeichen der Maske ein "+" oder "-" und diese Stelle eine führende Null enthält. Wird "S" angegeben, muß es immer vor eventuellen "Z"- oder "*"-Steuerzeichen angegeben werden. Ist das Steuerzeichen der Maske kein "+" oder "-", so hat "S" dieselbe Wirkung wie "Z" oder "*".
BWZ	Steuerzeichen für ein gesamtes Feld. Ist der Datenwert=0 oder eine leere Zeichenkette, dann wird bei der Bildschirmausgabe das gesamte Feld mit Leerzeichen belegt. Bei der Eingabe führt ein mit (einem oder mehreren) Leerzeichen belegtes Feld zusammen mit dem Steuerzeichen "BWZ" zu dem Datenwert 0. "BWZ" muß am Ende der Maske durch mindestens 1 Leerzeichen von der restlichen Maske getrennt angegeben werden.
9(n) Z(n) *(n) S(n)	Ersatzdarstellungen für die Angabe des jeweiligen Steuerzeichens das n-mal unmittelbar hintereinander dargestellt werden soll (0 < n).

Der ein- oder auszugebende Wert muß entsprechend den Konvertierungsregeln zur Maske passen. Ansonsten können nicht belegte Stellen aufgefüllt werden.

Zwischen den Maskensteuerzeichen und einem Hochkomma darf kein Leerzeichen sein.

Regeln für Datenfelder mit numerischem Datentyp

– "+" und "-" schließen sich gegenseitig aus. Innerhalb einer Maske kann deshalb nur eines dieser Zeichen vorkommen. Es muß dann das erste Zeichen in der Maske sein. Ist keines dieser Zeichen angegeben, wird kein Vorzeichen ausgegeben. Das Vorzeichen wird nicht durch ein Leerzeichen ersetzt.

– "Z" und "*" dürfen nicht zusammen innerhalb einer Maske auftreten. Rechts vom Dezimalpunkt-Steuerzeichen "P" darf kein "Z" oder "*" auftreten.
Ausnahme: Alle Zeichenstellen in der Maske bis auf "P" und Einfügesteuerzeichen sind "Z" oder "*".

– Eine Maske zur Exponentialdarstellung darf ausschließlich "E" mit Längenangabe enthalten.

Regeln für Datenfelder mit Zeit-Datentypen

– Die jeweiligen Steuerzeichen dürfen maximal einmal innerhalb einer Maske auftreten und müssen bündig hintereinander stehen. So dürfen bei YYYY-Steuerzeichen die vier "Y" nicht durch User-Text voneinander getrennt werden.

– Die gleichzeitige Angabe der Steuerzeichen für die numerische und verbale Ausgabe des Tages- oder Monatswertes ist erlaubt.

– Werden für eine Datenaus- und -eingabe zu wenige Q oder R-Steuerzeichen angegeben, so ist die interne Zuordnung des Tages- oder Monatswertes zur verbalen Eingabe nicht eindeutig. Es gelten dann folgende Regelungen:

  Sind in der Maske gleichzeitig DD/ZD- oder M0/Z0-Steuerzeichen vorhanden, so wird der Tages- oder Monatswert durch die Eingabe an diesen Stellen belegt.

  Sind keine DD/ZD- oder MO/ZO-Steuerzeichen angegeben, so erfolgt Programmabbruch. Eine Meldung wird ausgegeben, daß die Dateneingabe nicht eindeutig ist.

– Die in der Maske angegebenen Steuerzeichen müssen den Datums- und Zeitangaben der jeweiligen Variablen entsprechen. Bei der Dateneingabe ist darauf zu achten, daß der maskiert eingegebene Datenwert eindeutig der betreffenden DATE-, TIME- oder TIMESTAMP-Variablen zugeordnet werden kann.

Beispiele

Datenwert	Definition der Darstellung	Ausgabe
12.60	NUM(6,2) MASK 'ZZZZP99'	%%12.60
12.60	NUM(6,2) MASK '+SZZZP99'	%+%12.60
12.60	NUM(6,2) MASK '****P99'	**12.60
12.60	NUM(6,2) MASK 'ZZZZP99''%DM'''	%%12.60%DM
-12.60	NUM(6,2) MASK '-ZZZZP99'	-%%12.60
12.60	NUM(6,2) MASK '-ZZZZP99'	%%%12.60
12.60	NUM(6,2) MASK '99'	12
12.60	NUM(6,2) MASK '9999'	0012
1260	NUM(6,2) MASK 'ZZ,ZZZPZZ'	%1,260.00
-1260	NUM(6,2) MASK 'ZZ,ZZZPZZ'	%1,260.00
1260	NUM(6,2) MASK '-ZZ,ZZZPZZ'	%%1,260.00
126000	NUM(8,2) MASK 'ZZZBZZZPZZ'	126%000.00
12.60	NUM(8,2) MASK 'ZZZBZZZPZZ'	%%%%12.60
12.60	NUM(6,2) MASK 'ZZZZP99 BWZ'	%%12.60
0	NUM(6,2) MASK 'ZZZZP99 BWZ'	%%%%%%
12.60	NUM(6,2) MASK 'E9'	%1.2E+001
12.60	NUM(6,2) MASK 'E10'	%1.26E+001
12.60	NUM(6,2) MASK 'E16'	%1.26000000E+001
1260	INT MASK 'E9'	%1.2E+003
12	INT MASK 'ZZP99'	12.00
1260	INT MASK 'ZZP99'	Fehlermeldung
1260	REAL MASK 'ZZ99'	1260
12.60	REAL MASK 'ZZ99'	%%13
1260	REAL MASK 'E9'	%1.2E+003
1.2601260126	FLOAT MASK 'E16'	%1.26012601E+000
DER%PATE	CHAR (8) MASK '''DAS%BUCH:%''XXXXXXXX'	DAS%BUCH:%DER%PATE
DER%PATE	CHAR (8) MASK '''DAS%BUCH:%''X(8)'	DAS%BUCH:%DER%PATE
1789-07-14	DATE MASK 'DD''.''MO''.''YYYY'	14.07.1789
0622-06-15	DATE MASK 'ZD''.''ZO''.''ZZZY'	15.%6.%622
1982-08-13	DATE MASK 'Q(10)'',%DEN%''ZD''.'''	FREITAG%%%,%DEN%13.
20:15:26	TIME MASK 'HH''%UHR%''MI'	20%UHR%15
16:09:56.127	TIME(3) MASK 'HH''%UHR%''ZI'	16%UHR%%9

mengenfunktion
Mengenfunktionen

Eine Mengenfunktion berechnet einen Wert aus einer Menge von Sätzen.

```
mengenfunktion::={ SUM | AVG | MAX | MIN } ( [ ALL ] ausdruck )
```

SUM	berechnet die Summe der Werte einer Menge
AVG	berechnet das arithmetische Mittel aus einer Menge von Werten
MAX	bestimmt den größten Wert einer Menge
MIN	bestimmt den kleinsten Wert einer Menge
ALL	Alle Werte werden berücksichtigt, auch solche, die doppelt vorkommen. Bei MAX und MIN ist die Angabe syntaktisch erlaubt, hat aber keine Bedeutung.
ausdruck	Argument, auf das die Mengenfunktion angewendet wird. *ausdruck* muß ein nicht-strukturiertes Satzelement sein. Für AVG und SUM muß *ausdruck* numerisch sein. (Siehe Metavariable *ausdruck*)

Das Ergebnis der Mengenfunktion hat folgenden Datentyp:

Mengenfunktion	**Datentyp des Ergebnisses**
MIN und MAX	gleicher Datentyp wie *ausdruck*
SUM	Datentyp DECIMAL mit der Genauigkeit 15. Die Anzahl der Nachkommastellen entspricht der Anzahl der Nachkommastellen des angegebenen *ausdruck*.
AVG	Datentyp DECIMAL mit der Genauigkeit 15. Die Anzahl der Nachkommastellen entspricht der Anzahl der Nachkommastellen des angegebenen *ausdruck*.

SUM - Summe berechnen

SUM berechnet die Summe der Werte einer Menge.

```
SUM ( [ ALL ] ausdruck )
```

ALL	Vorbelegung Alle Werte werden berücksichtigt, auch solche, die doppelt vorkommen.
ausdruck	Ausdruck, der einen numerischen Wert ergibt.

AVG - Arithmetisches Mittel

AVG berechnet das arithmetische Mittel aus einer Menge von Werten.

```
AVG ( [ ALL ] ausdruck )
```

ALL	Vorbelegung Alle Werte werden berücksichtigt, auch solche, die doppelt vorkommen.
ausdruck	Ausdruck, der einen numerischen Wert ergibt.

MAX - Maximum bestimmen

MAX bestimmt den größten Wert einer Menge.

```
MAX ( [ ALL ] ausdruck )
```

ALL	Die Angabe ALL ist syntaktisch erlaubt, hat aber keine Bedeutung.
ausdruck	Ausdruck, der einen numerischen oder alphanumerischen Wert ergibt.

MIN - Minimum bestimmen

MIN bestimmt den kleinsten Wert einer Menge.

```
MIN ( [ ALL ] ausdruck )
```

ALL	Die Angabe ALL ist syntaktisch erlaubt, hat aber keine Bedeutung.
ausdruck	Ausdruck, der einen numerischen oder alphanumerischen Wert ergibt.

nullwert
Darstellung des NULL-Werts definieren

nullwert legt die Darstellung des NULL-Werts fest.

```
nullwert::=[ CHARTYPE=charliteral1 ] [ NUMTYPE=charliteral2 ]
```

CHARTYPE=charliteral1

Die NULL-Wertdarstellung wird für die Datentypen CHARACTER, DATE, TIME, TIME(3) und TIMESTAMP(3) (siehe Metavariable *grunddatentyp*) festgelegt (max. 1 Zeichen).

CHARTYPE darf nur einmal angegeben werden.

NUMTYPE=charliteral2

Die NULL-Wertdarstellung wird für die Datentypen NUMERIC, DECIMAL, INTEGER, SMALLINT und INTERVAL festgelegt (max. 1 Zeichen).

Folgende Zeichen sind zugelassen:
die Ziffern 0 bis 9, das Komma (,), der Punkt (.) und die Sonderzeichen * + – @

NUMTYPE darf nur einmal angegeben werden.

Bei Druckerausgaben ist die NULL-Wertdarstellung mit dem Punkt (.) vorbelegt.
Bei Bildschirmein-/ausgaben ist die NULL-Wertdarstellung mit dem Zeichen X'00' vorbelegt.

numausdruck
Numerischen Ausdruck definieren

numausdruck legt numerische Ausdrücke fest.

Die Grundrechenarten werden mit einer Genauigkeit von bis zu 32 Stellen (Datentyp XDEC) ausgeführt.

i Im Old-Style werden Grundrechenarten mit einer Genauigkeit von bis zu 15 Stellen (Datentyp NUMERIC) ausgeführt.

In Programmen, die mit dem DRIVE-Compiler DRIVE/WINDOWS-COMP übersetzt wurden, werden die Grundrechenoperationen über Festpunkt- oder Dezimalarithmetik durchgeführt. Soweit bei Addition, Subtraktion und Multiplikation nur Größen vom Typ INTEGER oder SMALLINT auftreten, erfolgt Festpunktarithmetik. In allen übrigen Fällen Dezimalarithmetik. (Siehe DRIVE-Compiler [16])

Weitere Ausdrücke siehe Metavariable *ausdruck*.

```
numausdruck::=numterm1 [ [ * | / | % | ** ] [ + | - ] numterm2 ]
```

---

numterm1, numterm2 Das erste Zeichen von *numterm* (siehe Metavariable *numterm*) darf nicht "+" oder "–" sein.

Der Datentyp von *numterm* muß numerisch sein, sonst gibt DRIVE/WINDOWS eine Fehlermeldung aus.

Bei Addition, Subtraktion, Multiplikation, Division und Prozentrechnung erhält *numausdruck* den Datentyp DECIMAL, bei Potenzierung den Datebtyp FLOAT.

Die Genauigkeit P und der Skalenfaktor S von *numausdruck* hängen von den Genauigkeiten P1 und P2, den Skalenfaktoren S1 und S2 und der Verknüpfung der *numterm* ab.

Rechenoperation	**P und S von *numausdruck***
`numterm1 * numterm2`	`P=MIN(15,P1+P2)` `S=MIN(15,S1+S2)`
`numterm1 / numterm2`	`P=15` `S=MAX(15-P1+S1-S2,0)`

`numterm1 { + | - } numterm2`	`P=MIN(15,MAX(P1-S1,P2-S2)`
	`  + MAX(S1,S2) + 1)`
	`S=MAX(S1,S2)`

Als Vorrangsregel gilt bei der Auswertung der Operatoren

Klammern
– vor Vorzeichenoperatoren
– vor Potenzierung
– vor Multiplikation, Division, Prozentrechnung
– vor Addition, Subtraktion.

Gleiche Operatoren werden von links nach rechts abgearbeitet.

Hat *numterm* den Wert NULL, hat auch *numausdruck* den Wert NULL.

* Multiplikationsoperator.

/ Divisionsoperator.

% Prozentoperator.

Der Skalenfaktor von *numausdruck* ist gleich der Summe der Skalenfaktoren von *numterm1* und *numterm2*.

In SQL-Anweisungen darf "%" nicht verwendet werden.

** Potenzierungsoperator.

Der Skalenfaktor von *numausdruck* ist 6.

Falls *numterm1* = 0, muß *numterm2* > 0 sein.

Falls *numterm1* < 0, muß *numterm2* eine Ganzzahl sein.

In SQL-Anweisungen darf "**" nicht verwendet werden.

\+ Vorzeichen- oder Summenoperator.

– Vorzeichen- oder Differenzoperator.

Als Vorzeichenoperator bewirkt "-" einen Vorzeichenwechsel von *numterm*.

numterm
Numerischen Term definieren

```
numterm::=[ + | - ] { wert |
                      satzelement |
                      mengenfunktion |
                      wertefunktion |
                      ( numausdruck ) |
                      intervallterm }
```

+	Vorzeichen- oder Summenoperator
–	Vorzeichen- oder Differenzoperator
	Als Vorzeichenoperator bewirkt "-" einen Vorzeichenwechsel von *numterm*. Das erste Zeichen vor oder nach monadischen Operatoren "+" und "-" darf nicht wieder "+" oder "-" sein. Der Datentyp von *numterm* muß numerisch ohne Nachkommastellen sein.
wert	siehe Metavariable *wert*
satzelement	Die Satzelemente, die in *numterm* angegeben werden, müssen aus derselben Basistabelle stammen. *satzelement* siehe SQL-Lexika [4-6]
mengenfunktion	siehe Metavariable *mengenfunktion*
wertefunktion	siehe Metavariable *wertefunktion*
numausdruck	siehe Metavariable *numausdruck*
intervallterm	siehe Metavariable *intervallterm* *intervallterm* wird als numerisch mit Genauigkeit 32 und Skalenfaktor 0 behandelt (siehe auch Metavariable *intervallausdruck*). In SQL-Anweisungen für SESAM und UDS darf *intervallterm* nicht angegeben werden.

programming
Anweisungen für den Verarbeitungsteil definieren

programming legt eine DRIVE-Anweisung für den Verarbeitungsteil eines Programms fest.

```
programming::={ anweisung; |
                CASE; END CASE; |
                CYCLE; END CYCLE; |
                DISPATCH; END DISPATCH; |
                IF; END IF; } ...
```

---

anweisung	Lineare Programmanweisung für den Verarbeitungsteil eines Programms (siehe DRIVE-Programmiersprache [2].
CASE	Programmieren einer bedingten Verzweigung (siehe Anweisung CASE).
CYCLE	Programmieren einer Schleife (siehe Anweisung CYCLE).
DISPATCH	Programmieren eines DISPATCH-Blocks (siehe Anweisung DISPATCH).
IF	Programmieren einer Bedingung (siehe Anweisung IF).

strukturtyp
Strukturierte Variable definieren

Man unterscheidet vier Arten von strukturierten Variablen:

– Vektoren (multiple Felder)
– Datengruppen
– Wiederholungsgruppen
– redefinierte Variablen

Zwei strukturierte Variablen sind gleich,

– wenn ihre Zerlegung in Komponenten identisch ist, und
– wenn die einander entsprechenden einfachen Komponenten vom selben Datentyp sind.

Weitere Datendefinitionen siehe Metavariable *datendef*.

i Verwenden Sie die REDEFINES-Klausel nicht bei Programmen, die Sie auf verschiedenen Rechnern einsetzen wollen, da sie maschinenabhängig ist. Auf verschiedenen Rechnern kann es zu unterschiedlichen Ergebnissen kommen.

```
strukturtyp::={ vektor |
                datengruppe |
                wiederholungsgruppe |
                usertyp |
                [ ( n ) ] [ REDEFINES { variable1 | komponente [ suffix ] } ]
                                                          LIKE variable2 }
```

---

vektor	siehe Metavariable *vektor*
datengruppe	siehe Metavariable *datengruppe*
wiederholungsgruppe	siehe Metavariable *wiederholungsgruppe*
usertyp	Benutzerdefinierter Datentyp, der mit DECLARE TYPE im DRIVE-Programm definiert sein muß.
n	Wiederholungsfaktor ($0 < n < 256$).
REDEFINES	Mit REDEFINES wird die Variable gekennzeichnet, die redefiniert wird (Basisvariable). Die redefinierende Variable darf nicht länger als die Basisvariable

sein. Ist die redefinierende Variable Komponente einer Struktur, muß die Basisvariable eine Komponente derselben Hauptstruktur (Struktur mit der Stufennummer 1) sein.

REDEFINES darf nicht in INIT- und USING-Klauseln angegeben werden.

variable1

Name der Basisvariablen, die redefiniert wird. Sie muß bereits definiert worden sein und darf selbst keine mit LIKE oder REDEFINES definierte Variable sein. Ist die Basisvariable eine Struktur, darf die redefinierende Variable keine Komponente dieser Struktur sein.

Zwischen dem Beginn der kleinsten Teilstruktur, die sowohl die Basisvariable als auch die redefinierende Variable enthält, und der Basisvariablen oder der redefinierenden Variablen dürfen keine Wiederholungsgruppen liegen.

komponente

Alphanumerische Zeichen für den Namen einer Komponente.

0 < Anzahl(*komponente*) < 32

suffix

siehe Metavariable *variable*

LIKE

Mit LIKE wird die Struktur einer Daten- oder Wiederholungsgruppe, die bereits definiert sein muß, komponentenweise in eine Variable kopiert. Bei einer Wiederholungsgruppe wird nur die Struktur und nicht der Wiederholungsfaktor übernommen. Die Stufennummern werden beim Kopieren entsprechend angepaßt. Die Angaben zu *n* und die REDEFINES-Klausel werden nicht übernommen.

Diese Variablen dürfen nur im Verarbeitungsteil von Programmen verwendet werden.

Wird die Struktur einer Variablen durch die LIKE-Klausel bestimmt, sind für diese Variable als zusätzliche Komponenten nur Komponenten der gleichen oder einer tieferen Stufennummer erlaubt.

LIKE-Angaben dürfen nicht geschachtelt werden.

variable2

Name der Variablen, deren Struktur kopiert wird. *variable2* muß bereits als Daten- oder Wiederholungsgruppe definiert worden sein. *variable2* darf qualifiziert aber nicht indiziert sein. Sie darf keine übergeordnete Struktur der zu definierenden Variablen sein.

Beispiele

REDEFINES innerhalb einer Struktur

```
DECLARE VARIABLE 1 &alpha,
                   2 a1,
                     3 a11 CHAR (10),
                     3 a12 CHAR (10),
                   2 a2 INT REDEFINES &a1,
                   2 a3 CHAR (2);
```

&a2 belegt den &a1 zugeordneten Bereich neu, d.h. den Bereich, der von &a11 und &a12 belegt ist; &a2 kann allerdings nur einen Bereich soweit redefinieren, wie es seine eigene Größe erlaubt: hier nur bis 4 Zeichen von &a11.

LIKE auf eine andere Gruppe (= Datengruppe oder Wiederholungsgruppe ohne Wiederjholungsfaktor)

Die Struktur der Datengruppe &v1 wird in die Variable &v2 übernommen. &v2 ist identisch mit &v1.

```
DECLARE VARIABLE 1 &v1,
                   2 v11 CHAR,
                   2 v12 INT,
                   &v2 LIKE &v1;  →  1 &v2,
                                       2 v11 CHAR,
                                       2 v12 INT;
```

LIKE innerhalb einer Gruppe

Die Komponente "v32" wird genauso definiert wie Komponente "v31".

```
DECLARE VARIABLE 1 &v3,                    1 &v3,
                   2 v31,                    2 v31,
                     3 v311 CHAR,              3 v311 CHAR,
                     3 v312 NUM (4,2),         3 v312 NUM (4,2),
                   2 v32 LIKE &v31;  →       2 v32,
                                               3 v311 CHAR,
                                               3 v312 NUM (4,2);
```

variable
Einfache Variable oder Komponente referenzieren

variable referenziert eine einfache Variable oder eine Komponente einer strukturierten Variablen. Die Liste aller Komponenten auf der nächsten Stufe kann mit der Teilqualifizierung ".*" angegeben werden. Der Name von *variable* muß mit dem "&"-Zeichen beginnen und darf insgesamt maximal 32 Zeichen lang sein.

```
variable::={ [ charliteral: ] varname1 [ suffix ] |

             [ charliteral: ]
             { varname2 [ ( { index1 | bereich1 }, { index2 | bereich2 } ) ] } }
```

---

charliteral

charliteral ist nur innerhalb einer Reportdefinition erlaubt und bezeichnet diejenige Satzart, auf die sich der Wert beziehen soll.

varname1

Name einer einfachen Variablen oder der ersten Qualifizierung für eine Komponente

varname1 muß mit dem "&"-Zeichen beginnen und darf insgesamt max. 32 Zeichen lang sein.

suffix

```
suffix::={ gruppenkomponente | indexkomponente }
```

Als Suffix kann eine Gruppenkomponente *gruppenkomponente* oder eine Indexkomponente *indexkomponente* angegeben werden.

```
gruppenkomponente::= . { * | komponente [ suffix ]}
```

. Qualifizierungsoperator

* abkürzende Schreibweise für die Liste aller Variablenkomponenten auf der nächst tieferen Stufe.
In SQL-Anweisungen darf "*" bei Satzelementen nicht angegeben werden.

komponente
Name eines Teils einer strukturierten Variablen (= Datengruppe und Wiederholungsgruppe) oder Name eines Teils eines Satzelements einer Datenbank (siehe Anweisung DECLARE VARIABLE ... LIKE CURSOR/TABLE).
Die Komponenten auf der untersten Hierarchiestufe, d.h. mit der höchsten Stufennummer (Level) werden einfache Komponenten oder Basiskomponenten genannt.

komponente wird ohne das "&"-Zeichen angegeben und darf insgesamt max. 31 Zeichen lang sein.
komponente [suffix] darf nicht Teil einer Wiederholungsgruppe sein.

```
indexkomponente::=
     { ( { index | varname [ .komponente ]... }
         [ .komponente [ suffix ] ) |
       ( bereich ) }
```

indexkomponente
muß angegeben werden, wenn für ein Satzelement eine Feldkomponente eines strukturierten Feldes mit Ausprägung oder eines multiplen Feldes spezifiziert werden soll.
Für die Angabe einer Feldkomponente eines multiplen Feldes muß *index* oder *bereich* mit einem Wert versorgt werden, d.h. die Angabe einer Index-Variablen ist nicht erlaubt.

index
bezeichnet die Ausprägung eines Vektors oder einer Wiederholungsgruppe.
Für *index* dürfen Werte angegeben werden, für die gilt:
0 < *index* ≤ *repetitions*, wobei *repetitions* die Angabe in der Typdefinition der Komponente ist (siehe Metavariable *vektor* und *wiederholungsgruppe*).

```
bereich::=index1 - index2
```

bereich
bezeichnet einen Ausprägungsbereich. Die Bereichsgrenzen dürfen keine Index-Variablen sein und es muß gelten:
0 < *index1* < *index2* ≤ *repetitions*, wobei *repetitions* die Angabe in der Typdefinition der Komponente ist.

varname2 — Name einer Matrix.

varname2 muß mit dem "&"-Zeichen beginnen und darf insgesamt max. 32 Zeichen lang sein.

index1, index2 — bezeichnet die Ausprägung einer Matrix, d.h. ein Matrixelement.

bereich1, bereich2 — bezeichnet einen Ausprägungsbereich einer Matrix, d.h. eine Untermatrix.
Die Bereichsgrenzen dürfen keine Index-Variablen sein und es muß für jede Grenze gelten: untergrenze < obergrenze ≤ *repetitions*, wobei *repetitions* die Angabe in der Typdefinition der Komponente ist (siehe Metavariable *datendef*).

Beispiel

"fremdsprache(1)" gibt die Feldkomponente der Ausprägung 1 an.
"fremdsprache(2-5)" gibt den Ausprägungsbereich von 2 bis 5 an.

In *suffix* dürfen maximal drei Indexkomponenten angegeben werden, weil in Datengruppen maximal drei Wiederholungsgruppen ineinander geschachtelt sein dürfen. Nur die letzte angegebene Indexkomponente darf ein Bereich sein.

Beispiel

```
DECLARE VARIABLE 1 &adresse
                   2 ort CHAR (10),
                   2 strasse CHAR (20);
```

&adresse.* ist eine abkürzende Schreibweise für &adresse.ort, &adresse.strasse.

Die Komponente "ort" kann folgendermaßen angesprochen werden:

```
SET  &adresse.ort='Muenchen';
```

oder als "eigenständige" Variable:

```
SET  &ort='Muenchen';
```

vektor
Vektor definieren

vektor legt für eine Variable den Datentyp "Vektor" oder "multiples Feld" fest. *vektor* besteht aus einer festen Anzahl von Komponenten, die alle denselben Grunddatentyp besitzen. Die Anzahl der Komponenten wird durch den Wiederholungsfaktor *n* festgelegt.

Weitere Strukturtypen siehe Metavariable *strukturtyp*.

```
vektor::=( n ) grunddatentyp basistyp
```

---

n	Wiederholungsfaktor (0 < *n* < 256)
grunddatentyp	siehe Metavariable *grunddatentyp*
basistyp	siehe Metavariable *basistyp*

Beispiel

Die Variable &umsatz (12) ist ein Vektor, der aus zwölf Feldern besteht. Jedes Feld hat 7 Vorkomma- und zwei Nachkommastellen und ist mit 0 vorbelegt.

```
DECLARE VARIABLE &umsatz (12) NUMERIC (9,2) INIT 0;
```

Dem siebten Feld wird der Wert "12345.67" zugewiesen.

```
SET &umsatz (7)=12345.67;
```

wert
Datenwert definieren

wert legt den Datenwert für eine Variable oder für eine Komponente einer Variablen fest.

i Weil der NULL-Wert nicht überall dort erlaubt ist, wo *wert* erlaubt ist, wurde er nicht in die Metavariable *wert* aufgenommen.

```
wert::={ charprim | variable | literal | aggregat | VALUE | $PI }
```

---

charprim — siehe Metavariable *charprim*.

variable — siehe Metavariable *variable*. *variable* darf in SQL-Anweisungen für UDS-Datenbanken nicht den NULL-Wert enthalten.

literal — siehe Metavariable *literal*.

aggregat

```
aggregat::= < { wert | NULL }, ... >
```

aggregat spezifiziert ein Aggregat, einen strukturierten Wert, dessen Komponenten durch *wert* festgelegt sind. *aggregat* darf nicht mehr als 255 *werte* enthalten. Für *wert* dürfen keine strukturierten Variablen angegeben werden.

wert
: Für *wert* dürfen Literale, Variablen und $PI angegeben werden. Bei strukturierten Variablen darf nur die unterste Struktur, aber nicht die gesamte Struktur angesprochen werden.

NULL
: Der Komponente von *aggregat* wird der NULL-Wert zugewiesen. Jeder Variablen und jeder Spalte einer Tabelle (ohne Nicht-NULL-Bedingung) kann der NULL-Wert zugeordnet werden (siehe Anweisungen SET im DRIVE-Lexikon [3] sowie INSERT und UPDATE in den SQL-Lexika [4-6].

Beispiel

```
adresse=<'Waldweg 4',80462,&ort>
```

VALUE — VALUE darf nur bei *check* angegeben werden. Dort kann die Variable, auf die sich die Prüfbedingung bezieht, referenziert werden, d.h. der Name der Variablen muß nicht wiederholt werden.

$PI	$PI steht für die Zahl 3,141592653 ...

wertefunktion
Wertefunktion

wertefunktion berechnet einen Wert, indem eine Wertefunktion auf genau ein Argument angewendet wird.

wertefunktion darf nicht in SQL-Anweisungen angegeben werden.

```
wertefunktion::=
    { { SIN | COS | TAN | LN | LG | ABS | EXP | SQR | SQRT } ( numausdruck ) |
      CHARLENGTH ( charausdruck1 ) |
      ROUND ( numausdruck1 [, s1 ] ) |
      TRUNC ( numausdruck2 [, s2 ] ) |
      NUMERIC ( charausdruck2 [, charliteral ] ) |
      POSITION ( charausdruck3 IN charausdruck4 [, n2 ] ) |
      LENGTH ( charausdruck5 ) |
      MODULO ( numausdruck3, numausdruck4 ) }
```

SIN	Die Sinusfunktion wird angewendet.
COS	Die Cosinusfunktion wird angewendet.
TAN	Die Tangensfunktion wird angewendet.
LN	Der natürliche Logarithmus wird angewendet.
LG	Der Zehnerlogarithmus wird angewendet,
ABS	Der Absolutbetrag wird angewendet.
EXP	Die Exponentialfunktion wird angewendet.
SQR	Die Quadratfunktion wird angewendet.
SQRT	Die Quadratwurzelfunktion wird angewendet.
numausdruck	Argument, auf das die Wertefunktion anzuwenden ist. Die trigonometrischen Funktionen beziehen sich auf Argumente im Bogenmaß. *numausdruck* darf kein Aggregat und keine strukturierte Variable sein.

CHARLENGTH

CHARLENGTH liefert die Länge der Zeichenkette *charausdruck1*.

Die Position des letzten alphanumerischen Zeichens von *charausdruck1* wird zurückgegeben. Dieses Zeichen kann auch ein Leerzeichen sein.

Hat *charausdruck1* den NULL-Wert, liefert CHARLENGTH den NULL-Wert.

charausdruck1

Der Datentyp von *charausdruck1* muß alphanumerisch sein.

ROUND

ROUND rundet numerische Werte auf die festgelegten Stellen.

Der Rundungsfaktor ist: $5 * 10^{-s1-1}$

Hat eines der Argumente den NULL-Wert, dann liefert ROUND den NULL-Wert.

numausdruck1

siehe Metavariable *numausdruck*

s1

s1 legt fest, auf wieviel Stellen *numausdruck1* gerundet werden soll. *s1* muß eine ganze Zahl sein.

Ohne Angabe von *s1* wird die erste Nachkommastelle gerundet. *numausdruck1* wird eine ganze Zahl.

Ist *s1* positiv, wird die Nachkommastelle *s1*+1 gerundet. *numausdruck1* erhält *s1* Nachkommastellen.

Ist *s1* negativ, wird die Vorkommastelle -(*s1*) gerundet.

Dabei gilt: -126 ≤ *s1* ≤ 128

Wenn *wertefunktion* aus *literal* oder *variable* besteht, gilt:
s1 ≤ Nachkommastellen von *numausdruck1* oder
-(*s1*) ≤ Vorkommastellen von *numausdruck1*

Beispiel

```
ROUND (3469.87126, 0)   = 3470
ROUND (3469.87126, 4)   = 3469,8713
ROUND (3469.87126, -3)  = 3000
```

TRUNC

TRUNC schneidet numerische Werte an einer festgelegten Stelle ab.

Ist diese Stelle eine Nachkommastelle, wird der Wert nach der Nachkommastelle abgeschnitten.

Ist diese Stelle eine Vorkommastelle, werden diese und alle folgenden Vorkommastellen mit 0 belegt.

Hat eines der Argumente den NULL-Wert, dann liefert TRUNC den NULL-Wert.

numausdruck2 — siehe Metavariable *numausdruck*

s2 — *s2* legt die Position fest, nach der *numausdruck2* abgeschnitten wird. *s2* muß eine ganze Zahl sein.

Ohne Angabe von *s2* wird der ganzzahlige Anteil von *numausdruck2* geliefert.

Ist *s2* positiv, wird *numausdruck2* nach der Nachkommastelle *s2* abgeschnitten.

Ist *s2* negativ, werden in *numausdruck2* alle Stellen ab der Vorkommastelle -(*s2*) mit 0 belegt.

Dabei gilt: -126 ≤ *s2* ≤ 128

Wenn *wertefunktion* aus *literal* oder *variable* besteht gilt:
s2 ≤ Nachkommastellen von *numausdruck1* oder
-(*s2*) ≤ Vorkommastellen von *numausdruck1*

Beispiel

```
TRUNC (3469.87126, 0)   = 3469
TRUNC (3469.87126, 4)   = 3469,8712
TRUNC (3469.87126, -3)  = 3000
```

NUMERIC — Das Argument *charausdruck2* wird in ein Ergebnis vom Typ NUMERIC umgewandelt.

Hat eines der Argumente den NULL-Wert, dann liefert NUMERIC den NULL-Wert.

charausdruck2 — *charausdruck2* muß sich durch *charliteral* (siehe Metavariable *literal*) beschreiben lassen.

Wenn *charausdruck2* einen numerischen Wert repräsentiert, dann muß *charliteral1* numerische Steuerzeichen enthalten. Dies kann zum Ablaufzeitpunkt mit dem Prädikat IS NUMERIC (siehe Metavariable *bedingung*) festgestellt werden.

charliteral — *charliteral* muß die Bedingungen von *mask* erfüllen (siehe Metavariable *mask*). Der Inhalt besteht aus Maskensteuerzeichen, die zum Datentyp passen müssen.

Entfällt die Angabe von charliteral, dann wird eine Standardmaske verwendet (siehe Metavariable *mask*).

POSITION

POSITION liefert die Position von Zeichenketten in Zeichenketten.

Hat eines der Argumente den NULL-Wert, dann liefert POSITION den NULL-Wert.

Wenn der Wiederholungsfaktor *n2* angegeben ist, liefert POSITION die Position an der *charausdruck3* zum *n2*-ten Mal in *charausdruck4* vorkommt. Es muß gelten:
Länge (*charausdruck3*) * *n2* ≤ Länge (*charausdruck4*)

Wenn *n2* nicht angegeben ist, liefert POSITION den Wert der ersten Position, an der *charausdruck3* in *charausdruck4* vorkommt. Es muß gelten:
Länge (*charausdruck3*) ≤ Länge (*charausdruck4*)

Ist *charausdruck3* nicht in *charausdruck4* enthalten oder nicht mit dem angegebenen Wiederholungsfaktor, dann liefert POSTION den Wert "0".

charausdruck3, charausdruck4

Der Datentyp von *charausdruck* muß alphanumerisch sein.

n2

n2 gibt den Wiederholungsfaktor an. *n2* muß numerisch, ganzzahlig und positiv sein.

LENGTH

LENGTH liefert die Position in einer Zeichenkette *charausdruck5*, der nur Leerzeichen folgen. An der Position selbst steht kein Leerzeichen.

Besteht *charausdruck5* nur aus Leerzeichen, dann liefert LENGTH den Wert 0.

Hat *charausdruck5* den NULL-Wert, dann liefert LENGTH den NULL-Wert.

charausdruck5

Der Datentyp von *charausdruck5* muß alphanumerisch sein.

MODULO

MODULO liefert den Restwert der Division von *numausdruck3* durch *numausdruck4*.

Sind *numausdruck3* und/oder *numausdruck4* Dezimalzahlen, so gilt: Die Anzahl der Nachkommastellen des Restwerts ist so groß wie die größere Anzahl der Nachkommastellen von *numausdruck3* und *numausdruck4*

Es besteht folgender Zusammenhang zur Funktion TRUNC:

MODULO(a,b) = a - (TRUNC(a/b) * b)

Hat eines der Argumente den NULL-Wert, dann liefert MODULO den NULL-Wert.

numausdruck3, numausdruck4
siehe Metavariable *numausdruck*

Regeln

– Die Wertefunktionen SIN, COS, TAN, LN, LOG, EXP, SQR und SQRT werden mit einer Genauigkeit von 15 und mit 6 Nachkommastellen berechnet.

– Die Wertefunktion ABS wird mit einer Genauigkeit von 15, aber mit unveränderter Nachkommastellenanzahl berechnet.

wiederholungsgruppe
Wiederholungsgruppe definieren

wiederholungsgruppe legt für eine Variable den Datentyp "Wiederholungsgruppe" fest. *wiederholungsgruppe* besteht aus einer festen Anzahl von Komponenten, die alle denselben Datentyp besitzen. Die Anzahl der Komponenten wird durch den Wiederholungsfaktor *n* festgelegt.

wiederholungsgruppe darf nur in SQL-Anweisungen für UDS-Datenbanken angegeben werden.

Weitere Strukturtypen siehe Metavariable *strukturtyp*.

```
wiederholungsgruppe::=( n ) datengruppe
```

---

n	Wiederholungsfaktor (0 < *n* < 31000).
datengruppe	siehe Metavariable *datengruppe*

6 Meldungen

In diesem Kapitel sind alle DRIVE-Meldungen aufgeführt, die DRIVE/WINDOWS auf den verschiedenen Plattformen (BS2000, SINIX und MS-Windows) ausgibt. Sie sind durch einen eindeutigen Schlüssel gekennzeichnet.

DRI0008 BITTE ANWEISUNG EINGEBEN

Bedeutung
DRIVE befindet sich im Dialog-Modus und erwartet eine Anweisung.

DRI0009 ANWEISUNG AUSGEFUEHRT

Maßnahme
Naechste Anweisung eingeben.

DRI0010 SPEICHERENGPASS

Maßnahme
Speicher freigeben, z.B.
- EDT-Arbeitsdateien loeschen
- kleinere Programme aufrufen
- im Dialog-Modus alle Views freigeben
- groessere Anlage (XS) verwenden.

DRI0011 SYNTAXFEHLER

Bedeutung
Die angegebene Anweisung entspricht nicht den Syntaxregeln. Syntaxerlaeuterungen sind mittels HELP-Anweisung erhaeltlich. Moegliche Fehlerursachen:
- Verwendeter Name ist ein DRIVE-Schluesselwort.
- Syntaxelement ist an der markierten Stelle unzulaessig.
- Literal ist zu lang bzw. nicht korrekt abgeschlossen.

Maßnahme
Mittels HELP-Anweisung Syntax korrigieren. Anschliessend Anweisung wiederholen.

DRI0012 VARIABLE DARF NICHT INDIZIERT SEIN

DRI0013 SPEICHERBEDARF DER VARIABLEN / DES AUSDRUCKS ZU GROSS

Bedeutung
Der Speicherbedarf der Variablen bzw. des Ausdrucks uebersteigt die maximal zulaessige Groesse von ca. 31 KByte.

DRI0014 OBJEKT IST NICHT DEFINIERT

Bedeutung
Das angesprochene Objekt wurde DRIVE nicht durch eine Deklaration bekannt gemacht, bzw. ein Datenbankobjekt ist fuer eine andere Umgebung deklariert.

Maßnahme
Objekt deklarieren bzw. angegebenen Namen ueberpruefen.

DRI0015 UNZULAESSIGER DATENTYP

Bedeutung
Der Datentyp ist an dieser Stelle unzulaessig. Bei Variablen vom Typ CHARACTER kann auch die Laenge unzulaessig sein.

DRI0016 UNZULAESSIGER WERT

Bedeutung
Der angegebene Wert ist nicht im zulaessigen Wertebereich.

DRI0017 OBJEKT BEREITS VORHANDEN

Bedeutung
Der Name des Objekts wurde bereits verwendet.

Maßnahme
Anderen Namen verwenden.

DRI0018 ANGABE NUR AUF OBERSTER STUFE ZULAESSIG

DRI0019 KEIN SPEICHERPLATZ FUER '(&00)' VERFUEGBAR

Bedeutung
(&00): Name des Systems, bei dem der Speicherengpass aufgetreten ist. Name der Datei, bei der der Speicherengpass aufgetreten ist.

DRI0020 MATHEMATISCHE OPERATION BZW. FUNKTION UNZULAESSIG

Bedeutung
Die verwendete Funktion bzw. mathematische Operation ist in der aktuellen Anweisung unzulaessig (z.B. in DB-Anweisungen sind numerischen Praedikate oder der Wert $PI unzulaessig).

DRI0021 NUR EINFACHE FELDER ZULAESSIG

Bedeutung
Strukturierte Felder koennen nur mit dem Vergleichsoperator '=' verwendet werden. Andere Vergleichsoperatoren sind nur mit einfachen Feldern zulaessig.

Maßnahme
Vergleich auf der Ebene der einzelnen Komponenten vornehmen.

DRI0022 '(&00)' MELDUNG: (&01) ((&02))

Bedeutung
(&00): BS2000/DMS/EDT/FHS/LMS/TIAM/TOM-REF/UTM
(&01),(&02): Meldungsnummern:

LMS: (&01): LMS-Returncode; (&02): PLAM- und DMS-Returncodes.
EDT: (&02): EDT-Returncode.
UTM: (&01): KDCS-Fehlercode; (&02): Interner UTM-Fehlercode.
BS2000: (&01): INTTRACE; (&02): DVS-Returncode.
FHS: (&01): Main Returncode; (&02): Category, Reason.
TIAM: (&02): TIAM-Returncode.
TOM-REF: (&02): Ein-/Ausgabe Zustand (siehe COB1-Handbuch).

Maßnahme
Die Returncode-Informationen sind den Handbuechern der entsprechenden Systeme zu entnehmen bzw. koennen im Systemmodus ueber das BS2000-Kommando /HELP-MESS erfragt werden.

DRI0023 ANWEISUNG IST GESPERRT

Bedeutung
Dem Anwender wurde diese Anweisung gesperrt.

Maßnahme
Anweisung nicht verwenden oder Sperre durch Administrator aufheben lassen.

DRI0024 UNZULAESSIGE REIHENFOLGE DER ANWEISUNGEN

Bedeutung
Zulaessige Anweisungsreihenfolge:
- OPTION-Anweisung(en)
- PROCEDURE-Anweisung
  Deklarative Anweisungen
  Ausfuehrbare Anweisungen
- END PROCEDURE-Anweisung.

Im Report-Fall: Vor FILL REPORT oder CLOSE REPORT muss ein OPEN REPORT erfolgt sein.

Maßnahme
Reihenfolge der Anweisungen korrigieren.
Im Report-Fall: OPEN REPORT - Anweisung einbringen.

DRI0025 OBJEKT IST NICHT VORHANDEN

Bedeutung
Das Objekt (z.B Bibliothek, Element) ist nicht bzw. nicht in der geforderten Form (z.B. Bibliothek keine PLAM-Bibliothek) vorhanden.

Maßnahme
Objekt anlegen bzw. anderes existierendes Objekt waehlen. Falls es sich bei dem Objekt um eine Bibliothek handelt, kann moeglicherweise eine Datei dieses Namens existieren, die jedoch keine PLAM-Bibliothek ist.

DRI0026 OBJEKT IST GESPERRT

Bedeutung
Das Objekt (z.B. Element einer PLAM-Bibliothek) ist gesperrt.

Maßnahme
Warten bis Objekt frei ist oder anderes Objekt waehlen.

DRI0027 ANWEISUNG IM '(&00)'-MODUS UNZULAESSIG

Bedeutung
Anweisung darf im angegebenen Modus nicht verwendet werden.
(&00): UTM
TIAM
PROGRAMM
DIALOG
IDP
ENTER
DISPATCH
AUFTRAGNEHMER
ASYNCHRON

DRI0028 MAXIMAL DREI INDEXSTUFEN ZULAESSIG

DRI0029 NAME NICHT EINDEUTIG

Bedeutung
Der angegebene Name muss eindeutig qualifiziert sein.

DRI0030 '(&00)' ERWARTET

Bedeutung
An der bezeichneten Stelle fehlt das angegebene DRIVE-Syntaxelement.
(&00): DRIVE-Syntaxelement.

DRI0031 NAME ZU LANG

DRI0032 '(&00)' ENTHAELT (&01) FEHLER

Bedeutung
(&00): Programmname
(&01): Anzahl der Fehler.

Maßnahme
Quellprogramm verbessern und erneut analysieren lassen.

DRI0033 DYNAMISCHE ANWEISUNG UNZULAESSIG

Bedeutung
Die Anweisung ist in der EXEC-Anweisung nicht zulaessig.

DRI0034 UNZULAESSIGE GRUPPENDEFINITION

Bedeutung
Die Definition einer Variablengruppe ist unzulaessig.

DRI0035 KOMPONENTE NICHT IN GLEICHER GRUPPE

DRI0036 VARIABLE LIEGT IN EINER UEBERGEORDNETEN GRUPPE

Bedeutung
Bei der REDEFINES- oder LIKE-Angabe darf die Variable nicht selbst in einer uebergeordneten Gruppe liegen.

DRI0037 NUR ANGABE EINER GRUPPE ZULAESSIG

Bedeutung
- LIKE-Angabe muss sich auf eine Gruppe beziehen.
- .* -Angabe ist nur fuer Variable vom Typ Gruppe zulaessig.

DRI0038 UNZULAESSIGE INDEXANGABE(N)

Bedeutung
- CHECK-Bedingung: Es darf kein Index angegeben werden, da die Pruefbedingung fuer alle Auspraegungen eines Vektors oder einer Wiederholungsgruppe gilt.
- Index darf nur konstante Angaben enthalten.
- Indizierte Variable ist kein Vektor bzw. keine Wiederholungsgruppe.
- Index muss vom Typ numerisch mit Skalenfaktor 0 sein.

DRI0039 CHECK-BEDINGUNG NICHT ERFUELLT

Bedeutung
Die in der CHECK-Klausel angegebene Bedingung ist nicht erfuellt.

Maßnahme
Wert entsprechend der in der Definition der Variablen angegebenen Check-Bedingung abaendern.

DRI0040 UNZULAESSIGE VARIABLENANGABE

Bedeutung
- In der CHECK-Bedingung ist nur die definierte Variable selbst zulaessig.
- In DB-Anweisungen ist die Angabe von Variablen als Index unzulaessig.

DRI0041 VARIABLE BEREITS REDEFINIERT

Bedeutung
Bei der Redefinition einer Variablen wurde eine selbst schon redefinierte Variable angesprochen.

DRI0042 WEITERE KOMPONENTEN IN MIT 'LIKE' ERZEUGTER GRUPPE UNZULAESSIG

Bedeutung
Eine Variable oder eine Komponente, die mit LIKE erzeugt werden soll, darf darueberhinaus keine weiteren Komponenten haben.

DRI0043 '(&00)' IST LEER

Bedeutung
(&00):
- EDT-Arbeitsdatei 0: Keine Anweisung enthalten.
- Formatname: Ein dynamisches Format benutzt weder eine TTITLE- noch eine BTITLE-Definition; vor der DISPLAY-Anweisung wurde keine FILL-Anweisung fuer das Format angegeben.
- LIST-Dateiname: LIST-Datei ist leer.
- DATA DICTIONARY.

Maßnahme
- EDT-Arbeitsdatei fuellen.
- Format mittels FILL-Anweisung fuellen.
- LIST-Datei fuellen.
- DATA DICTIONARY neu einrichten bzw. fuellen.

DRI0044 ANWEISUNG '(&00)' IN PROGRAMMBLOCK (&01) UNZULAESSIG

Bedeutung
Die angegebene Anweisung ist im angegebenen Programmblock der aktuellen Prozedur nicht zugelassen.
(&00): ADD WINDOW/NEW WINDOW/NEXT WINDOW
(&01): BODY/SCRIPT-INIT/SCRIPT-ON

Maßnahme
Anweisung/-teil entfernen/aendern

DRI0045 '(&00)' MIT AKT. 'DRIVE'-VERSION UNVEREINBAR

Bedeutung
Das angegebene Programm kann mit der verwendeten DRIVE-Version nicht kompiliert oder ausgefuehrt werden.
(&00): Programmname.
(&00): Programmname mit Zusatz CODE: Zwischencode.

Maßnahme
Programm entsprechend der aktuellen DRIVE-Version umsetzen oder Zwischencode neu erzeugen;

DRI0046 '(&00)' UEBERSCHREIBEN? ANTWORT: (Y=JA; N=NEIN)

Bedeutung
(&00): Programmname.

Maßnahme
Y: EDT-Arbeitsdatei 0 bzw. Quellprogramm wird ueberschrieben.
N: EDT-Arbeitsdatei 0 bzw. Quellprogramm wird nicht ueberschrieben.

DRI0047 *** FEHLER IN ZEILE (&00) DES EXPANDIERTEN ELEMENTS

Bedeutung
Im expandierten COPY-Element ist in der angegebenen Zeile ein Fehler aufgetreten. Die genaue Fehlersituation wird in EDT-Arbeitsdatei 9 abgelegt.

DRI0048 MAXIMAL ZULAESSIGE ZEILENANZAHL ((&00)) UEBERSCHRITTEN

DRI0049 MAXIMALE ZEILENLAENGE ((&00)) UEBERSCHRITTEN

Bedeutung
(&00): Maximal zulaessige Anzahl von Zeichen pro Zeile, z.B.
- 256 in EDT-Arbeitsdateien
- 80 auf Datensichtstation
- entsprechend der Angabe bei der DECLARE FORM-Anweisung.

Maßnahme
- Zeile kuerzen.
- Format mit laengeren Zeilen definieren.

DRI0050 '(&00)'-NAMENSKONVENTIONEN VERLETZT

Bedeutung
Namenskonventionen des Subsystems bzw. von DRIVE sind nicht erfuellt.
(&00): DRIVE
EDT
PLAM.

Maßnahme
Namenskonventionen im entsprechenden Handbuch nachschlagen.

DRI0051 PARAMETER '(&00)' BEREITS VERSORGT

Bedeutung
Bereits versorgte Werte duerfen nicht mehr geaendert werden.

DRI0052 PARAMETER '(&00)' NICHT VERSORGT

Bedeutung
(&00): Nicht versorgter Parameter.

Maßnahme
Parameter mittels PARAMETER-Anweisung versorgen.

DRI0053 '(&00)' IST NICHT ZUGREIFBAR

Bedeutung
Es wird auf eine Bibliothek (LIBRARY, FORMLIB) zugegriffen, die unter einer anderen USER-ID steht und nicht fuer Fremdzugriffe eingerichtet wurde.

Maßnahme
Bibliothek fuer Fremdzugriffe einrichten oder andere Bibliothek verwenden.

DRI0054 '(&00)' IST GESPERRT

Bedeutung
(&00): Programmname ohne Zusatz: Quellprogramm
Programmname mit Zusatz CODE: Zwischencode
Programmname mit Zusatz LIST: Uebersetzungsliste
Dateiname: Datei
Auf das angegebene Objekt kann von DRIVE aus nicht zugegriffen werden, da es durch einen anderen Anwender gesperrt ist.
(&00): DATA DICTIONARY.
Wegen parallelem Aktualisierungs-Zugriff mehrerer Tasks ist momentan kein Zugriff moeglich.

Maßnahme
Freigabe des Objekts veranlassen.

DRI0055 '(&00)' IST NICHT VORHANDEN

Bedeutung
(&00): Modulname: Modul konnte nicht nachgeladen werden.
Bibliotheksname: Bibliothek nicht vorhanden, nicht mehrbenutzbar (shareable) oder kann auf fremder Kennung nicht angelegt werden.
Elementname ohne Zusatz: Quellprogramm
Erfolgt die Meldung bei UNSAVE elementname und ist das Element nicht vorhanden, so wurden trotzdem Zwischencode, Uebersetzungsliste und Verwendungsnachweis im DATA DICTIONARY geloescht.
Elementname mit Zusatz CODE: Zwischencode
Elementname mit Zusatz LIST: Uebersetzungsliste
Objektname: Objektcode
Es konnte kein Objekt mit zum Laufzeitsystem passender Version angebunden werden.
DATA DICTIONARY: DATA DICTIONARY nicht vorhanden oder PARAMETER DD nicht eingeschaltet.
zusaetzlich im SINIX-Betriebssystem:
Dateiverzeichnis: Teil des Pfadnamens nicht vorhanden
Datei: unter dem angegebenen Namen ist keine Datei vorhanden

Maßnahme
- FHS-Module muessen in der Bibliothek mit Linkname FORMOML enthalten sein.
- Bibliothek bzw. Bibliothekselement anlegen.
- Objekt mit der zur Version des Laufzeitsystems passenden Version des Compilers erzeugen und in die Objektbibliothek einfuegen.
- DATA DICTIONARY einrichten.

zusaetzlich fuer das SINIX-Betriebssystem:
- Pfadnamen ueberpruefen und korrigieren
- Dateinamen ueberpruefen und korrigieren

DRI0056 LAGE VON '(&00)' IM XS-ADRESSRAUM INKONSISTENT ZU ANDEREN MODULEN

Bedeutung
Beim Nachladen eines Moduls an einer XS-Anlage wird unterschiedlicher Adressraum angesprochen.

Maßnahme
Administrator verstaendigen.

DRI0057 ANWEISUNG WURDE AUF ZULAESSIGE LAENGE GEKUERZT

Bedeutung
- Die erste Anweisung im COPY-Element passt nicht auf den Bildschirm und wurde deshalb auf die maximal zulaessige Laenge gekuerzt.
- Die mit REPEAT geholte Anweisung ist zu lang und wurde gekuerzt.

Maßnahme
Anweisung kuerzen (z.B. Leerzeichen loeschen).

DRI0058 VIEW-DEKLARATION AUF EINEN VIEW UNZULAESSIG

Bedeutung
Bei der Deklaration eines Views darf in der FROM-Klausel kein View angegeben werden.

DRI0059 NUR EIN VIEW ODER NUR BASISTABELLEN ZULAESSIG

Bedeutung
In der FROM-Klausel darf genau ein View bzw. eine oder mehrere Basistabellen angegeben werden.

DRI0060 ANGEGEBENE ANWEISUNG NICHT VOLLSTAENDIG

DRI0061 '(&00)' '(&01)' UNZULAESSIG ODER NICHT GENERIERT

Bedeutung
(&00): TAC:
Der Generierungsfehler ist abhaengig vom angegebenen Transaktionscode (&01):

DRISQL:	Ist nur als FIRST-TAC zugelassen.
DRISQF/SQLNEXT:	Sind nur als NEXT-TACs zugelassen.
SQLENTER/SQLLIST:	Sind nur als Asynchron-TACs zugelassen.
SQLRMT/SQLRMTA/SQLRET:	TACs fuer verteilte Transaktinsverarbeitung

sonstiger Transaktionscode:
- der Transaktionscode ist nicht generiert
- bei Dialogaufruf ist der Transaktioncode kein Dialog-TAC
- bei Asynchronaufruf ist der Transaktionscode kein Asynchron-TAC

(&00): LTERM:
(&01): LTERM-Name.

Maßnahme
Anwendung mit KDCDEF neu generieren bei Angabe des richtigen TAC-Typs bzw. LTERM-Namens.

DRI0062 UNZULAESSIGE K- ODER F-TASTE BETAETIGT

Bedeutung
Die Eingabe erfolgte ueber eine K- oder F-Taste, die nicht mit PARAMETER KFKEY definiert wurde oder nicht per UTM SFUNC-Makro bekanntgegeben wurde.

Maßnahme
Eingabe mit anderer Taste vornehmen.

DRI0063 DOLINE=(&00); PROGRAMM FORTSETZEN? ANTWORT: (Y=JA; N=NEIN)

Bedeutung
Die mittels PARAMETER DIAGNOSIS DOLINE eingestellte Anzahl von Programmanweisungen wurde durchlaufen.
(&00): Aktueller Wert des Parameters DOLINE.

Maßnahme
Y: Die Programmausfuehrung wird mit der naechsten Anweisung fortgesetzt.
N: Die Programmausfuehrung wird abgebrochen.

DRI0064 '(&00)' MIT '(&01)' ABGEBROCHEN

Bedeutung
Die Programm-Ausfuehrung wurde abgebrochen wegen
- EXIT-Angabe,
- BREAK-Anweisung,
- erreichen des DOLINE-Wertes und Prompting-Eingabe 'Nein',
- Eingabe von BREAK am Terminal, K1-Taste etc,
- Rechenueberlaufs oder Divisionsfehler.

(&00): Programmname ohne Zusatz: Quellprogramm
Programmname mit Zusatz CODE: Zwischencode
(&01): EXIT/BREAK/PROGRAMMFEHLER.

DRI0065 SPEICHERENGPASS BEI ZWISCHENCODE-ABSPEICHERUNG

Bedeutung
Analyse des Programms fehlerfrei abgeschlossen; gemaess der Option-Klausel zu generierende Objekte (z.B. Uebersetzungsliste, Verwendungsnachweis) sind erzeugt. Jedoch ist beim Zugriff auf PLAM-X-Element fuer den Zwischencode ein Speicherengpass aufgetreten.

Maßnahme
Speicher freigeben und COMPILE-Anweisung mit OPTION CODE=ON wiederholen.

DRI0066 OBJEKT GESPERRT BEI ZWISCHENCODE-ABSPEICHERUNG

Bedeutung
Programmanalyse fehlerfrei abgeschlossen; gemaess der Option-Klausel zu generierende Objekte (z.B. Uebersetzungsliste, Verwendungsnachweis) sind erzeugt. Jedoch ist das PLAM-X-Element fuer die Abspeicherung des Zwischencodes gesperrt.

Maßnahme
Abwarten bis Objekt nicht mehr gesperrt ist oder anderes Objekt waehlen.
Anschliessend COMPILE-Anweisung mit OPTION CODE=ON wiederholen.

DRI0067 '(&00)' MELDUNG: (&01) ((&02)) BEI ZWISCHENCODE-ABSPEICHERUNG

Bedeutung
Analyse des Programms fehlerfrei abgeschlossen; gemaess der Option-Klausel zu generierende Objekte (z.B. Uebersetzungsliste, Verwendungsnachweis) sind erzeugt. Jedoch ist beim Zugriff auf das PLAM-X-Element fuer den Zwischencode ein Statusfehler aufgetreten.
(&00): PLAM
(&01): PLAM-Returncode
(&02): DVS-Returncode.

Maßnahme
Die Returncode-Informationen sind den Handbuechern der entsprechenden Systeme zu entnehmen bzw. koennen im Systemmodus ueber das BS2000-Kommando /HELP-MESS erfragt werden. Anschliessend COMPILE-Anweisung mit OPTION CODE=ON wiederholen.

DRI0068 '(&00)'-ANGABE FEHLERHAFT

Bedeutung
(&00): Fehlerhafte Angabe, z.B.
- STATUS, FILE, LTERM, DEVICE bei LIST-Anweisung,
- LIST bei DRIVE-Formatierung,
- SCHEMA, PASSWORD, USERGROUP, USERNAME bei PERMIT-Anweisung,
- ITEM-Angabe

Maßnahme
Bei PRESELECT ITEM auf Einfachauswahlfelder nur Ziffern als Auswahlkennzeichen angeben.

DRI0069 UNZULAESSIGES ODER FEHLERHAFTES 'SYSTEM'-KOMMANDO

DRI0070 PARAMETER-UEBERGABEBEREICH GROESSER ALS (&00) BYTES

Bedeutung
Die Summe (Wertebereich, Beschreibung) der in der USING-Klausel angegebenen Parameter ueberschreitet den maximal zulaessigen Bereich.
(&00): Maximal zulaessige Groesse fuer den Uebergabebereich.

Maßnahme
Weniger bzw. kuerzere Parameter uebergeben.

DRI0071 FEHLER IM IMPLIZITEN 'COPY'-ELEMENT

Bedeutung
Anweisung DECLARE SCREEN:
EUA-Adressierungshilfe ist fehlerhaft oder keine Adressierungshilfe.
Anweisung USE VIEWS:
Die eingelesene View-Deklaration ist fehlerhaft.

Maßnahme
Adressierungshilfe neu erzeugen bzw. View-Deklaration neu hinterlegen.

DRI0072 REKURSIVER '(&00)' AUFRUF UNZULAESSIG

Bedeutung
DRIVE erlaubt keine rekursiven Programm- bzw. Unterprogrammaufrufe.
(&00): SUBPROCEDURE
Programmname ohne Zusatz: Quellprogramm
Programmname mit Zusatz CODE: Zwischencode.

DRI0073 '(&00)' ENTHAELT EIN-/AUSGABEANWEISUNGEN

Bedeutung
Programm als ENTER-Programm nicht verwendbar, da Ein- bzw. Ausgabeanweisungen dort unzulaessig sind.
(&00): Programmname ohne Zusatz: Quellprogramm
Programmname mit Zusatz CODE: Zwischencode.

DRI0074 SYSTEMPROGRAMM '(&00)' NICHT VORHANDEN/FEHLERHAFT

Bedeutung
(&00): Name des Systemprogramms.

Maßnahme
Administrator verstaendigen.

DRI0075 SYSTEMBIBLIOTHEK '(&00)' NICHT VORHANDEN

Bedeutung
(&00): Name der Systembibliothek.

Maßnahme
Administrator verstaendigen.

DRI0076 KEINE METAINFORMATION GEFUNDEN

Bedeutung
Bei einer SHOW-Anweisung wurde keine Information ueber den angegebenen View, Cursor usw. gefunden.

DRI0077 KEINE 'DRIVE'-ANWEISUNG GEFUNDEN

Bedeutung
- Es kann nichts analysiert werden, da entweder die EDT-Arbeitsdatei 0 oder das PLAM-Element keine DRIVE-Anweisung enthaelt.
- Bei der REPEAT-Anweisung wurde keine zuvor abgespeicherte Anweisung gefunden.

DRI0078 INTERNER 'DRIVE'-FEHLER '(&00)' IN PROZEDUR '(&01)'

Bedeutung
Interne Inkonsistenzen erzwingen einen Abbruch von DRIVE. Gleichzeitig werden Diagnoseunterlagen in Form eines Dumps erstellt.
(&00): Interne Fehlernummer
(&01): Name einer internen Prozedur.

Maßnahme
Diagnoseunterlagen an den Administrator weiterleiten.

DRI0079 ANGABE NUR BEI EINFACHEN VARIABLEN ZULAESSIG

Bedeutung
Die Klausel an der markierten Stelle ist nur zulaessig, wenn es sich nicht um einen Ausdruck handelt oder wenn die Variable, auf die sie sich bezieht, keine Gruppe ist.

Maßnahme
Angesprochene Klausel weglassen oder keine strukturierte Variable verwenden.

DRI0080 MARKIERTE KLAUSEL WURDE IN DER ANWEISUNG BEREITS ANGEGEBEN

Maßnahme
Klausel nur einmal angeben.

DRI0081 ANGABE NUR BEI FORMATEN ZULAESSIG

Bedeutung
Die IMAGE- oder INIT-Klausel ist nur bei Formaten, nicht aber bei Listen oder in der SEND MESSAGE-Anweisung zulaessig.

Maßnahme
IMAGE- bzw. INIT-Klausel loeschen.

DRI0082 NUR 'FHS'-FORMATE ZULAESSIG

Bedeutung
In einer DISPLAY-Anweisung, die mehrere Formatnamen enthaelt, sind nur FHS-Formate zulaessig.

Maßnahme
Formatname in DISPLAY-Anweisung aendern oder loeschen.

DRI0083 '(&00)'-ANGABE UNZULAESSIG

Bedeutung
- NEWLINE und NEWPAGE sind in der SEND MESSAGE-Anweisung unzulaessig.
- Die Angabe TABLE ist in Verbindung mit NEWLINE bzw. NEWPAGE unzulaessig.
- Innerhalb der COMMIT- und der STOP-Anweisung darf eine DISPLAY-Anweisung nicht zum Ziel haben, auf ein Listenformat zu schreiben.
- RETURN-Angabe ist nur bei Formaten zulaessig, nicht aber bei Listen.
- Im Anschluss an die Angabe NAMES ist die RETURN-Angabe unzulaessig.
- NEWPAGE ist in der DECLARE FORM-Anweisung unzulaessig.
- Die DISPLAY-Anweisung darf nur dann eine SCREENERROR-Klausel enthalten, wenn die Anweisung sich auf ein FHS-Format bezieht.
- USING-Angabe ist unzulaessig, wenn im gerufenen Programm keine USING-Klausel vereinbart ist.
- Im Anschluss an die Angabe RETURN ist die Attributangabe INVISIBLE unzulaessig.
- TRACE bei DEBUG ist unzulaessig, wenn keine aktuelle Uebersetzungsliste vorhanden ist oder Uebersetzungsliste und Quellprogramm nicht uebereinstimmen.
  (&00): TRACE

Maßnahme
Unzulaessige Angabe loeschen.
Neue Uebersetzungsliste erzeugen.

DRI0084 ANGABE NUR IM TRANSAKTIONSLOSEN ZUSTAND MOEGLICH

Bedeutung
- Im Dialog-Modus darf kein Programm aufgerufen werden, solange noch eine Transaktion geoeffnet ist.
- Die Anweisung PARAMETER DYNAMIC NORMSQL ist nur im transaktionslosen Zustand zulaessig.
- Die STOP-Anweisung wird nicht ausgefuehrt.

Maßnahme
Transaktion abschliessen und Angabe anschliessend wiederholen.

DRI0085 VARIABLE '(&00)' ENTHAELT NULL-WERT

Bedeutung
- Zum Ausfuehrungszeitpunkt darf eine DRIVE-Variable, die an das DB-System ueber geben werden soll, keinen NULL-Wert enthalten.
- Der Variablen darf kein NULL-Wert zugewiesen werden.

(&00): Variablenname.

Maßnahme
Variable mit zulaessigem Wert belegen.

DRI0086 TRANSAKTION WURDE BEENDET; BITTE 'DUE' EINGEBEN

DRI0087 FEHLER: '(&00)' ABGEBROCHEN

Bedeutung
(&00): Programmname ohne Zusatz: Quellprogramm
Programmname mit Zusatz CODE: Zwischencode
- Bei der Ausfuehrung des ueber DO bzw. ENTER gestarteten Programms traten Ablauffehler auf. Bei ENTER wurde die zugehoerige detaillierte Fehlerliste auf SYSLST ausgegeben.

(&00): Name der geladenen DRIVE-Fassung.
- DRIVE wurde aufgrund eines Fehlers abgebrochen. Eine diesbezuegliche Fehlermeldung wird zusaetzlich ausgegeben bzw. Diagnoseunterlagen werden erstellt.

Maßnahme
- Betroffenes ENTER-Programm korrigieren und anschliessend neu starten.
- Evtl. Diagnoseunterlagen an Administrator weiterleiten.

DRI0088 '(&00)' ORDNUNGSGEMAESS BEENDET

Bedeutung
- Normale Beendigung von DRIVE.
  (&00): Programmname der geladenen DRIVE-Fassung.
- Normale Beendigung des Anwenderprogramms (&00).
  (&00): Programmname ohne Zusatz: Quellprogramm
  Programmname mit Zusatz CODE: Zwischencode.

DRI0089 *** SCHWERWIEGENDER FEHLER. PROGRAMMANALYSE ABGEBROCHEN

Bedeutung
Waehrend der Programmanalyse ist ein schwerwiegender Fehler aufgetreten. Der restliche Teil des Programms konnte nicht mehr analysiert werden.

DRI0090 ENGPASS IM KLASSE-5-SPEICHER BEI DVS-MAKROAUSFUEHRUNG AUFGETRETEN

Maßnahme
- Administrator verstaendigen.
- Klasse-5-Speicher vergroessern.

DRI0091 UTM MELDUNG BEI '(&00)'. KCRC (&01) ((&02))

Bedeutung
(&00): UTM-Funktion (z.B. INIT, MGET, ...)
(&01): KCRCCC = UTM-Fehlercode
(&02): KCRCDC = Interner UTM-Fehlercode.
Weitere Informationen siehe UTM-Handbuch.

Maßnahme
Fehler beheben, z.B. Angaben in der KDCDEF aendern und DRIVE neu starten.

DRI0092 ANWEISUNG IN DIESEM BETRIEBSSYSTEM NICHT ERLAUBT

Bedeutung
Es wurde im SINIX- bzw. BS2000-Betriebssystem eine Anweisung eingegeben,die nur im BS2000- bzw. SINIX-Betriebssystem erlaubt ist.

DRI0093 KONTROLLVARIABLE DARF NICHT GEAENDERT WERDEN

Bedeutung
Es wurde versucht den Wert der Kontrollvariablen einer CYCLE FOR-Anweisung waehrend eines Schleifendurchlaufs zu aendern.
Fuer die Report-Generierung: Es wurde versucht den Wert einer, einen Report definierenden Variable zwischen OPEN REPORT und CLOSE REPORT zu veraendern.

Maßnahme
In der beanstandeten Anweisung (z.B. SET) die Kontrollvariable durch eine andere Variable ersetzen.

DRI0094 DD GESPERRT; IDDS-STATUS: (&00), (&01)

Bedeutung
Auf das DD bzw. ein Entity oder eine Relationship kann nicht zugegriffen werden
Moegliche Ursachen:
- Satz aufgrund paralleler Transaktion gesperrt
- Dictionary wurde deaktiviert

Die laufende Transaktion wurde in der Regel zurueckgesetzt; im DRIVE-Manual nachlesen.
(&00): Fehlernummer;
(&01): Fehlertext

Maßnahme
Der IDDS-Code gibt Auskunft ueber die genaue Ursache der Sperre. Nach Freigabe des gesperrten Objektes DD-Auftrag erneut starten

DRI0095 BETRIEBSMITTELENGPASS IM DD, IDDS-STATUS: (&00),(&01)

Bedeutung
Im ERMS ist ein Betriebsmittelengpass aufgetreten. Moegliche Ursachen:
- keine Cursor-id verfuegbar
- kein Speicherplatz verfuegbar

Die laufende Transaktion wurde zurueckgesetzt.
(&00): Fehlernummer
(&01): Fehlertext

Maßnahme
Bedeutung des aufgetretenen IDDS-Codes im ERMS-Manual nachschlagen. Es kann sein, dass der ERMS-Administrator eingreifen muss.

DRI0096 FEHLER IN '(&00)' IN ZEILE (&01):

Bedeutung
Kopfzeile fuer Auflistung von Fehlern.
Bei der Abarbeitung eines WHENEVER-Ereignisses ist ein weiterer Fehler im Programm (&00), Zeile (&01) aufgetreten (siehe Uebersetzungsliste).
Auf MS-Windows im RTS-Betrieb kann auch eine fehlerhafte Anweisung in einer Startdatei zu dieser Fehlermeldung fuehren.
(&00): Programmname ohne Zusatz: Quellprogramm
Programmname mit Zusatz CODE: Zwischencode
Name der Startdatei auf MS-Windows
(&01): Zeilennummer in der Uebersetzungsliste
Zeilennummer in der Startdatei

Maßnahme
Fehlerdiagnose mittels angegebenem Programmnamen und angegebener Zeilennummer der Uebersetzungsliste.
Fuer MS-Windows evtl. fehlerhafte Anweisung in der Startdatei korrigieren.

DRI0097 FEHLER ANZEIGEN? ANTWORT: (EDT=ANZEIGEN; BREAK=ABBRECHEN)

Bedeutung
Nach Programmanalyse in der EDT-Arbeitsdatei 0 kann mit der EDT-Anweisung in den Editor verzweigt werden. Dort ist in der Arbeitsdatei 0 das analysierte Quellprogramm zusammen mit den eingestreuten Fehlermeldungen, in der Arbeitsdatei 9 die ausfuehrliche Uebersetzungsliste abgelegt.

DRI0098 FORMAT '(&00)' NOCH NICHT AUSGEGEBEN

Bedeutung
- Eine parallele Bearbeitung mehrerer LIST-Formate ist nicht moeglich. Ein mit FILL-Anweisungen gefuelltes LIST-Format muss mittels DISPLAY-Anweisung ausgegeben werden, bevor das naechste LIST-Format gefuellt werden kann.
- Vor Verlassen eines mit DO aufgerufenen Programmes muessen alle Bildschirm-

Ausgaben abgeschlossen sein.
- Vor Verlassen eines mit CALL aufgerufenen Programmes muessen alle darin deklarierten temporaeren Formate abgeschlossen sein.

(&00): Formatname.

Maßnahme
- Das zuvor bearbeitete LIST-Format mittels einer DISPLAY-Anweisung ausgeben, bzw.
- in der FILL-Anweisung den eingegebenen Formatnamen auf den aktuellen Formatna men abaendern.

DRI0099 '(&00)' ENTHAELT GESPERRTE ANWEISUNGEN

Bedeutung
Das aufgerufene Programm enthaelt Anweisungen, die benutzerbezogen per PARAME-TER-Anweisung gesperrt sind.
(&00): Programmname ohne Zusatz: Quellprogramm
Programmname mit Zusatz CODE: Zwischencode.

Maßnahme
Aenderung der gesperrten Anweisungen ist nur in einer neuen DRIVE-Sitzung moeglich.

DRI0100 '(&00)' NUR PER 'CALL' AUSFUEHRBAR

Bedeutung
Ein Programm ist nur per CALL ausfuehrbar, wenn es z.B. Ausgabeparameter enthaelt.
(&00): Programmname ohne Zusatz: Quellprogramm
Programmname mit Zusatz CODE: Zwischencode

DRI0101 OFFENE TRANSAKTION IN '(&00)' WURDE ZURUECKGESETZT

Bedeutung
Vor Verlassen eines mit DO aufgerufenen Programmes darf keine Transaktion mehr geoeff-net sein, d.h. vor den Anweisungen END PROCEDURE, BREAK PROCEDURE, STOP oder einer Folge-DO-Anweisung muss dynamisch eine COMMIT WORK- oder ROLLBACK WORK-Anweisung durchlaufen worden sein.
(&00): Programmname ohne Zusatz: Quellprogramm
Programmname mit Zusatz CODE: Zwischencode.

DRI0102 UNZULAESSIGE GELTUNGSDAUER ('TEMPORARY', 'PERMANENT')

Bedeutung
Die redefinierte und die zu redefinierenden Variablen haben unterschiedliche Angaben: TEMPORARY oder PERMANENT.

Maßnahme
Beide Variablen entweder mit PERMANENT oder mit TEMPORARY deklarieren.

DRI0103 BEREICHSANGABEN NUR BEI DER LETZTEN KOMPONENTE ZULAESSIG

DRI0104 REDEFINIEREN DER 'LIKE'-KOMPONENTE UNZULAESSIG

Bedeutung
Die Gruppe enthaelt Komponenten, die wiederum andere Komponenten redefinieren.

Maßnahme
Die Variable ohne LIKE deklarieren oder die REDEFINES-Angaben aus der LIKE-Gruppe entfernen.

DRI0105 INDEXANGABE ERFORDERLICH

Bedeutung
Indexangabe ist erforderlich, da die Variable ein Vektor ist.

DRI0106 OPTION '(&00)' NUR BEI ANGABE EINES ELEMENTNAMENS ZULAESSIG

Bedeutung
Da kein Elementname bekannt ist, kann die angegebene Option in der COMPILE-Anweisung nicht erfuellt werden.
(&00): LISTING
CODE

Maßnahme
Elementname angeben oder Option weglassen.

DRI0107 '(&00)' BEREITS ALS EINGABEFELD VERWENDET

Bedeutung
In FILL-Anweisungen, die sich auf das gleiche DRIVE-Format beziehen, wird ein Eingabefeld mehrfach angegeben.
(&00): Variablenname.

DRI0108 DRIVE-MELDUNG '(&00)' FEHLERHAFT AUFGEBAUT

Bedeutung
Die in der Meldungsdatei enthaltene Meldung entspricht nicht den Konventionen, die fuer Meldungen mit Antwort gelten.
(&00): DRIVE-Meldungsnummer.

Maßnahme
DRIVE-Meldung den Konventionen entsprechend abaendern.

DRI0109 GESAMTLAENGE DER EINGABEDATEN ZU KLEIN

Bedeutung
Beim Einlesen von Werten in ein DRIVE-Format wurde
- die Endemarke in ein Eingabefeld gesetzt bzw.
- bei Verarbeitung mit SYSDTA-Datei ein zu kurzer Datensatz gelesen.

Maßnahme
- Endemarke entfernen bzw.
- Satz in der SYSDTA-Datei entsprechend der erwarteten Eingabelaenge vergroessern.

DRI0110 GESAMTLAENGE DER EINGABEDATEN ZU GROSS

Bedeutung
Beim Einlesen von Werten ueber SYSDTA-Datei in ein DRIVE-Format wurde ein zu langer Datensatz gelesen.

Maßnahme
Satz in der SYSDTA-Datei entsprechend der erwarteten Eingabelaenge verkuerzen.

DRI0111 '(&00)' HAT UNZULAESSIGE DATEIEIGENSCHAFTEN

Bedeutung
(&00): Linkname (DRILIST, INTTRACE).
Folgende Dateieigenschaften sind erforderlich:

Eigenschaft	DRILIST	INTTRACE
FCBTYPE	ISAM	ISAM
RECFORM	V	V
BLKSIZE	(STD,b) b<=16	(STD,16)
KEYPOS	5	5
KEYLEN	24	32
OPEN	INOUT	INOUT
SHAREUPD	YES	YES
SPACE	(b*2+1,b)	(33,16)

Maßnahme
Fehlerhafte Datei loeschen. DRIVE legt bei Bedarf die INTTRACE-Datei automatisch neu an.

DRI0112 EXITROUTINE KANN NICHT AUSGEFUEHRT WERDEN

Bedeutung
Exitroutine wurde in der F.EXITLIB-Bibliothek nicht gefunden.

DRI0113 ANGABE VON 'FHS'-FORMATEN UNZULAESSIG

Bedeutung
Die FILL-Anweisung ist bei FHS-Formaten unzulaessig.

DRI0114 '(&00)' IST NICHT EROEFFNET ODER GENERIERT

Bedeutung
Ein bei ADD WINDOW spezifiziertes PARENT-Window muss generiert und eroeffnet sein
(&00): Windowname

DRI0115 WINDOW '(&00)' IST NICHT DEFINIERT

Bedeutung
Ein ueber ADD/NEW/NEXT WINDOW zu oeffnendes Window existiert nicht

DRI0116 '(&00)' IM '(&01)'-MODUS NICHT AUSFUEHRBAR/UEBERSETZBAR

Bedeutung
Programm im angegebenen Terminalbetrieb nicht ausfuehrbar/uebersetzbar.
(&00): Programmname
(&01): 'WINDOW' oder 'ALPHA'

DRI0117 WINDOW '(&00)' BEREITS GEOEFFNET

Bedeutung
Ein ueber ADD/NEW/NEXT WINDOW zu oeffnendes Window ist bereits geoeffnet
(&00): Windowname

DRI0118 '(&00)' AUF BILDSCHIRM IM 'UTM-ASYNCHRON-BETRIEB' UNZULAESSIG

Bedeutung
Im UTM-ASYNCHRON-BETRIEB sind vom Anwender veranlasste PTRACE-Ausgaben auf den Bildschirm bzw. DOLINE-PROMPTING bei Erreichen des DOLINE-Werts unzulaessig.
(&00): PTRACE
DOLINE-PROMPTING

DRI0119 ZUGRIFFSRECHT AUF ANGESPROCHENE TABELLE NICHT VORHANDEN

Bedeutung
Im Dialog-Modus unter UTM muss, falls PARAMETER PERMISSION=ON gesetzt ist, das Zugriffsrecht auf die verwendeten Tabellen im DATA DICTIONARY hinterlegt sein. Falls das Zugriffsrecht (z.B. INSERT) fuer die verwendete Anweisung nicht ausreicht, wird diese abgewiesen.

Maßnahme
Ggf. Zugriffsrechte neu vergeben.

DRI0120 'OF'-TYP MIT 'CASE'-TYP NICHT VERTRAEGLICH

Bedeutung
Bei 'of' wurde 'value-expression' und bei 'case' 'search-condition' angegeben oder umgekehrt

DRI0121 ANGESPROCHENES OBJEKT IST NICHT DYNAMISCH

Bedeutung
DROP-Anweisung darf sich nur auf dynamisch erzeugte (EXECUTE-Anweisung) Objekte (View, Cursor) beziehen.

DRI0122 GESCHACHTELTE 'COPY'-AUFRUFE UNZULAESSIG

Bedeutung
In einem COPY-Element ist eine COPY-Anweisung unzulaessig.
Ausnahme: Die Anweisungen DECLARE SCREEN und USE VIEWS werden DRIVE-intern mittels COPY-Anweisungen realisiert und sind auch in COPY-Elementen zulaessig.

Maßnahme
Schachtelungshierarchie aufloesen.

DRI0123 VERSION UEBERSETZUNGSLISTE / QUELLPROGRAMM STIMMEN NICHT UEBEREIN

Bedeutung
PTRACE ist nicht ausfuehrbar, da das Datum in der Uebersetzungsliste nicht mit dem im aktuellen Programm uebereinstimmt.

Maßnahme
Neue Uebersetzungsliste erzeugen.

DRI0124 ANWEISUNG IN DER UEBERSETZUNGSLISTE NICHT VORHANDEN

Bedeutung
Die Anweisung wurde nicht in die Uebersetzungsliste uebernommen, z.B. weil die Anweisung im Quellprogramm erst in Spalte 256 beginnt. Daher kann die Anweisung auch nicht mittels PTRACE ausgegeben werden.

Maßnahme
Quellprogramm entsprechend aendern.

DRI0125 AKTUELLER 'SPAB'-BEREICH (=(&00)) ZU KLEIN

Bedeutung
Der SPAB-Bereich wurde zu klein generiert. Die ENTER-Anweisung kann deshalb nicht ausgefuehrt werden.

Maßnahme
SPAB-Bereich vergroessern.

DRI0126 'DRILOG' ABGEBROCHEN, 'LOG'-FUNKTION ZURUECKGESETZT

Bedeutung
Wegen eines ITC-Fehlers konnte der DRILOG-Auftrag nicht ausgefuehrt werden.

Maßnahme
Mit Administrator abklaeren, welches Problem bei der ENTER-Prozedur DRI.ENT.DRILOG aufgetreten ist. Entsprechenden Fehler beheben und im Anschluss daran PARAMETER-Anweisung zum Starten von DRILOG wiederholen (PARAMETER DYNAMIC LOG).

DRI0127 'DRILOG' NICHT GELADEN, 'LOG'-FUNKTION NICHT AUSFUEHRBAR

Bedeutung
Die ENTER-Prozedur DRI.ENT.DRILOG konnte nicht gestartet werden.

Maßnahme
- Ueberpruefen, ob die ENTER-Prozedur DRI.ENT.DRILOG und das Programm PRO.DRILOG in der Benutzerkennung vorhanden sind.
- Ueberpruefen, ob Batch wegen BS2000-Systemueberlastung nicht gestartet wurde. PARAMETER-Anweisung zum Starten von DRILOG wiederholen.

DRI0128 EDT-ARBEITSDATEI 0 NICHT LEER. 'DRIVE' BEENDEN? (Y=JA; N=NEIN)

Bedeutung
Zum STOP-Zeitpunkt befindet sich noch ein ungesicherter Inhalt in EDT-Arbeitsdatei 0.

Maßnahme
Y: DRIVE wird beendet; der ungesicherte Inhalt geht verloren.
N: DRIVE wird nicht beendet. SAVE-Anweisung kann eingegeben werden; im Anschluss daran ist erneut die STOP-Anweisung einzugeben.

DRI0129 MAXIMAL (&00) 'FHS'-FORMATE ZULAESSIG

Bedeutung
In der DISPLAY-Anweisung darf pro Bildschirmzeile maximal ein FHS-Format angegeben werden.
(&00): Maximale Anzahl zulaessiger Formate in einer DISPLAY-Anweisung.

Maßnahme
Anzahl der verwendeten Formate verringern.

DRI0130 FOLGENDE FEHLERMELDUNGEN KONNTEN NICHT ZUGEORDNET WERDEN:

Bedeutung
Da die Uebersetzungsliste in EDT-Arbeitsdatei 9 nicht vollstaendig eingelesen wurde, konnten die nachfolgenden Fehlermeldungen den fehlerhaften DRIVE-Anweisungen nicht korrekt zugeordnet werden (EDT-Arbeitsdatei 0).

Maßnahme
Programm kuerzen.

DRI0131 ANGEGEBENE BIBLIOTHEK UND 'OLD STYLE' PLAM-BIBLIOTHEK VERSCHIEDEN

Bedeutung
Im Mischbetrieb wurde versucht, ein OLD STYLE-Programm zu starten. Die Bibliothek, aus der dieses Programm gestartet werden soll, stimmt aber nicht mit der bereits zuvor definierten PLAM-Bibliothek des OLD STYLE Betriebs ueberein. Ein Start ist daher nicht moeglich, da im OLD STYLE Betrieb nur die Definition von genau einer PLAM-Bibliothek zulaessig ist.

Maßnahme
Bibliotheksangabe aendern und Programm neu starten.

DRI0132 VARIABLE IST NICHT TEIL EINER 'ADRESSIERUNGSHILFE'

Bedeutung
Die Variable muss implizit durch eine DECLARE SCREEN-Anweisung deklariert werden. Nur dann ist die ATTRIBUTE-Klausel zulaessig.

Maßnahme
Neue ADRESSIERUNGSHILFE mit IFG generieren oder ggf. korrigieren.

DRI0133 NUR EIN ATTRIBUT PRO ATTRIBUT-KLASSE ZULAESSIG

Bedeutung
In einer ATTRIBUTE-Klausel darf aus einer Attribut-Klasse maximal ein Attribut spezifiziert werden (z.B. ist es unzulaessig, gleichzeitig zwei Farben anzugeben).

DRI0134 'DEFAULT' MUSS ERSTES ATTRIBUT SEIN

Bedeutung
Bei Angabe von mehreren globalen Attributen muss DEFAULT an erster Stelle stehen.

DRI0135 NAME DER 'LIST'-DATEI WURDE VOR WIEDERANLAUF GEAENDERT

Bedeutung
Vor dem Absturz von DRIVE wurde bereits mit einer LIST-Datei gearbeitet, die einen anderen Namen als die beim Wiederanlauf angegebene Datei hatte.

Maßnahme
Alte LIST-Datei zuweisen.

DRI0136 FALSCHE VERSION DES 'TOM-REF'-MODULS

Bedeutung
Beim Nachladen von TOM-REF (DATA DICTIONARY-Anschluss soll erzeugt werden) wurde eine falsche Version festgestellt.

Maßnahme
Administrator verstaendigen.

DRI0137 'INDEX ERROR': '(&00)' NICHT IM INDEXBEREICH VON '(&01)'

Bedeutung
Der Wert der Indexvariable liegt nicht im Indexbereich der Variable (&01).
(&00): Indexvariablenname.
(&01): Variablenname.

Maßnahme
Wert der Indexvariable korrigieren (z.B. mittels Anweisung SET).

DRI0138 '(&00)' BEI REDEFINIERTER VARIABLE '(&01)'

Bedeutung
Der Wert der redefinierten Variable (&01) ist nur im Bezug auf eine Redefinition von (&01) korrekt.
(&00): CONVERSION ERROR:
Der Datenwert ist nicht konsistent mit dem Datentyp von (&01).
CHECK ERROR:
Der Datenwert von (&01) ist zwar konsistent, verletzt aber die CHECK-Klausel von (&01).

Maßnahme
Vor der beanstandeten Referenzierung von (&01) den Wert ueber eine SET-Anweisung korrigieren oder statt (&01) die redefinierte Variable verwenden.

DRI0139 DIMENSIONEN NICHT VERTRAEGLICH

Bedeutung
Bei Vektorarithmetik gilt: fuer +,- muessen die Dimensionen gleich sein; fuer * muss ein Faktor ein Skalar sein; fuer /,% muss der Divisor bzw. Prozentfaktor ein Skalar sein;

DRI0140 (&00). PARAMETER ENTHAELT NULL-WERT

Bedeutung
(&00): Position des fehlerhaften Aktualparameters in der Parameterleiste.
Im Mischbetrieb darf ein Parameter nicht den NULL-Wert annehmen.

DRI0141 SCHEMA NICHT BEKANNT

Maßnahme
Schemaname angeben oder mittels PARAMETER-Anweisung vorgeben.

DRI0142 GANZZAHLIGER AUSDRUCK ERWARTET

Bedeutung
Ausdruck darf z.B. keine Variablen vom Typ DECIMAL, NUMERIC oder numerische Literale mit Dezimalzeichen beinhalten.

DRI0143 PARAMETER MUSS AUS GENAU EINER VARIABLEN BESTEHEN

Bedeutung
Als Parameter ist an dieser Stelle nur eine Variable bzw. eine Variablenkomponente zulaessig (kein Literal, Ausdruck etc.).

DRI0144 (&00). PARAMETER NICHT ZUWEISUNGSVERTRAEGLICH

Bedeutung
Der Datentyp des angegebenen Aktualparameters ist nicht auf den Formalparameter zuweisbar.
(&00): Position des fehlerhaften Aktualparameters in der Parameterliste.

Maßnahme
Aktualparameter an formale Schnittstellen-Definition anpassen.

DRI0145 'RETURN'-ANGABE BEI (&00). PARAMETER UNZULAESSIG

Bedeutung
Der angegebene Aktualparameter darf nicht mit RETURN spezifiziert werden, da der entsprechende Formalparameter keine RETURN-Klausel besitzt.
(&00): Position des fehlerhaften Aktualparameters in der Parameterliste.

Maßnahme
Spezifikation des Aktualparameters an formale Schnittstellen-Definition anpassen.

DRI0146 '(&00)'-ANGABE BEI '(&01)'. PARAMETER FEHLT

Bedeutung
- Der Parameter muss mit RETURN spezifiziert werden, wenn der dazugehoerige Formalparameter eine RETURN-Klausel besitzt.
- Der Parameter muss mit INDICATOR spezifiziert werden, wenn ein NULL-Wert uebergeben werden soll.

(&00): INDICATOR
RETURN
(&01): Position des fehlerhaften Aktualparameters in der Parameterliste.

Maßnahme
- Spezifikation des Aktualparameters an die formale Schnittstellen-Definition anpassen. Schnittstellen-Definition anpassen.
- INDICATOR-Klausel einfuegen bzw. keinen NULL-Wert uebergeben.

DRI0147 ANZAHL DER AKTUAL- UND FORMALPARAMETER STIMMT NICHT UEBEREIN

Maßnahme
Spezifikation der Aktualparameter an die formale Schnittstellen-Definition anpassen.

DRI0148 '(&00)' MEHRFACH MIT 'RETURN' SPEZIFIZIERT

Bedeutung
Ein(e) mit RETURN spezifizierte(r) Parameter bzw. Parameterkomponente ist maximal einmal in einer USING-Klausel zulaessig.
(&00): Name des Aktualparameters.

Maßnahme
Die Parameter-Spezifikation dementsprechend ueberpruefen.

DRI0149 ANWEISUNG ABGEBROCHEN; (BREAK=BILDSCHIRM LOESCHEN)

Bedeutung
Programmanalyse bzw. Programmablauf war fehlerhaft.

Maßnahme
Vorgeschlagene BREAK-Anweisung mit DUE quittieren. Der Bildschirm wird geloescht und es koennen neue Anweisungen eingegeben werden.

DRI0150 NICHT ALLE DEKLARIERTEN DB-OBJEKTE KONNTEN FREIGEGEBEN WERDEN

Bedeutung
Die durch die Programmanalyse beim DB-System deklarierten Objekte konnten nicht alle freigegeben werden. Die Deklarationen sind noch gueltig. Der ggf. erstellte Zwischencode bleibt bestehen.

DRI0151 ANWEISUNG AUSGEFUEHRT; (BREAK=BILDSCHIRM LOESCHEN)

Maßnahme
Vorgeschlagene BREAK-Anweisung mit DUE quittieren. Der Bildschirm wird geloescht und es koennen neue Anweisungen eingegeben werden.

DRI0152 BITTE TRANSAKTIONSCODE EINGEBEN (&00)

Bedeutung
Der DRIVE-Vorgang wurde beendet. Im mit Leerzeichen vorbelegten Eingabefeld (&00) kann der Anwender den naechsten Transaktionscode eingeben.

DRI0153 FEHLER BEIM NACHLADEN DER 'OLD STYLE'-MODULE

Bedeutung
Moegliche Ursachen:
1. OLD STYLE-Betrieb ist nicht installiert, d.h. es ist kein Mischbetrieb moeglich.
2. OLD STYLE-Modulbibliothek wurde nicht zugewiesen.

Maßnahme
Zu 2: Modulbibliothek korrekt zuweisen und Mischbetrieb neu starten.

DRI0154 'DRIVE'-SYSTEMGRENZE ERREICHT ((&00);'(&01)')

Bedeutung
Moegliche Ursachen sind:
- Eine interne Tabelle wurde zu gross, z.B. bei
  - Deklaration einer sehr grossen Variablen,
  - Deklaration eines FHS-Formats mit mehr als 34 Ein-/Ausgabefeldern.
- Eine Anweisung ist zu tief geschachtelt bzw. zu komplex.
- Eine DB-Anweisung ist zu lang fuer die Schnittstelle zum DB-System.

(&00): Interne Fehlernummer, von interner Prozedur (&01) gemeldet.

Maßnahme
- Anweisung kuerzen bzw. vereinfachen.
- Weniger Ein- bzw. Ausgabefelder definieren.
- Administrator verstaendigen.

DRI0155 MISCHBETRIEB NUR MIT DB-SYSTEM 'SESAM' ZULAESSIG

Bedeutung
Mischbetrieb ist nur moeglich, wenn DRIVE zusammen mit dem DB-System SESAM laeuft.

DRI0156 MAXIMAL 128 PARAMETER ZULAESSIG

Bedeutung
Im Mischbetrieb koennen beim Aufruf eines OLD STYLE-Programms maximal 128 Parameter uebergeben werden.

DRI0157 ERMS-SESSION NICHT MOEGLICH; IDDS-STATUS: (&00),(&01)

Bedeutung
Die ERMS-Session konnte nicht geoeffnet werden aufgrund einer der folgenden Ursachen:
- falsche Eintraege in Security-Partition
- Subschema fuer DRIVE nicht installiert
- das ueber die Umgebungsvariable $DRIVE_DD oder Defaulteinstellung angegeben Dictionary ist nicht vorhanden oder gesperrt.
- Betriebsmittelengpass oder Fehler bei ERMS

(&00): Fehlernummer
(&01): Fehlertext

Maßnahme
Bedeutung des aufgetretenen IDDS-Codes im ERMS-Manual nachschlagen.
Sicherstellen, dass das richtige Dictionary ueber die Umgebungsvariable $DRIVE_DD eingestellt wird. Bei Installationsfehlern muss der ERMS- Administrator eingeschaltet werden

DRI0158 INKONSISTENZ IM DD, IDDS-STATUS: (&00),(&01)

Bedeutung
Bei einem DD-Zugriff trat der angezeigte IDDS-Code auf. Dieser hat die Bedeutung, dass im DD eine Inkonsistenz festgestellt wurde. Die laufende Transaktion wurde zurueckgesetzt. Es ist wahrscheinlich keine sinnvolle Weiterarbeit mit dem DD mehr moeglich.
(&00): Fehlernummer
(&01): Fehlertext

Maßnahme
Im ERMS-Manual genaue Bedeutung des angegebenen IDDS-Codes nachschlagen.
Wahrscheinlich muss der ERMS-Administrator fuer die Beseitigung der Inkonsistenz eingeschaltet werden.

DRI0159 ERMS-INSTALLATIONSFEHLER, IDDS-STATUS: (&00),(&01)

Bedeutung
Dieser Fehler ist waehrend einer geoeffneten Session aufgetreten. Moegliche Ursachen:
- Schema nicht korrekt installiert
- Subschema nicht korrekt installiert

Die laufende Transaktion wurde zurueckgesetzt.
(&00): Fehlernummer
(&01): Fehlertext

Maßnahme
ERMS-Administrator verstaendigen. Feststellen, ob Installationsfehler oder Fehler in den ausgelieferten Command-Files vorliegt.

DRI0160 ZIFFER- ODER 'X'-STEUERZEICHEN ENTSPRECHEN NICHT DER FELDDEFINITION

Bedeutung
In einer Maske ist nicht die korrekte Anzahl bzw. Reihenfolge an Ziffer-Steuerzeichen ('Z','*','S','9'), 'P'-Steuerzeichen (fuer numerische Datentypen) oder an 'X'-Steuerzeichen (fuer Datentyp CHARACTER) vergeben. Die Anzahl dieser Steuerzeichen muss der Laenge der Variablen entsprechen, fuer die die Maske angegeben wurde.

Maßnahme
Anzahl bzw. Reihenfolge der Maskensteuerzeichen anhand der Variablendefinition ueberpruefen.

DRI0161 (&00) UND (&01) IN EINER MASKE NICHT GLEICHZEITIG ZULAESSIG

Bedeutung
(&00),(&01): Steuerzeichen.

DRI0162 (&00) NUR ALS ERSTES STEUERZEICHEN IN DER MASKE ZULAESSIG

Bedeutung
(&00): Steuerzeichen.

Maßnahme
Maske auf falsch vorgegebenes '+' oder '-' ueberpruefen.

DRI0163 STEUERZEICHEN MEHRFACH VERWENDET

Bedeutung
Die Steuerzeichen 'P', '+', '-' sowie DATE-/TIME-Steuerzeichen duerfen jeweils maximal einmal in einer Maske verwendet werden.

Maßnahme
Steuerzeichen in der Maske ueberpruefen.

DRI0164 STEUERZEICHEN (&00) UNZULAESSIG

Bedeutung
In einer Maske ist ein unzulaessiges Zeichen bzw. Steuerzeichen angegeben worden.
(&00): Steuerzeichen.

Maßnahme
Maske auf unzulaessige Zeichen bzw. Steuerzeichen ueberpruefen.

DRI0165 ZULAESSIGE LAENGE ((&00)) FUER AUFBEREITETE MASKE UEBERSCHRITTEN

Bedeutung
(&00): Maximal zulaessige Laenge einer aufbereiteten Maske.

DRI0166 (&00) NACH ANGABE VON (&01) UNZULAESSIG

Bedeutung
1. In einer Maske ist Steuerzeichen 'Z' bzw. '*' nur links von '9' zulaessig.
2. In einer Maske ist Steuerzeichen 'Z' bzw. '*' nur rechts von 'P' zulaessig, wenn alle Steuerzeichen bis auf 'P' und Einfuegesteuerzeichen gleich 'Z' bzw. '*' sind.
3. In einer Maske sind rechts von Dezimalzeichen 'P' keine Einfuegesteuerzeichen ('.' bzw. 'B') zulaessig.

(&00), (&01): Steuerzeichen.

Maßnahme
Zu 1 und 2: 'Z' bzw. '*' durch ein anderes Ziffersteuerzeichen ersetzen.
Zu 3: Loeschen der Einfuegesteuerzeichen rechts von 'P'.

DRI0167 WIEDERHOLUNGSFAKTOR UNZULAESSIG ODER NICHT KORREKT

Bedeutung
In einer MASK-Klausel wurde fuer ein Steuerzeichen entweder ein Wiederholungsfaktor vergeben, obwohl dies unzulaessig ist (z.B. +(4)), oder die Angabe ist nicht korrekt (z.B. 9(0)).

Maßnahme
Wiederholungsfaktor korrigieren oder loeschen.

DRI0168 EINGABE DES NULL-WERTS MITTELS NIL-ZEICHEN UNZULAESSIG

Maßnahme
NULL-Wertzeichen mittels PARAMETER-Anweisung aendern und entsprechend das neue Zeichen als NULL-Wert eingeben.

DRI0169 TEXTDATEI FUER MONINFO FEHLERHAFT

Bedeutung
In einer Zeile des Listen-Layouts der Moninfo fehlt das erste Feldzeichen ('@') der zwei Feldzeichen, die den Identifikator einschliessen.

Maßnahme
Das Listen-Layout muss geaendert werden, d. h. das fehlende Feldzeichen ('@') muss nachgetragen werden.

DRI0170 ZEICHEN '(&00)' UNZULAESSIG; EINGABE KORRIGIEREN

Bedeutung
Bei einer Dateneingabe wurde ein unzulaessiges Zeichen (z.B. falsches Dezimalzeichen) eingegeben.
(&00): Unzulaessiges Zeichen.

DRI0171 EINGABEWERT NICHT MIT VERBALER EINGABE VERTRAEGLICH

Bedeutung
Eingabewert einer DATE- oder TIME-Eingabe (z.B. '01' fuer Monat) ist mit verbaler Eingabe nicht vertraeglich (z.B. 'FEBRUAR').

Maßnahme
Eingabewert und verbale Eingabe aufeinander abstimmen.

DRI0172 DATENEINGABE NICHT EINDEUTIG; EINGABE KORRIGIEREN

Bedeutung
In der Maske sind zu wenige Q- oder R-Steuerzeichen definiert, so dass die Eingabe nicht eindeutig interpretiert werden kann, z.B. R(2) und Eingabe 'JU' ('JU' kann als 'JUNI' oder 'JULI' interpretiert werden).

DRI0173 MASK-KLAUSEL UNZULAESSIG

Bedeutung
Die MASK-Klausel wurde fuer eine Variable angegeben, fuer die sie nicht zulaessig ist. Moegliche Gruende:
- Es handelt sich nicht um eine einfache Variable
- Variable ist von einem unzulaessigen Datentyp

Maßnahme
Ueberpruefen der Variablen und ggfs. Streichen der MASK-Klausel

DRI0174 OBJEKT '(&00)' BEREITS VORHANDEN

Bedeutung
(&00): Name des Objekts.

DRI0175 '(&00)' WURDE ABGESPEICHERT

Bedeutung
Das Programm wurde erfolgreich als Element in der Bibliothek abgespeichert.
(&00): Programmname.

DRI0176 '(&00)' WURDE NICHT ABGESPEICHERT

Bedeutung
Das Programm wurde nicht in der Bibliothek als Element abgespeichert.
(&00): Programmname.

DRI0177 '(&00)' IST NICHT KONSISTENT

Bedeutung
Der Zwischencode des Programms ist ausserhalb von DRIVE veraendert (z.B. gekuerzt) worden.
(&00): Programmname ohne Zusatz: Quellprogramm
Programmname mit Zusatz CODE: Zwischencode.

Maßnahme
Neuen Zwischencode mittels COMPILE-Anweisung erzeugen.

DRI0178 '(&00)' ENTHAELT UNZULAESSIGE ANWEISUNGEN FUER '(&01)'-MODUS

Bedeutung
Das Programm enthaelt Anweisungen bzw. Anweisungsklauseln, die im UTM-Modus nicht zulaessig sind (z.B. SYSTEM-Anweisung).
(&00): Programmname ohne Zusatz: Quellprogramm
Programmname mit Zusatz CODE: Zwischencode
(&01): UTM.
(&01): UTM-Asynchron
Remote-Zugriffe auf BS2000-Datenbanken sind im UTM-Asynchron-Betrieb nicht erlaubt.

DRI0179 COMPILER (&00) IST NICHT VORHANDEN

Bedeutung
Der genannte Compiler ist nicht in die DRIVE-Phase miteingebunden. Die Anweisung COMPILE OPTION OBJECT=ON kann daher nicht ausgefuehrt werden.
(&00): Version des Compilers

DRI0180 'CONVERSION ERROR' BEIM (&00). OPERANDEN DER OPERATION '(&01)'

Bedeutung
Bevor in einem Ausdruck die Operation (&01) ausgefuehrt werden konnte, ist bei der Konvertierung des (&00). Operanden von (&01) ein Ablauffehler aufgetreten.

Maßnahme
(&00). Operanden korrekt versorgen oder Operation (&01) zusammen mit deren Operanden aus dem Ausdruck entfernen.

DRI0181 'CALC OVERFLOW' ((&00)) BEI OPERATION '(&01)'

Bedeutung
Beim Ausfuehren der Operation (&01) innerhalb einer Ausdrucksberechnung ist ein Berechnungsueberlauf aufgetreten.
(&00): (MACHINE ERROR): Fehler bei Maschinenzahlarithmetik.

DRI0182 'DIVISION ERROR' BEI OPERATION '(&00)'

Bedeutung
In einem Ausdruck wurde bei der Operation (&00) versucht, durch 0 (nicht NULL-Wert NULL) zu teilen. Der Ausdruck konnte deshalb nicht berechnet werden.

DRI0183 WERT DES AUSDRUCKS ZU LANG BEI ZUWEISUNG AN '(&00)'

Bedeutung
Das Ergebnis des Ausdrucks ist zu lang fuer die Variable.
(&00): Variablenname.

DRI0184 '(&00)' BEI ZUWEISUNG AN '(&01)'

Bedeutung
Bei der Variablenzuweisung ist der Fehler (&00) aufgetreten.
(&00): CONVERSION ERROR
CHECK ERROR
(&01): Variablenname

DRI0185 UNZULAESSIGE GLEITPUNKTMASKE/-DARSTELLUNG

Bedeutung
- Eine vergebene Maske enthaelt 'E' als Steuerzeichen, kann jedoch nicht als korrekte Gleitpunktmaske erkannt werden.
- Bei einer Bildschirmeingabe oder NUMERIC-Funktion wurde 'e' im Datenwert erkannt, der Datenwert kann jedoch nicht als Gleitpunktwert interpretiert werden.

Maßnahme
Maske/Eingabe/NUMERIC-Ausdruck auf Korrektheit ueberpruefen und ggfs. korrigieren.

DRI0186 ANFRAGE LIEFERT MEHR ALS EINEN TREFFER

Bedeutung
Ein SELECT liefert mehr als einen Datensatz; DRIVE unterstuetzt aber nur den Einzel-SELECT.

Maßnahme
Formulieren der Anfrage, so dass nur ein Treffer geliefert wird, z.B. durch eine WHERE-Klausel

DRI0187 KEINE WEITEREN CURSOR-DEKLARATIONEN MOEGLICH

Bedeutung
Fuer Informix koennen nur maximal 63 Cursor vom Anwender deklariert werden.

Maßnahme
Nicht benoetigte Cursor schliessen und freigeben

DRI0188 '(&00)' UND '(&01)' UNVERTRAEGLICH IN EINEM PROGRAMMSYSTEM

Bedeutung
In einem Programmsystem duerfen sich zwei Programme, die fuer die genannten Datenbanksysteme uebersetzt sind, nicht gegenseitig aufrufen.
(&00),(&01): Datenbanksysteme

DRI0189 WECHSELN DER INFORMIX-DATENBANK IM UTM-BETRIEB NICHT ERLAUBT

DRI0190 VERFUEGBARE ZEILEN FUER '(&00)'-AUSGABE NICHT AUSREICHEND

Bedeutung
Die Anzahl der in der DECLARE FORM-Anweisung definierten Ausgabezeilen fuer TTITLE bzw. BTITLE ist groesser als die Anzahl der explizit oder implizit definierten Zeilen fuer den Ausgabebereich (Bildschirm oder Liste).
(&00): TTITLE
BTITLE

Maßnahme
Eine evtl. vorhandene LINES-Angabe entsprechend vergroessern oder die TTITLE- bzw. BTITLE-Angabe modifizieren.

DRI0191 FELD MIT EINGABEPFLICHT NICHT VERSORGT

Bedeutung
Das markierte Feld (EDIT STATE = MUST ERROR) muss versorgt sein.

DRI0192 WERT DES AUSDRUCKS ZU LANG

Bedeutung
- Maximal Laenge des Ausdrucks in der SYSTEM-Anweisung: 254
- Maximal Laenge fuer den Folgetac bei der STOP-Anweisung: 8.

DRI0193 '(&00)' MIT PROGRAMM IN EDT-ARBEITSDATEI 0 NICHT MOEGLICH

Bedeutung
Moegliche Ursachen:
- PTRACE benoetigt die Uebersetzungsliste. Beim Starten eines Programms in EDT-Arbeitsdatei 0 ist der Name des Programms und damit der Uebersetzungsliste nicht bekannt.
- UREF benoetigt einen Programmnamen, um die Verwendungsnachweise im DATA DICTIONARY ablegen zu koennen.

(&00): PTRACE
UREF.

DRI0194 WERT DES AUSDRUCKS ZU GROSS

Bedeutung
Der Wert des Ausdrucks kann wegen seiner Groesse am Bildschirm nicht ausgegeben werden.

DRI0195 '(&00)' KANN NICHT MIT SQL BEARBEITET WERDEN

Bedeutung
Folgende Eigenschaften sind unzulaessig:
- scale < 0
- scale > precision
- precision > 15

(&00): Name des Satzelements.

Maßnahme
Die Datenbank muss entsprechend geaendert werden.

DRI0196 KEIN TAC ODER FALSCHER TAC IN UPICFILE ANGEGEBEN

Bedeutung
upicfile ist fehlerhaft; TAC-Name in upicfile in BS2000-Anwendung nicht referenziert;

Maßnahme
upicfile ueberpruefen, TAC hinzufuegen

DRI0197 UPICFILE ODER TNS-EINTRAG FEHLERHAFT ODER NICHT VORHANDEN

Bedeutung
1. upicfile (side_info-Datei) nicht im aktuellen Directory oder fehlerhaft
2. fuer den angegebenen USER gibt es keinen entsprechenden TNS-Eintrag.

Maßnahme
1. upicfile ueberpruefen bzw. in aktuelles Dateiverzeichnis stellen
2. entsprechenden TNS-Eintrag erstellen lassen

DRI0198 NETZVERBINDUNG ABGEBROCHEN

Bedeutung
Die Netzverbindung zum Server ist abgebrochen
Moegliche Ursachen sind:
- ungueltiger TAC angegeben
- PEND ER im UTM-Vorgang aufgetreten
- UTM-Anwendungsende
- Verbindungsabbau durch UTM-Administration
- Verbindungsabbau durch das Transportsystem

Maßnahme
Angegebenen TAC ueberpruefen.
UTM-Anwendung ggfs. beenden und neu starten.
Netzadministrator benachrichtigen.

DRI0199 NICHT ALLE SELEKTIERTEN OBJEKTE SIND VORHANDEN

Bedeutung
Dieser Fehlerfall tritt bei der Funktion 'Anschauen' in der DRIVE-SPU auf. Da die angegebenen Objekte im Klassenfenster selektiert wurden, ist die dort angezeigte Liste nicht mehr aktuell.

Maßnahme
Ggfs. Liste im Klassenfenster aktualisieren

DRI0200 NAME IM '(&00)' ZU LANG

Bedeutung
Im DATA DICTIONARY koennen nur Bibliotheksnamen mit einer Laenge von maximal 32 Zeichen vergeben werden.
Auf SINIX: In Verwendungsnachweisen von Window-Programmen duerfen Resourcefilenamen maximal 54 Stellen lang sein.
(&00): DATA DICTIONARY

Maßnahme
Ggf. kuerzere Bibliotheksnamen verwenden.
Auf SINIX: Ggfs. kuerzere Resourcefilenamen verwenden

DRI0201 BIBLIOTHEKSNAME ZUM PROGRAMM IM '(&00)' NICHT EINDEUTIG

Bedeutung
Das aktuelle DRIVE-Programm ist laut DATA DICTIONARY in einer anderen als der aktuell angegebenen Bibliothek abgespeichert.
(&00): DATA DICTIONARY

Maßnahme
Eindeutigkeit des Quellprogramms zur Bibliothek herstellen.

DRI0202 'DATA DICTIONARY'-PARAMETERDATEI FEHLERHAFT ODER NICHT VORHANDEN

Bedeutung
Die mit Linkname TOMPAR zugewiesene Parameterdatei ist entweder leer, nicht vorhanden oder in ihr enthaltene Angaben sind fehlerhaft bzw. erforderliche Angaben fehlen ganz.

DRI0203 NAMENSVERGABE FUER OBJEKT '(&00)' IM '(&01)' NICHT EINDEUTIG

Bedeutung
Die Verwendungsnachweise koennen im DATA DICTIONARY nicht abgespeichert werden, da Namenseindeutigkeit nicht gegeben ist.
(&00): Objektklasse.
(&01): DATA DICTIONARY.

Maßnahme
Namen im DATA DICTIONARY ueberpruefen; unnoetige Objekte ggf. loeschen.

DRI0204 EIN ELEMENT IST ZU LANG

Bedeutung
- Der logische Bildschirmbereich des angesprochenen Formats ist nicht ausreichend, um ein in der FILL-Anweisung verwendetes Datenelement vollstaendig aufzunehmen.
- Ein in der SEND MESSAGE - Anweisung verwendetes Datenelement ist zu lang.

Maßnahme
- Eventuell vorhandene LINES- bzw. COLUMNS-Angaben anpassen.
- Fehlerhaftes Datenelement verkuerzen.

DRI0205 '(&00)'-FEHLER BEI MODUL/ENTRY '(&01)'

Bedeutung
Beim Initialisieren von DRIVE ist ein Fehler im Zusammenhang mit dem Binder-Lader-System aufgetreten.
(&00): Makro (TABLE/ITABL/LINK).
(&01): Modul- bzw. Entryname.

Maßnahme
Bei Fehlern mit dem LINK-Makro ist zu ueberpruefen, ob die DRIVE-Bibliothek vor dem Start von DRIVE richtig zugewiesen wurde. Ansonsten Administrator verstaendigen.

DRI0206 EINGABEFELD '(&00)' IN UEBERLAUFBILDSCHIRM UNZULAESSIG

Bedeutung
Bei einem DRIVE-Format duerfen in einem Ueberlaufbildschirm keine Eingabefelder liegen.
(&00): Variablenname.

DRI0207 ZUSTAND DER DB-TRANSAKTION NACH 'COMMIT WORK' UNBEKANNT.

Bedeutung
Netzverbindung wurde beendet;

Maßnahme
Bei erstem Aufruf: TAC ueberpruefen
Bei Folgeaufruf: UTM-Anwendung ueberpruefen (evtl. beendet ?)
Netzadministrator benachrichtigen

DRI0208 VERBINDUNG ZU 'UPIC' ABNORMAL BEENDET

Bedeutung
Die Verbindung zum UPIC-Prozess bzw. zu UPIC wurde abnormal beendet.
Moegliche Ursachen:
- UTM-Anwendungsende
- Verbindungsabbau durch UTM-Administration
- Verbindungsabbau durch das Transportsystem

Maßnahme
UTM-Anwendung ggfs. beenden.
Netzadministrator benachrichtigen.

DRI0209 DIE LIZENZ/SCHLUESSELINFORMATION FEHLT

Bedeutung
Anmelden an UPIC war nicht erfolgreich

Maßnahme
UTM-Installation ueberpruefen.

DRI0210 TRANSAKTION WURDE VOM DATENHALTUNGSSYSTEM ZURUECKGESETZT

Maßnahme
- DIALOG-Modus: SQL-Anweisungen ab dem letzten COMMIT WORK wiederholen.
- PROGRAMM-Modus: Programm neu starten (notwendig, da das Ruecksetzen in der ersten Transaktion des DRIVE-Programms erfolgte).

DRI0211 WIEDERAUFSETZEN IN DER ERSTEN TRANSAKTION NICHT MOEGLICH

DRI0212 PROGRAMM FORTSETZEN? ANTWORT: (Y=JA; N=NEIN)

Bedeutung
Die Transaktion wurde vom Datenhaltungssystem zurueckgesetzt.
Remote-Zugriff: wegen Netzproblemen ist der Zustand der Transaktion unbekannt

Maßnahme
Y: Das Programm wird auf den Stand des letzten COMMIT WORK zurueckgesetzt und die Verarbeitung mit der ersten Anweisung nach dem COMMIT WORK fortgesetzt.
Zusaetzlich gilt bei Remote-Zugriff:
wenn die Netzverbindung nicht wiederherzustellen ist, wird das Programm abgebrochen; der Zustand der Transaktion in der Datenbank ist unbekannt.

N: Das Programm wird abgebrochen.
Zusaetzlich gilt bei Remote-Zugriff: der Zustand der Transaktion in der Datenbank ist unbekannt.

DRI0213 WIEDERAUFSETZEN NICHT MOEGLICH; VORGANG ABGEBROCHEN

Bedeutung
Vor dem ersten COMMIT WORK im UTM-Vorgang wurde die Transaktion und damit der Vorgang zurueckgesetzt.

Maßnahme
Vorgang neu starten.

DRI0214 WIEDERANLAUF DES ABGEBROCHENEN 'UTM'-VORGANGS UNMOEGLICH

Bedeutung
Beim externen Wiederanlauf (z.B. nach Systemabsturz) trat erneut ein Fehler auf. Der Vorgang wurde beendet.

Maßnahme
Vorgang neu starten.

DRI0215 TRANSAKTION WEGEN SQL-CODE '(&00)' ZURUECKGESETZT

Bedeutung
Die Transaktion wurde vom Datenhaltungssystem zurueckgesetzt. SQL-Objekte haben den Stand des letzten Sicherungspunkts.
(&00): Vom DB-System gemeldeter SQL-CODE.

DRI0216 REDEFINIERTE BZW. REDEFINIERENDE VARIABLE NICHT ERLAUBT

Maßnahme
Variable ohne die genannten Eigenschaften benutzen

DRI0217 EVENT FUER WINDOW-OBJEKT NICHT ERLAUBT

Bedeutung
Die Kombination von diesem Event mit diesem Windowobjekt ist nicht moeglich

Maßnahme
Event aendern oder anderes Objekt angeben

DRI0218 LAENGE DER 'TITLE'-ANGABEN NACHTRAEGLICH GEAENDERT

Bedeutung
Bei der DISPLAY-Anweisung oder bei implizitem DISPLAY werden vor der Ausgabe die TITLE-Angaben aktualisiert. Die Laenge dieser TITLE-Angaben ist ungleich der urspruenglichen Laenge (z.B. durch Verwendung von Stringfunktionen bei der Zuweisung an TITLE-Variablen).

Maßnahme
Zuweisung nach dem ersten FILL ueberpruefen

DRI0219 KEINE VERBINDUNG MIT BS2000-UTM-ANWENDUNG; UTMRCS=(&00),(&01)

Bedeutung
Der APRO-Aufruf konnte nicht erfolgreich durchgefuehrt werden; moegliche Ursachen siehe UTM-Manual 'Anwendungen Programmieren', Kap. KDCS-Aufrufe, 'APRO';
(&00): UTM-Returncode
(&01): interner UTM-Fehlercode

Maßnahme
Systemverwalter informieren und UTM-Generierung ueberpruefen

DRI0220 'DRIVE-CACHE': RETURNCODE '(&00)' BEI MAKRO '(&01)'

Bedeutung
(&00): Returncode
(&01): BS2000-Makroname.

Maßnahme
Administrator verstaendigen.

DRI0221 'DRIVE-CACHE' KANN NICHT GESCHLOSSEN WERDEN

Maßnahme
Bei wiederholtem Auftreten des Fehlers ueberpruefen, ob beim Beenden von DRIVE die HALT-TSN (ENTER-Prozedur DRI.ENT.DRICACHE) noch laeuft. Wenn nicht, Ursache dafuer ermitteln. Wenn ja, Administrator verstaendigen.

DRI0222 'DRIVE-CACHE' EXISTIERT IN KENNUNG BEREITS MIT ANDEREN PARAMETERN

Bedeutung
Der angegebene DRIVE-CACHE existiert mit einer anderen Laenge als angegeben wurde. Beim Zugriff auf den gleichen CACHE muss die Laenge immer gleich sein.

Maßnahme
Moegliche Massnahmen:
- Eine zweite DRIVE-UTM-Anwendung in einer anderen Kennung starten
- CACHE-Laenge mittels Anweisung ACQUIRE MEMORY aendern.

DRI0223 'HALT-TSN' FUER 'DRIVE-CACHE' KANN NICHT AKTIV WERDEN

Bedeutung
Die ENTER-Prozedur DRI.ENT.DRICACHE (HALT-TSN) fuer den DRIVE-CACHE ist nicht innerhalb von 75 Sekunden angelaufen.

Maßnahme
- ENTER-Prozedur DRI.ENT.DRICACHE und Programm PRO.DRICACHE ueberpruefen.
- Ueberpruefen, ob die Batch-Queue des BS2000 ueberlastet ist und die ENTER-Prozedur deshalb nicht gestartet werden konnte.

DRI0224 NAMENSANGABE IM BENUTZERKENNSATZ SYNTAKTISCH FALSCH

Bedeutung
Der Bibliotheks- oder Elementname im Benutzerkennsatz wurde syntaktisch falsch angegeben.

Maßnahme
DRIVE-Handbuch zu Rate ziehen.

DRI0225 FEHLER BEI VERARBEITUNG DES BENUTZERKENNSATZES

Bedeutung
Das DRIVE-Programm, das bei der Verarbeitung des Benutzerkennsatzes gestartet wurde, enthaelt Analyse- oder Ablauffehler. Die entsprechende Fehlerliste wurde auf SYSLST ausgegeben.

Maßnahme
DRIVE-Programm korrigieren und UTM-Anwendung neu starten.

DRI0226 PROGRAMM FEHLERHAFT; 'DRIVE' BEENDET DA 'TEST=ALL' GESETZT

Bedeutung
Das zuletzt gestartete Programm enthaelt Analyse- oder Ablauffehler. Da Parameter TEST=ALL gesetzt ist, wurde DRIVE daraufhin beendet. Die entsprechende Fehlerliste wurde auf SYSLST ausgegeben.

DRI0227 KONSTANTE NICHT ERLAUBT

Bedeutung
Der Objektname bezeichnet eine Konstante, was in diesem Kontext nicht sinnvoll ist

Maßnahme
Eine Variable angeben

DRI0228 KEIN ZUGRIFF AUF '(&00)'

Bedeutung
Auf das bezeichnete Element besteht kein Zugriffsrecht (Lese-/Schreib-/Aus-fuehrungs- oder Suchrecht je nach Datei/Verzeichnis und versuchter Operation). Das fehlende Recht kann sich ggfs. auch auf Pfadnamenskomponenten beziehen. Klappt der Zugriff auf eine DRIVE-Systemressourcedatei nicht, kann auch ein Installationsfehler der Grund dafuer sein.
(&00): Dateiname

Maßnahme
Zugriffsrechte aendern
Liegt fuer eine DRIVE-Systemressourcedatei ein Installationsfehler vor, muss neu installiert werden.

DRI0229 ANGABE FUER DB-SYSTEM '(&00)' UNZULAESSIG

Bedeutung
Anweisung bzw. Teil einer Anweisung ist fuer die Zusammenarbeit mit SESAM/UDS nicht zulaessig
(&00): DB-System (SESAM/UDS)

Maßnahme
Fehlerhaften Anweisungsteil korrigieren oder entfernen oder DB-System INFORMIX waehlen

DRI0230 MITTELS 'QUERY' DEF. VIEW NICHT VERWENDBAR (SQL-CODE=(&00))

Bedeutung
Der mittels QUERY definierte View entspricht nicht den UDS- bzw. SESAM-Regeln (z.B. Gross-/Kleinschreibung, Zeichenvorrat fuer Viewname nicht beachtet).
(&00): Vom DB-System zurueckgemeldeter SQL-CODE.
SQL-CODE = 0: Der Fehler wurde bereits von DRIVE erkannt.
SQL-CODE < 0: Der Fehler wurde vom DB-System erkannt.

Maßnahme
Ggf. mittels QUERY den View neu definieren unter Beachtung der syntaktischen und semantischen Regeln des entsprechenden DB-Systems.

DRI0231 DAS (ZUGEHOERIGE) FENSTER IST NOCH NICHT GENERIERT

Bedeutung
Diese Window-4-GL-Anweisung ist erst dann zulaessig, wenn das Fenster, zu dem das angesprochene Objekt gehoert, generiert worden ist, d. h. nach der ersten ADD-/NEW-/NEXT-Anweisung fuer dieses Fenster

Maßnahme
Fenster vor Durchlaufen dieser Anweisung generieren

DRI0232 DER ANGEGEBENE ATTRIBUTWERT IST NICHT ZULAESSIG

Bedeutung
Der zu der SET ATTRIBUTE-Anweisung angegebene Wert
- liegt ausserhalb des zulaessigen Wertebereiches
- ist der NULL-Wert
- ist zu lang

Maßnahme
Datenquelle fuer den Attributwert korrigieren

DRI0233 DER ANGEGEBENE ITEMNAME IST NICHT ZULAESSIG

Bedeutung
Der Itemname ist
- der NULL-Wert
- zu lang (>255)
- mehrfach in der Itemliste vorhanden

Maßnahme
Itemname mit Nicht-NULL-Wert versorgen bzw. verkuerzen bzw. Duplikate aus der Itemliste entfernen.

DRI0234 DER ITEMNAME IST BEREITS VORHANDEN

Bedeutung
Alle Itemnamen innerhalb einer Auswahlliste muessen eindeutig sein.

Maßnahme
Datenquelle fuer den Itemnamen korrigieren

DRI0235 DIE 'POSITION'-ANGABE IN DER 'ADD ITEM'-KLAUSEL IST FEHLERHAFT

Bedeutung
Die Positionsangabe muss sich immer auf ein vorhandenes List-Item beziehen bzw. '0' sein

Maßnahme
Pruefen, ob der POSITION-Wert zwischen 0 und der Anzahl der List-Items liegt

DRI0236 DER ANGEGEBENE ITEMNAME EXISTIERT NICHT

Bedeutung
Bei DELETE ITEM duerfen nur die Namen von Items angegeben werden, die in der Liste vorhanden sind

Maßnahme
Datenquelle fuer den Itemnamen korrigieren

DRI0237 DER PFADNAME DES ANGESPROCHENEN WINDOW-OBJEKTS IST ZU LANG

Bedeutung
Dieses Objekt kann in keiner Window-4GL-Anweisung direkt angesprochen werden.

Maßnahme
Die einzelnen Objektnamen oder der komplette Pfadname muessen im Dialog Builder auf die zulaessige Maximallaenge (32768) verkuerzt werden.

DRI0238 ZUVIELE EINTRAEGE IN DER AKTUELLEN AUSWAHLLISTE

Bedeutung
Die maximale Anzahl von Listeneintraegen (32767) wird durch die beabsichtigte Neuaufnahme weiterer Eintraege ueberschritten

Maßnahme
Auswahlliste vor der Neuaufnahme durch Loeschen nicht mehr benoetigter Eintraege entsprechend verkleinern.

DRI0239 ANWEISUNG NUR MIT 'WITH DUPLICATES'-KLAUSEL ZULAESSIG

Bedeutung
Es wurde bereits eine ALTER CHOICE LIST-/ALTER COMBO BOX-Anweisung mit der Klausel "WITH DUPLICATES" fuer die aktuelle Auswahlliste/Kombobox ausgefuehrt. Ein Wechsel zu duplikatfreier Verarbeitung ist erst wieder nach einer erneuten Generierung des Fensters (durch ADD/NEW/NEXT WINDOW) moeglich.

Maßnahme
Die fehlerhafte Anweisung um die Klausel "WITH DUPLICATES" ergaenzen.

DRI0240 LISTENDEFINITION IN DER RESSOURCEDATEI IST FEHLERHAFT

Bedeutung
Mindestens einer der in der Ressourcedatei vordefinierten Eintraege fuer die Auswahlliste/Kombobox ueberschreitet die maximal zulaessige Laenge (255).

Maßnahme
Alle zu langen Listeneintraege entsprechend verkuerzen (Attribut "ITEMS" im Attributeditor des DialogBuilders).

DRI0241 SELEKTION VON MEHREREN EINTRAEGEN FUER EINE EINFACHAUSWAHLLISTE

Bedeutung
Die Selektionsliste fuer eine Einfachauswahlliste darf nur genau einen Eintrag enthalten. Nur bei Mehrfachauswahllisten koennen mehrere Eintraege gleichzeitig selektiert werden.

Maßnahme
Entweder die "select-item-clause" in der fehlerhaften "ALTER CHOICE LIST"-Anweisung korrigieren (nur einen einzigen Eintrag angeben) oder den Selektionsmodus der Auswahlliste im Attributeditor des DialogBuilders aendern, d.h. aus der Einfach- eine Mehrfachauswahlliste ("SELECTION_MODE =MULTIPLE") machen.

DRI0249 USEREVENT '(&00)' NICHT ZUSTELLBAR

Bedeutung
Das Zielwindow ist nicht am Bildschirm. Moegliche Ursachen hierfuer:
- das Window wurde noch nicht ausgegeben
- das Window wurde inzwischen durch eine CLOSE WINDOW-Anweisung oder explizit ueber den Fenstermenueknopf geschlossen.

(&00): Event

Maßnahme
Programmsourcen ueberpruefen; evtl. USEREVENT-Error durch entsprechendes WHENEVER abfangen;

DRI0250 AUSGABE ZU LANG, DA ZU VIELE INTERNE STEUERZEICHEN BENOETIGT WERDEN

Bedeutung
Die Ausgabe enthaelt zu viele sich in ihren Darstellungseigenschaften unterscheidende Felder (z.B. hell/normal/blinkend/ueberschreibbar/geschuetzt). Diese Eigenschaften werden intern ueber Steuerzeichen realisiert (terminalabhaengig), die die Ausgabe insgesamt zu lang machen.

Maßnahme
Ausgabe verkuerzen oder weniger Felder mit unterschiedlichen Darstellungseigenschaften verwenden.

DRI0251 '(&00)' KANN NICHT ERZEUGT WERDEN

Bedeutung
(&00): Dateiverzeichnis : Das Dateiverzeichnis kann mangels Speicherplatz nicht angelegt werden
(&00): Window/Dialogbox : Inkonsistenz im Resourcefile; Window/Dialogbox ist im angegebenen Resourcefile nicht vorhanden.

Maßnahme
Bei Speicherplatzmangel nicht benoetigte Dateien loeschen und Anweisung erneut ausfuehren. Bei Inkonsistenz im Resourcefile pruefen, ob das/die auszugebende Window/Dialogbox im angegebenen Resourcefile vorhanden ist.

DRI0252 KEIN ROOT-OBJEKT IM '(&00)' VORHANDEN

Bedeutung
Resourcefile wurde in unzulaessiger Weise modifiziert. Es kann nicht geoeffnet werden.
(&00): Resourcefile

Maßnahme
Resourcefile nur mit Dialog Builder erstellen.

DRI0253 FEHLER BEIM ANBINDEN VON CALLBACKS ZU RESOURCEFILE

Bedeutung
Das Resourcefile wurde in unzulaessiger Weise modifiziert. Es kann nicht geoeffnet werden.

Maßnahme
Callback-Angaben im Resourcefile entfernen und DRIVE-Programm neu uebersetzen

DRI0254 KEIN TOPLEVELOBJEKT IN RESOURCEFILE '(&00)'

Bedeutung
Das angegebene Resourcefile enthaelt kein Toplevelobjekt (Window/Dialogbox).
(&00): Resourcefile

Maßnahme
Namen des Toplevelobjekts und des Resourcefiles ueberpruefen

DRI0255 FEHLER BEIM OEFFNEN VON '(&00)'

Bedeutung
Das angegebene Resourcefile kann nicht geoeffnet werden. Moegliche Ursachen:
- Resourcefile nicht vorhanden
- Zugriffsrechte fehlen

(&00): Resourcefile

Maßnahme
Existenz und Zugriffsrechte des Resourcefiles ueberpruefen.

DRI0256 XLIB-FEHLER IM WINDOW-SYSTEM

Bedeutung
Xlib meldet einen Fehler, wenn ein Systemaufruf nicht ausgefuehrt werden kann. Z.B. wenn Verbindung zum Server abgebrochen ist.

DRI0257 FEHLER BEIM WINDOW-SYSTEM 'MOTIF'

Maßnahme
Die Meldung des Window-Systems wird in der Datei 'inttrace.idx' bzw. 'inttrace.dat' hinterlegt

DRI0258 OBJEKT '(&00)' ENTHAELT ZU VIELE KINDOBJEKTE

Bedeutung
Unter dem angegebenen Toplevelobjekt sind zu viele Kindobjekte. Die Folge ist ein interner Tabellenueberlauf
(&00): Toplevelobjekt

Maßnahme
Weniger Kindobjekte fuer das Toplevelobjekt definieren.

DRI0259 OBJEKT '(&00)' INKONSISTENT

Bedeutung
Moegliche Ursachen sind:
a) Objektklassentyp eines Windowobjekts ist inkorrekt
b) Datentyp eines Windowobjekts ist nicht vorhanden
c) Toplevelobjekt ist kein(e) Window, Dialogbox oder Messagebox
(&00): Name des Objekts

Maßnahme
Resourcefile auf moegliche Ursachen hin untersuchen

DRI0260 FEHLER BEIM SCHREIBEN DER 'DRIVE'-DATEN IN '(&00)'

Bedeutung
Schreibender Zugriff auf das Resourcefile nicht moeglich
(&00): Resourcefile

Maßnahme
Zugriffsrechte aendern

DRI0261 INKONSISTENZ IM RESOURCEFILE '(&00)'

Bedeutung
Das Resourcefile wurde zwischen Uebersetzung und Ausfuehrung des DRIVE-Programms vom Benutzer inkompatibel geaendert.
(&00): Fehlercode
Bedeutung der Fehlercodes:

1,2,3:	Windowobjekt bei SET ATTRIBUTE nicht vorhanden
4,5,6,7:	Eingabefeld nicht vorhanden
8,9,10,11,12,13:	Auswahlliste nicht vorhanden
14:	Windowobjekt bei GET ATTRIBUTE nicht vorhanden

Maßnahme
Resourcefile darf zwischen Uebersetzung und Ausfuehrung nicht inkompatibel geaendert werden

DRI0262 MAXIMALE ANZAHL DER FORMANTSITZUNGEN UEBERSCHRITTEN

Maßnahme
Systemverwaltung informieren.

DRI0263 FORMANT-RETURNCODE (&00) BEI FORMAT '(&01)'

Bedeutung
Das Format kann nicht geladen werden, denn die Formant-Formatdatei ist inkonsistent, nicht vorhanden, oder es kann nicht auf sie zugegriffen werden.
(&00): Formant-Returncode (siehe Formant-Manual)
(&01): Formatname

Maßnahme
Formant-Formatdatei und Formant-Adressierungshilfe ueberpruefen.

DRI0264 FORMANT-RETURNCODE (&00) BEI FELD '(&01)' IN FORMAT '(&02)'

Bedeutung
Der angegebene Feldname im angegebenen Format wurde nicht gefunden, d.h. der angegebene Feldname der Formant-Adressierungshilfe ist nicht in der Formant-Formatdateivorhanden.
(&00): Formant-Returncode (siehe Formant-Manual)
(&01): Feldname im Format
(&02): Formatname

Maßnahme
Formant-Formatdatei und Formant-Adressierungshilfe ueberpruefen.

DRI0265 FALSCHE FORMANT-VERSION INSTALLIERT

Bedeutung
Es ist nicht die richtige Formant-Version installiert

Maßnahme
Systemverwalter informieren

DRI0266 OBJEKT (&00) NICHT IN '(&01)'

Bedeutung
Das angegebene Resourcefile enthaelt nicht das angegebene Toplevelobjekt
(&00): Toplevelobjekt
(&01): Resourcefile

Maßnahme
Namen des Resourcefiles und des Toplevelobjekts ueberpruefen

DRI0267 VERBINDUNG ZU X-SERVER KANN NICHT HERGESTELLT WERDEN

Bedeutung
Die DISPLAY-Variable ist falsch oder nicht versorgt.

Maßnahme
DISPLAY-Variable richtig versorgen.

DRI0280 PROGRAMM '(&00)'

Bedeutung
Hinweistext in der SPU, dass das genannte Programm sich noch in Ausfuehrung befindet.
(&00): Programmname

DRI0281 UEBERSETZUNG VON '(&00)'

Bedeutung
Hinweistext in der SPU, dass das genannte Programm noch uebersetzt wird.
(&00): Programmname.

DRI0282 LOESCHEN VON '(&00)' '(&01)'

Bedeutung
Hinweistext in der SPU, dass das genannte inkl. aller zugehoeriger Objekte noch geloescht wird.
(&00): Programmname
(&01): Zeichen fuer nicht abbrechbaren Vorgang

DRI0283 EDITOR GELADEN MIT '(&00)'

Bedeutung
Hinweistext in der SPU, dass ein Editor mit der genannten Datei aufgerufen worden ist. Ggfs. ist die Datei noch nicht gesichert.
(&00): Dateiname

DRI0284 AUFTRAG '(&00)' IN WARTESCHLANGE EINGEREIHT

Bedeutung
Fuer den genannten Auftrag ist zur Zeit kein DRIVE-Kernprozess zur Bearbeitung frei.

DRI0285 DRUCK VON '(&00)'

Bedeutung
Hinweistext in der SPU, dass fuer die genannte Datei ein Druckauftrag angestossen wurde.
(&00): Dateiname

DRI0300 MAX. OBJEKTGROESSE DER COMPILIERTEN 'DRIVE'-PROZEDUR UEBERSCHRITTEN

Bedeutung
Der Befehlsteil des generierten Objektcodes ueberschreitet 400 Slices a 4KB.

Maßnahme
Pruefen, ob Option NULLVALUE=OFF oder CHECK=OFF anwendbar, sonst DRIVE-Prozedur in mehrere Teile zerlegen.

DRI0301 NULL SPEZIFIZIERT, OBWOHL NULL NICHT ZUGELASSEN IST

Bedeutung
Die Prozedur enthaelt den Wert NULL, aber als Compiler-Option wurde spezifiziert: NULL tritt nicht auf.

Maßnahme
Quelle oder Compiler-Optionen aendern.

DRI0302 KEINE AUFRUFART MOEGLICH

Bedeutung
Die Prozedur laesst sich in Verbindung mit den spezifizierten Compiler-Optionen weder als Startprozedur noch ueber DO/ENTER/CALL aufrufen. Der Fall tritt z. B. ein, wenn ein PERMIT-Bildschirm verlangt wird, die Prozedur aber Parameter enthaelt.

Maßnahme
Quelle oder Compiler-Optionen aendern

DRI0303 ZU VIEL PLATZ ERFORDERLICH FUER INTERNE PERMANENTE VARIABLE

Bedeutung
Interne permanente Variable werden angelegt
- zu SUBPROCEDURE,
- zu CYCLE FOR mit nicht konstantem STEP- oder END- Wert.
Deren erforderlicher Speicherbedarf uebersteigt 32 KB.

Maßnahme
Die Anzahl der obigen Anweisungen verringern.

DRI0304 ZU VIELE HILFSVARIABLE ERFORDERLICH (&00)

Bedeutung
Bei der Compilierung einer Anweisung in einer DRIVE-Prozedur werden zu viele Hilfsvariablen benoetigt. Der Einfuegetext enthaelt den Prozedurnamen und die Zeilennummer im expandierten Quell-Listing zu der betroffenen Anweisung.

Maßnahme
Die betroffene Anweisung in mehrere Anweisungen zerlegen.

DRI0305 FEHLER BEIM ANLEGEN ODER OEFFNEN DER HILFSDATEI ZUM OBJEKTCODE

Bedeutung
Der generierte Objektcode wird zunaechst in eine Hilfsdatei geschrieben, die intern angelegt wird.
Beim Anlegen oder Oeffnen dieser Hilfsdatei ist ein Fehler aufgetreten. Der DVS-Returncode wird auf dem Terminal ausgegeben.

Maßnahme
entsprechend DVS-Returncode

DRI0306 FEHLER BEIM SCHLIESSEN DER HILFSDATEI ZUM OBJEKTCODE

Bedeutung
Der generierte Objektcode wird zunaechst in eine Hilfsdatei geschrieben, die intern angelegt wird.
Beim Schliessen dieser Hilfsdatei ist ein Fehler aufgetreten. Der DVS-Returncode wird auf dem Terminal ausgegeben.

Maßnahme
entsprechend DVS-Returncode

DRI0307 FEHLER BEIM SCHREIBEN IN DIE HILFSDATEI ZUM OBJEKTCODE

Bedeutung
Der generierte Objektcode wird zunaechst in eine Hilfsdatei geschrieben, die intern angelegte wird.
Beim Schreiben in diese Hilfsdatei ist ein Fehler aufgetreten. Der DVS-Returncode wird auf dem Terminal ausgegeben.

Maßnahme
entsprechend DVS-Returncode

DRI0308 FEHLER BEI LMS - ZUGRIFF (&00)

Bedeutung
Beim Uebertragen des Objektcodes aus der temporaeren Hilfsdatei in die Zielbibliothek ist ein Fehler aufgetreten.
Der Einfuegetext enthaelt die LMS-Returncodes.

Maßnahme
entsprechend LMS-Returncodes

DRI0309 INTERNER FEHLER BEI COMPILIERUNG (&00)

Bedeutung
Bei der Compilierung einer DRIVE-Prozedur ist ein interner Fehler aufgetreten. Der Einfuegetext enthaelt den Prozedurnamen, die Zeilennummer zu der betroffenen Anweisung im expandierten Quell-Listing und eine interne Fehlernummer.

Maßnahme
Systemberatung verstaendigen, das expandierte Quell-Listing und die Angaben im Einfuegetext zur Verfuegung stellen. Gegebenenfalls werden weitere Unterlagen erforderlich.

DRI0310 VORGANG '(&00)' BEENDET. BITTE TRANSAKTIONSCODE EINGEBEN

Bedeutung
Beenden des Vorgangs im DRIVE/WINDOWS-Objektbetrieb.
(&00): Vorgangsname

Maßnahme
Transaktionscode eingeben

DRI0311 INTERNER FEHLER IM 'DRIVE'-OBJEKT '(&00)','(&01)'

Bedeutung
Interne Inkonsistenzen erzwingen einen Abbruch des DRIVE-Objekt-Laufs.
(&00): Ereignistyp
(&01): Systemcode zur naeheren Klassifizierung des Ereignisses

Maßnahme
Administrator verstaendigen

DRI0312 NULLWERT IM DRIVEOBJEKT AUFGETRETEN

Bedeutung
Bei einem, mit der OPTION NULLVALUE=OFF erzeugten Objekt ist ein Null-Wert aufgetreten. Dies kann geschehen bei Datenbankabfragen, Bildschirmeingaben oder Parameteruebergabe bei den Anweisungen CALL, DO oder ENTER.

Maßnahme
Prozedur mit der OPTION NULLVALUE=ON uebersetzen

DRI0390 DATEI (&00) KANN NICHT GELOESCHT WERDEN.

Bedeutung
Keine Schreibrechte auf das Verzeichnis oder Datei existiert nicht.
(&00): Dateiname

DRI0391 FEHLER BEIM SCHREIBEN DER DATEI (&00) AUFGETRETEN.

Bedeutung
Platzmangel auf dem Filesystem.
(&00): Dateiname

DRI0392 DATEI (&00) KANN NICHT ANGELEGT ODER GEOEFFNET WERDEN.

Bedeutung
Keine Schreibrechte auf das Verzeichnis oder existierende Datei darf nicht geoeffnet werden.
(&00): Dateiname

DRI0393 C-UEBERSETZUNGSVORGANG NICHT FEHLERFREI BEENDET.

Bedeutung
Uebersetzungsvorgang wurde durch Interrupt abgebrochen oder Systemgrenzen (Anzahl Prozesse, Speichergroesse) ueberschritten oder C-Programm fehlerhaft.

DRI0394 FEHLER BEIM OEFFNEN DER BIBLIOTHEK '(&00)'

Bedeutung
Unter dem angegebenen Namen ist keine Bibliothek vorhanden, das Zugriffsrecht auf die Bibliothek fehlt oder die Bibliothek ist gesperrt.
Zusaetzlich fuer das SINIX-Betriebssytem:
Umgebunsvariable LD_LIBRARY_PATH enthaelt nicht den Pfadnamen der Bibliothek.

Maßnahme
Korrekte Bibliothek erzeugen, Zugriffsrechte erwerben und sicherstellen, dass die Bibliothek zum Zugriffszeitpunkt nicht gesperrt ist (z.B. durch einen Uebersetzungsvorgang).
Zusaetzlich fuer das SINIX-Betriebssystem:
Pfadnamen der Bibliothek in Umgebungsvariable LD_LIBRARY_PATH aufnehmen.

DRI0399 (&00).

Bedeutung
Bei dem Bibliotheksnamen findet sich kein shared object (keine shared library).
(&00): Bibliotheksname

DRI0401 UNZULAESSIGER NAME

Bedeutung
In der DRIVE-SPU erfolgte in einem Eingabefeld eine unzulaessige Eingabe

Maßnahme
In der DRIVE-SPU kann ggfs. ueber den Hilfeknopf des Fensters Information ueber zulaessige Eingaben und gueltige Namen abgerufen werden.

DRI0402 FEHLER BEIM LOESCHEN VON OBJEKTEN

Bedeutung
Beim Loeschen von Objekten (Sourcen, Zwischencode, Listenelementen, Resourcefiles, Benutzerkennsaetzen) ist ein Fehler aufgetreten.

DRI0403 GENAU EIN OBJEKT SELEKTIEREN

Bedeutung
Die gewuenschte Funktion kann nur ausgefuehrt werden, wenn genau ein Objekt selektiert wurde

DRI0404 MINDESTENS EIN OBJEKT SELEKTIEREN

Bedeutung
Die gewuenschte Funktion wird nur ausgefuehrt, wenn mindestens ein Objekt selektiert wurde.

DRI0405 MINDESTENS EINEN AUSWAHLKNOPF DRUECKEN

DRI0410 AT-STATEMENT ZU LANG

Bedeutung
Die AT-Anweisung, die aus den Einstellungen in der Dialogbox aufgebaut wird, ist zu lang.

Maßnahme
Angaben in den Eingabefeldern kuerzen.

DRI0411 LIB-SPEC-ANGABE FEHLERHAFT

Bedeutung
Statt der 'lib-spec'-Angabe wurde ein Pfadname angegeben, d. h. das Element (Datei) wurde nicht in '(' und ')' eingeschlossen.

Maßnahme
Entweder Eingabe des Elementnamens (Dateiname) oder 'lib-spec'-Angabe. in der das Element in '(' und ')' eingeschlossen ist. Eine Pfadangabe ist unzulaessig.

DRI0450 UEBERGABEBEREICH UNTER VERTEILTER VERARBEITUNG FEHLERHAFT

Bedeutung
Der Uebergabebereich den DRIVE unter verteilter Verarbeitung von einem C/COBOL Partner als Input bekommt ist inkonsistent oder fehlerhaft

Maßnahme
Der Aufbau des Uebergabebereichs den DRIVE unter verteilter Verarbeitung von einem C/COBOL Partner erhaelt muss auf Inkonsistenzen ueberprueft werden.

DRI0451 NACHRICHTENLAENGE (&00) ZU KLEIN

Bedeutung
Die Speicherlaenge der vom Partner-Vorgang erhaltenen Nachricht ist zu klein. Der Header der Uebergabeinformationen konnte nicht gelesen werden.
(&00): Erhaltene Nachrichtenlaenge

Maßnahme
Laenge der Nachricht beim Senden ueberpruefen.

DRI0452 FELD '(&00)' IM UEBERGABEBEREICHS-HEADER FALSCH VERSORGT

Bedeutung
Das angegebene Feld im Header wurde vom Partner-Vorgang mit einem unzulaessigen Wert belegt.
(&00): fehlerhaft versorgtes Feld

Maßnahme
Versorgung des Header-Feldes ueberpruefen.

DRI0453 KEINE 'RETURN'-PARAMETERWERTE VOM (&00) ERHALTEN

Bedeutung
Der mit CALL aufgerufene Auftragnehmer gibt keine RETURN-Parameterwerte zurueck, obwohl in der USING-Klausel Parameter mit RETURN spezifiziert sind.
Per Newstyle-CALL wurde in den Oldstyle-Betrieb gewechselt. Das letzte durchlaufene Oldstyle-Programm hat keine USING-Leiste, beim Newstyle-CALL sind dagegen RETURN-Parameter in der USING-Leiste spezifiziert.

Maßnahme
Schnittstelle des Auftragnehmer-Programms/-Teilprogramms ueberpruefen.
USING-Leisten von Newstyle-CALL und Oldstyle-Programm aufeinander abstimmen.

DRI0454 RETURN-PARAMETERWERTE VOM AUFTRAGNEHMER UNZULAESSIG

Bedeutung
Der mit CALL aufgerufene Auftragnehmer gibt RETURN-Parameterwerte zurueck, obwohl in der USING-Klausel kein Parameter mit RETURN spezifiziert ist.

Maßnahme
Schnittstelle des Auftragnehmer-Programms/-Teilprogramms ueberpruefen.

DRI0455 AUFRUF ANWENDER-TAC '(&00)' MIT PARAMETERFEHLERN BEENDET

Bedeutung
Das aufgerufene Benutzer-Teilprogramm meldet Fehler beim Lesen der Header-Information im Uebergabebereich.
(&00): UTM-Transaktionscode

Maßnahme
Verteilinformationen fuer das betreffende Anwender-Teilprogramm ueberpruefen.

DRI0456 AUFRUF ANWENDER-TAC '(&00)' MIT FEHLERN BEENDET

Bedeutung
Das aufgerufene Benutzer-Teilprogramm meldet Fehler bei der Programm-Ausfuehrung.
(&00): Name des UTM-Transaktionscodes

Maßnahme
Ablauf des aufgerufenen Anwender-Teilprogramms und ggf. uebergebene USING-Daten ueberpruefen.

DRI0457 AUFRUF DRIVE-AUFTRAGNEHMER-VORGANG MIT PARAMETERFEHLERN BEENDET

Bedeutung
Der aufgerufene DRIVE-Auftragnehmervorgang meldet Fehler beim Lesen der Header-Informationen im Uebergabebereich.

Maßnahme
Verteilinformationen fuer das betreffende Auftragnehmer-DRIVE-Programm ueberpruefen.

DRI0458 AUFRUF DRIVE-AUFTRAGNEHMER-VORGANG MIT FEHLERN BEENDET

Bedeutung
Das aufgerufene DRIVE-Programm meldet Fehler bei der Programm-Ausfuehrung.

Maßnahme
Ablauf des aufgerufenen DRIVE-Programms und ggf. uebergebene USING-Daten ueberpruefen.

DRI0459 BIBLIOTHEKNAME '(&00)'

Bedeutung
Bibliothek eines ausgefuehrten DRIVE-Programms. Ergaenzungstext zu einer Parameterfehler- oder Ablauffehler-Meldung bzw. Statusinformation bei verteilter Verarbeitung.
(&00): Bibliotheksname

DRI0460 ELEMENTNAME '(&00)'

Bedeutung
Elementname eines ausgefuehrten DRIVE-Programms. Ergaenzungstext zu einer Parameterfehler- oder Ablauffehler-Meldung bzw. Statusinformation bei verteilter Verarbeitung.
(&00): Elementname

DRI0461 ANWENDUNGSNAME '(&00)'

Bedeutung
Name einer Auftragnehmer-Anwendung, in der ein DRIVE-Programm oder Anwender-Teilprogramm ausgefuehrt wurde. Ergaenzungstext zu einer Parameterfehler- oder Ablauffehler-Meldung bzw. Statusinformation bei verteilter Verarbeitung.
(&00): Anwendungsname

DRI0462 VERTEILTE VERARBEITUNG FEHLERHAFT

Bedeutung
Die Ausfuehrung eines oder mehrerer DRIVE-Programme bzw. Anwender-Teilprogramme als Auftragnehmer bei verteilter Verarbeitung wurde mit Fehlern beendet.

Maßnahme
Verteilinformationen, USING-Daten bzw. Programm-Ablauf ueberpruefen.

DRI0463 VORGANGSREGEL BEI VERTEILTER VERARBEITUNG VERLETZT

Bedeutung
Bei folgenden Situationen sind noch offene Auftragnehmer-Vorgaenge vorhanden:
- Anweisung 'DO PROCEDURE'
- Anweisung 'STOP' oder 'COMMIT WORK WITH STOP'
- Anweisung 'END PROCEDURE' in der obersten DRIVE-Programm-Stufe im obersten Auftraggeber-Vorgang einer verteilten Auftraggeber-Auftragnehmer-Hierarchie.

Maßnahme
Programmfluss im DRIVE-Programm ueberpruefen. Die Auftragnehmer muessen bei den genannten Anweisungen beendet sein.

DRI0464 TRANSAKTIONSREGEL BEI VERTEILTER VERARBEITUNG VERLETZT

Bedeutung
Ein oder mehrere DRIVE-Programme oder Anwender-Teilprogramme haben noch kein 'COMMIT WORK' durchlaufen bzw. kein Transaktionsende angefordert.

Maßnahme
Programmfluss im DRIVE-Programm ueberpruefen. Die Auftragnehmer muessen bei 'COMMIT WORK' ihre Transaktion abgeschlossen haben.

DRI0465 BOTTOM-UP-STRATEGIE BEI VERTEILTER VERARBEITUNG VERLETZT

Bedeutung
Ein DRIVE-Programm als Auftragnehmer wurde mit Transaktionsende-Anforderung aufgerufen.

Maßnahme
Verarbeitungsschritt im Teilprogramm mit 'PEND KP' abschliessen.

DRI0466 NOCH OFFENE AUFTRAGNEHMER MIT TRANSAKTIONSSTATUS 'P' VORHANDEN

Bedeutung
Beim fehlerhaften Programmabbruch eines Auftraggeber-DRIVE-Programms sind noch offene Auftragnehmer vorhanden, die bereits Transaktionsende angefordert haben.

Maßnahme
Die Anweisung 'COMMIT WORK' hinter CALL bzw. END DISPATCH verwenden, damit die Transaktionen in den Auftragnehmer-Umgebungen abgeschlossen werden.

DRI0467 AUFRUF ANWENDER-AUFTRAGNEHMER-VORGANG BEI 'PEND'-AUFTRAG FEHLERHAFT

Bedeutung
Beim Aufruf eines Anwender-Auftragnehmer-Vorgangs mit Aufforderung zum Beenden trat ein Fehler im Anwender-Teilprogramm auf oder es lag eine fehlerhafte Versorgung des Uebergabebereichs-Header durch das Anwender-Teilprogramm vor.

Maßnahme
Schnittstellen bzw. Verarbeitung der Anwender-Teilprogramme ueberpruefen.

DRI0468 ANWENDER-AUFTRAGNEHMER-VORGANG TROTZ 'PEND'-AUFTRAG NICHT BEENDET

Bedeutung
Beim Aufruf eines Anwender-Auftragnehmer-Vorgangs wurde die Aufforderung zum Beenden nicht beachtet.

Maßnahme
Schnittstellen bzw. Verarbeitung der Anwender-Teilprogramme ueberpruefen.

DRI0469 AUFTRAGNEHMER-TRANSAKTION TROTZ 'ROLLBACK'-AUFTRAG NICHT ZURUECKGESETZT

Bedeutung
Beim Aufruf eines Anwender-Auftragnehmer-Vorgangs wurde die Aufforderung zum Ruecksetzen der lokalen Transaktion nicht beachtet.

Maßnahme
Schnittstellen bzw. Verarbeitung der Anwender-Teilprogramme ueberpruefen.

DRI0470 VORGANGSSTATUS '(&00)', TRANSAKTIONSSTATUS '(&01)' VON DRIVE-VORGANG

Bedeutung
Nach Vorgangswiederanlauf bei verteilter Verarbeitung liegt von einem DRIVE-Auftragnehmer eine Statusinformation vor.
(&00): Auftragnehmer-Vorgangsstatus
(&01): Auftragnehmer-Transaktionsstatus

Maßnahme
Programm-Ablauf im Auftragnehmer-DRIVE-Programm ueberpruefen.

DRI0471 VORGANGSSTATUS '(&00)', TRANSAKTIONSSTATUS '(&01)' VON ANWENDER-TAC '(&02)'

Bedeutung
Nach Vorgangswiederanlauf bei verteilter Verarbeitung liegt von einem Anwender-Auftragnehmer-Vorgang eine Statusinformation vor.
(&00): Auftragnehmer-Vorgangsstatus
(&01): Auftragnehmer-Transaktionsstatus
(&02): Name des UTM-Transaktionscodes

Maßnahme
Programm-Ablauf im Auftragnehmer-Anwender-Teilprogramm ueberpruefen.

DRI0472 VORGANGSSTATUS '(&00)', TRANSAKTIONSSTATUS '(&01)' VON AUFTRAGNEHMER-VORGANG

Bedeutung
Nach Vorgangswiederanlauf bei verteilter Verarbeitung liegt von einem Auftragnehmer-Vorgang eine Statusinformation vor. Der Auftragnehmer-Vorgang wurde in der zurueckgesetzten verteilten Transaktion erstmals addressiert. Es liegen keine weiteren Auftragnehmer-Informationen vor.
(&00): Auftragnehmer-Vorgangsstatus
(&01): Auftragnehmer-Transaktionsstatus

Maßnahme
Auftragnehmer-Vorgaenge in verteilter Anwendung ueberpruefen.

DRI0473 AUFTRAGNEHMER-STATUSINFORMATIONEN NACH VORGANGSWIEDERANLAUF ERHALTEN

Bedeutung
Nach Vorgangswiederanlauf bei verteilter Verarbeitung liegt von einem Auftragnehmer-Vorgang eine Statusinformation vor.

Maßnahme
Ergaenzende DRIVE-Meldungen beachten.

DRI0474 ANWEISUNG NUR IN PROGRAMMSTUFE DES DISPATCH-BLOCKS ZULAESSIG

Bedeutung
Ein innerhalb eines DISPATCH-Blocks ueber lokales CALL PROCEDURE gerufenes DRIVE-Programm darf selbst keinen REMOTE CALL enthalten.

Maßnahme
REMOTE CALL in Uebersetzungseinheit des DISPATCH-Blocks verlagern.

DRI0475 UNZULAESSIGER DATENTYP BEI REMOTE-VERARBEITUNG

Bedeutung
An ein ueber CALL gerufenes DRIVE-Programm, das remote ausgefuehrt wird, koennen nur folgende Datentypen als USING-Parameter uebergeben werden:
- Literale
- einfache Variablen
- DRIVE-Ausdruecke
- Vektoren, Matrizen

Maßnahme
Schnittstellen bzw. Verarbeitung der Anwender-Teilprogramme ueberpruefen.

DRI0476 BEREITS EIN SERVER ADRESSIERT

Bedeutung
Bei einem Remote-Zugriff auf einen UTM-Server ist bereits ein anderer Server adressiert worden. Es ist zu einem Zeitpunkt nur eine Adressierung moeglich.

Maßnahme
Programmablauf ueberpruefen. Vor Remote-Zugriff den zuvor adressierten UTM-Server-Vorgang beenden. Generell schliessen sich Remote-Zugriffe auf SESAM- bzw. UDS-Datenbanken im BS2000 und Remote-CALLs innerhalb einer Session von DRIVE/WINDOWS aus.

DRI0477 UNGLEICHER SERVER-STAND ZUM LETZTEN SICHERUNGSPUNKT

Bedeutung
Zum Zeitpunkt eines ROLLBACK WORK WITH RESET in Client-Umgebung ist ein anderer Server adressiert als zum Zeitpunkt des letzten COMMIT WORK. Der Server-Stand des letzten COMMIT WORK kann nicht wiederhergestellt werden. Ein Ruecksetzen ist daher nicht moeglich.

Maßnahme
Programmablauf ueberpruefen. ROLLBACK WORK WITH RESET in Client-Umgebung moeglichst vermeiden.

DRI0478 WIEDERANLAUF IN SERVER-BETRIEB NICHT MOEGLICH

Bedeutung
In Server-Umgebung (nicht VTV) wurde die Anweisung ROLLBACK WORK WITH RESET durchlaufen oder von der lokalen Datenbank im Server wurde ein "ta_cancelled" gemeldet. Da von UTM bei einer Client-Server-Verbindung ohne VTV kein Vorgangswiederanlauf unterstuetzt wird, kann DRIVE/WINDOWS in dieser Situation das Server-Programm nicht zuruecksetzen.

Maßnahme
Ablauf und Zugriffe auf die lokalen Datenbestaende im Server-Programm ueberpruefen.

DRI0479 WIEDERANLAUF WIRD NICHT UNTERSTUETZT

Bedeutung
Die aktuelle DRIVE-Version unterstuetzt kein Wiederanlaufverhalten.

Maßnahme
Programm-Source ueberpruefen; die Anweisung ROLLBACK WORK WITH RESET ist nicht erlaubt.

DRI0480 FEHLER IM OLDSTYLE-PROGRAMM AUFGETRETEN

Bedeutung
Per Newstyle-CALL wurde ein Oldstyle-Programm gestartet, bei dessen Ablauf ein Fehler aufgetreten ist.

Maßnahme
Explizit in den Oldstyle wechseln und Programm analysieren und austesten.

DRI0481 UNZULAESSIGES (&00) IN EINEM MIT CALL GERUFENEN OLDSTYLE-PROGRAMM

Bedeutung
Per Newstyle-CALL wurde ein Oldstyle-Programm gestartet, das die angegebene unzulaessige Anweisung enthaelt.
(&00): DO, STOP

DRI0488 NOCH VORHANDENE DYNAMISCHE TEMPORAERE VIEWS WURDEN FREIGEGEBEN

Bedeutung
Vor Verlassen eines mit DO aufgerufenen Programms duerfen keine temporaeren, im Programm-Modus definierten Views mehr existieren, d.h. vor den Anweisungen END PROCEDURE, STOP oder einer Folge-DO-Anweisung muss dynamisch eine DROP TEMPORARY VIEWS-Anweisung durchlaufen worden sein.
Falls doch noch temporaere Programm-Views existieren, werden sie von DRIVE beim Datenbanksystem geloescht und es wird diese Meldung ausgegeben.

Maßnahme
Im Dialog-Modus DROP TEMPORARY VIEWS eingeben und anschliessend in den betroffenen Programmen mit temporaeren Views geeignete DROP TEMPORARY VIEWS-Anweisungen vorsehen.

DRI0489 ITEM-ANGABE MIT GENAU EINEM ITEM ERFORDERLICH

Bedeutung
Bei Einfachauswahlfeldern und der Anweisung SET SCREEN ATTRIBUTE PRESELECT muss 'ITEM' angegeben werden, darf aber nur ein Item enthalten.

Maßnahme
Falls ITEM-Angabe fehlt, nachtragen.
Falls mehr als ein Item angegeben, Items entfernen.

DRI0490 SESAMSQL UEBERSETZUNG BEI OFFENER TRANSAKTION NICHT MOEGLICH

Bedeutung
Es wird versucht, ein Programm mit SQL-Anweisungen ungleich COMMIT / ROLLBACK zu uebersetzen, obwohl bereits eine Transaktion offen ist.

Maßnahme
Das Programm separat uebersetzen und CALL- und CODE-Element ausfuehren oder vor einem CALL-Aufruf Transaktion beenden.

DRI0491 MAXIMALE ANZAHL ERLAUBTER DEKLARATIONEN UEBERSCHRITTEN

Bedeutung
Das DRIVE-Programm enthaelt zuviele Deklarationen einer Objektart, z.B. mehr als 20 DECLARE FILE ...

Maßnahme
Masshalten mit Deklarationen einer Objektart.

DRI0492 FORMAT IST KEIN 'FHS-DE'-FORMAT

Bedeutung
Die Anweisung kann nur mit FHS-DE-Formaten ausgefuehrt werden.

Maßnahme
Anderes Format verwenden oder die Eigenschaften dieses Formates mit IFG aendern.

DRI0493 WECHSEL DES FORMATTYPS NICHT ERLAUBT

Bedeutung
Alle Teilformate muessen vom gleichen Typ sein. Mischen von FHS-DE-Formaten mit Nicht-DE-Formaten ist nicht moeglich.

Maßnahme
Anderes Format verwenden oder die Eigenschaften dieses Formates mit IFG aendern.

DRI0494 FUER DIE (&00)-ANGABE SIND NUR POSITIVE WERTE ERLAUBT

Bedeutung
Es wurde bei ITEM oder LINES ein negativer Wert oder 0 angegeben
(&00): ITEM- oder LINES-Angabe

Maßnahme
Bei ITEM oder LINES nur positive Werte angeben.

DRI0495 DER (&00)-WERT HAT DIE ANZAHL (&01) UEBERSCHRITTEN

Bedeutung
(&00): LINES: Bei der Anweisung SET SCREEN ATTRIBUTE LINES hat der LINES-Wert die Anzahl der definierten Listenzeilen der Liste ueberschritten.
ITEM: - Bei SET SCREEN ATTRIBUTE PRESELECT ON/OFF ITEM hat der ITEM-Wert die Anzahl der definierten Listenzeilen der Liste ueberschritten.
- Bei SET SCREEN ATTRIBUTE LOCK/PRESELECT ON/OFF ITEM hat der ITEM-Wert die Anzahl der definierten Auswahlfelder des Einfachauswahlfeldes ueberschritten.
(&01): bei LINES: Listenelemente
bei ITEM: Listenelemente oder Auswahlfelder

Maßnahme
LINES- oder ITEM-Angabe ueberpruefen.

DRI0496 NEXT OHNE VORHERGEHENDES FIRST

Bedeutung
Es wurde ein GET NEXT MODIFIED INDEX auf ein Format gemacht, auf das es noch kein GET FIRST MODIFIED INDEX gab.

Maßnahme
- Pruefen, ob das richtige Format angegeben wurde.
- Zuerst ein GET FIRST MODIFIED INDEX auf das Format ausfuehren.

DRI0497 FUER DIE SCROLL-ANGABE SIND HOECHSTENS 4 ZEICHEN ERLAUBT

DRI0498 UNZULAESSIGE ZEICHEN BEI DER SCROLL-ANGABE

Bedeutung
Es wurden Zeichen ungleich '<', '>', '+', '-' oder Leerzeichen bei SCROLL angegeben.

Maßnahme
Fuer SCROLL nur die Zeichen '<', '>', '+', '-' oder Leerzeichen verwenden.

DRI0499 ZUGRIFF AUF DATEI NICHT MOEGLICH.

Bedeutung
Unvertraeglicher OPEN-verbietet den Zugriff auf die Datei.

Maßnahme
Datei mit dem entsprechenden Modus eroeffnen.

DRI0500 BEENDEN SIE BITTE DRIVE, BEVOR SIE WINDOWS BEENDEN

Bedeutung
Eine DRIVE-Anwendung ist noch aktiv.. Windows kann nur verlassen werden, wenn diese vorher beendet wird.

DRI0501 (&00)(&01)

Bedeutung
Fehler bei der Definition eines Reports entsprechend dem Meldungstext
(&00): Reportfehlernummer, siehe entsprechende DRIVE-Fehlernummer
(&01): Meldungstext

Maßnahme
Korrektur der Definition

DRI0502 RESOURCE IST NICHT DEFINIERT

Maßnahme
Name der Resourcedatei entweder in der Anweisung OPTION oder in der Anweisung DECLARE WINDOW angeben.

DRI0503 VERWENDUNG DES PRAEFIXES 'DRI#.' FUER DATEINAMEN UNZULAESSIG

Bedeutung
Das Praefix 'dri#.' ist fuer DRIVE-Zwecke reserviert

Maßnahme
Datei umbenennen;

DRI0504 ANZAHL DER ATTRIBUTE UNGLEICH ANZAHL DER VARIABLEN

Bedeutung
Anzahl der DRIVE-Variablen entspricht nicht der Anzahl der WINDOW-/INFORMIX-Attribute.

Maßnahme
Liste der angegebenen Attribute mit Liste der angegebenen Variablen vergleichen und korrigieren.

DRI0505 OBJEKT MUSS VOM TYP '(&00)' SEIN

Bedeutung
Anweisung kann mit dem angegebenen Objekt nicht ausgefuehrt werden.
(&00): - Auswahlliste
- Eingabefeld
- Eingabefeld/ Gruppe mit Eingabefeld(ern)

DRI0506 DAS ATTRIBUT IST HIER NICHT ZULAESSIG

Bedeutung
Bei DECLARE WINDOW: die Angabe REVERSE ON ist in Verbindung mit BACKGROUND bzw. FOREGROUND COLOUR unzulaessig.

DRI0507 (&00) BEREITS VORHANDEN

Bedeutung
(&00): Benutzerkennsatz:
Zu dem angegebenen Transaktionscode und Usernamen ist bereits ein Benutzerkennsatz vorhanden.
(&00): User:
Der ueber die PARAMETER-Anweisung eingegebene USER-Name wurde bereits vergeben

Maßnahme
Anderen Namen fuer Transaktionscode oder Benutzer verwenden. USER-Namen aendern. Nach einem Absturz von DRIVE die Datei DRIUSERNAME unter /tmp loeschen.

DRI0508 CONSTRAINT IM DD VERLETZT, IDDS-STATUS: (&00), (&01)

Bedeutung
Eine Wertigkeitsbedingung wurde verletzt. Moegliche Ursachen:
- Daten inkonsistent
- Installationsfehler
- interner Ablauffehler

(&00): Fehlernummer
(&01): Fehlertext

Maßnahme
Anhand der IDDS-Fehlernummer genaue Ursache ermitteln. Datenkonstellation im DD pruefen. Vom ERMS-Administrator pruefen lassen, ob eine falsche Constraint in den ausgelieferten Command Files definiert wurde.

DRI0509 ENTITY '(&00)' '(&01)' KANN NICHT ANGELEGT WERDEN

Bedeutung
Das Entity ist bereits im DD in einer Partition vorhanden, auf die DRIVE keinen Zugriff hat.
Falls Entitytyp = CAL, RLS : nur Name des gerufenen Programms.
Falls Entitytyp = RES : Name auf 32 Stellen gekuerzt.
(&00): Entitytyp
(&01): Entityname

Maßnahme
Anhand von Entitytyp und -name feststellen, wo der Fehler auftrat. Daten im DD bereinigen.

DRI0510 ENDE DER ERGEBNISTABELLE ERREICHT BZW. ERGEBNISTABELLE LEER

Bedeutung
INFORMIX-Anfrage liefert keinen oder keinen weiteren Treffer

DRI0511 WECHSEL DES DB-SYSTEMS NUR IM TRANSAKTIONSLOSEN ZUSTAND MOEGLICH

Bedeutung
Beim DB-Systemwechsel muss die Transaktion geschlossen sein.

DRI0512 AUFRUF DER 'DRIVE-SPU' NICHT ERLAUBT

Bedeutung
Die DRIVE-SPU darf nicht aus der DRIVE-SPU heraus aufgerufen werden.

DRI0513 DRIVE-PROZESSE PASSEN NICHT: '(&00)', '(&01)'

Bedeutung
Verschiedene, nicht zueinander passende DRIVE-Prozesse wurden installiert.
(&00): Version des Sendeprozesses
(&01): Version des Empfangsprozesses

Maßnahme
DRIVE-Installation vom Systemverwalter ueberpruefen lassen, ggfs. wiederholen.

DRI0514 FEHLER BEI '(&00)' (&01) (&02)

Bedeutung
Bei einem Systemmakro ist ein Fehler aufgetreten
(&00): Name des Systemmakros
(&01): Fehlernummer
(&02): Fehlertext

Maßnahme
Systemverwalter benachrichtigen

DRI0515 PROZESS-START NICHT MOEGLICH (&00) (&01)

Bedeutung
Entweder wurde die zulaessige Anzahl der Prozesse (systemweit oder fuer den Benutzer) ueberschritten oder es trat ein Speicherengpass auf.
(&00): Fehlernummer
(&01): Fehlertext

Maßnahme
Systemverwalter konsultieren oder z.B. beim DRIVE-SPU weniger Prozesse starten.

DRI0516 PROZESS '(&00)' KANN NICHT GESTARTET WERDEN : (&01)

Bedeutung
Fuer die aufgefuehrte DRIVE-Funktion ist kein DRIVE-Prozess installiert.
(&00): Prozessname
(&01): Fehlerursache

Maßnahme
DRIVE-Installation vom Systemverwalter ueberpruefen lassen, falls der Prozess vorhanden sein muss.

DRI0517 INKREMENTWERT IST 0 ODER DER NULL-WERT

Bedeutung
Zum Ausfuehrungszeitpunkt darf in einer CYCLE FOR-Schleife der Inkrementwert nicht 0 oder der NULL-Wert sein.

Maßnahme
Inkrementwert einen zulaessigen Wert zuweisen.

DRI0518 '(&00)' IST KEIN(E) (&01)

Bedeutung
(&00): Name der Datei bzw. des Dateiverzeichnisses
(&01): Datei oder Dateiverzeichnis

Maßnahme
Vorhandene Datei loeschen oder Dateiverzeichnis aendern und Anweisung erneut ausfuehren. Namen aendern;

DRI0519 FALSCHE EINGABE FUER (&00)

Bedeutung
(&00): Dateiverzeichnis: Es wurde bei der PARAMETER-Anweisung kein gueltiges Dateiverzeichnis angegeben.
Datei: Der angegebene Dateiname entspricht nicht den SINIX-Konventionen.

Maßnahme
Gueltiges Dateiverzeichnis angeben und Anweisung wiederholen.
Gueltigen Dateinamen angeben und Anweisung wiederholen.

DRI0520 DD-ZUSTAND DARF NICHT VERAENDERT WERDEN

Bedeutung
Der Parameterwert fuer DD soll auf OFF gesetzt werden, am Arbeitsplatz ist aber das Klassenfenster fuer Benutzerkennsaetze geoeffnet.

Maßnahme
Klassenfenster fuer Benutzerkennsaetze schliessen.

DRI0521 KONTROLLVARIABLE IN MEHREREN 'CYCLE-FOR'-STUFEN VERWENDET

Bedeutung
Die Kontrollvariable einer CYCLE-FOR-Schleife darf in keiner anderen CYCLE-FOR-Stufe benutzt oder veraendert werden.

Maßnahme
Bei Verwendung mehrerer CYCLE-FOR-Stufen fuer jede Stufe eine eigene Kontrollvariable definieren.

DRI0522 PARAMETERANZAHL VON SENDENDEM UND EMPFANGENDEM SCRIPT UNGLEICH

Bedeutung
Bei sendendem und empfangendem Script muss die Anzahl der Parameter uebereinstimmen

DRI0523 (&00). PARAMETER BEI SENDENDEM UND EMPFANGENDEM SCRIPT UNVERTRAEGLICH

Bedeutung
Der Datentyp des sendenden Scripts ist nicht auf den Datentyp des empfangenden Scripts zuweisbar.
(&00): Position des fehlerhaften Parameters

DRI0524 RETURN-ANGABE BEIM (&00). PARAMETER UNGLEICH

Bedeutung
RETURN-Angabe angleichen, d. h. bei sendendem und empfangendem Script muessen die RETURN-Angaben bei den jeweiligen Parametern gleich sein.
(&00): Position des fehlerhaften Parameters

DRI0525 OPTION '(&00)' MIT '(&01)' UNVERTRAEGLICH

Bedeutung
Anweisung oder Anweisungsteil fuer die Arbeit mit dem eingestellten DB-System nicht erlaubt

DRI0526 AKTUAL- UND FORMALPARAMETER SIND NICHT ZUWEISUNGSVERTRAEGLICH

Bedeutung
Der Datentyp eines Formalparameters ist nicht auf den Aktualparameter zuweisbar.

Maßnahme
Der Formalparameter muss an die aktuelle Schnittstellendefinition angepasst werden.

DRI0527 REMOTE-ZUGRIFF IM 1. DIALOGSCHRITT NICHT MOEGLICH; BITTE 'DUE' EINGEBEN

Bedeutung
Remote-Zugriff ist erst nach der ersten Bildschirmausgabe moeglich

Maßnahme
Programm neu starten

DRI0528 PFADNAME DES PROGRAMS ZU LANG

Bedeutung
Der Pfadname wurde bei der Installation zu lang angegeben

Maßnahme
DRIVE so installieren, dass der Pfadname kleiner als 254 Zeichen bleibt.

DRI0529 BUFFER-LAENGE NICHT AUSREICHEND

Bedeutung
Wenn der DIALOGBUILDER den DRIVE-Buffer fuellt, darf dieser nicht dynamisch erweitert werden.

Maßnahme
Buffer vergroessern und DRIVE erneut starten

DRI0530 EREIGNIS-OBJEKT-PAAR NICHT EINDEUTIG

Bedeutung
In einem SCRIPT sind mehrere identische ON-Bedingungen aufgefuehrt.

Maßnahme
Nur eindeutige ON-Bedingungen verwenden.

DRI0531 NAME DES DATA DICTIONARIES IST ZU LANG

Bedeutung
Ueber die Umgebungsvariable $DRIVE_DD wurde ein Name angegeben, der laenger als 10 Stellen ist.

Maßnahme
Maximal zehnstelligen Namen verwenden

DRI0532 UNZULAESSIGE EXPONENTIALMASKE

Bedeutung
Die Exponentialmaske fuer einen numerischen Datentyp ist falsch definiert worden. Gueltige Maske:
En, mit n > 8 und n < 25

Maßnahme
Maske entsprechend angeben

DRI0533 FEHLER BEIM INTERNEN WECHSEL/SCHLIESSEN DER INFORMIX-DATENBANK

Bedeutung
Aktuelle Datenbank ist noch gueltig. Fehlerursache siehe weitere Fehlermeldung.

DRI0534 LAUFZEITFEHLER '(&00)' BEI AUFRUF VON '(&01)'

Bedeutung
Ein PASCAL-Laufzeitfehler ist aufgetreten, welcher durch einen Schnittstellen- oder Codierfehler im fremdsprachigen Unterprogramm ausgeloest wurde.
(&00): PASCAL-Laufzeitfehler
(&01): Modulname

Maßnahme
- Genaue Spezifizierung der PASCAL-Laufzeitfehler im PASCAL-Benutzerhandbuch unter dem Kapitel "Laufzeitfehler und ihre Bedeutung".
- Laufzeitfehler durch beheben des Ablauffehlers im externen Unterprogramm beseitigen.

DRI0535 SCRIPT-DEKLARATION FUER '(&00)' FEHLT

Bedeutung
In der PROC-Anweisung gibt es eine SCRIPT-Klausel fuer die keine SCRIPT-Deklaration existiert.
(&00): Window-Name

Maßnahme
SCRIPT-Klausel in PROC-Anweisung streichen oder zu genanntem Window eine SCRIPT-Deklaration einbringen.

DRI0536 (&00)(&01)(&02)

Bedeutung
Bei einer SQL-Anweisung an das Datenbanksystem INFORMIX ist ein Fehler aufgetreten. Der Meldungstext hat folgenden Aufbau:
<INFORMIX-Fehlermeldungsnummer>[<(C-ISAM-Fehlercode>)]
<INFORMIX-Fehlermeldungs-text mit bis zu einem Einfuegetext>.

Maßnahme
Bedeutung und Massnahme der INFORMIX-Fehlermeldung ist dem Benutzerhandbuch "Fehlermeldungen fuer INFORMIX-Produkte" zu entnehmen.

DRI0537 '(&00)'-UMGEBUNGSVARIABLE NICHT BZW. FALSCH VERSORGT

Bedeutung
Die Umgebungsvariablen $SQLEXEC und $INFORMIXDIR fuer INFORMIX sind nicht oder falsch versorgt.
(&00): INFORMIX

Maßnahme
Umgebungsvariable (richtig) versorgen.

DRI0538 ANGABE (&00) IM PROGRAMMBLOCK (&01) UNZULAESSIG

Bedeutung
Diese Angabe ist in diesem Programmblock nicht erlaubt.

Maßnahme
Anweisung aendern/steichen.

DRI0539 PROZESS '(&00)' NICHT MEHR VORHANDEN

Bedeutung
Der genannte Prozess hat sich bereits beendet. Die Ausfuehrung der letzten Anweisung wurde abgebrochen. Moegliche Prozesstypen sind:
(&00): dri_mo: Monitorprozess
dri_op: Oberflaechenprozess
dri_dd: DD-Prozess
dri_dz: DD-Prozess fuer Benutzerkennsaetze;
Bei bereits erfolgter Beendigung des Monitorprozesses wird die SPU beendet. Ist der Prozess dri_dz bereits beendet, so wird das Klassenfenster fuer Benutzerkennsaetze geschlossen.

Maßnahme
Ein erneutes Ausfuehren der letzten Anweisung ist nicht mehr sinnvoll, mit DRIVE kann aber weitergearbeitet werden. Feststellen, warum der Prozess sich beendet hat.

DRI0540 BITTE QUITTIEREN

Maßnahme
Meldebox mit OK quittieren.

DRI0541 '(&00)' FEHLERFREI UEBERSETZT

Bedeutung
Das angegebene Programm wurde fehlerfrei uebersetzt.
(&00): Programm

DRI0542 '(&00)' ORDNUNGSGEMAESS GELOESCHT

Bedeutung
Das angegebene Programm wurde geloescht.
(&00): Programmname mit ggfs. allen zugehoerigen Objekten wie Liste, Zwischencode, etc.

DRI0543 FEHLENDE 'DATABASE'-ANWEISUNG IN UTM-STARTDATEI

Bedeutung
Informix-Programme koennen im UTM-Betrieb nur ausgefuehrt werden, wenn in der UTM-Startdatei die gleiche Datenbank bekanntgegeben wird.

Maßnahme
UTM-Startdatei aendern

DRI0544 '(&00)' FEHLERFREI VORUEBERSETZT

Bedeutung
Das angegebene Programm wurde fehlerfrei vom Precompiler (OPTION PRECOMPILE=ONLY) voruebersetzt
(&00): Programmname
Die Ergebnisdatei mit dem Suffix '.i' wurde erzeugt.

DRI0545 KEINE STARTDATEI VORHANDEN

Bedeutung
Eine Startdatei muss in folgenden Faellen zwingend existieren:
- BATCH-Betrieb von DRIVE;
- fuer DRIVE-RTS-Variante

Maßnahme
Startdatei erstellen

DRI0546 STARTDATEI FEHLERHAFT

Bedeutung
Die Startdatei enthaelt Fehler.

Maßnahme
Startdatei korrigieren

DRI0547 'DRIVE' MIT 'EXIT' ABGEBROCHEN

Bedeutung
Bei Ablauf als DRIVE-RTS-Variante enthaelt das auszufuehrende Programm eine EXIT-Anweisung

Maßnahme
Wenn Abbruch nicht erwuenscht, EXIT-Anweisung entfernen.

DRI0548 'DRIVE' MIT 'BREAK' ABGEBROCHEN

Bedeutung
Bei Ablauf als DRIVE-RTS-Variante enthaelt das auszufuehrende Programm eine BREAK-Anweisung.

Maßnahme
Wenn Abbruch nicht erwuenscht, BREAK-Anweisung entfernen.

DRI0549 ANWEISUNG ABGEBROCHEN;(EXIT=DRIVE BEENDET)

Bedeutung
Programmanalyse bzw. Programmablauf war fehlerhaft

Maßnahme
Vorgeschlagene EXIT-Anweisung mit <DUE> quittieren. DRIVE wird dann ordnungsgemaess beendet.

DRI0550 ANWEISUNG IN DRIVE-RTS-VARIANTE NICHT ERLAUBT

Bedeutung
Bei Ablauf als DRIVE-RTS-Variante ist in der Startdatei nur die DO-Anweisung erlaubt. Auf den Prompting-Text darf nur mit EXIT geantwortet werden.

Maßnahme
Unzulaessige Anweisung entfernen; EXIT bei Prompting-Text quittieren.

DRI0551 FEHLER BEIM KOPIEREN DER 'DRIVE'-FORMATDATEIEN

Bedeutung
Das Erstellen der Sicherungskopie der DRIVE-Formatdatei 'D@<usern>' bzw. das Rueckkopieren der Sicherungskopien 'D@<usern>.sav' auf die entsprechenden DRIVE-Formatdateien ist fehlgeschlagen. Moegliche Gruende: Speicherengpass, fehlende Zugriffsrechte

Maßnahme
Die Zugriffsrechte der DRIVE-Formatdateien sowie ihrer Sicherungskopien ueberpruefen ('read + write' noetig) oder genuegend Plattenspeicherplatz zur Verfuegung stellen.

DRI0552 FOLGEFEHLER BEI DATEIVERARBEITUNG

Bedeutung
Bei einer DISPLAY LIST-Anweisung tritt ein Fehler bei der Dateiverarbeitung auf. Als Reaktion darauf soll eine Fehlerliste geschrieben werden. Hierbei tritt der gleiche Fehler nochmals auf.

Maßnahme
Die Ursache, die zum ersten Fehler fuehrt beheben und Programm erneut starten.

DRI0553 BITTE DEBUG-ANWEISUNG EINGEBEN

Bedeutung
DRIVE befindet sich im Debug-Modus und erwartet die Eingabe einer Debuganweisung.

Maßnahme
Debug-Anweisung eingeben

DRI0554 DEBUG-ANWEISUNG AUSGEFUEHRT

Bedeutung
DRIVE hat die zuletzt eingegebene Debug-Anweisung ordnungsgemaess ausgefuehrt und erwartet erneut die Eingabe einer Debug-Anweisung.

Maßnahme
Naechste Debug-Anweisung eingeben.

DRI0555 UNGUELTIGE ZEILENNUMMER (&00)

Bedeutung
In einer AT-Anweisung wurde eine ungueltige Zeilennummer spezifiziert.
(&00): Zeilennummer

Maßnahme
Gueltige Zeilennummer ermitteln fuer neue AT-Anweisung

DRI0556 UNGUELTIGER ZEILENNUMMERNBEREICH '(&00)-(&01)'

Bedeutung
In einer AT-Anweisung wurde ein ungueltiger Zeilennummernbereich spezifiziert.
(&00): Anfangszeilennummer
(&01): Endzeilennummer

Maßnahme
Gueltigen Zeilennummernbereich ermitteln fuer neue AT-Anweisung.

DRI0557 ALPHA-AUSGABEN SIND OHNE FORMANT NICHT MOEGLICH

Bedeutung
DRIVE wurde mit dem Schalter '-w' gestartet. Hierbei sind nur Graphik- Ausgaben, aber keine Alpha-Ausgaben moeglich.
Hinweis: Die anstehende Alpha-Ausgabe kann auch durch einen Fehler in der Startdatei verursacht werden.

Maßnahme
Bei Start von DRIVE mit '-w' keine DRIVE-Programme mit Alpha-Ausgaben ausfuehren.

DRI0558 DATEI: '(&00)' FEHLER: (&01)

Bedeutung
Bei der Bearbeitung der angegebenen Datei trat der angegebene Fehler auf.
(&00): Dateiname
(&01): Fehlermeldung des Betriebssystems

Maßnahme
Angegebenen Fehler analysieren.

DRI0559 '(&00)' NICHT VORHANDEN: (&01)

Bedeutung
Angegebene Datei ist nicht vorhanden.
(&00): Dateiname
(&01): Fehlermeldung des Betriebssystems

Maßnahme
Angegebene Datei zur Verfuegung stellen.

DRI0560 '(&00)' IST GESPERRT: (&01)

Bedeutung
Auf die angegebene Datei kann nicht zugegriffen werden.
(&00): Dateiname
(&01): Fehlermeldung des Betriebssystems

Maßnahme
Zugriff auf die angegebene Datei ermoeglichen.

DRI0561 BITTE USING-PARAMETER VERSORGEN

Bedeutung
Der Anfangs-Haltepunkt wurde erreicht und es sind nicht alle USING-Parameter versorgt.

Maßnahme
Ueber SET-Anweisungen alle USING-Parameter versorgen oder mit BREAK DEBUG die Debug-Sitzung beenden.

DRI0562 NOCH (&00) USING-PARAMETER NICHT VERSORGT

Bedeutung
Am Anfangshaltepunkt sind noch die angegebene Anzahl USING-Parameter unversorgt geblieben.
(&00): Anzahl

Maßnahme
Ueber SET-Anweisungen die restlichen USING-Parameter versorgen oder mit BREAK DEBUG die Debug-Sitzung beenden.

DRI0563 USING-PARAMETER VOLLSTAENDIG VERSORGT

Bedeutung
Am Anfangshaltepunkt sind alle USING-Parameter versorgt.

Maßnahme
Beliebige Debuganweisung eingeben.

DRI0564 UNZULAESSIGE ANWEISUNG BEI PARAMETER-PROMPTING

Bedeutung
Waehrend eines Parameter-Promptings wurde eine Debuganweisung ungleich SET und ungleich BREAK DEBUG eingegeben.

Maßnahme
Parameter-Prompting beenden.

DRI0565 'SET'-ANWEISUNG NUR FUER USING-PARAMETER VON 'DO/CALL' ERLAUBT

Bedeutung
Am Anfangs-Haltepunkt sind noch nicht alle USING-Parameter versorgt und es wurde eine SET-Anweisung auf eine andere Variable versucht.

Maßnahme
USING-Parameter versorgen oder BREAK DEBUG eingeben

DRI0566 'SET'-ANWEISUNG MIT VARIABLER WERTZUWEISUNG NICHT ERLAUBT

Bedeutung
Am Anfangs-Haltepunkt sind noch nicht alle USING-Parameter versorgt und es wurde eine SET-Anweisung mit variabler Wertzuweisung versucht.

Maßnahme
USING-Parameter versorgen oder BREAK DEBUG eingeben.

DRI0567 ANWENDUNG MIT UNZULAESSIGER DATEI VERKNUEPFT

Bedeutung
Beim Start des DRIVE-Kernprozesses aus dem Dateimanager darf der DRIVE- Kernprozess nur mit einer Startdatei (Endung '.dri') oder einem Codeelement (Endung '.drx') verknuepft werden.

Maßnahme
Unzulaessige Verknuepfung berichtigen.

DRI0568 DEBUGAUSGABE ZU GROSS

Bedeutung
Bei einem Trace soll eine Anweisung ausgegeben werden, deren Zeilenanzahl (im Listingelement) zu gross ist; am Ende muessen daher einige Zeilen weggelassen werden.

Maßnahme
<ENTER>-Taste druecken

DRI0569 ZU VIELE FEHLERMELDUNGEN

Bedeutung
Es sind mehr Fehler aufgetreten, als ausgegeben werden koennen. Die Ausgabe von einigen der zuletzt aufgetretenen Fehler muss daher unterdrueckt werden.

Maßnahme
Naechste Debuganweisung eingeben

DRI0570 KEINE DEBUG-ANWEISUNG GEFUNDEN

Bedeutung
An einem Haltepunkt wurde nur die <ENTER>-Taste gedrueckt, anstatt eine Debug-Anweisung einzugeben.

Maßnahme
Gueltige Debug-Anweisung eingeben

DRI0571 ZU VIELE FEHLER ANALYSIERT

Bedeutung
Bei der Analyse eines DRIVE-Programms wurden mehr Fehler gefunden als in der internen Fehlertabelle eingetragen werden koennen. Moegliche weitere Fehler werden nicht mehr angezeigt.

Maßnahme
Die gemeldeten Fehler korrigieren und nochmals uebersetzten.

DRI0572 EXTENSION '(&00)' UNZULAESSIG

Bedeutung
Fuer DRIVE-Programme sind nur die Extensionen DRP oder DRX erlaubt.
(&00): Extension

Maßnahme
DRIVE-Programm umbenennen.

DRI0573 SPRUNG ZUM END-HALTEPUNKT

Bedeutung
Im Debug-Modus wurde nach Eingabe einer BREAK-Anweisung oder nach Druecken einer BREAK-Taste zum End-Haltepunkt verzweigt.

Maßnahme
BREAK DEBUG, DISPLAY FORM oder DISPLAY LIST eingeben.

DRI0574 BEREITS AM END-HALTEPUNKT

Bedeutung
Am End-Haltepunkt wurde eine nicht zugelassene Anweisung eingegeben. Zugelassen sind BREAK DEBUG, DISPLAY FORM und DISPLAY LIST.

Maßnahme
Zugelassene Debug-Anweisung eingeben

DRI0575 END-HALTEPUNKT ERREICHT

Bedeutung
Die Verarbeitung hat die END PROCEDURE-Anweisung der obersten Programmstufe (mit DEBUG aufgerufenes Programm) abgearbeitet und nach dieser Anweisung angehalten.

Maßnahme
Debug-Anweisung eingeben

DRI0576 ANFANGS-HALTEPUNKT ERREICHT

Bedeutung
Nach Eingabe der Anweisung DEBUG im Dialog wurde der Anfangs-Haltepunkt erreicht. Die Verarbeitung hat die PROCEDURE-Anweisung des mit DEBUG gerufenen Programms abgearbeitet und nach dieser Anweisung angehalten.

Maßnahme
Debug-Anweisung eingeben

DRI0577 ANFANGS-HALTEPUNKT ERREICHT - AUSFUEHRUNG VON DEBUG-ANWEISUNGEN

Bedeutung
Der Debugger wurde aus der SPU gestartet und die dort angegebenen Debuganweisungen werden ausgefuehrt. Anschliessend beginnt der Debug-Lauf mit oder ohne Ablaufverfolgung, je nachdem ob in der SPU eine TRACE-Anweisung abgesetzt wurde oder nicht.

DRI0578 AKTUELLER HALTEPUNKT: ZEILE (&00) IN PROZEDUR '(&01)'

Bedeutung
Der angegebene Haltepunkt wurde erreicht, d. h. die Programmausfuehrung steht vor einer Anweisung in der angegebenen Zeile der angegebenen Prozedur. Bei dieser Anweisung ggf. hinterlegte DEBUG-Aktionen vom Typ DISPLAY oder SET wurden bereits ausgefuehrt.
(&00): Zeilennummer
(&01): Prozedurname (incl. Bibliotheksangabe)

Maßnahme
Debug-Anweisung eingeben

DRI0579 FEHLER BEI AUSFUEHRUNG DER ANWEISUNG '(&00)'

Bedeutung
Es ist ein Ablauffehler aufgetreten beim Ausfuehren einer Programmanweisung oder einer dort hinterlegten Debug-Aktion. Die zugehoerige Zeilennummer und der Programmname koennen der anschliessend ausgegebenen Meldung DRI0578 entnommen werden. Weitere relevante Fehlermeldungen werden vor dieser Meldung DRI0579 ausgegeben (analog zur Fehlerliste im Programm-Modus).
(&00) : DRIVE-Anweisung

Maßnahme
Mit einer SET-Anweisung kann versucht werden, den Fehler zu beheben. Ist eine Behebung nicht moeglich, muss der Debug-Lauf mit BREAK DEBUG abgebrochen werden.

DRI0580 EINGABEN ZU LANG

Bedeutung
Der Satz ist laenger als die Eingabeliste. Die restlichen Zeichen werden ueberlesen.

Maßnahme
Eingabeliste erweitern.

DRI0581 DATEIAUSGABE ZU LANG

Bedeutung
Bei der Ausgabe auf eine Datei wird die Laenge von 32000 ueberschritten.

Maßnahme
Die Ausgabe auf mehrere Ausgabeanweisungen aufteilen.

DRI0582 NULLWERT NICHT DEFINIERT

Bedeutung
Es soll ein Nullwert auf Datei geschrieben werden, aber es ist kein Nullwert deklariert.

Maßnahme
Bei der Definition der Datei ist ein Zeichen fuer die Darstellung des Nullwertes anzugeben.

DRI0583 DATEI '(&00)' NOCH NICHT GEOEFFNET

Bedeutung
Die logische Datei wurde vom Programm noch nicht geoeffnet.
(&00): Dateiname

Maßnahme
Die logische Datei vor diesem Aufruf eroeffnen.

DRI0584 DATEI '(&00)' SCHON GEOEFFNET

Bedeutung
Die logische Datei wurde vom Programm schon geoeffnet.
(&00): Dateiname

Maßnahme
Die logische Datei vor diesem Aufruf schliessen.

DRI0585 ZUVIELE DATEIEN EROEFFNET

Bedeutung
Im UTM-Betrieb im BS2000 sind mehr als 20 'isam-shareupd/input'-Dateien eroeffnet.

Maßnahme
Anzahl der eroeffneten Dateien reduzieren.

DRI0586 DATEIENDE ERREICHT

Bedeutung
Es wurde das Ende der Datei erreicht.

DRI0587 EINGABEN ZU KURZ

Bedeutung
Der Satz ist kuerzer als die Eingabeliste.

DRI0588 NAME '(&00)' ZU LANG

Bedeutung
Der Bibliotheks-, Klassenverzeichnis- oder Elementname ist zu lang.
(&00): zu langes Element

DRI0589 ZUVIELE BOXEN SOLLEN ENTFERNT WERDEN

Bedeutung
Bei REPLACE oder REMOVE sollen mehr Boxen entfernt werden, als sich aktuell am Bildschirm befinden.

Maßnahme
Der Wert von 'number' in der REPLACE- oder REMOVE-Anweisung muss verkleinert werden.

DRI0590 AKTION '(&00)' BEREITS BEI ANWEISUNG IN ZEILE (&01) HINTERLEGT

Bedeutung
Die spezifizierte Debug-Aktion wurde bereits an der spezifizierten Zeilennummer hinterlegt und kann kein weiteresmal hinterlegt werden.
(&00): Debug-Aktion
(&01): Zeilennummer

Maßnahme
Naechste Debug-Anweisung eingeben

DRI0591 KEINE AKTIONEN GEFUNDEN

Bedeutung
Bei einer REMOVE-Anweisung wurden keine Aktionen gefunden.

Maßnahme
Naechste Debug-Anweisung eingeben

DRI0592 ZUWEISUNG AN EINZELNE KOMPONENTEN UNZULAESSIG

Bedeutung
In einem Parameter-Prompting beim Debug-Modus ist eine Zuweisung an eine Komponente eines USING-Parameters nicht moeglich.

Maßnahme
Soll beim Parameter-Prompting im Debug-Modus eine Zuweisung an eine strukturierte Variable durchgefuehrt werden, so muss dieses durch eine Aggregatzuweisung erfolgen.

DRI0593 ABLAUFVERFOLGUNG BEENDET: ZEILE (&00) IN PROZEDUR '(&01)'

Bedeutung
Bei einer Trace-Protokollierung auf dem Bildschirm wurde die vorgesehene Anzahl von Einzelschritten durchgefuehrt und dann ein Haltepunkt nach der zuletzt ausgefuehrten Anweisung eingenommen, d. h. die Programmausfuehrung steht nach einer Anweisung in der angegebenen Zeile der angegebenen Prozedur. Eine bei dieser Anweisung ggf. hinterlegte DEBUG-Aktion COUNT wurde bereits ausgefuehrt.
(&00): Zeilennummer
(&01): Prozedurname (inkl. Bibliotheksname)

Maßnahme
Debug-Anweisung eingeben

DRI0594 ANWEISUNG NUR BEI VARIABLEM CURSOR ERLAUBT

Bedeutung
Die Anweisung (z.B. DROP) ist nur erlaubt, wenn der Cursor variabel (d.h. ohne FOR SELECT ... Klausel) deklariert ist.

Maßnahme
- Cursordeklaration aendern
- DROP streichen oder mit EXEC ausfuehren

DRI0595 DIALOGSCHRITTWECHSEL BEI 'REPORT'-VERARBEITUNG UNTER UTM

Bedeutung
Im UTM-Betrieb steht eine Bildschirmausgabe an und es ist noch mindestens ein REPORT zum fuellen geoeffnet.

Maßnahme
Programmablauf ueberpruefen. Vor Bildschirmausgaben unter UTM alle REPORT-Verarbeitungen mit CLOSE REPORT abschliessen.

DRI0596 '(&00)' NUR AUF OBERSTER PROGRAMMSTUFE AUSFUEHRBAR

Bedeutung
Ein Programm ist ausfuehrbar:
- nur per DO-Anweisung oder
- in Auftragnehmerumgebung als die Programmstufe, die direkt vom Auftraggeber aufgerufen wird.

Ein Programm ist nur per DO ausfuehrbar, wenn es z. B. eine STOP-Anweisung enthaelt.
(&00): Programmname ohne Zusatz: Quellprogramm
Programmname mit Zusatz CODE: Zwischencode

DRI0597 ANWEISUNG BEI PREFETCH CURSOR NICHT ERLAUBT

Bedeutung
Die Anweisung (z.B. FETCH PRIOR) ist nicht erlaubt, wenn der Cursor mit PREFETCH deklariert wurde.

Maßnahme
Cursordeklaration aendern.

DRI0598 LAUFENDES PROGRAMM ABGEBROCHEN

Bedeutung
Das laufende DRIVE-Programm wurde auf Anforderung des Benutzers abgebrochen, z.B. ueber den Menueeintrag 'Abbrechen' in der SPU.

DRI7001 Fehler bei Hauptspeicherplatzanforderung.

DRI7002 Systemfehler!

DRI7003 Interner Fehler!

DRI7004 Report Services ist nicht installiert.

DRI7005 Kein Zugriff auf die Fehlermeldungsdatei.

DRI7006 Der angegebene Zeiger bezeichnet kein gueltiges Report-Objekt.

DRI7007 Der angegebene Zeiger bezeichnet kein geoeffnetes Report-Objekt.

DRI7008 Aktuelle Aufruf-Parameter fehlen.

DRI7009 Falsche Laenge des aktuellen Parametersatzes.

DRI7010 Fehler bei Hauptspeicherplatzanforderung.

DRI7011 Systemfehler (Funktion: (&00) Nummer: (&01)).

DRI7012 Interner Fehler in Zeile (&00).

DRI7013 Ein LMS Fehlerist aufgetreten (LMS: (&00) DMS: (&00)).

DRI7014 Interner PMC Fehler.

DRI7015 Fehler beim Zugriff auf NLS-Konstante.

DRI7016 Versions-Fehler im Report-Objekt.

DRI7017 Falscher Parameter.

DRI7018 Die Trennzeichen im Benutzerprofil sind nicht eindeutig.

DRI7019 Die Benutzerfunktion ist nicht definiert.

DRI7020 NOCH NICHT IMPLEMENTIERT!

DRI7021 Die Funktion ist im angegeben Open-Modus nicht erlaubt.

DRI7022 Das zu erzeugende Report-Objekt existiert bereits.

DRI7023 Das zu oeffnende Report-Objekt ist gesperrt.

DRI7024 Kein Zugriff auf Report-Objekt oder Nettodaten-File.

DRI7025 Die Anweisung (&00) ist hier nicht erlaubt.

DRI7026 Die Anweisung (&00) ist bereits vorhanden.

DRI7027 Die Angabe (&00) hier nicht erlaubt.

DRI7028 Die Angabe (&00) ist bereits vorhanden.

DRI7029 (&00) ist ungueltig fuer die Angabe (&01).

DRI7030 Fehler bei Reihenfolge fuer (&00).

DRI7031 Fehlender Name fuer uebergeordnete Struktur.

DRI7032 Das Gruppenfeld ist nicht im Sortier-Kriterium enthalten.

DRI7033 Angabe fehlt.

DRI7034 Das Report-Objekt enthaelt keine gueltige Satzbeschreibung.

DRI7035 Das Report-Objekt enthaelt bereits eine Layoutbeschreibung.

DRI7036 Der angegebene Puffer ist zu klein fuer Wert.

DRI7037 Das Symbol (&00) ist ungueltig.

DRI7038 Das Symbol (&00) ist nicht eindeutig.

DRI7039 Nur Konstante oder lokale Variable sind als Array-Index zulaessig.

DRI7040 Ein Array-Index ist nicht im gueltigen Wertebereich.

DRI7041 Fuer (&00) fehlt eine obligatorische Anweisung.

DRI7042 Die Angabe (&00) fuer (&01) fehlt.

DRI7043 Fuer (&00) sind keine Angaben definiert.

DRI7044 Ungleich viele Detailbeschreibungen und Satzarten sind definiert.

DRI7045 Das Report-Objekt enthaelt keine Layoutbeschreibung.

DRI7046 Keine Satzbeschreibung ist definiert.

DRI7047 Doppelter (&00) derselben Hierarchie ist definiert.

DRI7048 Die Redefinition ueberschreitet die Laenge des redefinierten Feldes.

DRI7049 Fehler bei der Typauswertung eines Ausdrucks.

DRI7050 Typ-Konflikt im Ausdruck.

DRI7051 Falsche Gruppennummer (&00) in Gruppenkopf und/oder -fuss.

DRI7052 Ein Parameter der Benutzerfunktion ist nicht definiert.

DRI7053 Datentyp des Array-Index nicht zulaessig.

DRI7054 Kontrolblock (&00) darf nur einmal definiert werden.

DRI7055 Diese Angabe ist im Seiten-Hintergrundmuster nicht erlaubt.

DRI7056 Die Eingabedaten ueberschreiten die maximale Satzlaenge.

DRI7057 Die Formatbeschreibung ist ungueltig.

DRI7058 NULL_ALLOWED ist fuer dieses Feld nicht definiert.

DRI7059 Die angegebene Laenge des Nettodatensatzes ist falsch.

DRI7060 Die Satzart ist nicht definiert.

DRI7061 Falsche Laenge des Nettodatenfeldes.

DRI7062 Falscher Datentyp des Nettodatenfeldes.

DRI7063 Fehler beim Aufruf einer INFORMIX Konversions-Funktion.

DRI7064 Fehler beim Aufruf einer INFORMIX Arithmetik-Funktion.

DRI7065 Der Detailbereich ist zu klein.

DRI7066 Die Bedingung hat als Ergebnis NULL.

DRI7067 Zeilenbereich fuer den Seitenkopf ist zu klein definiert.

DRI7068 Zeilenbereich fuer den Seitenfuss ist zu klein definiert.

DRI7069 Fuer diesen Kontrollblock sind keine Tabulatoren definiert.

DRI7070 Typ-Inkompatibilitaet bei einer Zuweisung.

DRI7071 Der Datentyp eines Benutzer-Funktions-Parameters ist falsch.

DRI7072 Der Parameter einer Benutzer-Funktion ist nicht deklariert.

DRI7073 Division durch 0 ist nicht erlaubt.

DRI7074 Die numerische String-Konstante enthaelt keine Zahl.

DRI7075 Der Wert ist ausserhalb des Wertebereichs in der Zuweisung.

DRI7076 Fehler in der Benutzerfunktion: (&00).

DRI7077 Laenge der Raster-Grafik-Daten ist falsch.

DRI7078 Die Image-Daten muessen in Hexadezimal-Form sein.

DRI7079 Das Geraeteprofil ist nicht vorhanden.

DRI7080 Die angegebene Ergebnis-Datei kann nicht geoeffnet werden.

DRI7081 Der SINIX-Spool kann nicht aktiviert werden.

DRI7082 Das Geraeteprofil ist nicht vollstaendig definiert.

DRI7083 Im Geraeteprofil fehlt der Name des RDI-Konverters.

DRI7084 Konversionsfehler im Geraeteprofil (Zeile (&00)).

DRI7085 Ungueltiges Zeichen im Geraeteprofil (Zeile (&00)).

DRI7086 Unvollstaendige Zeile im Geraeteprofil (Zeile (&00)).

DRI7087 Ungueltige Zeile im Geraeteprofil (Zeile (&00)).

DRI7088 Ungueltiger Wert im Geraeteprofil (Zeile (&00)).

DRI7089 Zu langer Wert im Geraeteprofil (Zeile (&00)).

DRI7090 Falscher Initialwert im Geraeteprofil (Zeile (&00)).

DRI7091 Ungueltige Metawort-Klasse im Geraeteprofil (Zeile (&00)).

DRI7092 Fehlende Kommandozeile im Geraeteprofil (Zeile (&00)).

DRI7093 Fehlende Kommandozeile im Geraeteprofil (Zeile (&00)).

DRI7094 Fehlende Kommandozeile im Geraeteprofil (Zeile (&00)).

DRI7095 Falsche Kontrollwort-Sequenz im Geraeteprofil (Zeile (&00)).

DRI7096 Fehlende "valid value"-Zeile im Geraeteprofil (Zeile (&00)).

DRI7097 Falsche "valid value"-Liste im Geraeteprofil (Zeile (&00)).

DRI7098 Fehlende "define connected"-Zeile im Geraeteprofil (Zeile (&00)).

DRI7099 Falsche "connected value"-Zeile im Geraeteprofil (Zeile (&00)).

DRI7100 Ungueltige "connected value"-Zeile im Geraeteprofil (Zeile (&00)).

DRI7101 Falscher Inkrementwert im Geraeteprofil (Zeile (&00)).

DRI7102 Metawort im Geraeteprofil nicht definiert (Zeile (&00)).

DRI7103 Nur "CHARTYPE" im Geraeteprofil erlaubt (Zeile (&00)).

DRI7104 Ein falscher Parametertyp im Geraeteprofil (Zeile (&00)).

DRI7105 Defaultwert im Geraeteprofil nicht definiert (Zeile (&00)).

DRI7106 Ein Ersatzwert im Geraeteprofil ist nicht erlaubt (Zeile (&00)).

DRI7107 Ein Fehler wurde vom Spool-System gemeldet.

DRI7108 Ziel und Quelle sind identisch.

DRI7109 Fehlerhafte Parameter-Indikatoren im Geraeteprofil (Zeile (&00)).

DRI7110 Zu viele Parameter-Indikatoren im Geraeteprofil (Zeile (&00)).

DRI7111 Nur "NUMTYPE" im Geraeteprofil eraubt (Zeile (&00)).

DRI7112 Falsches Geraeteprofil oder RDI-Konverter.

DRI7113 Die Eingabedatei (&00) kann nicht geoeffnet werden.

DRI7114 Falsche "magic number" in der Eingabedatei.

DRI7115 Usage: (&00) ... (siehe Manual)!

DRI7116 Das Prologfile ist nicht vorhanden.

DRI7117 Source-Datei hat keine Bounding Box.

DRI7118 Kein Benutzerprofil ist vorhanden.

DRI7119 Syntax-Fehler im Benutzerprofil (Kommando (&00)).

DRI7120 (&00) in dieser Ausbaustufe nicht verfuegbar.

DRI7121 Datenstruktur im Standard-Layout ausgelassen.

DRI7122 Zu viele aktuelle Parameter sind angegeben.

DRI7123 Falscher Datentyp fuer Wiederholungsfaktor eines konstanten Strings.

DRI7124 Die beim Source-Statement angegebene Datei existiert nicht.

DRI7125 Die Formatbeschreibung auf Seite (&00) Zeile (&01) ist zu kurz.

DRI7126 Die Druckposition auf Seite (&00) Zeile (&01) ist ungueltig.

DRI7127 Kein Tabulator auf Seite (&00) Zeile (&01) vorhanden.

DRI7128 Rotation wird vom Ausgabegeraet nicht unterstuetzt.

DRI7129 Eine implizite Sortierung wird durchgefuehrt.

DRI7130 Kein Text - nicht verwendet.

DRI8000 MONTAG/DIENSTAG/MITTWOCH/DONNERSTAG/FREITAG/SAMSTAG/SONNTAG/

Bedeutung
Wochentagsnamen, die bei einer Maskierung als Wochentagsstrings verwendet werden. Maximale Laenge pro String: 40 Byte.

DRI8001 JANUAR/FEBRUAR/MAERZ/APRIL/MAI/JUNI/JULI/AUGUST/SEPTEMBER/OKTOBER/NOVEMBER/DEZEMBER/

Bedeutung
Monatsnamen, die bei einer Maskierung als Monatsstrings verwendet werden. Maximale Laenge pro String: 40 Byte.

DRI8002 LISTE DES AKTUELLEN VORGANGS FUER BENUTZER '(&00)' MIT VORGANGSSTATUS '(&01)'

Bedeutung
Listenkopf bei UTM-Druckausgaben.

DRI8003 LISTE ALLER VORGAENGE FUER BENUTZER '(&00)' MIT VORGANGSSTATUS '(&01)'

Bedeutung
Listenkopf bei UTM-Druckausgaben.

DRI8004 VORGANGSDATUM: (&00) VORGANGSZEIT: (&01)

DRI8005 LISTENAUSDRUCK BEENDET AM: (&00)/(&01)

Bedeutung
(&00): Datum
(&01): Uhrzeit.

DRI8010 (NEXT/PRIOR/FIRST/LAST/BREAK=SCROLL;+/-/+-/++/--=IM SATZ BLAETTERN)

Bedeutung
SCROLL = Navigieren innerhalb der Cursormenge:
NEXT: Naechsten Cursorsatz lesen.
PRIOR: Vorherigen Cursorsatz lesen.
FIRST: Ersten Cursorsatz lesen.
LAST: Letzten Cursorsatz lesen.
BREAK: Keinen weiteren Cursorsatz lesen.
Im Satz blaettern:
+ : Naechsten Bildschirm anzeigen.

- : Vorherigen Bildschirm anzeigen.
+- : Aktuellen Bildschirm wiederholen.
++ : Ende des Satzes anzeigen.
-- : Anfang des Satzes anzeigen.

DRI8011 (NEXT/BREAK=SCROLL;+/-/+-/++/--=IM SATZ BLAETTERN)

Bedeutung
SCROLL = Navigieren innerhalb der Cursormenge:
NEXT: Naechsten Cursorsatz lesen.
BREAK: Keinen weiteren Cursorsatz lesen.
Im Satz blaettern:
+ : Naechsten Bildschirm anzeigen.
- : Vorherigen Bildschirm anzeigen
+- : Aktuellen Bildschirm wiederholen
++ : Ende des Satzes anzeigen
-- : Anfang des Satzes anzeigen

DRI8012 (BREAK=ABBRECHEN;+/-/+-/++/--=IM SATZ BLAETTERN)

Bedeutung
Im Satz blaettern:
+ : Naechsten Bildschirm anzeigen
- : Vorherigen Bildschirm anzeigen
+- : Aktuellen Bildschirm wiederholen
++ : Ende des Satzes anzeigen
-- : Anfang des Satzes anzeigen

DRI8013 (NEXT/PRIOR/FIRST/LAST/CURRENT/ABSOLUTE/RELATIVE/BREAK=SCROLL)

Bedeutung
SCROLL = Navigieren innerhalb der Cursormenge:
NEXT: Naechsten Cursorsatz lesen.
PRIOR: Vorherigen Cursorsatz lesen.
FIRST: Ersten Cursorsatz lesen.
LAST: Letzten Cursorsatz lesen.
CURRENT: Aktuellen Cursorsatz nochmal lesen.
RELATIVE: Relatives Positionieren ueber nachfolgend abgefragte Satznummer.
ABSOLUTE: Absolutes Positionieren ueber nachfolgend abgefragte Satznummer.
BREAK: Keinen weiteren Cursorsatz lesen.

DRI8014 (SATZNR)

Bedeutung
Folgemeldung auf RELATIVE- oder ABSOLUTE-Eingabe bei der Meldung DRI8013

Maßnahme
Satznummer fuer FETCH RELATIVE/ABSOLUTE eingeben.

DRI8020 SEITE/ZEILE/QUELLE/NEST/DURCH KOMMANDO/DURCH VOREINSTELLUNG/IM QUELLCODE/ UEBERSETZUNGSOPTIONEN/ANZAHL FEHLER/QUERVERWEISLISTE/PROGRAMM-NAME/ELEMENT/ IDP/START UP/PROGRAMM/

Bedeutung
Angaben fuer Uebersetzungsliste. Maximale Laenge pro String: 40 Byte.

DRI8021 LITERAL/ATTRIBUT-SPEZIFIKATION/PRAEDIKAT/USER-TEXT/

Bedeutung
Maximale Laenge pro String: 40 Byte.

DRI8022 NR/DATUM/ZEIT/ANWENDER/AUS/FEHLERLISTE/FEHLERHINWEISE/GERUFEN VON/DYNAMISCHE PROGRAMM-AUFRUFKETTE/ENDE DER FEHLERLISTE/DISTANZ/BIBLIOTHEKS-TABELLE/

Bedeutung
Maximale Laenge pro String: 40 Byte.

DRI8023 ENDE QUERVERWEISLISTE/VON/VOM TYP/IN/VORDEFINIERT IM DD/BASISTABELLE/ BASISTABELLEN/SATZELEMENT/DATEI/VARIABLE/FORMAT/CALL MOD/BIBLIOTHEK/PROGRAMM/

Bedeutung
Angaben fuer UREF/XREF. Maximale Laenge pro String: 40 Byte.

DRI8024 EDT-DATEI 0/PROGRAMM IN EDT-DATEI 0/SYSTEMBIBLIOTHEK/TASKBIBLIOTHEK/DATA DICTIONARY/VEKTOR/MATRIX/PROGRAMMFEHLER/

Bedeutung
Maximale Laenge pro String: 40 Byte.

DRI8025 FORMATEINGABE/FORMATAUSGABE/ANWEISUNGEN/KONSTANTEN/WERTE/ZWISCHENCODE/ANZAHL/ NAME/GROESSE/SUMME/

Bedeutung
Angaben zur Ausgabe von Tabellengroessen. Maximale Laenge pro String: 40 Byte.

DRI8026 DATEIVERZEICHNIS/OBJEKT/BENUTZER-KENNSATZ/COPY-ELEMENT/KONSTANTE/WINDOW-OBJEKT/IN/DEBUGLISTE/

DRI8027 DIALOG/ALPHA/GRAFIK/UTM-ASYNCHRON/MASKEN/PRUEFBEDINGUNGEN/WINDOW-ATTRIBUTE/

DRI8028 PROZESS NICHT STARTBAR/EDITOR NICHT STARTBAR/VERZEICHNISWECHSEL NICHT DURCHFUEHRBAR/VERZEICHNIS NICHT ERZEUGBAR/PFADNAME ZU LANG/PFADNAME FALSCH/ TILDE NICHT INTERPRETIERBAR/

Bedeutung
Inserttexte, die in Meldungen zu den Editoren eingesetzt werden.

Maßnahme
Bei Aenderungen bitte beachten: maximale Stringlaenge einer Einheit ist 40 Bytes

DRI8029 MASCHINEN FEHLER/BEI ARGUMENT/ASYNCHRON/AUFTRAGNEHMER/AUFTRAGGEBER/BEGINN/ENDE/

Bedeutung
Inserttexte, die in Meldungen zum Ausdrucksberechner eingereiht werden.

Maßnahme
Bei Aenderung bitte maximale Stringlaenge von 40 Zeichen beachten.

DRI8030 VERSION/DIREKTIVE/AUFRUF-SCHLUESSEL/GESAMTLAENGE/LAENGE BIBLIOTHEKSNAME/LAENGE ELEMENTNAME/STUFE/ZAEHLER/

Bedeutung
Inserttexte fuer Meldungen

DRI8100 VORWAERTS

Bedeutung
Ausgabetext eines DRIVE-Systemprogramms

DRI8101 RUECKWAERTS

Bedeutung
Ausgabetext eines DRIVE-Systemprogramms

DRI8102 ABBRECHEN

Bedeutung
Ausgabetext eines DRIVE-Systemprogramms

DRI8103 AUSWAHL DER PARAMETER-ANWEISUNG

Bedeutung
Ausgabetext eines DRIVE-Systemprogramms

DRI8104 EINSTELLEN DER PROTOKOLLIERUNG

Bedeutung
Ausgabetext eines DRIVE-Systemprogramms

DRI8105 FESTLEGEN VON DYNAMISCHEN PARAMETERN

Bedeutung
Ausgabetext eines DRIVE-Systemprogramms

DRI8106 BELEGEN VON K- BZW. F-TASTEN

Bedeutung
Ausgabetext eines DRIVE-Systemprogramms

DRI8107 FESTLEGEN VON STATISCHEN PARAMETERN

Bedeutung
Ausgabetext eines DRIVE-Systemprogramms

DRI8108 SPERREN VON ANWEISUNGEN

Bedeutung
Ausgabetext eines DRIVE-Systemprogramms

DRI8109 VERARBEITUNG

Bedeutung
Ausgabetext eines DRIVE-Systemprogramms.

DRI8110 KEINE AKTION

Bedeutung
Ausgabetext eines DRIVE-Systemprogramms

DRI8111 A

Bedeutung
Ausgabetext eines DRIVE-Systemprogramms

DRI8112 V

Bedeutung
Ausgabetext eines DRIVE-Systemprogramms

DRI8200 /********* DEKLARATION DER BEARBEITETEN WINDOWS *********/

DRI8201 /************** EREIGNISGESTEUERTER ABLAUF **************/

DRI8202 /**************** INITIALISIERUNGSBLOCK ****************/

DRI8203 /******************** PROGRAMMBODY ********************/

DRI8204 /*********** FEHLERROUTINE FUER WINDOW-ERROR ***********/

DRI8205 <Kein Treffer>

Bedeutung
Fuer die Auswahlliste eines Klassenfensters wurde kein passendes Objekt gefunden. Die Liste ist in diesem Fall gesperrt. Ebenfalls gesperrt sind die Funktionen des Menues 'Objekt', bis auf 'Erzeugen als' und 'Schliessen'.

Maßnahme
Mit einer der moeglichen Funktionen weiterarbeiten, z. B. Fenster schliessen, Verzeichnis wechseln, Erzeugen als.

DRI8206 Laufende Prozesse abbrechen (Ja/Nein)

Bedeutung
Waehrend das Klassenfenster geschlossen werden soll, sind noch Prozesse amlaufen, die beim Schliessen abgebrochen werden.

DRI8207 Ja

Bedeutung
Auswahltext zur vorangegangenen Meldung

DRI8208 Nein

Bedeutung
Auswahltext zur vorangegangenen Meldung

DRI8209 <Kein Unterverzeichnis vorhanden>

Bedeutung
Das aktuelle Dateiverzeichnis enthaelt kein Unterverzeichnis. Ein Wechsel des Dateiverzeichnisses ist nur in bzw. ein uebergeordnetes Dateiverzeichnis moeglich.

DRI8210 <Keine Vorgaenge aktiv>

Bedeutung
Es laufen im Hintergrund keine Vorgaenge wie Uebersetzungen, Programmausfuehrungen, etc.

DRI8211 Pfadangabe/Bibliotheksangabe:

Bedeutung
Text fuer Ueberschrift in einer Dialogbox

DRI8212 Neuer Objektname:

Bedeutung
Text fuer Ueberschrift in einer Dialogbox

DRI8213 Alter Objektname:

Bedeutung
Text fuer Ueberschrift in einer Dialogbox

DRI8214 Mindestens zwei Objekte selektieren

Bedeutung
Beim Binden einer Phase muessen mindestens zwei Objekte aus der Objektliste selektiert werden.

DRI9000 ANWEISUNG ERFOLGREICH AUSGEFUEHRT

DRI9010 SATZ VON FREMDER TRANSAKTION GESPERRT

DRI9021 BENUTZERIDENTIFIKATION FALSCH

DRI9022 ZUGRIFFSRECHTE VERLETZT

DRI9100 ENDE DER ERGEBNISTABELLE ERREICHT BZW. ERGEBNISTABELLE LEER

DRI9116 SCHEMA-NAME '(&00)' NICHT KORREKT

DRI9118 SCHEMA-NAME '(&00)' NICHT EINDEUTIG

DRI9119 SCHEMA-NAME '(&00)' ZU LANG

DRI9121 TABELLE '(&00)' NICHT EINFACH

DRI9123 TABELLE '(&00)' KEINE BASISTABELLE

DRI9126 TABELLEN-NAME '(&00)' NICHT KORREKT

DRI9127 TABELLE '(&00)' NICHT DEFINIERT

DRI9128 TABELLEN-NAME '(&00)' NICHT EINDEUTIG

DRI9129 TABELLEN-NAME '(&00)' ZU LANG

DRI9131 SATZELEMENT FEHLERHAFT ANGEGEBEN

DRI9135 FEHLER BEI SATZELEMENT-SPEZIFIKATION

DRI9136 SATZELEMENT-NAME '(&00)' NICHT KORREKT

DRI9137 SATZELEMENT NICHT DEFINIERT

DRI9138 SATZELEMENT-NAME NICHT EINDEUTIG

DRI9139 SATZELEMENT-NAME ZU LANG

DRI9141 CURSOR IST GESCHLOSSEN

DRI9142 CURSOR IST GEOEFFNET

DRI9143 CURSOR NICHT POSITIONIERT

DRI9144 CURSOR KANN NICHT POSITIONIERT WERDEN

DRI9146 CURSOR-NAME '(&00)' NICHT KORREKT

DRI9147 CURSOR NICHT DEFINIERT

DRI9148 CURSOR-NAME '(&00)' NICHT EINDEUTIG

DRI9149 CURSOR-NAME '(&00)' ZU LANG

DRI9156 KORRELATIONS-NAME '(&00)' NICHT KORREKT

DRI9158 KORRELATIONS-NAME NICHT EINDEUTIG

DRI9159 KORRELATIONS-NAME '(&00)' ZU LANG

DRI9210 NULLWERTBEDINGUNG VERLETZT

DRI9220 EINDEUTIGKEITSBEDINGUNG VERLETZT

DRI9230 REFERENZBEDINGUNG VERLETZT

DRI9300 INDIKATORVARIABLE NICHT ZULAESSIG

DRI9310 INDIKATORVARIABLE NICHT SPEZIFIZIERT

DRI9320 ANFRAGE LIEFERT MEHR ALS EINEN TREFFER

DRI9330 WERTELISTE UNVOLLSTAENDIG ODER FEHLERHAFT

DRI9335 WERT FEHLERHAFT

DRI9336 PRIMAERSCHLUESSEL IN 'SET'-KLAUSEL ANGEGEBEN

DRI9337 PRIMAERSCHLUESSEL NICHT VOLLSTAENDIG ANGEGEBEN

DRI9338 MENGENFUNKTION NICHT ZULAESSIG

DRI9339 ZUVIELE ELEMENTE IM AGGREGAT

DRI9340 WERTUEBERLAUF / -UNTERLAUF

DRI9345 DATENTYP FEHLERHAFT

DRI9350 DATENTYP UNVERTRAEGLICH

DRI9360 UNVERTRAEGLICHKEIT BEI KONVERTIERUNG

DRI9365 SATZELEMENT IN DER MENGENFUNKTION FEHLERHAFT

DRI9370 MUSTER IN DER 'LIKE'-KLAUSEL NICHT KORREKT

DRI9371 NUR EIN ELEMENT IN DER 'IN'-KLAUSEL

DRI9372 STANDARD-WERT NICHT ZULAESSIG

DRI9380 FEHLER IN DER BEDINGUNG

DRI9384 MEHR ALS 2 TABELLEN ANGESPROCHEN

DRI9385 MEHR ALS 6 SORT-KRITERIEN

DRI9390 KEINE SYNONYME FUER SATZELEMENTE DES VIEWS ANGEGEBEN

DRI9400 FEHLER IN 'ORDER BY'-KLAUSEL

DRI9420 FEHLER IN 'GROUP BY'-KLAUSEL

DRI9440 FEHLER IN 'INTO'-KLAUSEL

DRI9450 SATZ VON EIGENER TRANSAKTION BEREITS GELOESCHT

DRI9500 ANGABE DES OBJEKTS IN 'SHOW'-ANWEISUNG FEHLERHAFT

DRI9600 ANWEISUNGSREIHENFOLGE IST FEHLERHAFT

DRI9630 POSITIONIERANGABE UNZULAESSIG

DRI9700 KURZFRISTIGE ZUGRIFFSSPERRE

DRI9701 ABBRUCH WEGEN 'CANCEL' ODER 'INTR'-AUFRUF

DRI9702 ABBRUCH WEGEN FEHLER BEIM SORTIEREN

DRI9710 DATENBANKSYSTEM KURZFRISTIG UEBERLASTET

DRI9720 EIN-/AUSGABE-FEHLER IM DATENBANKSYSTEM

DRI9730 TASK-DEADLOCK

DRI9740 VORGANG UNBEKANNT WEGEN ADMINISTRATIONS-EINGRIFF ODER ENGPASS

DRI9745 VORGANG UNBEKANNT WEGEN DBMS-NEUSTART

DRI9750 INSERT-ANWEISUNG AUF ANGEGEBENE BASISTABELLE UNZULAESSIG

DRI9760 UPDATE-ANWEISUNG AUF ANGEGEBENE BASISTABELLE UNZULAESSIG

DRI9770 ZUGRIFF AUF BASISTABELLE UNZULAESSIG

DRI9775 FEHLER BEIM EROEFFNEN DER DATENBANK

DRI9780 ZUGRIFF AUF SCHEMA UNZULAESSIG

DRI9785 ZUGRIFF AUF SATZELEMENT UNZULAESSIG

DRI9790 INNERHALB EINER TA AUF ZWEI SCHEMATA EINER DATENBANK ZUGEGRIFFEN

DRI9800 ZUGRIFF VORUEBERGEHEND UNZULAESSIG

DRI9810 AENDERUNGS-ZUGRIFF UNZULAESSIG

DRI9820 DATENBANKSYSTEM WURDE NORMAL VOM ADMINISTRATOR BEENDET

DRI9830 DBH NOCH NICHT BZW. NICHT MEHR VORHANDEN

DRI9840 SYSTEMKOMPONENTE FUER DATENBANKSYSTEM NICHT VERFUEGBAR

DRI9850 WEGEN ADMINISTRATIONS-EINGRIFF ZUR ZEIT NEUE TA UNZULAESSIG

DRI9860 FEHLER IN KONFIGURATIONSDATEI

DRI9900 PROGRAMMIERFEHLER IM DATENBANKSYSTEM

DRI9910 DATENBANK-INTEGRITAET VERLETZT ODER PROGRAMMIERFEHLER

DRI9920 SYSTEMGRENZEN DES DATENBANKSYSTEMS ERREICHT

DRI9930 FEHLER IN ANWEISUNGS-REPRAESENTATION

DRI9940 ANWEISUNG/ANWEISUNGSKLAUSEL NICHT IMPLEMENTIERT

DRI9990 FEHLERHAFTE OPERATION

Returncodes von SQL

SESAM V1 und UDS

SQL-Codes werden von den Datenbank-Systemen SESAM V1 und UDS übernommen und von DRIVE/WINDOWS als Fehlermeldung in der Form DRI9xxx ausgegeben. Bei SQL-Code -121 wird also die DRIVE-Meldung DRI9121 ausgegeben.

Beim Auftreten der Fehler mit dem SQL-Code -920 (Fehler in der Anweisungsrepräsentation) und -990 (Fehlerhafte Operation) wird DRIVE/WINDOWS abgebrochen.

Die SQL-Returncodes finden Sie in der SQL-Sprachbeschreibung des jeweiligen Datenbanksystems.

SESAM V2

SQL-Codes des Datenbanksystems SESAM V2 entnehmen Sie bitte dem Handbuch SESAM/SQL-Server: Meldungen [24]

7 Anhang

Namenskonventionen

Für Namen dürfen Buchstaben, Ziffern und der Tiefstrich (_) angegeben werden. Groß- und Kleinbuchstaben werden nicht unterschieden.

Jeder Name muß mit einem Buchstaben beginnen und darf nur bei UDS-Datenbanken mit einem Tiefstrich enden.

Namen müssen den Konventionen des jeweiligen Systems genügen:

INFORMIX-Namen müssen den INFORMIX-Konventionen genügen,

SESAM-Namen den SESAM-Konventionen,

UDS-Namen den UDS-Konventionen,

IFG-Formatnamen den IFG-Konventionen,

FHS-Formatnamen den FHS-Konventionen,

BS2000-Dateinamen den BS2000-Konventionen,

SINIX-Dateinamen den SINIX-Konventionen,

MS-Windows-Dateinamen den MS-Windows-Konventionen,

Modulnamen den Konventionen des Bibliothekssystems.

Innerhalb einer DRIVE-Sitzung müssen die Namen benennbarer, gleichartiger Objekte eindeutig sein. Z.B. dürfen nicht zwei Prozeduren gleichzeitig aktiv sein, die den gleichen Namen haben.

benutzergruppe

Name für eine Anwendergruppe (max. 8 Zeichen). Der Name muß den UDS-Konventionen genügen.

Benutzerkennsatz

Mit einem Benutzerkennsatz kann für einen DRIVE-Anwender ein Vorlauf-Programm gestartet werden. Der Benutzerkennsatz ist aufgebaut aus dem Transaktionscode (TAC) und der UTM-Benutzerkennung (USER). Der Benutzerkennsatz darf max. 31 Zeichen lang sein.

bibliothek

In einem BS2000-System:

Name einer DRIVE-Bibliothek (max. 54 Zeichen). Enthält der Name Sonderzeichen, so muß er in Anführungszeichen (") gesetzt werden. *bibliothek* kann auch der Dateikettungsname der DRIVE-Bibliothek (entsprechend den BS2000-Konventionen) sein. DRIVE/WINDOWS interpretiert *bibliothek* zuerst als Dateikettungsnamen, dann als Bibliotheksnamen.

Die DRIVE-Bibliothek *bibliothek* kann mit der Anweisung PARAMETER DYNAMIC LIBRARY voreingestellt werden.

In einem SINIX-System:

Relativer oder absoluter Pfadname eines Dateiverzeichnisses, das die DRIVE-Bibliothek darstellt (max. 54 Zeichen).

Ein relativer Pfadname bezieht sich auf das Dateiverzeichnis, in dem DRIVE/WINDOWS gestartet wurde.

Die Angaben *bibliothek* und *elemname* werden mit der Angabe *class_name* aus der Anweisung PARAMETER DYNAMIC CLASS oder mit der Angabe der Standard-Vorbelegung zu einem Datei-Pfadnamen ergänzt: *bibliothek/class_name/elemname*.

Die DRIVE-Bibliothek *bibliothek* kann mit der Anweisung PARAMETER DYNAMIC LIBRARY voreingestellt werden.

In einem MS-Windows-System:

Relativer oder absoluter Pfadname eines Dateiverzeichnisses, das die DRIVE-Bibliothek darstellt (max. 54 Zeichen).

Ein relativer Pfadname bezieht sich auf das Dateiverzeichnis, in dem DRIVE/WINDOWS gestartet wurde.

Bei absoluten Pfadnamen ist die Angabe eines Laufwerks erlaubt.

Die Angaben *bibliothek* und *elemname* werden mit der Angabe *class_name* aus der Anweisung PARAMETER DYNAMIC CLASS oder mit der Angabe der Standard-Vorbelegung zu einem Datei-Pfadnamen ergänzt: *bibliothek\class_name\elemname*.

Die DRIVE-Bibliothek *bibliothek* kann mit der Anweisung PARAMETER DYNAMIC LIBRARY voreingestellt werden.

catalog

Name einer SESAM V2-Datenbank (max. 18 Zeichen).

cursor

Name eines Cursors (max. 18 Zeichen.

Wird *cursor* von DRIVE/WINDOWS vergeben, kann er die Werte DRIVE0000000001 bis DRIVE999999999999 erhalten. Innerhalb einer Übersetzungseinheit müssen alle Cursornamen eindeutig sein.

elemname

In einem BS2000-System:

Name eines Elements in der DRIVE-Bibliothek (max. 31 Zeichen).

In einem SINIX-System:

Name einer Datei, die ein Element in einer DRIVE-Bibliothek bestimmt (max. 31 Zeichen).

(max. 8 Zeichen). Die Dateinamenserweiterung besteht aus bis zu 4 Zeichen, wobei das erste Zeichen ein Punkt ist.

In einem MS-Windows-System:

Name einer Datei, die ein Element in einer DRIVE-Bibliothek bestimmt (max. 8 Zeichen). Die Dateinamenserweiterung kann zusätzlich aus bis zu 4 Zeichen bestehen, wobei das erste Zeichen ein Punkt ist.

filename

In einem BS2000-System:

In der Anweisung PARAMETER DYNAMIC LOGFILE die Namenserweiterung der Dialog-Protokolldatei (max. 20 Zeichen).

In der Anweisung LIST ... [INTO FILE] die Namenserweiterung der zentralen Druckdatei (max. 20 Zeichen).

In einem SINIX-System:

Relativer oder absoluter Pfadname einer Datei (max. 54 Zeichen).

Ein relativer Pfadname bezieht sich auf das Dateiverzeichnis, in dem DRIVE/WINDOWS gestartet wurde.

In einem MS-Windows-System:

Relativer oder absoluter Pfadname einer Datei (max. 54 Zeichen).

Ein relativer Pfadname bezieht sich auf das Dateiverzeichnis, in dem DRIVE/WINDOWS gestartet wurde.

formatbibliothek

Name der Formatbibliothek (max. 54 Zeichen).

formatname

Name des DRIVE-Formats (max. 31 Zeichen).

listname

Name des Listenformats (max. 31 Zeichen).

passwort

In der Anweisung PARAMETER DYNAMIC LOGPASSWORD das Kennwort für die Dialog-Protokolldatei (max. 4 Zeichen).

In der Anweisung PERMIT ... PASSWORD die Benutzeridentifikation für ein Schema einer UDS- oder SESAM-Datenbank (UDS: max. 48 Zeichen, SESAM max. 3 Zeichen).

pfadname

In einem SINIX-System:

Absoluter oder relativer Pfadname einer Datei (max. 254 Zeichen).

Ein relativer Pfadname bezieht sich auf das Dateiverzeichnis, in dem DRIVE/WINDOWS gestartet wurde.

In einem MS-Windows-System:

Absoluter oder relativer Pfadname einer Datei (max. 254 Zeichen).

Ein relativer Pfadname bezieht sich auf das Dateiverzeichnis, in dem DRIVE/WINDOWS gestartet wurde.

Bei absoluten Pfadnamen ist die Angabe eines Laufwerks erlaubt.

progname

Programmname (max. 31 Zeichen). Er muß nicht identisch sein mit dem Elementnamen, unter dem das Programm gespeichert ist.

Bei Namen für externe Unterprogramme gelten die Regeln der jeweiligen Programmiersprache, in der das Programm geschrieben ist. Der Name darf keine reservierten Wörter der Unterprogrammsprache oder Schlüsselwörter von DRIVE/WINDOWS enthalten.

In einem SINIX-System muß der Name eines Unterprogramms exakt seinem Aufrufnamen im Rahmenprogramm entsprechen, z.B. auch in der Groß- und Kleinschreibung.

Der Name für Programme mit Zugriffen auf UDS-Datenbanken muß aus Großbuchstaben bestehen, da DRIVE/WINDOWS Cursor- und Viewnamen mit einem Präfix für das Programm versieht und UDS für Namen nur Großbuchstaben zuläßt.

satzelement

Name eines Satzelements in einer Tabelle (max. 31 Zeichen). Innerhalb einer Tabelle müssen alle Satzelementnamen eindeutig sein. In verschiedenen Tabellen dürfen identische Namen vergeben werden. Bei identischen Namen muß zur eindeutigen Identifizierung der Tabellenname mitangegeben werden: *tabelle.satzelement*.

schemaname

Name eines Schemas einer UDS-, SESAM V1 oder V2-Datenbank (UDS: max. 30 Zeichen, SESAM V1: max. 18 Zeichen, SESAM V2: max. 31 Zeichen).

screenformat

Name eines FHS-Teilformats (max. 7 Zeichen).

string

Zeichenfolge (max. 256 Zeichen), deren Datentyp alphanumerisch ist. *string* muß von Hochkommas (') eingeschlossen werden.

subprogname

Name eines internen Unterprogramms (max. 31 Zeichen).

synonym

In Verbindung mit einem *select-ausdruck* ein Name für eine Basistabelle oder für einen View. Dieses Synonym gilt nur für die Dauer des SELECTs. Der Name darf maximal 18 Zeichen lang sein.

Synonyme können insbesondere dazu verwendet werden, kürzere oder sprechendere Namen für eine Tabelle zu definieren.

tabelle

Name einer Basistabelle oder einer Ergenistabelle wie View, Cursor. Der Name einer Basistabelle darf bei SESAM V1-Datenbanken maximal 18 Zeichen lang sein, bei SESAM V2 max. 31 und bei UDS-Datenbanken max. 30 Zeichen. Der Name eines Views darf maximal 18 Zeichen lang sein.

tabelle muß zur eindeutigen Identifizierung angegeben werden, wenn in einer Anweisung identische Namen für Satzelemente/Spalten aus verschiedenen Tabellen vorkommen. *tabelle* bezeichnet eine Basistabelle oder einen View. Wird *tabelle* mit [*schemaname*.] qualifiziert, so handelt es sich um eine Basistabelle. Bei UDS ist diese Basistabelle im SQLU-Schema schemaname definiert. Bei SESAM gilt für diese Basistabelle, daß *schemaname* und Tabellenname identisch sein müssen.

Bezeichnet *tabelle* eine Basistabelle, muß *tabelle* bei UDS in der Form [*schemaname*.] *basistabelle* angegeben werden. Bezeichnet *tabelle* einen View, darf kein *schemaname* angegeben werden.

username

Name, der in der Systemvariablen &USER hinterlegt wird. Er darf maximal 8 Zeichen lang sein.

Im TIAM-Betrieb kann *username* nur vergeben werden, solange noch keine SQL-Anweisung eingegeben wurde. Danach wird für *username* die TSN eingetragen. USER darf nur im TIAM-Betrieb explizit angegeben werden.

Im UTM-Betrieb wird der bei KDCSIGN angegebene Name in &USER eingetragen.

variable

Name einer einfachen Variablen oder einer Komponente einer strukturierten Variablen. Die Komponenten können mit qualifiziertem Namen ("." oder "*") angegeben werden. Der Name von *variable* muß das Präfix "&" haben und darf max. 32 Zeichen lang sein (einschließlich "&").

varname

Name einer einfachen Variablen. Sie muß das Präfix "&" haben und darf max. 32 Zeichen lang sein (einschließlich "&").

view

Name eines Views (max. 18 Zeichen). Innerhalb einer Übersetzungseinheit müssen alle Namen von Views eindeutig sein und sich insbesondere von Namen für Basistabellen unterscheiden.

Schlüsselwörter

Die folgende Liste zeigt die Wörter, die als Schlüsselwörter reserviert sind, und – soweit vorhanden – ihre Abkürzung. Die Wörter dürfen nicht als Namen in Anweisungen verwendet werden.

Schlüsselwort	**Abkürzung**
$PI	
ABS	
ABSOLUTE	
ACCELERATOR	
ACCOUNT	
ACQUIRE	
ACTION	
ACTIVATE	ACT
ADD	
ALARM	
ALIGNMENT	
ALL	
ALTER	
AND	
ANGLE	
ANSI	
ANY	
APPLICATION	APPL
AS	
ASCENDING	ASC
AT	
ATTRIBUTE	ATTR
AUTHORIZATION	
AVG	
BACKGROUND	
BASE	
BEFORE	

Schlüsselwort	Abkürzung
BEGIN	
BETWEEN	
BIN	
BLANK	
BOLD	
BLUE	
BORDER	
BOTH	
BOTTOM	
BOX	
BREAK	
BSSYSTEM	
BS2000	
BTITLE	
BTYPE	
BUFFERED	
BUTTON	
BWIDTH	
BY	
C	
CALL	
CANCEL	
CAPITAL	
CASE	
CATALOG	
CENTER	
CHARACTER	CHAR
CHARLENGTH	CHARLN
CHARTYPE	
CHECK	
CHOICE	

Schlüsselwort	Abkürzung
CLASS	
CLEAR	
CLICK	
CLIPPED	
CLOSE	
CLUSTER	
CM	
COBOL	
CODE	
COLOUR	COLOR
COLUMN	
COLUMNS	
COMBO	
COMMIT	
COMMITTED	
COMPILE	
COMPRESS	
COMTRACE	
CONCAT	
CONNECT	
CONSISTENCY	CONSIS
CONSTANT	
CONSTRAINT	
CONTINUE	CON
COPY	
COPYSOURCE	
COS	
COUNT	
CREATE	CRE
CROSS	
CURRENT	

Schlüsselwort	Abkürzung
CURSOR	
CURSORS	
CYAN	
CYCLE	
DATA	
DATABASE	
DATE	
DATETIME	
DAY	
DAYS	
DBA	
DBSERVER	
DBSYSTEM	
DBTRACE	
DBUTRACE	
DCSYSTEM	
DEACTIVATE	
DEBUG	
DECFLOAT	
DECIMAL	DEC
DECIMALSIGN	DECSIGN
DECLARE	DCL
DEFAULT	DEF
DELETE	DEL
DELSTRING	DELSTR
DENSITY	
DESCENDING	DESC
DESELECT	
DETAIL	
DEVICE	
DEVICETABLE	DEVTAB

Schlüsselwort	Abkürzung
DIAGNOSIS	DIAG
DIALOG	
DICTIONARY	DD
DIRTY	
DISPATCH	
DISPLAY	
DISTANCE	
DISTINCT	DIST
DISTRIBUTION	DIS
DMSTRACE	
DO	
DOUBLE	
DROP	
DUPLICATES	
DYNAMIC	DYN
EDITABLE	
EDITOR	
EDT	
ELEMENT	
ELSE	
EMPTY	
END	
ENTER	
ERROR	
ERRORATTRIBUTE	ERRATTR
ESCAPE	
EXCLUSIVE	
EXECUTE	EXEC
EXISTS	
EXIT	
EXP	

Schlüsselwort	Abkürzung
EXPANSION	
EXPERT	
EXPLAIN	
EXTEND	
EXTENDED	
EXTENT	
EXTERNAL	
FETCH	F
FILE	
FILL	
FILLER	
FILTER	
FIRST	
FIRSTPAGE	
FLOAT	
FLUSH	
FOCUS	
FONT	
FOR	
FOREGROUND	
FOREIGN	
FORM	
FORMAT	
FORMLIB	
FRACTION	
FRACTIONS	
FREE	
FROM	
FULL	
FUNCTION	
GET	

Schlüsselwort	Abkürzung
GLOBAL	
GRANT	
GRAPHICEDITOR	
GREEN	
GROUP	
GROUPNUM	
HARDCOPY	HC
HAVING	
HEADER	
HEIGHT	
HELP	
HIGHINTENSITY	HINT
HOLD	
HORIZONTAL	
HOUR	
HOURS	
ICON	
IF	
IMAGE	
IN	
INCH	
INDEX	
INDICATOR	IND
INFORMIX	
INIT	
INNER	
INOUT	
INPUT	
INSERT	INS
INTEGER	INT
INTERVAL	IV

Schlüsselwort	Abkürzung
INTO	
INTTRACE	
INVALID	
INVERSE	
INVISIBLE	INVIS
IOTRACE	
IS	
ISAM	
ISOLATION	
ITALIC	
ITEM	
ITEMS	
JOIN	
KEY	
KFKEY	
LANDSCAPE	
LAST	
LASTPAGE	
LAYOUT	
LEASY	
LEFT	
LENGTH	
LETTERS	
LEVEL	
LG	
LIBRARY	LIB
LIKE	
LINE	
LINES	
LIST	
LISTING	

Schlüsselwort	Abkürzung
LISTTYPE	
LN	
LOCATE	
LOCK	
LOG	
LOGFILE	
LOGPASSWORD	LOGPSW
LOWERSTRING	
LTERM	
LTYPE	
LWIDTH	
MAGENTA	
MANAGE	
MANDATORY	
MARGIN	
MASK	
MATCHES	
MAX	
MEMORY	MEM
MEMTRACE	
MESSAGE	MSG
MIN	
MINIMUM	
MINUTE	
MINUTES	
MNEMONIC	
MODE	
MODIFY	
MODULE	
MODULO	
MONEY	

Schlüsselwort	Abkürzung
MONINFO	
MONTH	
MONTHS	
MOVE	
MSGSTRING	MSGSTR
MUST	
NAMES	
NATIONAL	
NEED	
NEW	
NEWLINE	NL
NEWPAGE	NP
NEXT	
NOCHECK	
NOCOLOUR	NOCOLOR
NOCURSOR	NOCURS
NOINIT	
NOINVERS	
NORMALINPUT	NORMIN
NORMALINTENSITY	
NORMSQL	
NOT	
NOUNDERLINE	NOUL
NULL	
NULLVALUE	
NUMBER	
NUMERIC	NUM
NUMFLOAT	
NUMTYPE	
OBJECT	
OBJECTNAME	

Schlüsselwort	Abkürzung
OF	
OFF	
OK	
OLDSTYLE	
ON	
ONLY	
OPEN	
OPTION	
OR	
ORDER	
OUT	
OUTER	
OUTIN	
OUTPUT	
OVERLAY	
PAGE	
PAPER	
PARAMETER	PAR
PARENT	
PASSWORD	PSW
PATTERN	
PERMANENT	PERM
PERMISSION	
PERMIT	
PIXMAP	
PORTRAIT	
POSITION	
POSITIONED	
POTMUST	
PRAGMA	
PRECISION	

Schlüsselwort	Abkürzung
PRECOMOPT	
PRECOMPILE	
PREFETCH	
PRESELECT	
PRESSED	
PRIMARY	
PRINT	
PRIOR	
PRIVILEGES	
PROCEDURE	PROC
PROMPT	
PROPORTIONAL	
PUBLIC	
PROTECTED	PROT
PUBLIC	
PUT	
READ	
REAL	
RECORD	
RED	
REDEFINES	REDEF
REFERENCES	
RELATIVE	
REMOTE	
REMOVE	
RENAME	
REPEAT	R
REPEATABLE	
REPLACE	
REPORT	
RESET	

Schlüsselwort	Abkürzung
RESOLUTION	
RESOURCE	
RESOURCELIB	
REST	
RESTART	
RESTORE	
RESULT	
RETURN	RET
REVERSE	
REVOKE	
RIGHT	
ROLLBACK	
ROLLFORWARD	
ROTATION	
ROUND	
ROW	
ROWCOL	
ROWID	
ROWS	
SAM	
SAVE	
SCHEMA	
SCREEN	
SCREENCHECK	
SCREENERROR	SCREENERR
SCRIPT	
SCROLL	
SEARCHED	
SECOND	
SECONDS	
SELECT	S

Schlüsselwort	Abkürzung
SELECTABLE	
SELECTED	
SEND	
SEQUENCE	SEQ
SERIAL	
SERIALIZABLE	
SESAM	
SESAMSQL	
SESSION	
SET	
SETVALUE	
SHARE	
SHIFTLEFTSTRING	SLSTR
SHOW	
SIDEINFO	
SIGN	
SIN	
SINIX	
SITENAME	
SIZE	
SMALLINT	SMINT
SOME	
SORT	
SOURCE	
SPACE	
SPACING	
SPECIAL	
SQLCODE	
SQR	
SQRT	
STABILITY	

Schlüsselwort	Abkürzung
STANDARD	STD
START	
STATIC	
STATISTICS	
STATUS	
STOP	
STORE	
SUBPROCEDURE	SUBPROC
SUBSCRIPT	
SUBSTRING	SUBSTR
SUM	
SUPERSCRIPT	
SYNONYM	
SYSTEM	
TABLE	
TABLES	
TABULATOR	TAB
TAC	
TAN	
TASKTYPE	
TEMPORARY	TEMP
TERMINAL	
TERMINATE	TERM
TEST	
TEXT	
THEN	
TIAM	
TIME	
TIMESTAMP	
TITLE	
TO	

Schlüsselwort	Abkürzung
TODAY	
TOGGLE	
TOP	
TRACE	T
TRAILER	
TRANSACTION	TA
TRSTRING	
TRUNC	
TTITLE	
TYPE	
UDS	
UNCHANGED	
UMCOMMITTED	
UNDERLINE	UL
UNION	
UNIQUE	
UNITS	
UNPROTECTED	UNPR
UNSAVE	
UPARROW	
UPDATE	UPD
UPDSTRING	UPDSTR
UPPERSTRING	
UREF	
USE	
USER	
USEREVENT	USEV
USERGROUP	
USERLABEL	
USERMSGFILE	
USERNAME	

Schlüsselwort	Abkürzung
USING	
UTM	
UTMRC	
VALID	
VALUE	
VALUES	
VARCHAR	
VARIABLE	VAR
VARYING	
VERSIONMIX	
VERTICAL	
VIEW	
VIEWS	
VISIBLE	VIS
WAIT	
WEEKDAY	
WHENEVER	
WHERE	
WHILE	
WHITE	
WIDTH	
WINDOW	
WITH	
WITHOUT	
WORK	
WRITE	
XDEC	
XREF	
YEAR	
YEARS	
YELLOW	

Literatur

[1] **DRIVE/WINDOWS** (BS2000)
Programmiersystem
Benutzerhandbuch

Zielgruppe
Anwendungsprogrammierer
Inhalt
Einführung in das Programmiersystem DRIVE/WINDOWS und Erläuterung der Funktionen des Dialog-Modus' sowie Beschreibung der Installation, der Generierung und der Administration von DRIVE/WINDOWS

[2] **DRIVE/WINDOWS** (BS2000)
Programmiersprache
Sprachbeschreibung

Zielgruppe
Anwendungsprogrammierer
Inhalt
Beschreibung der Programmerstellung einschließlich Bildschirm- und Listenformaten und Reports. Beschreibung des Transaktionskonzepts und der Verteilten Transaktionsverarbeitung. Beispiele.

[3] **DRIVE/WINDOWS** (BS2000)
Lexikon der DRIVE-Anweisungen
Referenzhandbuch

Zielgruppe
Anwendungsprogrammierer
Inhalt
Syntax und Funktionsumfang aller DRIVE-Anweisungen, Meldungen und Schlüsselwörter von DRIVE/WINDOWS

[4] **DRIVE/WINDOWS** (BS2000/SINIX)
Lexikon der DRIVE-SQL-Anweisungen für SESAM V1.x
Referenzhandbuch

Zielgruppe
Anwendungsprogrammierer
Inhalt
Syntax und Funktionsumfang aller DRIVE-SQL-Anweisungen für SESAM V1.x in Kurzform.

[5] **DRIVE/WINDOWS** (BS2000/SINIX)
Lexikon der DRIVE-SQL-Anweisungen für SESAM V2.x
Referenzhandbuch

Zielgruppe
Anwendungsprogrammierer
Inhalt
Syntax und Funktionsumfang aller DRIVE-SQL-Anweisungen für SESAM V2.x in Kurzform.

[6] **DRIVE/WINDOWS** (BS2000/SINIX)
Lexikon der DRIVE-SQL-Anweisungen für UDS
Referenzhandbuch

Zielgruppe
Anwendungsprogrammierer
Inhalt
Syntax und Funktionsumfang aller DRIVE-SQL-Anweisungen für UDS in Kurzform.

[7] **DRIVE/WINDOWS** (MS-Windows)
Software-Produktionsumgebung (SPU)
Benutzerhandbuch

Zielgruppe
Anwendungsprogrammierer
Inhalt
Erläuterung der Funktionen der Software-Produktionsumgebung (Arbeitsplatz). Einsatzvorbereitung für DRIVE/WINDOWS, den Remote-Zugriff auf BS2000- und SINIX-Datenbanken und für Client-Server-Anwendungen.

[8] **DRIVE/WINDOWS** (MS-Windows)
Programmiersprache
Sprachbeschreibung

Zielgruppe
Anwendungsprogrammierer
Inhalt
Beschreibung der Programmerstellung einschließlich Grafik-Bildschirmformaten und Client-Server-Anwendungen.

[9] **DRIVE/WINDOWS** (MS-Windows)
Lexikon der DRIVE-Anweisungen
Referenzhandbuch

Zielgruppe
Anwendungsprogrammierer
Inhalt
Syntax und Funktionsumfang aller DRIVE-Anweisungen, Meldungen und Schlüsselwörter von DRIVE/WINDOWS.

[10] **DRIVE/WINDOWS** (SINIX)
Software-Produktionsumgebung (SPU)
Benutzerhandbuch

Zielgruppe
Anwendungsprogrammierer
Inhalt
Erläuterung der Funktionen der Software-Produktionsumgebung (Arbeitsplatz) und des Expertenmodus. Einsatzvorbereitung für Remote-Zugriff auf BS2000-Datenbanken, für das Erstellen von Anwendungen für das BS2000 und für DRIVE/WINDOWS allgemein.

[11] **DRIVE/WINDOWS** (SINIX)
Programmiersprache
Sprachbeschreibung

Zielgruppe
Anwendungsprogrammierer
Inhalt
Beschreibung der Programmerstellung einschließlich Grafik- und Alpha-Bildschirmformaten sowie Listenformaten mit DRIVE und Report Generator.

[12] **DRIVE/WINDOWS** (SINIX)
Lexikon der DRIVE-Anweisungen
Referenzhandbuch

Zielgruppe
Anwendungsprogrammierer
Inhalt
Syntax und Funktionsumfang aller DRIVE-Anweisungen. Meldungen und Schlüsselwörter von DRIVE/WINDOWS.

[13] **DRIVE/WINDOWS** (SINIX)
Lexikon der DRIVE-SQL-Anweisungen für INFORMIX
Referenzhandbuch

Zielgruppe
Anwendungsprogrammierer
Inhalt
Syntax und Funktionsumfang aller DRIVE-SQL-Anweisungen für INFORMIX in Kurzform.

[14] **DRIVE V5.1** (BS2000)
Teil 1: Benutzerhandbuch

Zielgruppe
– DV-Laien in der Fachabteilung
– Anwendungsprogrammierer
Inhalt
– Allgemeiner Überblick über das System DRIVE in OLD-Style
– Erläuterung der DRIVE-Komponenten
– Beschreibung möglicher Anwendungsfälle anhand einführender Beispiele
– Generierung und Administration von DRIVE im UTM-Betrieb

[15] **DRIVE V5.1** (BS2000)
Teil 2: LEXIKON

Zielgruppe
– DV-Laien in der Fachabteilung
– Anwendungsprogrammierer
Inhalt
– Syntax und Funktionsumfang aller DRIVE-Anweisungen in OLD-Style
– Meldungen und Schlüsselwörter von DRIVE

[16] **DRIVE/WINDOWS-COMP** (BS2000)
Benutzerhandbuch

Inhalt
Beschreibung der Sprachabweichungen zu DRIVE/WINDOWS V1.1 und Darstellung des Compilierungsvorganges. Beschreibung des Generierens und Startens von Anwendungen von compilierten DRIVE-Objekten (TIAM- und UTM-Betrieb) unter besonderer Berücksichtigung des Versionsmischbetriebes.

[17] **SQL für SESAM/SQL**
Sprachbeschreibung

Zielgruppe
Programmierer, die mit SQL-Anweisungen auf SESAM-Datenbanken zugreifen wollen.
Inhalt
SQL-Anweisungen für den Zugriff auf SESAM-Datenbanken.

[18] **SESAM-SERVER** (BS2000/OSD)
SQL-Sprachbeschreibung Teil 1: SQL-Anweisungen
Benutzerhandbuch

Zielgruppe
Zur Zielgruppe gehören alle Personen, die eine SQL-Datenbank mit SQL-Anweisungen bearbeiten.
Inhalt
Das Handbuch beschreibt die Programmeinbettung von SQL-Anweisungen und die SQL-Sprachelemente. In einem alphabetischen Nachschlagteil sind alle SQL-Anweisungen ausführlich dargestellt.

[19] **SESAM/SQL-Server** (BS2000/OSD)
SQL-Sprachbeschreibung Teil 2: Utilities
Benutzerhandbuch

Zielgruppe
Zur Zielgruppe gehören alle Personen, die mit der Verwaltung einer SESAM/SQL-Datenbank befaßt sind.
Inhalt
Das Handbuch enthält eine alphabetische Beschreibung der Utility-Anweisungen; Utility-Anweisungen sind Anweisungen in SQL-Syntax und realisieren die Dienstprogrammfunktionen von SESAM/SQL.

[20] **SESAM/SQL-Server** (BS2000/OSD)
Basishandbuch
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich an alle Anwender und alle, die sich über SESAM/SQL informieren wollen.
Inhalt
Das Handbuch gibt einen Überblick über das Datenbanksystem und beschreibt Grundlagen, Konzepte und Zusammenhänge. Es ist die Basis für das Verständnis der weiteren SESAM/SQL-Handbücher.

[21] **SESAM/SQL-Server** (BS2000/OSD)
SESAM/SQL-Server Utility-Monitor
Benutzerhandbuch

Zielgruppe
Das Handbuch ist für den Datenbankverwalterund den Systemverwalter von SESAM/SQL-Server bestimmt.
Inhalt
Das Handbuch beschreibt die Bedienung des Utility-Monitors. Mit dem Utility-Monitor können DB-Verwaltungs- und Administrationsaufgaben u.a. im maskengeführten Dialog ausgeführt werden.

[22] **SESAM/SQL-Server** (BS2000/OSD)
Umstellen von SESAM-Datenbanken u.-Anwendungen auf SESAM/SQL-Server
Benutzerhandbuch

Zielgruppe
Das Handbuch richtet sich an alle, die sich für SESAM/SQL-Server V2.0 interessieren.
Inhalt
Dieses Handbuch beschreibt die neuen Konzepte und Funktionen im Überblick. Im Vordergrund steht der Bezug zu Vorgängerversionen, um bisherigen SESAM/SQL-Anwendern den Umstieg in die neue Welt von SESAM/SQL-Server V2.0 zu erleichtern.

[23] **SESAM/SQL-Server** (BS2000/OSD)
CALL-DML-Anwendungen
Benutzerhandbuch

Zielgruppe
Das Handbuch richtet sich an alle CALL-DML-Anwendungen-Progammierer.
Inhalt
Es enthält die Beschreibung der CALL-DML-Schnittstelle mit den DML-Anweisungen und dazugehörigen Beispielen. Außerdem sind unter anderem Binden, Laden, Anwenden im Teilhaberbetrieb und die CALL-DML-Dienstprogramme beschrieben.

[24] **SESAM/SQL-Server** (BS2000/OSD)
Meldungen
Benutzerhandbuch

Zielgruppe
Zur Zielgruppe gehören alle SESAM/SQL-Anwender.
Inhalt
Das Handbuch enthält sämtliche Meldungen zu SESAM/SQl nach Meldungsnummern sortiert.

[25] **SQL für UDS/SQL**
Sprachbeschreibung

Zielgruppe
Programmierer, die mit SQL-Anweisungen auf UDS-Datenbanken zugreifen wollen.
Inhalt
SQL-Anweisungen für den Zugriff auf UDS-Datenbanken.

[26] **UDS/SQL** (BS2000)
Verwalten und Bedienen
Benutzerhandbuch

Zielgruppe
Datenbankadministrator
Inhalt
– Verwaltungs- und Bedienungsarbeiten wie Sichern der Datenbank,
– Datenbankbetrieb
– Ausgeben von Datenbankinformationen
– Reorganisieren der Datenbank Datenbankinformationen,
– Konsistenzprüfprogramm
Einsatz
Datenbankadministration im laufenden Betrieb

[27] **UDS/SQL** (BS2000)
Aufbauen und Umstrukturieren
Benutzerhandbuch

Zielgruppe
Datenbankadministrator
Inhalt
– Übersicht über die von UDS benötigten Dateien
– UDS-Dienstprogramme, die zum Aufbauen der UDS-Datenbank nötig sind
– Dienstprogramme zum Umstrukturieren
Einsatz
Datenbankadministrator beim Aufbauen einer Datenbank

[28] **IFG für FHS** (TRANSDATA)
Benutzerhandbuch

Zielgruppe
Datenstationsbenutzer, Anwendungsdesigner und Programmierer
Inhalt
Der Interaktive Formatgenerator (IFG) ist ein System zur komfortablen und einfachen Erstellung und Verwaltung von Formaten an Datensichtstationen. Diese Formate können zusammen mit FHS im Verarbeitungsrechner eingesetzt werden. Das Benutzerhandbuch beschreibt, wie die Formate erstellt, geändert und verwaltet werden sowie die neuen Funktionen von IFG V8.1.

[29] **FHS** (TRANSDATA)
Benutzerhandbuch

Zielgruppe
Programmierer
Inhalt
Programmschnittstellen von FHS für TIAM-, DCAM- und UTM-Anwendungen. Erstellen, Einsatz und Verwalten von Formaten.

[30] **UTM** (BS2000/OSD)
Anwendungen generieren und administrieren
Benutzerhandbuch

Zielgruppe
Organisierer, Einsatzplaner und Administratoren von UTM-Anwendungen.
Inhalt
– Installation von UTM
– Einrichten, Bedienen und Verwalten von UTM-Anwendungen
– UTM-Benutzerkommandos

[31] **UTM** (TRANSDATA)
Anwendungen programmieren
Benutzerhandbuch

Zielgruppe
Programmierer von UTM-Anwendungen
Inhalt
– Sprachunabhängige Beschreibung der Programmschnittstelle KDCS,
– Aufbau von UTM-Programmen
– KDCS-Aufrufe
– Testen von UTM-Anwendungen
– Alle Informationen, die der Programmierer von UTM-Anwendungen benötigt
Einsatz
BS2000-Transaktionsbetrieb

[32] **UTM** (SINIX)
Formatierungssystem

Zielgruppe
UTM(SINIX)-Anwender, die mit Formaten arbeiten wollen, C-Programmierer und COBOL-Programmierer
Inhalt
Einsetzen der Formatsteuerung FORMANT in UTM(SINIX)-Teilprogrammen, Erstellen von Formaten, konvertieren von Formaten zwischen BS2000 und SINIX.

[33] **EDT** (BS2000/OSD)
Anweisungen
Benutzerhandbuch

Zielgruppe
EDT-Einsteiger und EDT-Anwender
Inhalt
Bearbeiten von SAM- und ISAM-Dateien und Elementen aus Programm-Bibliotheken und POSIX-Dateien.

[34] **LMS** (BS2000)
ISP-Format
Beschreibung

Zielgruppe
BS2000-Anwender
Inhalt
Beschreibung der Anweisungen zum Erstellen und Verwalten von PLAM-Bibliotheken und darin enthaltenen Elementen.
Häufige Anwendungsfälle werden an Hand von Beispielen erklärt.

[35] **BS2000/OSD-BC**
Kommandos Band 1 - 3
Benutzerhandbuch

Zielgruppe
Die Handbücher wenden sich sowohl an den nichtprivilegierten Anwender als auch an die Systembetreuung.
Inhalt
Sie enthalten die BS2000/OSD-Kommandos (BS2000/OSD-Grundausbau und ausgewählte Produkte) mit der Funktionalität für alle Privilegien. Die Einleitung gibt Hinweise zur Kommandoeingabe.

[36] **BS2000/OSD-BC**
Systeminstallation
Benutzerhandbuch

Zielgruppe
BS2000/OSD-Systemverwaltung
Inhalt
Das Handbuch beschreibt
– die Generierung der Hardware- und Software-Konfiguration mit UGEN
– die Installationsdienste
  – Plattenorganisation mit MPVS
  – Programmsystem SIR
  – Datenträgerinstallation mit SIR
  – Configuration Update (CONFUPD)
  – Dienstprogramm IOCFCOPY.

[37] **BS2000/OSD-BC**
Einführung in das DVS
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich an den nichtprivilegierten Anwender und an die Systembetreuung.
Inhalt
Es beschreibt die Dateiverarbeitung im BS2000.
Themenschwerpunkte:
- Datei- und Katalogverwaltung
- Dateien und Datenträger
- Datei- und Datenschutz
- OPEN-, CLOSE-, EOV-Verarbeitung
- DVS-Zugriffsmethoden (SAM, ISAM,...)

[38] BS2000
Einführung in die Systemanwendung
Benutzerhandbuch

Zielgruppe
BS2000-Anwender
Inhalt
– Einführung ins BS2000
– Beschreibung der meistgebrauchten Benutzerkommandos bis BS2000 V8.5A
– Einführung in die Benutzung der Dienstprogramme und Softwareprodukte EDT, SORT, ARCHIVE, TSOSLNK, LMS, PERCON
– Hinweise für den programmierenden Benutzer
Einsatz
BS2000-Dialogbetrieb und -Stapelbetrieb

[39] **FORMANT** (SINIX)
Beschreibung

Zielgruppe
– C-Programmierer
– COBOL-Programmierer
– Anwendungsplaner
Inhalt
FORMANT ist eine Maskensteuerung für alle SINIX-Systeme. Das Manual enthält:
– Einführung in FORMANT
– Beschreibung von FORMANTGEN
– Beschreibung der Bedienerschnittstelle
– Programmschnittstellen in C und COBOL
– Beispiele zur Programmierung

[40] **OMNIS** (TRANSDATA, BS2000)
Administration und Programmierung
Benutzerhandbuch

Zielgruppe
– OMNIS-Administrator
– Programmierer
Inhalt
Beschreibung der Grundlagen der Administration von OMNIS, der OMNIS-Dienstprogramme sowie der Anwendungsschnittstelle zur Erweiterung des Funktionsumfanges von OMNIS.

[41] **DRIVE/WINDOWS-COMP** (SINIX)
Compiler
Benutzerhandbuch
Zielgruppe
Anwendungsprogrammierer und Systemverwalter
Inhalt
Beschreibung des Compilierungsvorgangs durch den DRIVE-Compiler.

[42] **INFORMIX-NET** (SINIX)
INFORMIX-STAR (SINIX)
Benutzerhandbuch

Zielgruppe
INFORMIX-Benutzer und Systemverwalter
Inhalt
Das Handbuch beschreibt die Arbeit mit den INFORMIX-Netzprodukten INFORMIX-NET und INFORMIX-STAR.
Mit den INFORMIX-Netzprodukten können INFORMIX-Anwendungen von einem lokalen Rechner aus Datenbanken auf fernen Rechnern erstellen und bearbeiten.

[43] **DRIVE/WINDOWS** (SINIX)
Ergänzungsband
Benutzerhandbuch

Zielgruppe
Anwendungsprogrammierer
Inhalt
Der Ergänzungsband enthält die funktionalen Änderungen von DRIVE/WINDOXS (SINIX) V1.1. Die Handbücher der Version 1.0 werden benötigt.

Bestellen von Handbüchern

Die aufgeführten Handbücher finden Sie mit ihren Bestellnummern im *Druckschriftenverzeichnis* der Siemens Nixdorf Informationssysteme AG. Neu erschienene Titel finden Sie in den *Druckschriften-Neuerscheinungen*.

Beide Veröffentlichungen erhalten Sie regelmäßig, wenn Sie in den entsprechenden Verteiler aufgenommen sind. Wenden Sie sich bitte hierfür an Ihre zuständige Geschäftsstelle. Dort können Sie auch die Handbücher bestellen.

Stichwörter

D

E

Q

R

S

V

Inhalt

DRIVE/WINDOWS (BS2000) V2.1

Lexikon der DRIVE-Anweisungen

Referenzhandbuch

Zielgruppe

Anwendungsprogrammierer

Inhalt

Syntax und Funktionsumfang aller DRIVE-Anweisungen, Meldungen und Schlüsselwörter von DRIVE/WINDOWS

Ausgabe: Februar 1996

Datei: DRV_LEX.PDF

Information on this document

On April 1, 2009, Fujitsu became the sole owner of Fujitsu Siemens Computers. This new subsidiary of Fujitsu has been renamed Fujitsu Technology Solutions.

This document from the document archive refers to a product version which was released a considerable time ago or which is no longer marketed.

Please note that all company references and copyrights in this document have been legally transferred to Fujitsu Technology Solutions.

Contact and support addresses will now be offered by Fujitsu Technology Solutions and have the format *...@ts.fujitsu.com*.

The Internet pages of Fujitsu Technology Solutions are available at *http://ts.fujitsu.com/*...
and the user documentation at *http://manuals.ts.fujitsu.com*.

Hinweise zum vorliegenden Dokument

Zum 1. April 2009 ist Fujitsu Siemens Computers in den alleinigen Besitz von Fujitsu übergegangen. Diese neue Tochtergesellschaft von Fujitsu trägt seitdem den Namen Fujitsu Technology Solutions.

Das vorliegende Dokument aus dem Dokumentenarchiv bezieht sich auf eine bereits vor längerer Zeit freigegebene oder nicht mehr im Vertrieb befindliche Produktversion.

Bitte beachten Sie, dass alle Firmenbezüge und Copyrights im vorliegenden Dokument rechtlich auf Fujitsu Technology Solutions übergegangen sind.

Kontakt- und Supportadressen werden nun von Fujitsu Technology Solutions angeboten und haben die Form *...@ts.fujitsu.com*.

Die Internetseiten von Fujitsu Technology Solutions finden Sie unter *http://de.ts.fujitsu.com/*..., und unter *http://manuals.ts.fujitsu.com* finden Sie die Benutzerdokumentation.